Term Rewriting with Prefix Context Constraints and Bottom-Up Strategies

Florent Jacquemard, Yoshiharu Kojima, Masahiko Sakai

Term Rewriting with Prefix Context Constraints and Bottom-Up Strategies

Florent Jacquemard[1], Yoshiharu Kojima[2], and Masahiko Sakai[2]

[1] INRIA & Ircam, 1 place Igor Stravinsky, 75004 Paris, France
florent.jacquemard@inria.fr
[2] Graduate School of Information Science, Nagoya University
Furo-cho, Chikusa-ku, Nagoya, 464-8603 Japan
{kojima@trs.cm.,sakai@}is.nagoya-u.ac.jp

Abstract. We consider an extension of term rewriting rules with context constraints restricting the application of rewriting to positions whose prefix (*i.e.* the sequence of symbols from the rewrite position up to the root) belongs to a given regular language. This approach, well studied in string rewriting, is similar to node selection mechanisms in XML transformation languages, and also generalizes the context-sensitive rewriting. The systems defined this way are called prefix constrained TRS (pCTRS), and we study the decidability of reachability of regular tree model checking and the preservation of regularity for some subclasses. The two latter properties hold for linear and right-shallow standard TRS but not anymore when adding context constraints. We show that these properties can be restored by restricting derivations to bottom-up ones, and moreover that it implies that left-linear and right-ground pCTRS preserve regularity and have a decidable regular model checking problem.

1 Introduction

Term rewriting systems (TRS) are a rule-based computation model for the definition of ranked trees (*terms*) transformations. In the context of formal verification, they can be used to model the dynamics of a system whose configurations are represented by terms. The rewrite relation represents the transitions between configurations. For instance, functional programs manipulating structured data values with pattern matching can be described by rewrite rules [15,17] such that the rewriting relation represents the program evaluation. This approach can also be applied to communication protocols, distributed algorithms [16], or imperative programs [2,13] modifying some parts of tree shaped data structures in place, while leaving the rest unchanged.

Regular model checking (RMC)[1] is a useful approach for the automatic reachability and flow analysis of programs or systems modeled by TRS. This technique works by constructing an automaton-based finite representation of the set of reachable configurations of the system analyzed, and uses this representation to detect possible erroneous reachable configurations. Tree automata (TA [3]) appear to be appropriate for this purpose. A sufficient condition for the

decision of RMC is the effective *preservation of regularity*: given a TRS $\mathcal{R}$ satisfying some restrictions, and a TA recognizing a set of terms L_{in} which represents initial configurations, can we compute a TA recognizing the rewrite closure of L_{in} by $\mathcal{R}$, *i.e.* the set of reachable configurations? *Static type checking* of XML transformations can sometimes be solved with similar techniques (see *e.g.* [21]).

Standard TRSs are a Turing-complete low-level formalism with a simple definition by pattern matching and subterm replacement: one rewrite rule can be applied at any position in a term, provided that the left-hand-side of the rule matches the subterm at this position in the term. For instance, a rule with left-hand-side $a(x)$ can be applied at positions labelled by a.

For some applications, one may need to add *context conditions* for the application of rewriting, for instance: rename the label a into b at some position π in a term with the rewrite rule $a(x) \to b(x)$, provided that there exist more than one occurrence of b above π. This is analogous to XML node selection in *e.g.* XQuery update[3] expressed by languages such as XPath. Of course, context conditions can be encoded with additional rewrite rules but this way, small programs or systems will have complex TRS representations, making the modeling process tedious and error prone, and the verification with RMC complicated.

The goal of this paper is to study an expressive extension of TRS with context conditions, which eases modeling, while preserving decidability of RMC under restrictions. More precisely, we study a class called pCTRS (prefix controlled TRS) where term rewriting rules are extended with conditions restricting the application of rewriting to positions π whose path (i.e. sequence of symbols and directions from the root down to rewrite position π) belongs to a given regular language. Such context constraints have been studied intensively for string rewriting [26,4] but very few results are known in the case of terms.

First, we show that regularity preservation does not hold with prefix constraints, already for rewrite systems with strong restrictions such as linearity and flatness of left or right hand sides of rules (Section 3.1) which are known to ensure the preservation of regularity in the case of unconstrained TRS [25]. Moreover, for linear TRS whose rule have the form of production rules of context-free tree grammars [3], reachability becomes undecidable when adding prefix constraints (Section 3.2), whereas it is polynomially decidable in the unconstrained case.

We consider next a natural restriction ensuring effective regularity preservation by bottom-up derivations [6] for linear and right-shallow pCTRS (Section 4). Considering bottom-up strategy is quite natural in the context of the applications mentioned above. Left-linear and right-ground pCTRS enforce bottom-up derivations (Section 5), and hence effectively preserve regularity.

2 Preliminaries

Terms. We use the standard notations for terms and positions, see [20]. A *signature* Σ is a finite set of function symbols with fixed arity. We denote the

[3] `http://www.w3.org/TR/xquery-update-10/`

arity of $f \in \Sigma$ as $ar(f)$ and the maximal arity of a symbol of Σ as $max(\Sigma)$. Given an infinite set $\mathcal{X}$ of variables, the set of terms built over Σ and $\mathcal{X}$ is denoted $\mathcal{T}(\Sigma, \mathcal{X})$, and the subset $\mathcal{T}(\Sigma, \emptyset)$ of *ground* terms is denoted $\mathcal{T}(\Sigma)$. The set of variables occurring in a term $t \in \mathcal{T}(\Sigma, \mathcal{X})$ is denoted $var(t)$. A signature is called *unary* (resp. *strictly unary*) if all its symbols have arity at most 1 (resp. arity exactly 1). In the following, given a strictly unary signature Σ, a string $a_1\, a_2 \ldots a_n \in \Sigma^*$ is represented by the term $a_1(a_2(\ldots a_n(x)))$, where $x \in \mathcal{X}$.

A term $t \in \mathcal{T}(\Sigma, \mathcal{X})$ can be seen as a function from its set of positions $\mathcal{P}os(t)$ into $\Sigma \cup \mathcal{X}$. Positions in terms are denoted by sequences of natural numbers, ε is the empty sequence (root position), and $\pi \cdot \pi'$ denotes the concatenation of positions π and π'. The concatenation is naturally extended to sets of positions. The *subterm* of t at position π is denoted $t|_\pi$ defined by $t|_\varepsilon = t$ and $f(t_1, \ldots, t_m)|_{i \cdot \pi} = t_i|_\pi$. The *size* $\|t\|$ of a term t is the cardinality of $\mathcal{P}os(t)$. We write $|s|$ for the length of a finite sequence s. The *depth* of a symbol that occurs in a term at a position π is $|\pi|$. Note that for a string s and its associated term representation t (over a strictly unary signature), $|s| = \|t\| - 1$. A term t is *linear* if no variable occurs more than once in t, *flat* if its depth is at most one and *shallow* if every variable of $var(t)$ occurs at depth at most one in t.

A *substitution* is a mapping from variables of $\mathcal{X}$ into terms of $\mathcal{T}(\Sigma, \mathcal{X})$. It is called *grounding for* $V \subseteq \mathcal{X}$ if the codomain of the restriction $\sigma|_V$ is a set of ground terms. The application of a substitution σ to a term t is denoted as $t\sigma$.

A *context* is a term $C \in \mathcal{T}(\Sigma, \mathcal{X})$ with one distinguished variable x_C occurs exactly once in C. Given a context C and one terms $t \in \mathcal{T}(\Sigma, \mathcal{X})$, we write $C[t]_\pi$ to denote $C\sigma$, where σ is the substitution associating t to x_C. and π is the (unique) position of x_C in C. The notation $s = C[t]_\pi$ may also be used to emphasize that $s|_\pi$ is t.

Controlled Term Rewriting Systems. We propose a formalism that strictly extends standard term rewriting systems by forcing, for every rewrite position π in a term t, the path in t from the root into π to belong to a given regular language. For this purpose we use a notion of path carrying both the labels (in Σ) and directions (in $1..\max(\Sigma)$). More precisely, let $\mathcal{D}ir(\Sigma) = \{\langle g, i\rangle \mid g \in \Sigma, 0 < i \leq ar(g)\}$; we associate with a ground term $t = g(t_1, \ldots, t_{ar(g)}) \in \mathcal{T}(\Sigma)$ and a position $\pi \in \mathcal{P}os(t)$, a *path* in $\mathcal{D}ir(\Sigma)$ defined recursively by

$$path(g(t_1, \ldots, t_{ar(g)}), \varepsilon) = \varepsilon,$$
$$path(g(t_1, \ldots, t_{ar(g)}), i \cdot \pi) = \langle g, i\rangle \cdot path(t_i, \pi) \quad (\text{with } 1 \leq i \leq ar(g)).$$

In the case of unary signatures, we may omit the direction (which is always 1) from the path notation, *i.e.* we write g instead of $\langle g, 1\rangle$.

A *prefix controlled term rewriting system* (pCTRS) over a signature Σ is a finite set $\mathcal{R}$ of prefix controlled *rewrite rules* of the form $L : \ell \to r$, where $L \subseteq \mathcal{D}ir(\Sigma)^*$ is a regular language over $\mathcal{D}ir(\Sigma)$, $\ell \in \mathcal{T}(\Sigma, \mathcal{X}) \setminus \mathcal{X}$ (the left-hand side, or *lhs*), and $r \in \mathcal{T}(\Sigma, var(\ell))$ (the right-hand side, or *rhs*). We use a finite automaton $\mathcal{A}_L$ or a regular expression to present the regular language L.

A term t is rewritten to t' in one step by a pCTRS $\mathcal{R}$, denoted by $t \xrightarrow[\mathcal{R}]{} t'$, if there exist a controlled rewrite rule $L : \ell \to r \in \mathcal{R}$, a position $\pi \in \mathcal{P}os(t)$ such

that $path(t,\pi) \in L$, and a substitution σ such that $t|_\pi = \ell\sigma$ and $t' = t[r\sigma]_\pi$. The reflexive and transitive closure of $\xrightarrow[\mathcal{R}]{}$ is denoted by $t_1 \xrightarrow[\mathcal{R}]{*} t_n$, which we call *a derivation* by $\mathcal{R}$. The size of a pCTRS rule $L : \ell \to r$ is the sum of the sizes of the given automaton $\mathcal{A}_L$ defining the control language L, the lhs ℓ and the rhs r. The size $\|\mathcal{R}\|$ of a pCTRS $\mathcal{R}$ is the sum of the sizes of its rules.

A controlled rewrite rule $L : \ell \to r$ is *ground*, *flat*, *linear*, *shallow* if ℓ and r are so. It is *right-flat*, *etc* (resp. *left-flat*) if r (resp. ℓ) is. It is *collapsing* if $r \in \mathcal{X}$, and otherwise *non-collapsing*. A pCTRS is *flat*, *etc* if all its rules are so.

Example 1. Let us consider the pCTRS

$$\mathcal{R} = \big\{\langle h,1\rangle^+ : a \to c,\ \langle g,2\rangle^+ : b \to d,\ \big(\langle h,1\rangle \mid \langle h,2\rangle\big)^* : h(x,y) \to g(x,y)\big\}.$$

The rewriting $h(a,b) \to h(c,b)$ is possible with the first rule of $\mathcal{R}$, and $h(a,b) \to g(a,b) \to g(a,d)$ with the third and then the second rule of $\mathcal{R}$, but $g(a,b) \to g(c,b)$ is not possible with the first rule of $\mathcal{R}$, because of its control language. $\diamond$

Related Work: TRS with Context Constraints. Standard (uncontrolled) TRSs [20] are particular cases of pCTRSs with rules of the form $\mathcal{D}ir(\Sigma)^* : \ell \to r$. Rewrite systems with context constraints expressed with regular languages have been studied in the case of string rewriting, see [26], and also [4] for the case of conditional context-free (string) grammars.

In [12], we studied a class called CntTRS more general than pCTRS. The context constraints in pCTRS are specified, for each rewrite rule, by a selection automaton which defines a set of positions in a term based on tree automata computations. Reachability is undecidable for ground CntTRS, whereas we show here that it is decidable for left-linear and right-ground pCTRS (Section 5).

Under the *context-sensitive rewriting* [11] the rewrite positions are selected according to priorities on the evaluation of arguments of function symbols. More precisely, let us call CS TRS over Σ a pair $\langle\mathcal{R},\mu\rangle$ made of an uncontrolled TRS $\mathcal{R}$ over Σ and a mapping μ associating to every symbol of Σ the subset of the indexes of its argument that can be rewritten. It means that the positions selected for rewriting in a term $f(t_1,\ldots,t_n)$ are defined recursively as the root position and all the positions selected in every t_i such that $i \in \mu(f)$. In the above definition, a *path* in $\mathcal{D}ir(\Sigma)$ contains information both of the symbols and directions. It follows that CS TRSs are particular case of pCTRSs.

Proposition 1. *For all CS* TRS $\langle\mathcal{R},\mu\rangle$ *over* Σ*, there exists a* pCTRS $\mathcal{R}'$ *over* Σ *such that the rewrite relations defined by* $\langle\mathcal{R},\mu\rangle$ *and* $\mathcal{R}'$ *coincide.*

Consequently, the results below for pCTRSs (Corollary 10) extend to CS TRS.

Automata. A finite (string) automaton (FSA) $\mathcal{B}$ over an alphabet Γ with state set P is presented as a tuple $\langle P, p_0, G, \Theta\rangle$ where $p_0 \in P$ is the initial state (denoted $init(\mathcal{B})$) and $G \subseteq P$ (denoted $final(\mathcal{B})$) and the set of transitions $\Theta \subseteq P \times \Gamma \times P$. A transition $\langle p,a,p'\rangle \in \Theta$ is denoted $p \xrightarrow{a} p'$. The size of $\mathcal{B}$ is $\|\mathcal{B}\| = 3 * |\Theta|$.

A *tree automaton* (TA) $\mathcal{A}$ over a signature Σ is a tuple $\langle Q, F, \Delta \rangle$ where Q is a finite set of nullary state symbols, disjoint from Σ, $F \subseteq Q$ is the subset of final states and Δ is a set of transition rules of the form: $g(q_1, \ldots, q_{ar(g)}) \to q$, or $q_1 \to q$ (ε-transition) where $q_1, \ldots, q_{ar(g)}, q \in Q$. Sometimes, the components of a TA $\mathcal{A}$ are written with $\mathcal{A}$ as subscript, like in $Q_\mathcal{A}$ to indicate that Q is the state set of $\mathcal{A}$. The size of the transition $g(q_1, \ldots, q_{ar(g)}) \to q$ (resp. ε-transition) is $ar(g)+2$ (resp. 2), and the size $\|\mathcal{A}\|$ of $\mathcal{A}$ is the sum of the sizes of its transitions.

The transition set of a TA $\mathcal{A}$ over Σ is an (uncontrolled) ground TRS, hence we can define a TA transition from $s \in \mathcal{T}(\Sigma \cup Q_\mathcal{A})$ into $t \in \mathcal{T}(\Sigma \cup Q_\mathcal{A})$ as a rewrite step, denoted $s \xrightarrow[\mathcal{A}]{} t$. The *language* $\mathcal{L}(\mathcal{A}, q)$ of $\mathcal{A}$ in the state $q \in Q_\mathcal{A}$ is the set of terms $t \in \mathcal{T}(\Sigma)$ such that $t \xrightarrow[\mathcal{A}]{*} q$. A TA $\mathcal{A}$ is called *clean* if for all $q \in Q_\mathcal{A}$, $\mathcal{L}(\mathcal{A}, q) \neq \emptyset$. The language of $\mathcal{A}$ is $\mathcal{L}(\mathcal{A}) = \bigcup_{q \in F_\mathcal{A}} \mathcal{L}(\mathcal{A}, q)$. A set of terms $L \subseteq \mathcal{T}(\Sigma)$ is called *regular* if it is the language of a TA.

Regular (tree) languages are effectively closed by intersection, union and complement. The problems of emptiness (given a TA $\mathcal{A}$, does it hold that $\mathcal{L}(\mathcal{A}) = \emptyset$?) and membership (given a TA $\mathcal{A}$ and a ground term t, does it hold that $t \in \mathcal{L}(\mathcal{A})$?) are decidable in deterministic time respectively linear and quadratic.

Rewrite Closure and Decision Problems. The *rewrite closure* of a set of ground terms L by a pCTRS $\mathcal{R}$ is $\mathcal{R}^*(L) = \{t \mid \exists s \in L, s \xrightarrow[\mathcal{R}]{*} t\}$. *Reachability* is the problem to decide, given two terms $s, t \in \mathcal{T}(\Sigma, \mathcal{X})$ and a pCTRS $\mathcal{R}$ whether $s \xrightarrow[\mathcal{R}]{*} t$. *Regular model checking* (RMC) is the problem to decide, given two regular tree languages L_{in} and L_{err} and a pCTRS $\mathcal{R}$ whether $\mathcal{R}^*(L_{\mathsf{in}}) \cap L_{\mathsf{err}} = \emptyset$. Note that non-reachability corresponds to the particular case where $L_{\mathsf{in}} = \{s\}$ and $L_{\mathsf{err}} = \{t\}$. The name RMC is coined after state exploration techniques for checking safety properties. In this setting, L_{in} and L_{err} represent (possibly infinite) sets of initial, respectively error, states. This problem is also related to the problem of *typechecking* tree transformations, see *e.g.* [21].

3 Regularity Preservation for pCTRSs

A pCTRS $\mathcal{R}$ is said to *preserve regularity* if for every regular language $L \subseteq \mathcal{T}(\Sigma)$, the closure $\mathcal{R}^*(L)$ is regular. The preservation is *effective* if moreover a TA recognizing $\mathcal{R}^*(L)$ can be constructed. Thanks to the closure and decidability properties of TAs, the effective preservation is a sufficient condition for RMC.

There have been many studies of regularity preservation for various classes of standard TRS see *e.g.* [8] for a sum-up. Some works have also considered this problem, or the decidability of RMC, for (unconstrained) term rewriting under some strategies, see Section 4.1 for references.

3.1 Linear and flat pCTRSs

Every linear and right-flat (uncontrolled) TRS effectively preserves regularity [22]. This property does not hold when adding prefix control.

Proposition 2. *Linear and flat pCTRSs do not preserve regularity.*

Proof. Let us consider the unary signature $\Sigma = \{a, a', b, b', c, d, \bot\}$ where $\bot$ has arity 0 and all other symbols have arity 1, and the linear and flat pCTRS $\mathcal{R}$ over Σ containing the 4 following rules

$$c^* : \ c(x) \to a'(x), \qquad c^*a'a^*b^* : \ d(x) \to b'(x),$$
$$c^* : a'(x) \to a(x), \qquad c^*a^*b^* : b'(x) \to b(x).$$

For the sake of readability, given a string $w \in (\Sigma \setminus \{\bot\})^*$, we simply write below w for the term $w\sigma_0$ where σ_0 is the substitution associating $\bot$ to the (single) variable of the term representing w. The intersection of the regular term set a^*b^* and the rewrite closure of c^*d^* by $\mathcal{R}$ is $\{a^nb^m \mid n \geq m\}$, which is context free (CF) and not regular. Indeed, the control language $c^*a'a^*b^*$ imposes a pairing between rewritings of d into b and rewritings of c into a: for each rewriting of d into b', there must have been one (and only one) earlier rewriting of c into a', as illustrated by the following rewrite sequence $ccdd \xrightarrow[\mathcal{R}]{} ca'dd \xrightarrow[\mathcal{R}]{} ca'b'd \xrightarrow[\mathcal{R}]{} cab'd \xrightarrow[\mathcal{R}]{} cabd \xrightarrow[\mathcal{R}]{} a'abd \xrightarrow[\mathcal{R}]{} a'abb' \xrightarrow[\mathcal{R}]{} aabb' \xrightarrow[\mathcal{R}]{} aabb$. □

We can generalize the principle of the construction of Proposition 2, in order to build a linear and flat pCTRS producing a rewrite closure of the form $\{a^nb^mc^p \mid n \geq m \geq p\}$ (after intersection with the regular language $a^*b^*c^*$), starting from a regular set of the form $d^*e^*f^*$ and using a flat pCTRS. Since the produced language is context-sensitive (CS), it follows that there is no hope for a polynomial time decision procedure (congruence closure like) for the decision of reachability for linear and flat pCTRS.

3.2 Left-(linear and flat) pCTRSs

Let us consider the situation where the linearity and flatness restrictions apply only to left-hand-side of rewrite rules. In the literature, (uncontrolled) TRSs with such syntactical restrictions are called *inverse-monadic*. They also have the same expressiveness as production rules of CF Tree Grammars.

When restricting to strictly unary signatures, these TRSs correspond to string rewriting rules with lhs of length exactly one. It is folklore knowledge that this kind of string rewriting systems transform CF languages into CF languages. This result generalizes to trees (see *e.g.* [12]). It follows that reachability and RMC are decidable for left-(linear and flat) TRSs. This does not hold when extending the expressiveness with prefix control, even in the case of strings. This is a direct consequence of the following lemma, based on a transformation of CS grammars into Pentonnen normal form [23].

Lemma 3. *For every CS (resp. recursively enumerable (RE)) language L over a strictly unary signature Σ, there exists a linear, left-flat and non-collapsing (resp. linear and left-flat) pCTRS $\mathcal{R}$ over an extended strictly unary signature $\Sigma' \supset \Sigma$ such that $L = \mathcal{R}^*(\{s\}) \cap \mathcal{T}(\Sigma)$ for some term $s \in \mathcal{T}(\Sigma', \mathcal{X})$.*

Proof. Assume that $L \subseteq \Sigma^*$ is a CS language, and let $\mathcal{G} = \langle \mathcal{N}, \Sigma, S, P \rangle$ be a CS grammar generating L, with non-terminal set $\mathcal{N}$, set of terminals Σ, $S \in \mathcal{N}$, and let $\Sigma' = \Sigma \cup \mathcal{N}$ (where the symbols of $\mathcal{N}$ are unary). We can assume that the

production rules of $\mathcal{G}$ are in Pentonnen normal form [23]: $AB \to AC$, $A \to BC$, $A \to a$ where $A, B, C \in \mathcal{N}$ and $a \in \Sigma$. Transforming any CS grammar into a grammar of this form can be done in PTIME. It follows that L is the intersection between $\mathcal{T}(\Sigma)$ and the rewrite closure of $\{S(x)\}$ by the linear, left-flat and non-collapsing pCTRS $\mathcal{R}$ simulating the production rules of $\mathcal{G}$, as described in Figure 1. Note that the size of $\mathcal{R}$ is linear in the size of $\mathcal{G}$.

$\mathcal{G}$	$\mathcal{R}$
$A \to BC$	$(\mathcal{N} \cup \Sigma)^* : A(x) \to B(C(x))$
$AB \to AC$	$(\mathcal{N} \cup \Sigma)^* A : B(x) \to C(x)$
$A \to a$	$(\mathcal{N} \cup \Sigma)^* : A(x) \to a(x)$
$A \to \varepsilon$	$(\mathcal{N} \cup \Sigma)^* : A(x) \to x$

Fig. 1. Construction of a linear CF pCTRS for the proof of Proposition 4.

Every RE language can be generated by a CS grammar as above, completed with some deleting rules of the form $A \to \varepsilon$. It corresponds to the collapsing rewrite rule $A(x) \to x$ (last line of Figure 1). □

Proposition 4. *Over strictly unary signatures, (i) reachability is undecidable for linear and left-flat p*CTRS*s, and (ii) reachability is PSPACE-complete and regular model checking is undecidable for linear, left-flat, non-collapsing p*CTRS*s.*

Proof. The undecidability of the reachability problem for linear and left-flat pCTRSs (Claim (i)) follows from Lemma 3, and undecidability of the membership problem of RE languages.

For Claim (ii), the PSPACE-hardness of the reachability problem and undecidability of RMC for linear, left-flat, non-collapsing pCTRSs follow from Lemma 3 and, respectively, the PSPACE-completeness of the membership problem and undecidability of emptiness problem for CS languages.

The PSPACE upper bound for reachability follows immediately from the fact that for a left-flat and non-collapsing pCTRS $\mathcal{R}$ over a strictly unary signature, the size of every rhs of rule of $\mathcal{R}$ is larger or equal to the size of the corresponding lhs. Hence, $s \xrightarrow[\mathcal{R}]{*} t$ can be checked by a backward exploration of the ancestors of t *wrt* $\mathcal{R}$, and they are all smaller than or equal to t. □

To sum up, (unconstrained) TRSs with syntactic restrictions of flatness and linearity benefit good results of regularity preservation and decision, but these results are lost when adding prefix constraints. The reason is that these constraints permit to test the context of rewrite positions and therefore simulate computations of Turing Machines (Proposition 4(i)) or Linear Bounded Automata (Proposition 4(ii) and remark after Proposition 2). We can observe that in the simulations, it is important to rewrite alternatively in two directions, top-down and bottom-up (see for instance the rewrite sequence presented in the proof of Proposition 2). A key property of regularity preservation results such as [18] is that in every step $C[t] \xrightarrow[\mathcal{R}]{} D[t]$, either all redexes in t are preserved

or all are inactivated. For an ordinary step of a left-linear right-shallow pCTRS, such a property does not hold in general, because a reduction in context C may activate a redex in t. However, if we restrict to bottom-up rewriting, the above properties are recovered for linear and right-shallow pCTRSs. We show this in the next sections, and show consequently regularity preservation and decision results for right-ground pCTRSs (Section 5) pCTRSs.

4 Bottom-Up Rewrite Strategy

We show in this section that when we restrict to bottom-up derivations [6], the preservation of regularity holds for linear and right-shallow pCTRSs.

4.1 Definition

We define a bottom-up derivation on terms by introducing a bottom-up marked rewriting on marked terms, where the latter is called *weakly* bottom-up in [6]. Following the definition of [6], we use a marked copy of the signature $\overline{\Sigma} = \{\bar{g} \mid g \in \Sigma\}$. A *marked term* is a term in $\mathcal{T}(\Sigma \cup \overline{\Sigma}, \mathcal{X})$. Given a term of $t \in \mathcal{T}(\Sigma, \mathcal{X})$, we use the notation $\bar{t}$ to represent a marked term in $\mathcal{T}(\Sigma \cup \overline{\Sigma}, \mathcal{X})$ associated with t in a way that t is obtained from $\bar{t}$ by replacing each symbol $\bar{g}$ by g. Moreover, $\widetilde{t}$ denotes the unique marked term associated with t which belongs to $\mathcal{T}(\overline{\Sigma} \cup \mathcal{X})$. This notation is extended to contexts and substitutions as expected.

The *bottom-up marked rewriting* relation for a pCTRS $\mathcal{R}$ is defined as

$$\overline{C}[\overline{\ell\sigma}]_\pi \xrightarrow[\mathcal{R}]{bu} \overline{C}[r\,\widetilde{\sigma}]_\pi$$

for a context $\overline{C}$ if $L : \ell \to r \in \mathcal{R}$, $path(C, \pi) \in L$ and the root symbol of $\overline{\ell}$ is in Σ (the other symbols may be marked or not). We say that the derivation $s \xrightarrow[\mathcal{R}]{*} t$ on terms is *bottom-up* if there exist a marking $\bar{t}$ and a bottom-up marked rewriting sequence $s \xrightarrow{*} \bar{t}$. In this case, we write $s \Rightarrow^{bu}_{\mathcal{R}} t$. Note that the derivation $\Rightarrow^{bu}_{\mathcal{R}}$ on terms is not transitive.

Example 2. Let $\mathcal{R}$ contain the two following prefix controlled rules $\varepsilon : h(x) \to g(x)$ and $\langle h, 1\rangle : a \to b$. For the controlled rewrite derivation $h(a) \xrightarrow[\mathcal{R}]{*} g(b)$, there exists a bottom-up marked rewriting sequence $h(a) \xrightarrow[\mathcal{R}]{bu} h(b) \xrightarrow[\mathcal{R}]{bu} g(\bar{b})$. Thus the former sequence is bottom-up, *i.e.* $h(a) \Rightarrow^{bu}_{\mathcal{R}} g(b)$. ⋄

Example 3. Let $\mathcal{R} = \{\varepsilon : h(x) \to g(x), \langle g, 1\rangle : a \to b\}$. The controlled rewrite derivation $h(a) \xrightarrow[\mathcal{R}]{*} g(b)$ is not bottom-up. Indeed, following the above definition of the bottom-up marked rewriting, we have $h(a) \xrightarrow[\mathcal{R}]{bu} g(\bar{a})$ but we do not have $g(\bar{a}) \xrightarrow[\mathcal{R}]{bu} g(\bar{b})$ because $\bar{a}$ is not in Σ. ⋄

Related Rewrite Strategies. The notion of bottom-up derivations was firstly introduced as the basic narrowing [20]. The bottom-up marked rewriting BU

of [6] (that we shall call BU[6] to avoid confusions) is defined with integer marking. It is more general than the above bottom-up marked rewriting, the latter being roughly the restriction of BU[6] using 1 marker.

In [6], a result of regularity preservation is proved for the subclass of linear TRSs such that every rewrite derivation can be simulated by a BU[6] rewrite derivation (such TRSs are called BU). It is shown in [6] that linear and right-flat TRSs are BU. We have seen (Proposition 2) that with prefix control, regularity is not preserved by linear and right-flat pCTRSs. However, we will prove in the next section that regularity is preserved by linear and right-flat pCTRSs when restricting to bottom-up derivations.

There have been studies on regularity preservation, or the decidability of RMC, for (unconstrained) term rewriting under other strategies. It is show in [18] that regularity is preserved by rewriting with linear and right-shallow TRS under the *context-sensitive* strategy. The *innermost* rewriting $\xrightarrow[\mathcal{R}]{in}$ [20] corresponds to the *call by value* computation for programming languages, where arguments are fully evaluated before the function application. More precisely, a rewrite rule can be applied to a subterm at position π if all the proper subterms at children positions of π are normal forms. It is easily shown that $\xrightarrow[\mathcal{R}]{in,*} \subseteq \Rightarrow^{bu}_{\mathcal{R}}$, where the relation is proper for most of TRSs. Regularity preservation have been shown for innermost rewriting with linear right-shallow term rewriting systems [18], and with constructor based systems with additional restrictions [24].
The *one-pass leaf-started derivation* $\Rightarrow^{1pls}_{\mathcal{R}}$ [10] is defined using an auxiliary symbol which acts as a token passed from leaves to root. It is shown in [10] that RMC is decidable for left-linear TRS with one-pass leaf-started rewriting (but regularity is not necessarily preserved). It is shown in [5] that regularity is preserved for *one IO rewrite pass* [7] (denoted $\Rightarrow^{IO}_{\mathcal{R}}$ for its reflexive extension) by linear TRS $\mathcal{R}$. It can be observed that $\Rightarrow^{1pls}_{\mathcal{R}} \subseteq \Rightarrow^{IO}_{\mathcal{R}} \subseteq \Rightarrow^{bu}_{\mathcal{R}}$, where the relations are proper for most of TRSs.

To our knowledge, our approach of studying closure under bottom-up derivations for rewrite rules with context constraints is original.

4.2 Tree Automata Completion

We show now that linear and right-shallow pCTRSs effectively preserve regularity when used with bottom-up rewriting. For this purpose we use a procedure completing a given TA $\mathcal{A}$ with respect to a given pCTRS $\mathcal{R}$. Assuming *wlog* that the initial TA $\mathcal{A}$ is clean and is given without ε-transitions, we complete it first into a TA $\mathcal{A}'$, with one new state $\underline{v}$ for every ground subterm v of a *rhs* of $\mathcal{R}$ and with appropriate transitions such that $\mathcal{L}(\mathcal{A}', \underline{v}) = \{v\}$. Note that $\mathcal{A}'$ can also be assumed to be clean and without ε-transitions.

For each rule $L : \ell \to r$ in $\mathcal{R}$, we assume given an FSA $\mathcal{C}_L$ over $\mathcal{D}ir(\Sigma)$ recognizing L. The respective state sets of all these FSAs are assumed disjoint. We define an automaton $\mathcal{C}_0 = \langle 2^P, \mathcal{D}ir(\Sigma), S, 2^G, \Theta\rangle$ that simulates all the control automata $\mathcal{C}_L$ as follows: P (resp. G) is the union of all the state sets (resp. final state sets) of the $\mathcal{C}_L$'s, S is the set of all the initial states of $\mathcal{C}_L$'s, and Θ contains

all the transitions of the form $s \xrightarrow{\langle g,i\rangle} s'$ where $s' = \{p' \in P \mid \exists p \in s\ \exists L : \ell \to r \in \mathcal{R}$ s.t. $p \xrightarrow{\langle g,i\rangle} p'$ is a transition of $\mathcal{C}_L\}$. Note that $\mathcal{C}_0$ is deterministic.

Let $\mathcal{A}_0 = \langle Q, F, \Delta_0\rangle$ where $Q = Q_{\mathcal{A}'} \times 2^P$, $F = \{\langle q, S\rangle \mid q \in F_{\mathcal{A}'}\}$, and Δ_0 is the set of transitions of the form:

$$g(\langle q_1, s_1\rangle, \ldots, \langle q_m, s_m\rangle) \to \langle q_0, s_0\rangle$$

with $g \in \Sigma$ and such that $g(q_1, \ldots, q_m) \to q_0$ is a transition of $\mathcal{A}'$, and $s_0 \xrightarrow{\langle g,i\rangle} s_i \in \Theta$ for all i with $1 \leq i \leq m$. Intuitively, a term $C[\ell\sigma] \in \mathcal{A}$ and the displayed redex $\ell\sigma$ is reducible by a rule $L : \ell \to r$, if and only if there exists a transition $C[\ell\sigma] \xrightarrow[\mathcal{A}_0]{*} C[\langle q_0, s_0\rangle] \xrightarrow[\mathcal{A}_0]{*} \langle q, s\rangle \in F$ such that $s_0 \cap \mathit{final}(\mathcal{C}_L) \neq \emptyset$.

We show now how to complete an automaton $\mathcal{A}_k = \langle Q, F, \Delta_k\rangle$, for $k \geq 0$, into $\mathcal{A}_{k+1} = \langle Q, F, \Delta_{k+1}\rangle$, in order to simulate one bottom-up rewrite step with $\mathcal{R}$. At each construction step $k \geq 0$, we construct Δ_{k+1} by adding rules.

(Rules 1) For all $L : \ell \to g(r_1, \ldots, r_m)$ in $\mathcal{R}$, with $m \geq 0$, for all substitutions θ from $\mathcal{X}$ into Q grounding for $\mathit{var}(\ell)$, such that $\ell\theta \xrightarrow[\mathcal{A}_k]{*} \langle q_0, s_0\rangle$, the last step of this derivation is not an ε-transition, and $s_0 \cap \mathit{final}(\mathcal{C}_L) \neq \emptyset$, we add to Δ_k all the following rules:

$$g\left(\langle q_1, s_1\rangle, \ldots, \langle q_m, s_m\rangle\right) \to \langle q_0, s_0\rangle$$

such that for all $1 \leq j \leq m$, if r_j is a variable then $\langle q_j, s_j\rangle = r_j\theta$, otherwise, $q_j = \underline{r_j}$, and $s_0 \xrightarrow{\langle g,j\rangle} s_j \in \Theta$.

(Rules 2) For all $L : \ell \to x$ in $\mathcal{R}$ with $x \in \mathit{var}(\ell)$, for all substitutions θ from $\mathcal{X}$ into Q grounding for $\mathit{var}(\ell)$ such that $\ell\theta \xrightarrow[\mathcal{A}_k]{*} \langle q_0, s_0\rangle$, and $s_0 \cap \mathit{final}(\mathcal{C}_L) \neq \emptyset$, we add to Δ_k the following rule:

$$x\theta \to \langle q_0, s_0\rangle$$

The completion terminates with a fixpoint Δ_k, and the TA $\mathcal{A}^* = \langle Q, F, \Delta_k\rangle$ recognizes the bottom-up rewrite closure of $\mathcal{L}(\mathcal{A})$ by $\mathcal{R}$.

Example 4. Let us consider the pCTRS of Example 2, and two FSA describing the control languages of its two rules: the first FSA has one state v_0, both initial and final, and no transitions, and the second FSA has two states v_1 (initial) and v_2 (final) and one transition $v_1 \xrightarrow{\langle h,1\rangle} v_2$. Let the initial $\mathcal{A}$ recognize the singleton language $\{h(a)\}$, with the two transitions $a \to q_a$ and $h(q_a) \to q$ (q is the only final state). The automaton $\mathcal{A}_0$ contains the following transitions:

$$\begin{aligned} a &\to \langle q_a, s\rangle \text{ for all } s \subseteq \{v_0, v_1, v_2\},\\ h\big(\langle q_a, \{v_2\}\rangle\big) &\to \langle q, s\rangle \text{ for all } s \text{ with } \{v_1\} \subseteq s \subseteq \{v_0, v_1, v_2\},\\ h\big(\langle q_a, \emptyset\rangle\big) &\to \langle q, s\rangle \text{ for all } s \subseteq \{v_0, v_2\}. \end{aligned}$$

The completion process adds the following transitions to $\mathcal{A}^*$, by the case (Rules 1):

$$\begin{aligned} b &\to \langle q_a, s\rangle \text{ for all } s \text{ with } \{v_2\} \subseteq s \subseteq \{v_0, v_1, v_2\},\\ g\big(\langle q_a, \{v_2\}\rangle\big) &\to \langle q, s\rangle \text{ for all } s \text{ with } \{v_0, v_1\} \subseteq s \subseteq \{v_0, v_1, v_2\},\\ g\big(\langle q_a, \emptyset\rangle\big) &\to \langle q, s\rangle \text{ for all } s \text{ with } \{v_0\} \subseteq s \subseteq \{v_0, v_2\}. \end{aligned}$$

The only final state of $\mathcal{A}^*$ is $\langle q, \{v_0, v_1\}\rangle$. Then $g(a)$ and $g(b)$ which are both in the bottom-up closure of $h(a)$ are recognized by $\mathcal{A}^*$ with the derivation $g(a) \xrightarrow[\mathcal{A}_0]{} g\big(\langle q_a, \{v_2\}\rangle\big) \xrightarrow[\mathcal{A}^*]{} \langle q, \{v_0, v_1\}\rangle$ and $g(b) \xrightarrow[\mathcal{A}^*]{} g\big(\langle q_a, \{v_2\}\rangle\big) \xrightarrow[\mathcal{A}^*]{} \langle q, \{v_0, v_1\}\rangle$. $\diamond$

Example 5. With the pCTRS of Example 3, we have a first FSA for control identical to the one-state FSA of Example 4 and a second one with two states v_1 (initial) and v_2 (final) and one transition $v_1 \xrightarrow{\langle g,1\rangle} v_2$. Let us consider the same initial automaton $\mathcal{A}$ as in Example 4. The automaton $\mathcal{A}_0$ contains now the transitions: $a \to \langle q_a, s\rangle$ and $h\big(\langle q_a, \emptyset\rangle\big) \to \langle q, s\rangle$ for all $s \subseteq \{v_0, v_1, v_2\}$ (q is final). We obtain the following additional transitions in $\mathcal{A}^*$ by the case (Rules 1):

$$\begin{aligned} b &\to \langle q_a, s\rangle \text{ for all } s \text{ with } \{v_2\} \subseteq s \subseteq \{v_0, v_1, v_2\},\\ g\big(\langle q_a, \emptyset\rangle\big) &\to \langle q, s\rangle \text{ for all } s \text{ with } \{v_0\} \subseteq s \subseteq \{v_0, v_1, v_2\}. \end{aligned}$$

The term $g(a)$ is in the bottom-up rewrite closure of $h(a)$ by $\mathcal{R}$. It is recognized by $\mathcal{A}^*$ with the following derivation $g(a) \xrightarrow[\mathcal{A}_0]{} g\big(\langle q_a, \emptyset\rangle\big) \xrightarrow[\mathcal{A}^*]{} \langle q, \{v_0, v_1\}\rangle$.
The term $h(b)$ is not in the bottom-up rewrite closure of $h(a)$ by $\mathcal{R}$; it holds that $h(b) \xrightarrow[\mathcal{A}^*]{} h(\langle q_a, s\rangle)$ if $v_2 \in s$, but $\mathcal{A}^*$ cannot compute from such configurations. The situation is similar for $g(b)$, which is neither in the bottom-up rewrite closure of $h(a)$ by $\mathcal{R}$ (see Example 3). $\diamond$

Note that the number of states of the automaton $\mathcal{A}^*$ is exponential in the number of states of the control automata used in the definition of $\mathcal{R}$, and polynomial in $|Q_{\mathcal{A}}|$. When we do not account the size of control automata in the evaluation of the size of $\mathcal{R}$, then the size of $\mathcal{A}^*$ is polynomial and the above construction is PTIME (as well as the decidability results in the next corollary).

Theorem 5. *Given a TA $\mathcal{A}$ and a linear and right-shallow pCTRS $\mathcal{R}$ over Σ, one can construct in EXPTIME a TA over Σ recognizing the bottom-up rewrite closure of $\mathcal{L}(\mathcal{A})$ by $\mathcal{R}$, and whose size is exponential in the size of $\mathcal{A}$ and $\mathcal{R}$.*

In the rest of the section we prove the theorem, by establishing the correctness and completeness of the construction of $\mathcal{A}^*$. For this purpose, we use a relation defined as $t \xrightarrow[\mathcal{R},s]{\pi} t'$, with $\pi \in \mathcal{P}os(t)$ and $s \subseteq P$, iff there exist $L : \ell \to r \in \mathcal{R}$ and a substitution σ such that $t|_\pi = \ell\sigma$, $t' = t[r\sigma]_\pi$, $s \xrightarrow[\Theta]{path(t,\pi)} s'$ and s' contains a final state of C_L. The suffix π in $\xrightarrow[\mathcal{R},s]{\pi}$ might be dropped. We associate to this relation its bottom-up marked counterpart $\xrightarrow[\mathcal{R},s]{bu}$ as above. Note that $\xrightarrow[\mathcal{R}]{} = \xrightarrow[\mathcal{R},S]{}$ and $\xrightarrow[\mathcal{R}]{bu} = \xrightarrow[\mathcal{R},S]{bu}$. The next lemma follows immediately from the definition of the relation $\xrightarrow[\mathcal{R},s]{}$.

Lemma 6. *For all $u, t \in \mathcal{T}(\Sigma)$, $i \cdot \pi \in \mathcal{P}os(u)$, and $s \subseteq P$, $u \xrightarrow[\mathcal{R},s]{i\cdot\pi} t$ iff there exist $s_i \subseteq P$ and $g \in \Sigma \cup \overline{\Sigma}$ such that $u = g(u_1, \ldots, u_m)$, $t = g(t_1, \ldots, t_m)$, $u_i \xrightarrow[\mathcal{R},s_i]{\pi} t_i$, and $s \xrightarrow[\Theta]{\langle g,i\rangle} s_i$.*

The next lemma follows from the construction of $\mathcal{A}_0$, as $\mathcal{A}_0$ embeds both $\mathcal{A}'$ and the control automata $\mathcal{C}_L$ in the first, resp. second, components of its states.

Lemma 7. *For all $t \in \mathcal{T}(\Sigma)$,*

i. if $t \xrightarrow[\mathcal{A}']{*} q$, then for all $s \subseteq P$ there exists $\overline{t}$ such that $\overline{t} \xrightarrow[\mathcal{A}_0]{*} \langle q, s\rangle$.
ii. if $\overline{t} \xrightarrow[\mathcal{A}_0]{*} \langle q, s\rangle$, then $t \xrightarrow[\mathcal{A}']{*} q$.

The correctness of the construction, *i.e.* the inclusion of $L(\mathcal{A}^*)$ in the bottom-up closure of $L(\mathcal{A})$ by $\mathcal{R}$, results from the following lemma.

Lemma 8. *For all $t \in \mathcal{T}(\Sigma)$ and all state $\langle q, s\rangle \in Q$ such that $t \xrightarrow[\mathcal{A}^*]{*} \langle q, s\rangle$, there exist u and $\overline{t}$ such that (i) $u \xrightarrow[\mathcal{A}_0]{*} \langle q, s\rangle$ and (ii) $u \xrightarrow[\mathcal{R},s]{bu,*} \overline{t}$.*
Moreover, if the last step in $t \xrightarrow[\mathcal{A}^*]{*} \langle q, s\rangle$ is not an ε-transition, then the top symbol of $\overline{t}$ is in Σ.

Proof. Let the *index* of a transition rule γ of $\mathcal{A}^*$ be 0 if γ is a transition of $\mathcal{A}_0$ and otherwise, the minimal $k > 0$ such that γ is a transition of $\mathcal{A}_k$ and not a transition of $\mathcal{A}_{k-1}$. We do a proof by induction on the multiset of the indexes of transition rules of $\mathcal{A}^*$ used in the derivation $t \xrightarrow[\mathcal{A}^*]{*} \langle q, s\rangle$, which we call ρ.

We illustrate only an interesting case where the rule γ used in its last step is not an ε-transition and is nor in Δ_0, *i.e.* we assume that the derivation ρ has the following form for $k > 0$:

$$\rho : \ t = g(t_1, \ldots, t_m) \xrightarrow[\mathcal{A}^*]{*} g\big(\langle q_1, s_1\rangle, \ldots, \langle q_m, s_m\rangle\big) \xrightarrow[\mathcal{A}_k]{} \langle q, s\rangle \qquad (\rho_1)$$

This means that $t_j \xrightarrow[\mathcal{A}^*]{*} \langle q_j, s_j\rangle$ for all j with $1 \leq j \leq m$, and then by induction hypothesis, there exist u_j and $\overline{t_j}$ such that (i_0) $u_j \xrightarrow[\mathcal{A}_0]{*} \langle q_j, s_j\rangle$, and (ii_0) $u_j \xrightarrow[\mathcal{R},s_j]{bu,*} \overline{t_j}$.

In this case γ has been added by the case (Rules 1) of the construction, because there exist a rewrite rule $L : \ell \to r \in \mathcal{R}$, a substitution θ from $\mathcal{X}$ into Q, grounding for $var(\ell)$, such that $\ell\theta \xrightarrow[\mathcal{A}_{k-1}]{*} \langle q, s\rangle$, the last step of this derivation is not an ε-transition, and $s \cap final(\mathcal{C}_L) \neq \emptyset$. Moreover, letting $r = g(r_1, \ldots, r_m)$, it holds that for all $1 \leq j \leq m$, if r_j is a variable then $\langle q_j, s_j\rangle = r_j\theta$, and otherwise, $q_j = \underline{r_j}$ and $s \xrightarrow{\langle g,j\rangle} s_j \in \Theta$.

Without loss of generality, let us assume that for some i, $r_1, \ldots, r_i$ are ground terms and $r_{i+1}, \ldots, r_m$ are distinct variables (remember that $\mathcal{R}$ is linear and right-shallow). Let us now construct a substitution σ from $\mathcal{X}$ into $\mathcal{T}(\Sigma, \mathcal{X})$, grounding for $var(\ell)$. For each $x \in var(\ell) \cap var(r)$, there exists $i + 1 \leq j \leq m$ such that $x = r_j$, and we let $x\sigma = t_j$. For each $x \in var(\ell) \setminus var(r)$, we let $x\sigma$ be an arbitrary ground term in $L(\mathcal{A}_0, x\theta)$ (such a term exists by assumption that $\mathcal{A}_0$ is clean). One can check, using (ρ_1), the construction of γ, and the linearity of the rewrite rules of $\mathcal{R}$, that $\ell\sigma \xrightarrow[\mathcal{A}^*]{*} \ell\theta \xrightarrow[\mathcal{A}_{k-1}]{*} \langle q, s\rangle$, where the last step is not an ε-transition. This derivation is strictly smaller that ρ *wrt* the induction ordering. Thus, by induction hypothesis, there exist u and $\overline{\ell\sigma}$ such that (i_1) $u \xrightarrow[\mathcal{A}_0]{*} \langle q, s\rangle$ and (ii_1) $u \xrightarrow[\mathcal{R},s]{bu,*} \overline{\ell\sigma}$, and moreover, the top symbol of $\overline{\ell\sigma}$ is in Σ.

By construction of γ, $s \cap final(\mathcal{C}_L) \neq \emptyset$, hence, using the rule $L : \ell \to r \in \mathcal{R}$, it holds that: $\overline{\ell\sigma} \xrightarrow[\mathcal{R},s]{bu} g(r_1, \ldots, r_i, \widetilde{t_{i+1}}, \ldots, \widetilde{t_m})$. Moreover, for all j with $1 \leq j \leq i$,

it holds by construction of γ that $q_j = \underline{r_j}$, hence (i_0) and Lemma 7(ii) imply that $u_j = r_j$, hence $r_j \xrightarrow[\mathcal{R},s_j]{bu,*} \overline{t_j}$ by (ii_0). Using Lemma 6, it follows that

$$u \xrightarrow[\mathcal{R},s]{bu,*} \overline{\ell\sigma} \xrightarrow[\mathcal{R},s]{bu} g(r_1,\ldots,r_i,\widetilde{t_{i+1}},\ldots,\widetilde{t_m}) \xrightarrow[\mathcal{R},s]{bu,*} g(\overline{t_1},\ldots,\overline{t_i},\widetilde{t_{i+1}},\ldots,\widetilde{t_m}).$$

Letting $\overline{t} = g(\overline{t_1},\ldots,\overline{t_i},\widetilde{t_{i+1}},\ldots,\widetilde{t_m})$, we can conclude for (ii) in this case. Note that the top symbol of $\overline{t}$ is in Σ. The proof of the other cases can be found in Appendix A. □

The following lemma implies the completeness of the construction of $\mathcal{A}^*$.

Lemma 9. *For all $u \in \mathcal{T}(\Sigma)$, $\overline{t} \in \mathcal{T}(\Sigma \cup \overline{\Sigma})$, state $\langle q,s\rangle \in Q$, if $u \xrightarrow[\mathcal{R},s]{bu,*} \overline{t}$, and $u \xrightarrow[\mathcal{A}_0]{*} \langle q,s\rangle$, then $t \xrightarrow[\mathcal{A}^*]{*} \langle q,s\rangle$. Moreover, if the top symbol of $\overline{t}$ is in Σ, then the last step in $t \xrightarrow[\mathcal{A}^*]{*} \langle q,s\rangle$ is not an ε-transition.*

Proof. We do a proof by induction on the lexical combination of the length of the derivation $u \xrightarrow[\mathcal{R},s]{bu,*} \overline{t}$ and the structure of u.

We illustrate only an interesting case that some rewrite steps are performed at the root position, and the last rewrite step performed at the root position involves a non-collapsing rule.

Since $u \xrightarrow[\mathcal{R},s]{bu,*} \overline{t}$ is a bottom-up marked rewriting, no earlier derivation is performed at the root position with a collapsing rule (because the root symbol of every redex in a bottom-up marked derivation must be in Σ), and the top symbol of $\overline{t}$ is in Σ. We can write the rewrite sequence as follows:

$$u \xrightarrow[\mathcal{R},s]{bu,*} \overline{\ell\sigma} \xrightarrow[\mathcal{R},s]{bu} g(r_1,\ldots,r_m)\,\widetilde{\sigma} \xrightarrow[\mathcal{R},s]{bu,*} \overline{t}$$

where $L : \ell \to g(r_1,\ldots,r_m) \in \mathcal{R}$ and $\varepsilon \in L$. Without loss of generality, we assume that $r_1,\ldots,r_i$ are ground terms and $r_{i+1},\ldots,r_m$ are variables for some $i \le m$. By induction hypothesis, it holds that $\ell\sigma \xrightarrow[\mathcal{A}^*]{*} \langle q,s\rangle$ and the last step of this derivation is not an ε-transition. This rewrite sequence can be decomposed into $\ell\sigma \xrightarrow[\mathcal{A}^*]{*} \ell\theta \xrightarrow[\mathcal{A}^*]{*} \langle q,s\rangle$ where θ is a substitution from $\mathcal{X}$ into Q, grounding for $var(\ell)$. Note that we use the assumption that $\mathcal{R}$ is linear in order to construct this θ. Moreover, $s \cap \mathit{final}(\mathcal{C}_L) \neq \emptyset$. Then from the construction case (Rules 1), $\mathcal{A}^*$ contains the transition rule $g(\langle q_1,s_1\rangle,\ldots,\langle q_m,s_m\rangle) \to \langle q,s\rangle$ where

- for all j with $1 \le j \le i$, $q_j = \underline{r_j}$,
- for all j with $i < j \le m$, $\langle q_j, \underline{s_j}\rangle = r_j\theta$,
- for all j with $1 \le j \le m$, $s \xrightarrow[\Theta]{\langle g,j\rangle} s_j$.

For all j with $1 \le j \le i$, $r_j \xrightarrow[\mathcal{A}_0]{*} \langle q_j,s_j\rangle$ by the construction of $q_j = \underline{r_j}$ and Lemma 7(i), and for all j with $i \le j \le m$, $r_j\sigma \xrightarrow[\mathcal{A}^*]{*} r_j\theta$. Therefore

$$t = g(r_1,\ldots,r_i,r_{i+1}\sigma\ldots,r_m\sigma) \xrightarrow[\mathcal{A}_0]{*} g(\langle q_1,s_1\rangle,\ldots,\langle q_i,s_i\rangle,r_{i+1}\sigma\ldots,r_m\sigma)$$
$$\xrightarrow[\mathcal{A}^*]{*} g(\langle q_1,s_1\rangle,\ldots,\langle q_i,s_i\rangle,\langle q_{i+1},s_{i+1}\rangle,\ldots,\langle q_m,s_m\rangle) \xrightarrow[\mathcal{A}_0]{*} \langle q,s\rangle.$$

Hence $t = x\sigma \xrightarrow[\mathcal{A}^*]{*} x\theta \xrightarrow[\mathcal{A}^*]{*} \langle q,s\rangle$. The proof of the other cases can be found in Appendix B. □

Corollary 10. *Reachability and RMC wrt. bottom-up rewriting are decidable in EXPTIME for linear and right-shallow pCTRSs.*

Proof. Given a linear and right-shallow pCTRS $\mathcal{R}$, the reachability problem $s \xrightarrow{bu,*}_{\mathcal{R}} t$ *wrt* bottom-up rewriting is equivalent to $t \in L(\mathcal{A}^*)$ where $\mathcal{A}^*$ is the TA constructed from a TA recognizing $\{s\}$ as in Theorem 5. The RMC problem $\mathcal{R}^*(L_{\mathsf{in}}) \cap L_{\mathsf{err}} = \emptyset$, *wrt* bottom-up rewriting, is equivalent to $L(\mathcal{A}^*_{\mathsf{in}}) \cap L_{\mathsf{err}} = \emptyset$, where $\mathcal{A}^*_{\mathsf{in}}$ is the TA constructed from a TA recognizing L_{in} as in Theorem 5.

Both problems can be decided in PTIME in the size of $\mathcal{A}^*$ and t on one hand and $\mathcal{A}^*_{\mathsf{in}}$ and a TA recognizing L_{err} on the other hand. □

5 Left-Linear and Right-Ground pCTRSs

It can be observed that every rewrite sequence with a right-ground pCTRS is bottom-up. Hence the following corollary immediately follows.

Corollary 11. *Given a TA $\mathcal{A}$ and a left-linear and right-ground pCTRS $\mathcal{R}$ over Σ, one can construct in EXPTIME a TA over Σ recognizing the rewrite closure of $\mathcal{L}(\mathcal{A})$ by $\mathcal{R}$, and whose size is exponential in the size of $\mathcal{A}$ and $\mathcal{R}$. Reachability and RMC are decidable for left-linear and right-ground pCTRSs.*

The following proposition establishes a lower bound for the construction.

Proposition 12. *Reachability is PSPACE-hard for ground pCTRSs.*

Proof. We make a reduction of the intersection emptiness problem for regular string languages. Let $L_1, \ldots, L_n (n \geq 2)$ be regular languages, and let

$$\begin{array}{ll} \mathcal{R} = \{\Sigma^* : \sharp_1 \to a(\sharp_1) \mid a \in \Sigma\} \cup \{L_i : \sharp_i \to \sharp_{i+1} \mid 1 \leq i \leq n-1\} \\ \quad \cup \{L_n : \sharp_n \to \flat\} & \quad \cup \{\Sigma^* : a(\flat) \to \flat \mid a \in \Sigma\} \end{array}$$

It can be easily checked that $\sharp_1 \xrightarrow[\mathcal{R}]{*} \flat$ iff $L_1 \cap \cdots \cap L_n \neq \emptyset$. □

Given a linear and right-shallow (uncontrolled) TRS $\mathcal{R}$, for every rewrite sequence $s \xrightarrow[\mathcal{R}]{*} t$ there exists a bottom-up rewrite sequence $s \xrightarrow[\mathcal{R}]{*} t$ [6]. This is however not the case in presence of prefix control, since linear and right-shallow pCTRSs do not preserve regularity (Proposition 2).

6 Conclusion

This work could be extended in several directions. A question is whether the results of Section 4 still hold when weakening the linearity restriction into right-linearity. Note that regularity preservation has been established for right-linear and right-shallow (unconstrained) TRSs in [22]. An alternative approach might be to construct automata recognizing regular over-approximating of the closures, for larger classes of pCTRS, like in [8]. The completion algorithms of this paper terminate because the shallowness of the rhs ensure that no new state needs

to be added to the automata; other automata completion methods [8] accept non-shallow rhs and thus need to normalize the new transitions by adding new states, they ensure their termination by merging states, at the cost of precision.

Finally, a difficult problem is the generalization of the problems presented in this paper to unranked tree rewriting [14], where variables are instantiated by forests (i.e. finite sequences of trees) instead of terms.

References

1. P. A. Abdulla, B. Jonsson, P. Mahata, and J. d'Orso. Regular tree model checking. In *Proceedings of the 14th International Conference on Computer Aided Verification*, CAV '02, pages 555–568, London, UK, UK, 2002. Springer-Verlag.
2. A. Bouajjani, P. Habermehl, A. Rogalewicz, and T. Vojnar. Abstract regular tree model checking of complex dynamic data structures. In K. Yi, editor, *Static Analysis*, volume 4134 of *Lecture Notes in Computer Science*, pages 52–70. Springer Berlin / Heidelberg, 2006.
3. H. Comon, M. Dauchet, R. Gilleron, F. Jacquemard, C. Löding, D. Lugiez, S. Tison, and M. Tommasi. *Tree Automata Techniques and Applications*. `http://tata.gforge.inria.fr`, 2007.
4. J. Dassow, G. Paun, and A. Salomaa. Grammars with controlled derivations. In *Handbook of Formal Languages*, volume 2, chapter 3, pages 101–154. Springer, 1997.
5. M. Dauchet and F. De Comite. A gap between linear and non-linear term-rewriting systems. In *On Rewriting Techniques and Applications*, pages 95–104, London, UK, UK, 1987. Springer-Verlag.
6. I. Durand and G. Sénizergues. Bottom-up rewriting is inverse recognizability preserving. In F. Baader, editor, *Proceedings of the 18th International Conference on Term Rewriting and Applications (RTA)*, volume 4533 of *Lecture Notes in Computer Science*, pages 107–121. Springer, 2007.
7. J. Engelfriet and E. M. Schmidt. IO and OI. II. *J. Comput. Syst. Sci.*, 16(1):67–99, 1978.
8. G. Feuillade, T. Genet, and V. V. T. Tong. Reachability analysis over term rewriting systems. *Journal of Automated Reasoning*, 33(3-4):341–383, 2004.
9. Z. Fülöp, E. Jurvanen, M. Steinby, and S. Vágvölgyi. On one-pass term rewriting. In L. Brim, J. Gruska, and J. Zlatuška, editors, *Mathematical Foundations of Computer Science 1998*, volume 1450 of *Lecture Notes in Computer Science*, pages 248–256. Springer Berlin Heidelberg, 1998.
10. Z. Fülöp, E. Jurvanen, M. Steinby, and S. Vágvölgyi. On one-pass term rewriting. *Acta Cybernetica*, 14(1):83–98, Feb. 1999.
11. K. Futatsugi, J. A. Goguen, J.-P. Jouannaud, and J. Meseguer. Principles of obj2. In *Proceedings of the 12th ACM SIGACT-SIGPLAN Symposium on Principles of Programming Languages*, POPL '85, pages 52–66, New York, NY, USA, 1985. ACM.
12. F. Jacquemard, Y. Kojima, and M. Sakai. Controlled term rewriting. In C. Tinelli and V. Sofronie-Stokkermans, editors, *Proceedings of the 8th International Symposium Frontiers of Combining Systems (FroCoS)*, volume 6989 of *Lecture Notes in Artificial Intelligence*, pages 179–194. Springer, 2011.

13. F. Jacquemard and M. Rusinowitch. Rewrite-based verification of xml updates. In *Proceedings of the 12th ACM SIGPLAN International Symposium on Principles and Practice of Declarative Programming (PPDP)*, PPDP '10, pages 119–130, New York, NY, USA, 2010. ACM.
14. F. Jacquemard and M. Rusinowitch. Rewrite Closure and CF Hedge Automata. In *7th International Conference on Language and Automata Theory and Application*, Lecture Notes in Computer Science, Bilbao, Espagne, 2013. Springer.
15. N. D. Jones and N. Andersen. Flow analysis of lazy higher-order functional programs. *Theoretical Computer Science*, 375(1-3):120 – 136, 2007.
16. Y. Kesten, O. Maler, M. Marcus, A. Pnueli, and E. Shahar. Symbolic model checking with rich assertional languages. *Theor. Comput. Sci.*, 256:93–112, April 2001.
17. J. Kochems and C.-H. L. Ong. Improved functional flow and reachability analyses using indexed linear tree grammars. In M. Schmidt-Schauß, editor, *22nd International Conference on Rewriting Techniques and Applications (RTA'11)*, volume 10 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 187–202, Dagstuhl, Germany, 2011. Schloss Dagstuhl–Leibniz-Zentrum fuer Informatik.
18. Y. Kojima and M. Sakai. Innermost reachability and context sensitive reachability properties are decidable for linear right-shallow term rewriting systems. In A. Voronkov, editor, *Proceedings of the 19th International Conference on Rewriting Techniques and Applications (RTA)*, volume 5117 of *Lecture Notes in Computer Science*, pages 187–201. Springer, 2008.
19. S. Lucas. Context-sensitive rewriting strategies. *Information and Computation*, 178:294–343, October 2002.
20. R. d. V. Marc Bezem, Jan Willem Klop, editor. *Term Rewriting Systems by Terese*, volume 55 of *Cambridge Tracts in Theoretical Computer Science*. Cambridge University Press, 2003.
21. T. Milo, D. Suciu, and V. Vianu. Typechecking for xml transformers. *Journal of Computer and System Sciences*, 66(1):66–97, 2003.
22. T. Nagaya and Y. Toyama. Decidability for left-linear growing term rewriting systems. *Information and Computation*, 178(2):499–514, 2002.
23. M. Penttonen. One-sided and two-sided context in formal grammars. *Information and Control*, 25:371–392, 1974.
24. P. Réty and J. Vuotto. Tree automata for rewrite strategies. *Journal of Symbolic Computation*, 40:749–794, July 2005.
25. K. Salomaa. Deterministic tree pushdown automata and monadic tree rewriting systems. *Journal of Computer and System Sciences*, 37(3):367–394, 1988.
26. G. Sénizergues. Some decision problems about controlled rewriting systems. *Theoretical Computer Science*, 71(3):281–346, 1990.

A Complete Proof of Lemma 8

We present in this appendix the rest of the case analysis for the proof by induction of Lemma 8.

Lemma 8. *For all $t \in \mathcal{T}(\Sigma)$ and all state $\langle q, s\rangle \in Q$ such that $t \xrightarrow[\mathcal{A}^*]{*} \langle q, s\rangle$, there exist u and $\overline{t}$ such that (i) $u \xrightarrow[\mathcal{A}_0]{*} \langle q, s\rangle$ and (ii) $u \xrightarrow[\mathcal{R},s]{bu,*} \overline{t}$.*
Moreover, if the last step in $t \xrightarrow[\mathcal{A}^*]{*} \langle q, s\rangle$ is not an ε-transition, then the top symbol of $\overline{t}$ is in Σ.

Proof. Let the index of a transition rule γ of $\mathcal{A}^*$ be 0 if γ is a transition of $\mathcal{A}_0$ and otherwise, the minimal $k > 0$ such that γ is a transition of $\mathcal{A}_k$ and not a transition of $\mathcal{A}_{k-1}$. We do a proof by induction on the multiset of the indexes of transition rules of $\mathcal{A}^*$ used in the reduction $t \xrightarrow[\mathcal{A}^*]{*} \langle q, s\rangle$, which we call ρ.

Case 1. Assume first that the rule γ used in its last step is not an ε-transition, *i.e.* assume that the reduction ρ has the following form:

$$\rho : \; t = g(t_1, \ldots, t_m) \xrightarrow[\mathcal{A}^*]{*} g\big(\langle q_1, s_1\rangle, \ldots, \langle q_m, s_m\rangle\big) \xrightarrow[\mathcal{A}^*]{} \langle q, s\rangle \qquad (\rho_1)$$

This means that $t_j \xrightarrow[\mathcal{A}^*]{*} \langle q_j, s_j\rangle$ for all j with $1 \leq j \leq m$, and then by induction hypothesis, there exists u_j and $\overline{t_j}$ such that (i_0) $u_j \xrightarrow[\mathcal{A}_0]{*} \langle q_j, s_j\rangle$, and (ii_0) $u_j \xrightarrow[\mathcal{R},s_j]{bu}{}^* \overline{t_j}$.

Case 1.0. Assume that the rule γ is in Δ_0. Let $u = g(u_1, \ldots, u_m)$ and $\overline{t} = g(\overline{t_1}, \ldots, \overline{t_m})$. Obviously, $u \xrightarrow[\mathcal{A}_0]{*} g(\langle q_1, s_1\rangle, \ldots, \langle q_m, s_m\rangle) \xrightarrow[\gamma]{} \langle q, s\rangle$ from (i_0), hence (i) holds, and $u = g(u_1, \ldots, u_m) \xrightarrow[\mathcal{R},s]{bu}{}^* g(\overline{t_1}, \ldots, \overline{t_m}) = \overline{t}$, by (ii_0) and Lemma 6, since $s \xrightarrow[\Theta]{\langle g,j\rangle} s_j$ for all j with $1 \leq j \leq m$, by construction of $\mathcal{A}_0$. Hence (ii) also holds.

Case 1.1. Assume now that γ is not in Δ_0 but in Δ_k for some $k > 0$, and has been added by the case (Rules 1) of the construction, because there exists a rewrite rule $L : \ell \to r \in \mathcal{R}$, a substitution θ from $\mathcal{X}$ into Q, grounding for $var(\ell)$, such that $\ell\theta \xrightarrow[\mathcal{A}_{k-1}]{*} \langle q, s\rangle$, the last step of this reduction is not an ε-transition, and $s \cap \mathit{final}(\mathcal{C}_L) \neq \emptyset$. Moreover, letting $r = g(r_1, \ldots, r_m)$, it holds that for all $1 \leq j \leq m$, if r_j is a variable then $\langle q_j, s_j\rangle = r_j\theta$, and otherwise, $q_j = q_{r_j}$ and $s \xrightarrow{\langle g,j\rangle} s_j \in \Theta$.

Without loss of generality, let us assume that for some i, $r_1, \ldots, r_i$ are ground terms and $r_{i+1}, \ldots, r_m$ are distinct variables (remember that $\mathcal{R}$ is linear and right-shallow). Let us now construct a substitution σ from $\mathcal{X}$ into $\mathcal{T}(\Sigma, \mathcal{X})$, grounding for $var(\ell)$. For each $x \in var(\ell) \cap var(r)$, there exists $i + 1 \leq j \leq m$ such that $x = r_j$, and we let $x\sigma = t_j$. For each $x \in var(\ell) \setminus var(r)$, we let $x\sigma$ be an arbitrary ground term in $L(\mathcal{A}_0, x\theta)$ (such a term exists by assumption that

$\mathcal{A}_0$ is clean). One can check, using (ρ_1), the construction of γ, and the linearity of the rewrite rules of $\mathcal{R}$, that

$$\ell\sigma \xrightarrow[\mathcal{A}^*]{*} \ell\theta \xrightarrow[\mathcal{A}_{k-1}]{*} \langle q, s\rangle$$

where the last step is not an ε-transition. This reduction is strictly smaller that ρ *wrt* the induction ordering. Thus, by induction hypothesis, there exists u and $\overline{\ell\sigma}$ such that (i_1) $u \xrightarrow[\mathcal{A}_0]{*} \langle q, s\rangle$ and (ii_1) $u \xrightarrow[\mathcal{R},s]{bu}{}^* \overline{\ell\sigma}$, and moreover, the top symbol of $\overline{\ell\sigma}$ is in Σ.

By construction of γ, $s \cap \mathit{final}(\mathcal{C}_L) \neq \emptyset$, hence, using the rule $L : \ell \to r \in \mathcal{R}$, it holds that:

$$\overline{\ell\sigma} \xrightarrow[\mathcal{R},s]{bu} g(r_1, \ldots, r_i, \widetilde{t_{i+1}}, \ldots, \widetilde{t_m}).$$

Moreover, for all j with $1 \leq j \leq i$, it holds by construction of γ that $q_j = q_{r_j}$, hence (i_0) and Lemma 7(ii) imply that $u_j = r_j$, hence $r_j \xrightarrow[\mathcal{R},s_j]{bu}{}^* \overline{t_j}$ by (ii_0). Using Lemma 6, it follows that

$$u \xrightarrow[\mathcal{R},s]{bu}{}^* \overline{\ell\sigma} \xrightarrow[\mathcal{R},s]{bu} g(r_1, \ldots, r_i, \widetilde{t_{i+1}}, \ldots, \widetilde{t_m}) \xrightarrow[\mathcal{R},s]{bu}{}^* g(\overline{t_1}, \ldots, \overline{t_i}, \widetilde{t_{i+1}}, \ldots, \widetilde{t_m}).$$

Letting $\overline{t} = g(\overline{t_1}, \ldots, \overline{t_i}, \widetilde{t_{i+1}}, \ldots, \widetilde{t_m})$, we can conclude for (ii) in this case. Note that the top symbol of $\overline{t}$ is in Σ.

Case 2. Finally, assume that the rule γ used in the last step of ρ is an ε-transition, *i.e.* that ρ has the following form:

$$\rho : \; t = g(t_1, \ldots, t_m) \xrightarrow[\mathcal{A}^*]{*} \langle q', s'\rangle \xrightarrow[\mathcal{A}^*]{} \langle q, s\rangle. \qquad (\rho_2)$$

Since by assumption, $\mathcal{A}_0$ does not contain ε-transitions, it follows that γ is in Δ_k for some $k > 0$, and has been added by the case (Rules 2) of the construction, because there exists a rewrite rule $L : \ell \to x \in \mathcal{R}$ with $x \in \mathit{var}(\ell)$, a substitution θ from $\mathcal{X}$ into Q grounding for $\mathit{var}(\ell)$, such that $\ell\theta \xrightarrow[\mathcal{A}_{k-1}]{*} \langle q, s\rangle$, $s \cap \mathit{final}(\mathcal{C}_L) \neq \emptyset$, and $\langle q', s'\rangle = x\theta$.

We construct a substitution σ grounding for $\mathit{var}(\ell)$ like in Case 1.1: $x\sigma = t$, and for each $y \in \mathit{var}(\ell)\backslash\{x\}$, $y\sigma$ is an arbitrary ground term in $L(\mathcal{A}_0, x\theta)$. Again, we can apply the induction hypothesis to $\ell\sigma \xrightarrow[\mathcal{A}^*]{*} \ell\theta \xrightarrow[\mathcal{A}_{k-1}]{*} \langle q, s\rangle$ and find u and $\overline{\ell\sigma}$ such that (i_2) $u \xrightarrow[\mathcal{A}_0]{*} \langle q, s\rangle$ and (ii_2) $u \xrightarrow[\mathcal{R},s]{bu}{}^* \overline{\ell\sigma}$. Since $s \cap \mathit{final}(\mathcal{C}_L) \neq \emptyset$, it holds that $\overline{\ell\sigma} \xrightarrow[\mathcal{R},s]{bu} \widetilde{t}$, and we can conclude with $\overline{t} = \widetilde{t}$. □

B Proof of Lemma 9

We present the rest of the case analysis for the proof by induction of Lemma 9.

Lemma 9. *For all $u \in \mathcal{T}(\Sigma)$, $\overline{t} \in \mathcal{T}(\Sigma \cup \overline{\Sigma})$, state $\langle q, s\rangle \in Q$, if $u \xrightarrow[\mathcal{R},s]{bu,*} \overline{t}$, and $u \xrightarrow[\mathcal{A}_0]{*} \langle q, s\rangle$, then $t \xrightarrow[\mathcal{A}^*]{*} \langle q, s\rangle$. Moreover, if the top symbol of $\overline{t}$ is in Σ, then the last step in $t \xrightarrow[\mathcal{A}^*]{*} \langle q, s\rangle$ is not an ε-transition.*

Proof. We do a proof by induction on the lexical combination of the length of the reduction $u \xrightarrow[\mathcal{R},s]{bu}{}^* \overline{t}$ and the structure of u.

Case 0. If there is no reduction at the root position, then we can write $u = g(u_1, \ldots, u_n)$ and $\overline{t} = g(\overline{t_1}, \ldots, \overline{t_n})$ and we can assume *wlog* that the rewrite reduction has the form

$$u = g(u_1, \ldots, u_n) \xrightarrow[\mathcal{R},s]{bu}{}^* g(\overline{t_1}, u_2, \ldots, u_n) \xrightarrow[\mathcal{R},s]{bu}{}^* \cdots \xrightarrow[\mathcal{R},s]{bu}{}^* g(\overline{t_1}, \ldots, \overline{t_n}) = \overline{t}.$$

Let us decompose the reduction $u \xrightarrow[\mathcal{A}_0]{*} \langle q, s\rangle$ into

$$u = g(u_1, \ldots, u_n) \xrightarrow[\mathcal{A}_0]{*} g\big(\langle q_1, s_1\rangle, \ldots, \langle q_n, s_n\rangle\big) \xrightarrow[\mathcal{A}_0]{} \langle q, s\rangle.$$

For all i, $1 \leq i \leq n$, it holds that $s \xrightarrow[\Theta]{\langle g,i\rangle} s_i$ by construction of $\mathcal{A}_0$, and $u_i \xrightarrow[\mathcal{R},s_i]{bu}{}^* \overline{t_i}$ by Lemma 6. Hence by induction hypothesis $t_i \xrightarrow[\mathcal{A}^*]{*} \langle q_i, s_i\rangle$. It follows that $t = g(t_1, \ldots, t_n) \xrightarrow[\mathcal{A}^*]{*} g\big(\langle q_1, s_1\rangle, \ldots, \langle q_n, s_n\rangle\big) \xrightarrow[\mathcal{A}_0]{} \langle q, s\rangle$.

Case 1. If some rewrite steps are performed at the root position, then we have two cases.

Case 1.1. First, we consider that the last rewrite step performed at the root position involves a non-collapsing rule.

Since $u \xrightarrow[\mathcal{R},s]{bu}{}^* \overline{t}$ is a bottom-up rewrite sequence, no earlier reduction is performed at the root position with a collapsing rule (because the root symbol of every redex in a bottom-up reduction must be in Σ), and the top symbol of $\overline{t}$ is in Σ. We can write the reduction sequence as follows:

$$u \xrightarrow[\mathcal{R},s]{bu}{}^* \overline{\ell\sigma} \xrightarrow[\mathcal{R},s]{bu} g(r_1, \ldots, r_m)\,\widetilde{\sigma} \xrightarrow[\mathcal{R},s]{bu}{}^* \overline{t}$$

where $L : \ell \to g(r_1, \ldots, r_m) \in \mathcal{R}$ and $\varepsilon \in L$. Without loss of generality, we assume that $r_1, \ldots, r_i$ are ground terms and $r_{i+1}, \ldots, r_m$ are variables for some $i \leq m$. By induction hypothesis, it holds that $\ell\sigma \xrightarrow[\mathcal{A}^*]{*} \langle q, s\rangle$ and the last step of this reduction is not an ε-transition. This reduction sequence can be decomposed into $\ell\sigma \xrightarrow[\mathcal{A}^*]{*} \ell\theta \xrightarrow[\mathcal{A}^*]{*} \langle q, s\rangle$ where θ is a substitution from $\mathcal{X}$ into Q, grounding for $var(\ell)$. Note that we use the assumption that $\mathcal{R}$ is linear in order to construct this θ. Moreover, $s \cap final(\mathcal{C}_L) \neq \emptyset$. Then from the construction case (Rules 1), $\mathcal{A}^*$ contains the transition rule $g(\langle q_1, s_1\rangle, \ldots, \langle q_m, s_m\rangle) \to \langle q, s\rangle$ where

– for all j with $1 \le j \le i$, $q_j = q_{r_j}$,
– for all j with $i < j \le m$, $\langle q_j, s_j \rangle = r_j\theta$,
– for all j with $1 \le j \le m$, $s \xrightarrow[\Theta]{\langle g,j \rangle} s_j$.

For all j with $1 \le j \le i$, $r_j \xrightarrow[A_0]{*} \langle q_j, s_j \rangle$ by the construction of $q_{r_j} = q_j$ and Lemma 7(i), and for all j with $i \le j \le m$, $r_j\sigma \xrightarrow[\mathcal{A}^*]{*} r_j\theta$. Therefore

$$\begin{aligned} t &= g(r_1, \ldots, r_i, r_{i+1}\sigma \ldots, r_m\sigma) \\ &\xrightarrow[\mathcal{A}_0]{*} g(\langle q_1, s_1 \rangle, \ldots, \langle q_i, s_i \rangle, r_{i+1}\sigma \ldots, r_m\sigma) \\ &\xrightarrow[\mathcal{A}^*]{*} g(\langle q_1, s_1 \rangle, \ldots, \langle q_i, s_i \rangle, \langle q_{i+1}, s_{i+1} \rangle, \ldots, \langle q_m, s_m \rangle) \\ &\xrightarrow[\mathcal{A}_0]{*} \langle q, s \rangle. \end{aligned}$$

Case 1.2. Now, we consider the case where the last rewrite step performed at the root position involves a collapsing rule. Then the rewrite sequence has following form:

$$u \xrightarrow[\mathcal{R},s]{bu}{}^{*} \overline{\ell\sigma} \xrightarrow[\mathcal{R},s]{bu} x\widetilde{\sigma} = \bar{t}$$

where $L : \ell \to x \in \mathcal{R}$ and $\varepsilon \in L$. Similarly to the Case 1.1, by induction hypothesis, it holds that $\ell\sigma \xrightarrow[A^*]{*} \langle q, s \rangle$ and the last step of this reduction is not an ε-transition. This reduction sequence can be decomposed into $\ell\sigma \xrightarrow[\mathcal{A}^*]{*} \ell\theta \xrightarrow[\mathcal{A}^*]{*} \langle q, s \rangle$ where θ is a substitution from $\mathcal{X}$ into Q grounding for $var(\bar{\ell})$. It means that $x\sigma \xrightarrow[\mathcal{A}^*]{*} x\theta$. From the completion step (Rules 2), since $s \cap \mathit{final}(\mathcal{C}_L) \neq \emptyset$, $\mathcal{A}^*$ contains the ε-transition rule $x\theta \to \langle q, s \rangle$. Hence $t = x\sigma \xrightarrow[\mathcal{A}^*]{*} x\theta \xrightarrow[\mathcal{A}^*]{} \langle q, s \rangle$. □