Controlled Term Rewriting[⋆]

Florent Jacquemard[1], Yoshiharu Kojima[2,3], and Masahiko Sakai[2]

[1] INRIA & LSV, ENS Cachan, 61 av. Pdt Wilson 94230 Cachan, 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
[3] Research Fellow of the Japan Society for the Promotion of Science

Abstract. Motivated by the problem of verification of imperative tree transformation programs, we study the combination, called controlled term rewriting systems (CntTRS), of term rewriting rules with constraints selecting the possible rewrite positions. These constraints are specified, for each rewrite rule, by a selection automaton which defines a set of positions in a term based on tree automata computations.
We show that reachability is PSPACE-complete for so-called monotonic CntTRS, such that the size of every left-hand-side of every rewrite rule is larger or equal to the size of the corresponding right-hand-side, and also for the class of context-free non-collapsing CntTRS, which transform Context-Free (CF) tree language into CF tree languages.
When allowing size-reducing rules, reachability becomes undecidable, even for flat CntTRS (both sides of rewrite rules are of depth at most one) when restricting to words (*i.e.* function symbols have arity at most one), and for ground CntTRS (rewrite rules have no variables).
We also consider a restricted version of the control such that a position is selected if the sequence of symbols on the path from that position to the root of the tree belongs to a given regular language. This restriction enables decision results in the above cases.

Introduction

Term rewriting is a rule-based formalism for describing computations in ranked terms. In the context of formal verification, term rewriting systems (TRS) can be used to provide a finite abstraction of the dynamics of a system whose configurations are represented by ranked terms. In this case, the rewrite relation represents the transitions between configurations. For instance, functional programs manipulating (tree) structured data values with pattern matching can be described by rewrite rules [18] such that the rewriting relation represents the program evaluation. This approach can also be applied to systems [1] or imperative programs [4,17] modifying some parts of tree shaped data structures in

⋆ This work has been partly supported by the ERC research project FoX under grant agreement FP7-ICT-233599 and by the INRIA ARC project ACCESS.

place, while leaving the rest unchanged. These update operations can be modeled by rewrite rules or similarly, tree transducers. A crucial problem for the automatic reachability and flow analysis of programs is to find finite and decidable representations of the closure the above TRS representations of sets of terms representing configurations. This approach is related to *static type checking* of XML transformations (see *e.g.* [22]), which is the problem to verify that a program always converts valid source trees (documents) into valid output trees (where types are defined by TA). It has also been shown useful for the analysis of consistency of XML read and write access control policies [17].

A rewrite rule is an oriented equation, whose left-hand-side (*lhs*) describe a pattern to be replaced in a term, and whose right-hand-side (*rhs*) is the new term for replacement. It can be applied at any position in a term, providing that the *lhs* matches the subterm at this position. For instance, a rule with *lhs* $a(x_1, x_2)$ can be applied at any position labelled by a. This simple approach is in general relevant in domains like theorem proving or algebraic computation. For some applications however, like the analysis of programs for XML document transformations or of access-control policies, it is important to be able to express explicitly some *context conditions* to be checked before applying a rewrite rule. For instance, one may want to rename the label at some position with a rewrite rule $a(x) \rightarrow b(x)$ providing that there is no occurrence of b above the position to be rewritten (position labeled with a). Some standard XML transformation languages like XSLT or XQuery update [6], use the path specification language XPath or a XQuery expressions in order to define the position where the transformation can be applied.

In this paper, we study the so called *controlled term rewriting systems* (CntTRS) in the context of regular tree model checking. They are defined by the combination of term rewriting rules with some constraints (called control) specifying the possible rewrite positions. In order to define the constraints, we have chosen a model similar to the selection tree automata (SA) of [13], which, intuitively, select positions in a term based on the computations of a tree automaton. This gives a powerful selection mechanism, with the same expressiveness as the monadic second order logic of the tree, or monadic Datalog [14]. We consider also a restriction of the SA where a position p in a term t is selected if the sequence of symbols on the path from p to the root of t belongs to a given regular language. The corresponding restricted rewriting model is called prefix CntTRS, or pCntTRS. It turns out quickly (Examples at the end of Section 1) that even the restricted pCntTRS are actually too powerful for preserving regularity, even for very simple rewrite rules.

Therefore, we consider in Section 2 the classes of *context-free* (CF) and *context-sensitive* (CS) tree languages, which are both strictly larger than the class of TA languages (also called *regular* tree languages). We also define the so called CF and *monotonic* classes of rewrite systems (without control). A rewrite rule is CF if its *lhs* is of the form $f(x_1, \ldots, x_n)$ where $x_1, \ldots, x_n$ are distinct variables, and monotonic if the size of its *lhs* is larger or equal to the size of its *rhs*. We show that CF and monotonic TRS respectively preserve CF tree languages and CS tree languages.

The monotonic uncontrolled rewrite rules already have the full power of CS tree grammars. Adding control with SA does not improve their expressiveness, and it follows that reachability is PSPACE-complete and model checking undecidable for monotonic CntTRS (Section 3.1). Similar results also hold for CF CntTRS without collapsing rules (projection rules of the form $f(x_1, \ldots, x_n) \to x_i$), even when restricting to prefix control (Section 3.2).

When allowing depth-reducing rules (Section 3.3), reachability becomes undecidable, even for flat CntTRS (*lhs* and *rhs* of rewrite rules are of depth at most one) and in the case of words (*i.e.* function symbols of arity at most one). Similarly, reachability is undecidable for ground CntTRS (rewrite rules have no variables). When restricting to words, prefix control and flat rewrite rules, reachability is decidable in PSPACE.

Finally, we consider in Section 4 a relaxed form of the prefix control of rewrite rules, where the selection is done by considering the term in input modulo the rewrite relation. We obtain a regularity preservation result for this recursive form of prefix controlled rewriting, using alternating tree automata with ε-transitions.

Related Work. Controlled rewrite systems have been introduced in the case of word rewriting, see [28] for a survey, and also [9] for the case of conditional context-free (word) grammars, which are mentioned in Section 3.

Regarding term rewriting, there have been many studies for finding syntactical restriction on term rewrite rules ensuring the preservation of regularity, see *e.g.* [12]. This is the case for instance of linear and flat as well as ground TRS without control, in contrast to the results of Section 3.3 for their controlled counterpart.

Many strategies have been proposed for term rewriting, most often for efficiency purposes (our goal here is rather to study the expressiveness and decidability results for controlled rewriting). We cannot mention all of them, and will just give some elements of comparison with controlled rewriting. The *innermost* strategy, which is the analogous of *call-by-value* for functional languages (a subterm can be rewritten only when all its proper subterms are no more rewritable) can be expressed in controlled rewriting for left-linear TRS, because the set of non-rewritable terms (the *normal forms*) for such TRS are recognizable by tree automata (see *e.g.* [8]). Some results of preservation of regularity for the innermost and the outermost and leftmost rewrite strategies can be found in [26] In the *context sensitive* strategy [21] (which is not related to the context-sensitive term languages presented in this paper), the rewriting positions are selected according to a mapping μ associating to every symbol of the signature the subset of the indexes of its argument that can be rewritten. More precisely, the positions selected for rewriting in a term $f(t_1, \ldots, t_n)$ are the root position and all the positions selected in every t_i such that $i \in \mu(f)$. This set of positions can be defined by SA, *i.e.* context-sensitive rewriting is a particular case of controlled rewriting. It is a strict subcase because with the context-sensitive strategy, the root position is always rewritable whereas this is not the case for controlled rewriting.

A result of preservation of regularity for this rewrite strategy was established in [19]. Another related topic is the optimal *call-by-need* rewrite strategies for TRS. In [7] and [10], it is shown how to select the *needed redexes* (positions at which rewriting must be performed in order to transform a term to its normal form) using monadic second order logic formulae, which is equivalent as using SA.

Top-down tree transducers with regular look-ahead [11] are an extension of top-down tree transducers, where a transition can be fired provided that the current subtree belongs to some given regular tree language. This is similar to our notion of control for rewrite systems. A notable difference however is that the transducers transform terms in one (top-down) pass, whereas we consider here the terms computed by an arbitrary iteration of controlled rewrite rules.

1 Preliminaries

Terms. We use the standard notations for terms and positions, see [2]. A *signature* Σ is a finite set of function symbols with arity. We denote the subset of function symbols of Σ of arity n as Σ_n. 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 of *ground* terms (terms without variables) is denoted $\mathcal{T}(\Sigma)$. The set of variables occurring in a term $t \in \mathcal{T}(\Sigma, \mathcal{X})$ is denoted $\mathit{vars}(t)$. A signature is called *unary* if it contains only symbols of arity 1 and one symbol $\perp$ of arity 0. We make no distinction below between ground terms over a unary signature Σ and words of Σ_1^*. More precisely, a term $a_1(a_2(\ldots a_n(\perp)))$ will be represented by the string $a_1\, a_2 \ldots a_n$ (the constant symbol $\perp$ is forgotten in this representation).

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 $p.p'$ denotes the concatenation of positions p and p'. The set $\mathcal{P}os(t)$ of positions of the term t is defined recursively as $\mathcal{P}os\big(f(t_1, \ldots, t_m)\big) = \{\varepsilon\} \cup \{i.p \mid i \in \{1, \ldots, m\} \wedge p \in \mathcal{P}os(t_i)\}$. The *height* of a term t, denoted $h(t)$, is the maximal length of a position of $\mathcal{P}os(t)$. The *size* $\|t\|$ of a term t is the cardinality of $\mathcal{P}os(t)$.

The *subterm* of t at position p is denoted $t|_p$ defined by $t|_\varepsilon = t$ and $f(t_1, \ldots, t_m)|_{i.p} = t_i|_p$. The term obtained from t by replacing subterm of t at position p by s, is denoted $t[s]_p$. The notation $t = t[s]_p$ may also be used to emphasize that $t|_p$ is s.

A *substitution* is a mapping $\mathcal{X} \to \mathcal{T}(\Sigma, \mathcal{X})$. Substitutions can also be applied to arbitrary terms by homomorphically extending its application to variables. The application of a substitution σ to a term t, denoted as $t\sigma$, is defined as follows for non-variable terms: $f(t_1, \ldots, t_n)\sigma = f(t_1\sigma, \ldots, t_n\sigma)$. A *variable renaming* is a substitution from variables to variables.

A term is *linear* if no variable occurs more than once in it. A term is *shallow* if each occurrence of variables is at most depth one, and *flat* if its height is at most one.

A *context* of dimension n is a linear term $C \in \mathcal{T}(\Sigma, \{x_1, \ldots, x_n\})$. When $C = x_1$, it is called the *empty context*. Given a context C of dimension n and n terms $t_1, \ldots, t_n \in \mathcal{T}(\Sigma, \mathcal{X})$, we write $C[t_1, \ldots, t_n]$ to denote $C\sigma$ where σ is the substitution $\{x_1 \mapsto t_1, \ldots, x_n \mapsto t_n\}$.

Tree Automata. 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 is a set of states of Q called final states, Δ is a set of transition rules of the form: $f(q_1, \ldots, q_n) \to q$, where $f \in \Sigma_n$, and $q_1, \ldots, q_n, q \in Q$. Sometimes, we shall refer to $\mathcal{A}$ as a subscript of its components, like in $Q_\mathcal{A}$ to indicate that Q is the state set of $\mathcal{A}$. The size of $\mathcal{A}$ is $\|\mathcal{A}\| = \sum_{f(q_1,\ldots,q_n)\to q \in \Delta}(n+2)$. Transition from s to t in one step by a TA $\mathcal{A}$ is denoted by $s \xrightarrow[\mathcal{A}]{} t$, and its reflexive and transitive closure is denoted by $s \xrightarrow[\mathcal{A}]{*} t$.

A *run* of $\mathcal{A}$ on a term $t \in \mathcal{T}(\Sigma)$ is a mapping ρ from $\mathcal{P}os(t)$ into $Q_\mathcal{A}$ such that for all $p \in \mathcal{P}os(t)$, $t(p)\big(\rho(p.1), \ldots, \rho(p.n)\big) \to \rho(p)$ is in $\Delta_\mathcal{A}$, where n is the arity of the symbol $t(p)$ in Σ. The run ρ is called *successful* (or *accepting*) if $\rho(\varepsilon)$ is in $F_\mathcal{A}$. The set of successful runs of $\mathcal{A}$ on t is denoted *sruns*$(\mathcal{A}, t)$. For the sake of conciseness, we shall sometimes apply term-like notations (subterm, replacement...) to runs. The language $\mathcal{L}(\mathcal{A})$ of $\mathcal{A}$ is the set of terms t for which *sruns*$(\mathcal{A}, t)$ is not empty.

Selection Automata. Besides being able to recognize terms, tree automata can also be used to select positions in a term [13,24]. We propose here a definition of position selection by TA very close to [13].

A *selection automaton* (SA) $\mathcal{A}$ over a signature Σ is a tuple $\langle Q, F, S, \Delta \rangle$ where $\langle Q, F, \Delta \rangle$ is a tree automaton denoted *ta*$(\mathcal{A})$ and S is a set of states of Q called selection states. Given a SA $\mathcal{A}$ and a term $t \in \mathcal{T}(\Sigma)$, the set of positions of t selected by $\mathcal{A}$ is defined as

$$sel(\mathcal{A}, t) = \{p \in \mathcal{P}os(t) \mid \exists \rho \in sruns(ta(\mathcal{A}), t), \rho(p) \in S\}.$$

Note that it is required that t is recognized by $\mathcal{A}$ in order to select positions.

We shall consider below a restricted kind of selection by TA, where a position p in a term t is selected only according to its strict prefix (*i.e.* the sequence of symbols labeling the path from the root of t down to the immediate ancestor of p), which is tested for membership to a regular (word) language. More precisely, a selection automaton $\mathcal{A} = \langle Q, F, S, \Delta \rangle$ is called *prefix* if Q contains two special states: q_0 (universal state) and q_s (selection state), $F \subseteq Q \setminus \{q_0\}$, $S = \{q_s\}$, and Δ contains $f(q_0, \ldots, q_0) \to q_0$ and $f(q_0, \ldots, q_0) \to q_s$ for all $f \in \Sigma$, and Δ contains some other transition rules of the form $f(q_1, \ldots, q_n) \to q$ where $q \in Q \setminus \{q_0, q_s\}$ and there exists exactly one $i \leq n$ such that $q_i \neq q_0$. Intuitively, assume that we are given a finite automaton $\mathcal{B}$ defining the strict prefixes of selected positions. Then q_s is the initial state of $\mathcal{B}$, F is the set of final states of $\mathcal{B}$, and for all $a \in \Sigma_n$, Δ contains n rules $a(q_0, \ldots, q_0, q', q_0, \ldots, q_0) \to q$ for each transition $q' \xrightarrow{a} q$ of $\mathcal{B}$. Note that with this definition, the root position is always selected by a by a prefix selection automaton.

Controlled Term Rewriting Systems. We propose a formalism that strictly extends standard term rewriting systems [2] by the restricting the possible rewrite positions to positions selected by a given SA. Formally, a *controlled term rewriting system* (CntTRS) $\mathcal{R}$ over Σ is a finite set of *controlled rewrite rules* of the form $\mathcal{A} : \ell \to r$, made of a SA $\mathcal{A}$ over Σ and a *rewrite rule* $\ell \to r$ such that $\ell \in \mathcal{T}(\Sigma, \mathcal{X}) \setminus \mathcal{X}$ (the left-hand side of the rule), and $r \in \mathcal{T}(\Sigma, \mathit{vars}(\ell))$ (the right-hand side). Ths size of $\mathcal{R}$ is the number of the rewrite rules inn $\mathcal{R}$.

A term s rewrites to t in one step by an CntTRS $\mathcal{R}$, denoted by $s \xrightarrow[\mathcal{R}]{} t$, if there exists a controlled rewrite rule $\mathcal{A} : \ell \to r \in \mathcal{R}$, a position $p \in \mathit{sel}(\mathcal{A}, s)$, and a substitution σ such that $s|_p = \ell\sigma$ and $t = s[r\sigma]_p$. In this case, s is said to be $\mathcal{R}$-*reducible*, and otherwise s is called an $\mathcal{R}$-*normal form*. The reflexive and transitive closure, and reflexive, symmetric and transitive closure of $\xrightarrow[\mathcal{R}]{}$ are denoted as $\xrightarrow[\mathcal{R}]{*}$ and $\xleftrightarrow[\mathcal{R}]{*}$, and $\xrightarrow[\mathcal{R}]{=}$ denotes the union of $\xrightarrow[\mathcal{R}]{}$ and $=$.

Example 1. Let us consider the CntTRS $\mathcal{R} = \{(1)\, \mathcal{A}_1 : a \to c, (2)\, \mathcal{A}_2 : b \to c, (3)\, \mathcal{A}_3 : f(x,y) \to g(x,y)\}$ where each SA is as follows ($Q = \{q_1, q_2, q_f\}$):

$$\begin{aligned}\mathcal{A}_1 &= \langle Q, \{q_f\}, \{q_1\}, \{a \to q_1, b \to q_2, f(q_1, q_2) \to q_f\}\rangle\\ \mathcal{A}_2 &= \langle Q, \{q_f\}, \{q_2\}, \{c \to q_1, b \to q_2, g(q_1, q_2) \to q_f\}\rangle\\ \mathcal{A}_3 &= \langle Q, \{q_f\}, \{q_f\}, \{c \to q_1, b \to q_2, f(q_1, q_2) \to q_f\}\rangle\end{aligned}$$

Then, the following rewriting is possible with $\mathcal{R}$ $f(a,b) \xrightarrow[(1)]{} f(c,b)$ since $\mathit{sel}(\mathcal{A}_1, f(a,b)) = \{1\}$ and the subterm of $f(a,b)$ at the position 1 is a. Similarly, we have $f(c,b) \xrightarrow[(3)]{} g(c,b) \xrightarrow[(2)]{} g(c,c)$ where $\mathit{sel}(\mathcal{A}_3, f(c,b)) = \{\varepsilon\}$, and $\mathit{sel}(\mathcal{A}_2, g(c,b)) = \{2\}$.

We call *prefix controlled term rewriting system* (pCntTRS), resp. *term rewriting systems* (TRS), the special cases of CntTRS such that every SA in a controlled rewrite rule is a prefix SA, resp. is the *universal* SA $\mathcal{A}_0 = \langle\{q_0\}, \{q_0\}, \{q_0\}, \{f(q_0, \ldots, q_0) \to q_0 \mid f \in \Sigma\}\rangle$. In the latter case, $\mathcal{A}_0$ may be dropped when defining the rewrite rules, *i.e.* we present a TRS as a finite set of uncontrolled rewrite rules, as usual.

A controlled rewrite rule $\mathcal{A} : \ell \to r$ is ground, *flat*, *linear*, *shallow* if ℓ and r are so. It is *collapsing* if r is a variable. A CntTRS is *flat*, *linear*, etc if all its rules are so.

Decision Problems. The *closure* of a ground term set L by a CntTRS $\mathcal{R}$ is $\{t \mid \exists s \in L, s \xrightarrow[\mathcal{R}]{*} t\}$ (it is sometimes denoted $\mathcal{R}^*(L)$).
Ground reachability is the problem to decide, given two ground terms $s, t \in \mathcal{T}(\Sigma)$ and a CntTRS $\mathcal{R}$ whether $s \xrightarrow[\mathcal{R}]{*} t$. *Regular Model checking* (RMC) is the problem to decide, given two TA languages L_{in} and L_{err} and a CntTRS $\mathcal{R}$ whether $\mathcal{R}^*(L_{\mathsf{in}}) \cap L_{\mathsf{err}} = \emptyset$. The name of the problem 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.

Example 2. Let us consider the following CntTRS $\mathcal{R}$ over the unary signature Σ with $\Sigma_1 = \{a, b, c, d\}$ and $\Sigma_0 = \{\bot\}$. Let $\mathcal{R}$ be the CntTRS containing the following controlled rewrite rules. The SA of these rules select one position per term, and they are represented by a regular expression where the selected letter is underlined.

$$\begin{array}{llll} (1)\ c^*a^*d^*\underline{d}\,b^* & : d(x) \to b'(x) & (2)\ c^*\underline{c}\,a^*d^*b'b^* : & c(x) \to a'(x) \\ (3)\ c^*a'a^*d^*\underline{b'}\,b^* & : b'(x) \to b(x) & (4)\ c^*\underline{a'}\,a^*d^*b^* & : a'(x) \to a(x) \end{array}$$

More precisely, the SA for the above rules are respectively (Q is the state set $\{q_a, q_b, q_c, q_d, q\}$)

$$\begin{aligned} \mathcal{A}_1 = \big\langle Q, \{q_c, q_a, q_d, q\}, \{q\}, \{&\perp \to q_b, b(q_b) \to q_b, d(q_b) \to q, d(q|q_d) \to q_d, \\ &a(q|q_d|q_a) \to q_a, c(q_a|q_c|q) \to q_c\}\big\rangle \\ \mathcal{A}_2 = \big\langle Q, \{q_c, q\}, \{q\}, \{&\perp \to q_b, b(q_b) \to q_b, b'(q_b) \to q_d, d(q_d) \to q_d, \\ &a(q_d|q_a) \to q_a, c(q_a) \to q, c(q|q_c) \to q_c\}\big\rangle \\ \mathcal{A}_3 = \big\langle Q, \{q_c\}, \{q\}, \{&\perp \to q_b, b(q_b) \to q_b, b'(q_b) \to q, d(q|q_d) \to q_d, a(q|q_d) \to q_a, \\ &a'(q|q_d|q_a) \to q_c, c(q_c) \to q_c\}\big\rangle \\ \mathcal{A}_4 = \big\langle Q, \{q_c, q\}, \{q\}, \{&\perp \to q_b, b(q_b) \to q_b, d(q_b|q_d) \to q_d, a(q_b|q_d|q_a) \to q_a, \\ &a'(q_b|q_d|q_a) \to q, c(q|q_c) \to q_c\}\big\rangle \end{aligned}$$

Note that these SA are all deterministic. The SA $\mathcal{A}_1$ selects the last d (starting from the top), $\mathcal{A}_2$ selects the last c when there is a b', $\mathcal{A}_3$ selects the b' when there is a a', and $\mathcal{A}_4$ selects the a' when there is no b'. The closure of the regular tree language $L = c^+(d^+(\perp))$ by the CntTRS is such that $R^*(L) \cap a^*(b^*(\perp)) = \{a^nb^n \leq n \geq 0\}$. Therefore, $R^*(L)$ is a non regular tree language (it is a context-free tree language).

Example 3. Using the same signature as in Example 2, we can obtain a context-free set of descendants with a flat pCntTRS. Indeed, intersection of a^*b^* and the closure of c^*d^* by the following set of rewrite rules is $\{a^nb^m \mid n \geq m\}$ which is CF and not regular. In the rewrite rules, we use an informal description of the languages of the prefix allowed, instead of giving explicitly the prefix SA.

no a', no a :	$c(x) \to a'(x)$,	exactly one a' :	$d(x) \to b'(x)$,
no a', no a :	$a'(x) \to a(x)$,	no a', no b', no b :	$b'(x) \to b(x)$.

It is not difficult to generalize the construction of Example 3 in order to obtain a context sensitive rewrite closure of the form $\{a^nb^mc^p \mid n \geq m \geq p\}$, starting from a regular set of the form $c^*d^*e^*$ and using a flat pCntTRS.

2 CF and CS Tree Languages and TRS

In this section, we define the context-free and context-sensitive sets of terms, and give properties of their closure under term rewriting.
A rewrite rule over Σ is called

context-free (CF) if it is of the form $f(x_1, \ldots, x_n) \to r$ where $r \in \mathcal{T}\big(\Sigma, \{x_1, \ldots, x_n\}\big)$, $x_1, \ldots, x_n$ are distinct variables and $f \in \Sigma_n$. Recall that when $r = x_i$ for some $i \leq n$, then the rule is called collapsing.

monotonic if it is of the form $C[x_1, \ldots, x_n] \to D[x_1, \ldots, x_n]$ where C and D are two contexts over Σ and such that $\|C\| \leq \|D\|$ and $x_1, \ldots x_n$ are distinct variables (note that it implies that the rule is linear).

A *tree grammar* (TG, see *e.g.* [8]) is a tuple $\mathcal{G} = \langle \mathcal{N}, S, \Sigma, P \rangle$ where $\mathcal{N}$ is a finite set of *non-terminal* symbols, each with an arity, $S \in \mathcal{N}$ has arity 0, it is called the *axiom* of $\mathcal{G}$, Σ is a signature disjoint from $\mathcal{N}$, (its elements are also called *terminal* symbols) and P is a set of (uncontrolled) rewrite rules, called *production rules*, of the form $\ell \to r$ where ℓ, r are terms of $\mathcal{T}(\Sigma \cup \mathcal{N}, \mathcal{X})$ such that ℓ contains at least one non-terminal. The tree grammar $\mathcal{G}$ is *regular* if all non-terminal symbols of $\mathcal{N}$ have arity 0 and all production rules of P have the form $A \to r$, with $A \in \mathcal{N}$ and $r \in \mathcal{T}(\Sigma \cup \mathcal{N})$. It is *context-free* (CFTG), resp. *context-sensitive* (CSTG) if all production rules are CF, resp. monotonic. In the two later cases, we assume from now on *wlog* that every production rule either has the form $A(x_1, \ldots, x_n) \to a(x_1, \ldots, x_n)$ where $A \in \mathcal{N}$ and $a \in \Sigma_n$, or it does not contain terminal symbols of Σ, by introducing the non-terminal symbol $\langle b \rangle$, the production rule $\langle b \rangle \to b$, and replace all b in the other production rules by $\langle b \rangle$.

The *language* generated by $\mathcal{G}$, denoted by $L(\mathcal{G})$, is the set of terms of $\mathcal{T}(\Sigma)$ which can be reached by successive applications of the production rules, starting from the axiom, *i.e.* $L(\mathcal{G}) = \{t \in \mathcal{T}(\Sigma) \mid S \xrightarrow[P]{*} t\}$. A tree language is called regular (resp. CF, CS) if it is the language of a regular (resp. CF, CS) grammar.

Note that the classical cases of word languages are particular cases of the above, if the symbols of $\mathcal{N} \cup \Sigma$ are unary symbols of a unary signature.

The regular tree grammars are equivalent in expressiveness to TA. There exists a model of pushdown TA equivalent to the CF tree grammars [15].
The *membership* problem is, given a tree grammar $\mathcal{G}$ over Σ and a term $t \in \mathcal{T}(\Sigma)$, to decide whether $t \in \mathcal{L}(\mathcal{G})$.
The *emptiness* problem is, given a tree grammar $\mathcal{G}$, to decide whether $\mathcal{L}(\mathcal{G}) = \emptyset$.

Proposition 1. *Membership and emptiness are decidable in PTIME for CFTG.*

The following result (perhaps a folklore knowledge) is almost immediate from the above definitions.

Proposition 2. *Given a CFTG $\mathcal{G}$ and a CF TRS $\mathcal{R}$, one can construct in PTIME a CFTG generating the closure of $L(\mathcal{G})$ by $\mathcal{R}$, and whose size is polynomial in the size of $\mathcal{G}$ and $\mathcal{R}$.*

Proof. Let $\mathcal{G} = \langle \mathcal{N}, S, \Sigma, P \rangle$ be a CFTG and $\mathcal{R}$ a CF TRS over Σ. For all $a \in \Sigma$, we create a new non-terminal N_a with the same arity as a. Let $\mathcal{N}' = \mathcal{N} \cup \{N_a \mid a \in \Sigma\}$, and let P' be obtained from P by replacing every production rule $A(x_1, \ldots, x_n) \to a(x_1, \ldots, x_n)$, with $A \in \mathcal{N}$ and $a \in \Sigma$, by $A(x_1, \ldots, x_n) \to N_a(x_1, \ldots, x_n)$. Moreover, we add to P' the rules obtained from the rules of $\mathcal{R}$ by replacing every symbol $a \in \Sigma$ by N_a. The CFTG $\mathcal{G}' = \langle \mathcal{N}', S, \Sigma, P' \rangle$ generates the closure of $L(\mathcal{G})$ by $\mathcal{R}$. $\square$

Corollary 1. *Reachability and RMC are decidable in PTIME for CF TRS.*

Proof. For the RMC, we use the fact that the intersection of the languages of a CF tree grammar $\mathcal{G}$ and a TA $\mathcal{A}$ is the language of a CF tree grammar whose size is the product of the respective sizes of $\mathcal{G}$ and $\mathcal{A}$. □

Note however that *joinability*, the problem to decide, given two ground terms $s, t \in \mathcal{T}(\Sigma)$ and a TRS $\mathcal{R}$, whether there exists $u \in \mathcal{T}(\Sigma)$ such that $s \xrightarrow[\mathcal{R}]{*} u \xleftarrow[\mathcal{R}]{*} t$, is undecidable for CF TRS [5], because the emptiness of intersection is undecidable for CF tree languages.

We can also observe that the CF TRS are left linear, but in general not right linear. They are the symmetric the so called right-linear, monadic and non-collapsing TRS, whose rules have the form $f(x_1, \ldots, x_n) \to r$, where $r \in \mathcal{T}(\Sigma, \{x_1, \ldots, x_n\}) \setminus \mathcal{X}$. It has been shown that these TRS preserve regularity [23]: the closure of a regular tree language by such a TRS is a regular tree language. The decidability of reachability for CF TRS is already a consequence of this former result. It has been observed, see *e.g.* [16], that in several cases, one class of word rewrite system preserves regularity and its symmetric class preserves CF languages. We have here an example of such a situation in the case of terms.

To our knowledge, the case of CSTG and monotonic TRS was not studied before but it is not very surprising.

Proposition 3. *Membership is PSPACE-complete for CSTG.*

Proof. The hardness is an immediate consequence of the same result for CS (word) grammars [20], which are a particular case of CSTG. For the decision algorithm, let $\mathcal{G}$ be a CSTG over Σ and let $t \in \mathcal{T}(\Sigma)$ be given. We can observe that if two terms s and s' over the non-terminal and terminal symbols of the CSTG $\mathcal{G}$ are successive in a derivation starting from the axiom S of $\mathcal{G}$, then the size $\|s'\|$ is larger or equal to $\|s\|$. Hence, if there is a derivation from S to t by $\mathcal{G}$, then all the terms in this derivation have a size smaller or equal to $\|t\|$. Hence, it is possible to construct a linear bounded automata which, starting from t, will search backward (non deterministically) a derivation from S. The detailed construction is given in Appendix A. □

Proposition 4. *Emptiness is undecidable for CSTG.*

Proof. It is a consequence of the same result for CS (word) grammars. □

Proposition 5. *Given a CSTG $\mathcal{G}$ and a monotonic TRS $\mathcal{R}$, one can construct in PTIME a CSTG generating the closure of $L(\mathcal{G})$ by $\mathcal{R}$, and whose size is polynomial in the size of $\mathcal{G}$ and $\mathcal{R}$.*

Proof. The construction is similar as the one in the proof of Proposition 2. □

3 Controlled Term Rewriting

3.1 Monotonic CntTRS

The result of Proposition 5 can be extended from uncontrolled to controlled monotonic TRS. Intuitively, monotonic TRS are powerful enough to be able to simulate a control with uncontrolled rewrite rules.

Theorem 1. *Given a CSTG $\mathcal{G}$ and a monotonic CntTRS $\mathcal{R}$, one can construct in PTIME a CSTG generating the closure of $L(\mathcal{G})$ by $\mathcal{R}$, and whose size is linear in the size of $\mathcal{G}$ and $\mathcal{R}$.*

Proof. (sketch) In order to prove this theorem, we show how to construct a CSTG $\mathcal{G}_*$ that accepts the set of terms reachable by $\mathcal{R}$ from the terms in $L(\mathcal{G})$. The production rules P_* of $\mathcal{G}_*$ consists in two sets: the rules P of $\mathcal{G}$ and P_1, that simulate rewriting by $\mathcal{R}$. The basic idea for the construction of $\mathcal{G}_*$ is the introduction of non-terminals of the form $\langle f, q\rangle$ where f is a symbol and q is a state of some SA.

First, we produce the term $\langle t\rangle$ where $t \in L(\mathcal{G})$ and $\langle t\rangle$ is the term obtained by replacing each symbol f by the non-terminal $\langle f\rangle$. Next, we simulate the rewriting of $\mathcal{R}$ by P_1. We simulate a transition $f(q_1, \ldots, q_n) \to q$ of a SA by some production rules in P_1 of the form $\langle f\rangle(\langle f_1, q_1\rangle(\overline{x_1}), \ldots, \langle f_n, q_n\rangle(\overline{x_n})) \to \langle f, q\rangle(\langle f_1, q_1\rangle(\overline{x_1}), \ldots, \langle f_n, q_n\rangle(\overline{x_n}))$. Finally, if a final state occurs at the root position of a term and a rewrite rule matches the subterm where a selection state appears, then we rewrite the term. A detailed proof of Theorem 1 can be found in Appendix B, as well as an illustration in the following example. $\square$

Example 4. Consider the CntTRS $\mathcal{R}$ in Example 1 and the CSG $\mathcal{G}$ such that $L(\mathcal{G}) = \{f(a, b)\}$. We construct the CSG $\mathcal{G}_*$ such that $L(\mathcal{G}_*) = \{f(a, b), f(c, b), g(c, b), g(c, c)\}$. Let the set of production rules P of $\mathcal{G}$ be $\{S \to \langle f\rangle(\langle a\rangle, \langle b\rangle), \langle a\rangle \to a, \langle b\rangle \to b, \langle f\rangle(x_1, x_2) \to f(x_1, x_2)\}$, and mark i to each component of SA to distinguish each SA $\mathcal{A}_i$. Let the axiom of $\mathcal{G}_*$ be S^λ. We define the set of production rules P_* of $\mathcal{G}_*$ as $P_* = P \cup P' \cup P_{\mathcal{A}} \cup P_{\mathsf{fin}} \cup P_{\mathcal{R}} \cup P_{\mathsf{re}}$ where

$$
\begin{aligned}
P' &= \{S^\lambda \to \langle f\rangle^\lambda(\langle a\rangle, \langle b\rangle), \langle f\rangle^\lambda(x_1, x_2) \to f(x_1, x_2)\}\\
P_{\mathcal{A}} &= \{\langle c_1\rangle \to \langle c_1, q^i\rangle \mid c_1 \to q^i \in \Delta_i \text{ for some } i\} \cup\\
&\quad \left\{ \begin{array}{l} \langle f_1\rangle^\lambda(\langle c_1, q_1^i\rangle, \langle c_2, q_2^i\rangle) \\ \to \langle f_1, q^i\rangle^\lambda(\langle c_1, q_1^i\rangle, \langle c_2, q_2^i\rangle) \end{array} \middle| \begin{array}{l} c_1, c_2 \in \{a, b, c\}, \\ f_1(q_1^i, q_2^i) \to q^i \in \Delta_i \text{ for some } i \end{array} \right\}\\
P_{\mathsf{fin}} &= \{\langle f_1, q^i\rangle^\lambda(x_1, x_2) \to \langle f_1, q^i\rangle^\lambda_{\mathsf{fin}}(x_1, x_2) \mid f_1 \in \{f, g\}, q^i \in F_i \text{ for some } i\} \cup\\
&\quad \left\{ \begin{array}{l} \langle f_1, q^i\rangle^\lambda_{\mathsf{fin}}(\langle c_1, q_1^i\rangle, \langle c_2, q_2^i\rangle) \\ \to \langle f_1\rangle^\lambda(\langle c_1, q_1^i\rangle_{\mathsf{fin}}, \langle c_2, q_2^i\rangle_{\mathsf{fin}}) \end{array} \middle| \begin{array}{l} f_1 \in \{f, g\}, \\ c_1 \in \{a, b, c\}, \\ f(q_1^i, q_2^i) \to q \in \Delta_i \text{ for some } i \end{array} \right\}\\
P_{\mathcal{R}} &= \{\langle a, q_1^1\rangle_{\mathsf{fin}} \to \langle c\rangle\} \cup \{\langle b, q_2^2\rangle_{\mathsf{fin}} \to \langle c\rangle\} \cup \{\langle f, q_f^3\rangle^\lambda_{\mathsf{fin}}(x_1, x_2) \to \langle g\rangle^\lambda(x_1, x_2)\}\\
P_{\mathsf{re}} &= \{\langle c_1, q^i\rangle \to \langle c_1\rangle, \langle c_1, q^i\rangle_{\mathsf{fin}} \to \langle c_1\rangle \mid c_1 \in \{a, b, c\}, q^i \in Q_i \text{ for some } i\} \cup\\
&\quad \left\{ \begin{array}{l} \langle f_1, q^i\rangle^\lambda(x_1, x_2) \to \langle f_1\rangle^\lambda, \\ \langle f_1, q^i\rangle^\lambda_{\mathsf{fin}}(x_1, x_2) \to \langle f_1\rangle^\lambda(x_1, x_2) \end{array} \middle| \begin{array}{l} f_1 \in \{f, g\}, \\ q^i \in Q_i \text{ for some } i \end{array} \right\}
\end{aligned}
$$

The term $f(c, b)$ is produced by $\mathcal{G}_*$ with the production $S^\lambda \xrightarrow[P']{} \langle f\rangle^\lambda(\langle a\rangle, \langle b\rangle) \xrightarrow[P_{\mathcal{A}}]{*} \langle f, q_f\rangle^\lambda(\langle a, q_1\rangle, \langle b, q_2\rangle) \xrightarrow[P_{\mathsf{fin}}]{*} \langle f\rangle^\lambda(\langle a, q_1\rangle_{\mathsf{fin}}, \langle b, q_2\rangle_{\mathsf{fin}}) \xrightarrow[P_{\mathcal{R}}]{} \langle f\rangle^\lambda(\langle c\rangle, \langle b, q_2\rangle_{\mathsf{fin}}) \xrightarrow[P_{\mathsf{re}}]{} \langle f\rangle^\lambda(\langle c\rangle, \langle b\rangle) \xrightarrow[P \cup P']{*} f(c, b)$.

Corollary 2. *Reachability is PSPACE-complete for monotonic CntTRS.*

From Proposition 4, it immediately holds that regular model checking is undecidable for monotonic CntTRS. Moreover, the following lower bounds already hold in the very restricted case of controlled rewrite rules over words, and where each side of every rule has depth exactly one.

Proposition 6. *Reachability is NLINSPACE-complete and regular model checking is undecidable for monotonic and flat CntTRS over unary signatures.*

Proof. We reduce the acceptance (for reachability) and emptiness (for regular model checking) problems for a linear bounded automaton (LBA) $\mathcal{M}$ [20]. The detailed reduction can be found in Appendix C, let us present here the general idea. Every configuration of $\mathcal{M}$ will be represented by a term of the form $\|{:}\, a_1 \ldots a_{j-1} a_j^p a_{j+1} \ldots a_n {:}\|$, where $\|{:}$, ${:}\|$ are the symbols for left and right end-markers, $a_1 \ldots a_n$ is the content of the tape of $\mathcal{M}$, p is its current state and a_j^p marks the current position (j) of the head.

To every transition θ of $\mathcal{M}$ stating that in state p, reading a, $\mathcal{M}$ changes state to p', write b and moves left, we associate the four following monotonic and flat controlled rules (Γ is the input alphabet of $\mathcal{M}$, the selected position in the regular expression is obvious, it is the *lhs* of the rule)

$$\begin{array}{llll}
\|{:}\, \Gamma^* c\, a^p \Gamma^* {:}\| & : a^p(x) \to \langle a^p, \theta\rangle(x), & \|{:}\, \Gamma^* \langle c, \theta\rangle\, \langle a^p, \theta\rangle \Gamma^* {:}\| & : \langle a^p, \theta\rangle(x) \to b(x), \\
\|{:}\, \Gamma^* c\, \langle a^p, \theta\rangle \Gamma^* {:}\| & : c(x) \to \langle c, \theta\rangle(x), & \|{:}\, \Gamma^* \langle c, \theta\rangle\, b \Gamma^* {:}\| & : \langle c, \theta\rangle(x) \to c^{p'}(x).
\end{array}$$

The rules for a transition moving to the right are similar. □

Some remarks about the above result. In the above construction, the selection of the rewrite position by the SA is not necessary. Only the selection of the rewritable terms by TA is needed (a weaker condition). Note also that linear and flat TRS preserve regularity, with a PTIME construction of the TA recognizing the closure (see *e.g.* [27]). Hence reachability is decidable in PTIME in the uncontrolled case.

The *conditional grammars* of [9] can be redefined in our settings as (word) grammars whose production rules are CF controlled rewrite rules (and derivations are defined using the controlled rewrite relation). It is shown in [9] that the class of languages of conditional grammars without collapsing rules coincide with CS (word) languages. Hence, it also holds that reachability is PSPACE-complete and regular model checking is undecidable for CF non-collapsing CntTRS over unary signatures.

3.2 Prefix control

Some other former results in the case of words imply that the above lower bounds still hold when control is limited to prefix SA. It is shown in [25] that every CS word language can be generated by a CS grammar with production rules of the form $AB \to AC$, $A \to BC$, $A \to a$ (where A, B, C are non-terminal and a is a terminal). It follows that every CS word language is the closure of a constant symbol under a CF non-collapsing pCntTRS (over a unary signature).

Proposition 7. *For all CS tree language L over a unary signature Σ, there exists a CF non-collapsing pCntTRS $\mathcal{R}$ over $\Sigma' \supset \Sigma$ such that $L = \mathcal{R}^*(\{c\}) \cap \mathcal{T}(\Sigma)$ for some constant $c \in \Sigma'_0 \setminus \Sigma$.*

Proof. Since L is the language over unary symbols, L can be regarded as a word language. Moreover, we can easily construct a pCntTRS that has the rule $c \to S(\bot)$ where S is the start symbol of the grammar for L and inverse of every production rule. □

Corollary 3. *Reachability is PSPACE-complete and regular model checking undecidable for CF non-collapsing pCntTRS over unary signatures.*

Another consequence of Proposition 7 is that deterministic top-down SA (which are incomparable with prefix SA in general but more general than prefix SA in unary signatures) already capture CS languages, for unary signatures.

To add a final remark, we can observe that following Example 3, there is no hope of regularity preservation even for very simple CF CntTRS containing only flat and monotonic rules, and even restricting to prefix control.

3.3 Non-monotonic rewrite rules

It is also shown in [9] that the class of languages of conditional grammars with collapsing rules coincide with recursively enumerable languages. As a consequence, reachability is undecidable for CF CntTRS (with collapsing rules) already in the case of unary signatures. Actually, the following propositions shows that flat (but not monotonic) controlled rewrite rules are sufficient for the simulation of Turing machines.

Proposition 8. *Reachability is undecidable for flat CntTRS over unary signatures.*

Proof. Flat CntTRS is a super class of monotonic and flat CntTRS. We can extend the flat CntTRS $\mathcal{R}$ that simulates the moves of LBA in the proof for Proposition 6, by adding some flat rules of the form $:\| \to \flat{:}\|$ and $\flat{:}\| \to :\|$ to $\mathcal{R}$ ($:\|$ denotes the right endmarker), in order to simulate all the moves of a TM. □

Note that when the signature is unary, all the rewrite rules are necessary linear. Again, this result is in contrast with the case of uncontrolled rewriting, because linear and flat TRS preserve regularity, and hence have a decidable reachability problem. Restricting the control to prefix permits to obtain a decidability result for non-monotonic rewrite rules, as long as they are not collapsing.

Theorem 2. *Reachability is decidable in PSPACE for flat non-collapsing pCntTRS over unary signatures.*

Proof. We show the following claim: u rewrites to v iff $u = u_0 \xrightarrow[\mathcal{R}]{} u_1 \xrightarrow[\mathcal{R}]{} \cdots \xrightarrow[\mathcal{R}]{} u_k = v$ with $\|u_0\|, \ldots, \|u_k\| \leq max(\|u\|, \|v\|)$.

The "if" direction is trivial. For the "only if" direction, assume given a reduction $u = w_1 \xrightarrow[\mathcal{R}]{*} w_n = v$, and let $max(\|u\|, \|v\|) = M$. We make an induction on the number of strings w_i longer than M. Suppose that the reduction contains one w_i such that $||w_i|| > M$. Then there exists a sub-sequence $w_k \xrightarrow[\mathcal{R}]{+} w_m$ such that

$\|w_k\| = \|w_m\| = M$, with $k < m$. It holds that $w_k = w_k[\bot]_M \xrightarrow[\mathcal{R}]{*} w_m[\bot]_M = w_m$ because we consider only prefix control. This reduces the number of string w_i longer than M. □

Non-monotonicity is also a source of undecidability of reachability for Cnt-TRS even in the case of ground controlled rewrite rules.

Proposition 9. *Reachability is undecidable for ground CntTRS.*

Proof. By representing the words $a_1, \ldots, a_n$ as right combs $f(a_1, f(\cdots f(a_n, \bot)))$, we can construct a ground CntTRS that simulates the moves of Turing Machine. Like in the proof of Propositions 8 and 6, one move of the of the TM is simulated by several rewrite steps, controlling the context left or right of the current position of the TM's head. Here, the controlled rewrite rule will have the form $\mathcal{A} : a \to a'$ were a and a' are constant symbols, and $\mathcal{A}$ controls c and d in a configuration $f(\ldots f(c, f(a, f(d(, \ldots)))))$, where a is at the rewrite position. □

This result is in contrast to uncontrolled ground TRS, for which reachability is decidable in PTIME.

4 Recursive Prefix Control

We propose a relaxed form of control, where, in order to select the positions of application of a controlled rewrite rule, the term to be rewritten is tested for membership in the closure of a regular language L, instead of membership to L directly. The idea is somehow similar to conditional rewriting (see *e.g.* [2]) where the conditions are equations that have to be solved by the rewrite system.

The definition is restricted to control with prefix SA, and a recursive pCnt-TRS $\mathcal{R}$ is defined as a pCntTRS. In order to define formally the rewrite relation, let us recall first that in the computations of a prefix SA $\mathcal{A}$, the states below a selection state q_s are universal (q_0), *i.e.* that we can have any subterm at a selected position (only the part of the term *above* the selected position matters). Following this observation, we say that the variable position p in a context $C[x_1]$ over Σ is selected by the prefix SA $\mathcal{A}$ if p is selected in $C[c]$ where c is an arbitrary symbol of Σ_0. A term s rewrites to t in one step by a recursive pCntTRS $\mathcal{R}$, denoted by $s \xrightarrow[\mathcal{R}]{} t$, if there exists a controlled rewrite rule $\mathcal{A} : \ell \to r \in \mathcal{R}$, where $\mathcal{A}$ is a prefix SA, a substitution σ, a position $p \in \mathcal{P}os(s)$, and a context $C[x_1]$ such that $C[x_1] \xrightarrow[\mathcal{R}]{*} s[x_1]_p$ and the position of x_1 is selected in $C[x_1]$ by $\mathcal{A}$, such that $s|_p = \ell\sigma$ and $t = s[r\sigma]_p$. This definition is well-founded because of the restriction to prefix control (remember that the root position is always selected by prefix SA).

Example 5. Let $\Sigma = \{a, b, c, d, \bot\}$ be a unary signature, and let $\mathcal{R}$ be the flat recursive pCntTRS containing the rules $\mathcal{C}_1 : a(a(x)) \to b(x)$, and $\mathcal{C}_2 : c(x) \to d(x)$, where the SA $\mathcal{C}_1$ selects the position after a prefix aa, and $\mathcal{C}_2$ selects the position after a prefix $aaaa$. Then we have with $\mathcal{R}$ (we omit the parentheses and the tail $\bot$, and underline the part of the term which is rewritten)

$$aa\underline{aa}c \xrightarrow[\mathcal{R}]{} aab\underline{c} \xrightarrow[\mathcal{R}]{} aabd$$

Note that for the last step, we have use the fact that $aaaa \xrightarrow[\mathcal{R}]{} aab$, *i.e.* there exists $C[x_1] = aaaa(x_1)$ with $C[x_1] \xrightarrow[\mathcal{R}]{} aab(x_1)$ and the position of x_1 in $C[x_1]$ is selected by $\mathcal{C}_2$. The last rewrite step would not be possible if $\mathcal{R}$ would not be recursive, because aab is not a prefix admitted by $\mathcal{C}_2$.

Theorem 3. *Regular model-checking is decidable in EXPTIME for linear and right-shallow recursive pCntTRS.*

Proof. (sketch) We show that, given a right-shallow and linear recursive pCnt-TRS $\mathcal{R}$ and given the language $L \subseteq \mathcal{T}(\Sigma)$ of a TA $\mathcal{A}_L$ we can construct an alternating tree automaton with epsilon-transitions (ε-ATA) $\mathcal{A}'$ recognizing the rewrite closure $\mathcal{R}^*(L)$. Intuitively, an alternating tree automata $\mathcal{A}$ is a top-down tree automaton that can spawn in several copies during computation on a term t. Formally, an ε-ATA over a signature Σ is a tuple $\mathcal{A} = \langle Q, q_0, \delta \rangle$ where Q is a finite set of states, $q_0 \in Q$ is the initial state and δ is a function which associates to every state $q \in Q$ a disjunction of conjunctions of propositional variables of the following form $a \in \Sigma$, or $\langle q', \varepsilon \rangle$, for $q' \in Q \setminus \{q\}$, or $\langle q', i \rangle$, for $q' \in Q$ and $1 \leq i \leq m$ where m is the maximal arity of a symbol in Σ.
A *run* of $\mathcal{A}$ on $t \in \mathcal{T}(\Sigma)$ is a function ρ from $\mathcal{P}os(t)$ into 2^Q such that for all position $p \in \mathcal{P}os(t)$, with $t(p) = a \in \Sigma_n$ ($n \geq 0$), and for all state $q \in \rho(p)$, it holds that $a, \langle \rho(p.1), 1 \rangle, \ldots, \langle \rho(p.n), n \rangle, \langle \rho(p), \varepsilon \rangle \models \delta(q)$ where $\langle S, p \rangle$ is a notation for all the variables $\langle q, p \rangle$ with $q \in S$, and $\models$ denotes propositional satisfaction, while assigning true to the propositional variables on the left of $\models$.
The language $L(\mathcal{A})$ of $\mathcal{A}$ is the set of terms $t \in \mathcal{T}(\Sigma)$ on which there exists a run ρ of $\mathcal{A}$ such that $q_0 \in \rho(\varepsilon)$ (terms *recognized* by $\mathcal{A}$).

Roughly, the principle of the construction of $\mathcal{A}'$ is to start with $\mathcal{A}_L$ and the SA of $\mathcal{R}$, casted into ATA and merged, and to complete the transition functions in order to reflect the effect of the possible rewrite steps. The detailed construction can be found in Appendix D. Let us present below the construction on an example, based on the recursive pCntTRS $\mathcal{R}$ of Example 5. The transitions of the prefix SA $\mathcal{C}_1$ and $\mathcal{C}_2$, for control in R are explicitly the following

$$\mathcal{C}_1 : \bot \to q^1_s \mid q^1_0, f(q^1_0) \to q^1_s \mid q^1_0, a(q^1_s) \to q^1_1, a(q^1_1) \to q^1_2 \quad (f \in \{a, b, c, d\})$$
$$\mathcal{C}_2 : \bot \to q^2_s \mid q^1_0, f(q^2_0) \to q^2_s \mid q^2_0, a(q^2_s) \to q^2_1, a(q^1_1) \to q^2_2, a(q^2_2) \to q^2_3, a(q^2_3) \to q^2_4$$

They are casted into the following transition functions

$$\delta^1 : q^1_2 \mapsto a \wedge \langle q^1_1, 1 \rangle, q^1_1 \mapsto a \wedge \langle q^1_s, 1 \rangle,$$
$$\delta^2 : q^2_4 \mapsto a \wedge \langle q^2_3, 1 \rangle, q^2_3 \mapsto a \wedge \langle q^2_2, 1 \rangle, q^2_2 \mapsto a \wedge \langle q^2_1, 1 \rangle, q^2_1 \mapsto a \wedge \langle q^2_s, 1 \rangle$$

with moreover $\delta^i(q^i_s) = \bot \vee \bigvee_{f=a,b,c,d} (f \wedge \langle q^i_0, 1 \rangle)$ and $\delta^i(q^i_0) \mapsto \bot \vee \bigvee_{f=a,b,c,d} (f \wedge \langle q^i_0, 1 \rangle)$ for $i = 1, 2$. The initial language $L = \{aaaac\}$ is recognized by a ε-ATA

$\mathcal{A}'_L$ with the transition function $\delta_L : q_5 \mapsto a \wedge \langle q_4, 1\rangle, q_4 \mapsto a \wedge \langle q_3, 1\rangle, q_3 \mapsto a \wedge \langle q_2, 1\rangle, q_2 \mapsto a \wedge \langle q_1, 1\rangle, q_1 \mapsto c \wedge \langle q_0, 1\rangle, q_0 \mapsto \perp$.

Let δ_0 be the union of δ^1, δ^2 and δ_L. It holds that, starting from q_3, δ_0 permits to "reach" $a(a(q_1))$, denoted $a(a(q_1)) \models_0 q_3$. Formally, the relation $t \models_i q$ holds in the following cases, for $t = a(t_1, \ldots, t_n) \in \mathcal{T}(\Sigma \cup Q)$, with $n \geq 0$ and the symbols of Q are assumed of arity 0

$\delta_i(q) = \phi_1 \vee \phi_2$ and $t \models_i \phi_1$ or $t \models_i \phi_2$
$\delta_i(q) = \phi_1 \wedge \phi_2$ and $t \models_i \phi_1$ and $t \models_i \phi_2$
$\delta_i(q) = \langle q', \varepsilon\rangle$ and $t \models_i q'$
$\delta_i(q) = \langle q', j\rangle$, with $1 \leq j \leq n$ and $t_j \models_i q'$
$\delta_i(q) = a$.

In order to simulate the first rule of $\mathcal{R}$, we let $\delta_1(q_3) = \delta_0(q_3) \vee \big(b \wedge \langle q_1, 1\rangle \wedge \langle q_s^1, \varepsilon\rangle\big)$. The term $a(a(q_1))$ corresponds to the *lhs* of this rule, $b(q_1)$ is matching the *rhs* and $\langle q_s^1, \varepsilon\rangle$ ensures that the control is satisfied (following the recursive definition). Similarly, $a(a(q_s^2)) \models_1 q_2^2$, and we can let $\delta_2(q_2^2) = \delta_0(q_2^2) \vee \big(b \wedge \langle q_s^2, 1\rangle \wedge \langle q_s^1, \varepsilon\rangle\big)$. Moreover, $c(q_0) \models_2 q_1$, and with the second rule of $\mathcal{R}$, we can let

$$\delta_3(q_1) = \delta_0(q_1) \vee \big(d \wedge \langle q_0, 1\rangle \wedge \langle q_s^2, \varepsilon\rangle\big).$$

With these transitions, we have the following runs for 2 descendants of L

$$\begin{array}{ccccc} a & a & b & c & \perp \\ \left\{\begin{array}{c} q_5 \\ q_2^1 \end{array}\right\} & \left\{\begin{array}{c} q_4 \\ q_1^1 \end{array}\right\} & \left\{\begin{array}{c} q_3 \\ q_s^1 \end{array}\right\} & \left\{\begin{array}{c} q_1 \\ q_0^1 \end{array}\right\} & \left\{\begin{array}{c} q_0 \\ q_0^1 \end{array}\right\} \end{array} \qquad \begin{array}{ccccc} a & a & b & d & \perp \\ \left\{\begin{array}{c} q_5 \\ q_4^2 \\ q_2^1 \end{array}\right\} & \left\{\begin{array}{c} q_4 \\ q_3^2 \\ q_1^1 \end{array}\right\} & \left\{\begin{array}{c} q_3 \\ q_2^2 \\ q_s^1 \end{array}\right\} & \left\{\begin{array}{c} q_1 \\ q_s^2 \\ q_0^1 \end{array}\right\} & \left\{\begin{array}{c} q_0 \\ q_0^2 \\ q_0^1 \end{array}\right\} \end{array}$$

The ε-ATA $\mathcal{A}'$ can be casted into a TA $\mathcal{A}$ whose size is exponential in the size of $\mathcal{A}'$ and such that $L(\mathcal{A}) = L(\mathcal{A}') = \mathcal{R}^*(L)$, and hence we can decide the problem of regular model checking for $\mathcal{R}$ in exponential time. □

Conclusion

We have proposed a definition of controlled term rewrite systems based on selection automata for the specification of authorized rewrite position, and a restriction where the selection of a position depends only on the labels on its prefix path. We have shown that reachability is PSPACE-complete for controlled monotonic (non-size-reducing) rewrite rules, using context-sensitive tree languages, and for prefix-controlled context-free non-collapsing rewrite rules. When allowing size decreasing rules, reachability becomes undecidable, even for flat and linear or ground rules. Finally, we have presented a relaxed form of prefix control called recursive prefix control which permits to obtain preservation of regular tree languages, hence decidability of reachability and regular model checking (in EXPTIME). The proof involves the construction of alternating tree automata with ε-transitions, and we believe that this technique could be useful for computing the rewrite closure of other classes of automata with local constraints.

The proof of undecidability for ground CntTRS (Proposition 9) does not work when restricting to prefix control. It could be interesting to know whether reachability is decidable for ground pCntTRS. Also, we are interested in knowing how the decidability result of Theorem 2 can be generalized to non-unary signatures.

In [9], the conditional grammars (*i.e.* controlled (in the above sense) context-free word grammars) are related to grammars with a restriction on the possible production sequences (the list of names of production rules used must belong to a regular language). It could be interesting to establish a similar comparison for term rewriting. In particular, results on the restriction defined by authorized sequences of rewrite rules could be useful for the analysis of languages for the extensional specification of the set of possible rewrite derivations like in [3].

Rewriting of unranked ordered labeled tree has been much less studied that its counterpart for ranked terms. We would like to study controlled rewriting in this case, in particular in the context of update rules for XML [6,17].

Acknowledgements The authors wish to thanks Olivier Ly for his suggestion for the proof of Proposition 9, and Sylvain Schmitz, Géraud Senizergues and Hubert Comon-Lundh for their useful remarks and recommendations.

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. F. Baader and T. Nipkow. *Term Rewriting and All That.* Cambridge University Press, New York, 1998.
3. P. Borovansky, H. Kirchner, C. Kirchner, and C. Ringeissen. Rewriting with strategies in elan: a functional semantics. *International Journal of Foundations of Computer Science (IJFCS)*, 12(1):69–95, 2001.
4. 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.
5. J. Chabin and P. Réty. Visibly pushdown languages and term rewriting. In *Proceedings 6th International Symposium on Frontiers of Combining Systems (FroCos 2007)*, volume 4720 of *Lecture Notes in Computer Science*, pages 252–266. Springer, 2007.
6. D. Chamberlin and J. Robie. Xquery update facility 1.0. W3C Candidate Recommendation. `http://www.w3.org/TR/xquery-update-10/`, 2009.
7. H. Comon. Sequentiality, second order monadic logic and tree automata. In *Proceedings, Tenth Annual IEEE Symposium on Logic in Computer Science (LICS)*, pages 508–517. IEEE Computer Society, 1995.
8. 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.

9. J. Dassow, G. Paun, and A. Salomaa. *Handbook of Formal Languages*, volume 2, chapter Grammars with Controlled Derivations, pages 101–154. Springer, 1997.
10. I. Durand and A. Middeldorp. Decidable call-by-need computations in term rewriting. *Information and Computation*, 196(2):95 – 126, 2005.
11. J. Engelfriet. Top-down tree transducers with regular look-ahead. *Mathematical Systems Theory*, 10:289–303, 1977.
12. G. Feuillade, T. Genet, and V. V. T. Tong. Reachability analysis over term rewriting systems. *J. Autom. Reasoning*, 33(3-4):341–383, 2004.
13. M. Frick, M. Grohe, and C. Koch. Query evaluation on compressed trees (extended abstract). In *Proceedings of the 18th Annual IEEE Symposium on Logic in Computer Science*, pages 188–, Washington, DC, USA, 2003. IEEE Computer Society.
14. G. Gottlob and C. Koch. Monadic datalog and the expressive power of languages for web information extraction. *Journal of the ACM*, 51(1):74–113, 2004.
15. I. Guessarian. Pushdown tree automata. *Mathematical Systems Theory*, 16(1):237–263, 1983.
16. D. Hofbauer and J. Waldmann. Deleting string rewriting systems preserve regularity. *Theor. Comput. Sci.*, 327(3):301–317, 2004.
17. 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.
18. N. D. Jones and N. Andersen. Flow analysis of lazy higher-order functional programs. *Theoretical Computer Science*, 375(1-3):120 – 136, 2007.
19. 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.
20. S. Y. Kuroda. Classes of languages and linear-bounded automata. *Information and Control*, 7:207–223, 1964.
21. S. Lucas. Context-sensitive rewriting strategies. *Inf. Comput.*, 178:294–343, October 2002.
22. T. Milo, D. Suciu, and V. Vianu. Typechecking for xml transformers. *J. of Comp. Syst. Sci.*, 66(1):66–97, 2003.
23. T. Nagaya and Y. Toyama. Decidability for left-linear growing term rewriting systems. *Inf. Comput.*, 178(2):499–514, 2002.
24. F. Neven and T. Schwentick. Query automata over finite trees. *Theoretical Computer Science*, 275(1-2):633–674, 2002.
25. M. Penttonen. One-sided and two-sided context in formal grammars. *Information and Control*, 25:371–392, 1974.
26. P. Réty and J. Vuotto. Tree automata for rewrite strategies. *J. Symb. Comput.*, 40:749–794, July 2005.
27. K. Salomaa. Deterministic tree pushdown automata and monadic tree rewriting systems. *J. Comput. Syst. Sci.*, 37(3):367–394, 1988.
28. G. Sénizergues. Formal languages and word-rewriting. In H. Comon and J.-P. Jouannaud, editors, *Term Rewriting, Advanced Course of French Spring School of Theoretical Computer Science, Font Romeux, France*, volume 909 of *Lecture Notes in Computer Science*, pages 75–94. Springer, 1993.

A Proof of Proposition 3

We give here a non-deterministic linear space decision algorithm for the membership problem for CSTG, using a simulation of CSTG by linear bounded automata.

Let α be the transformation of terms of $\mathcal{T}(\Sigma)$ into strings defined as follows: $\alpha(x) =$ "(x)" for $x \in X$ and $\alpha(f(t_1, \ldots, t_n)) =$ "$(f($"$\alpha(t_1) \cdots \alpha(t_n)$"$))$" for $n \geq 0$.

Lemma 1. $\alpha(\mathcal{C}[t_1, \ldots, t_n]) = \alpha(\mathcal{C})[\alpha(t_1), \ldots, \alpha(t_n)]$ *(where* $\alpha(\mathcal{C}) = \alpha(\mathcal{C}[x_1, \ldots, x_n])$*).*

Proof. We show this lemma by induction on the size of $\mathcal{C}$.
If $||\mathcal{C}|| = 0$, then this lemma holds trivially.
Otherwise, let $p_1, \ldots, p_n$ be the positions where t_i's occur in $\mathcal{C}[t_1, \ldots, t_n]$. We take $\mathcal{C}[t_1, \ldots, t_n]$ as $f(t'_1, \ldots, t'_m)$ and then we have $\alpha(C[t_1, \ldots, t_n]) = \alpha(f(t'_1, \ldots, t'_m)) =$ "$(f($"$\alpha(t'_1) \cdots \alpha(t'_m)$"$))$". For j such that there exists p_i where $jp' = p_i$ for some p', t'_j is represented as $\mathcal{C}_j[t_i]$ by some context $\mathcal{C}_j$ and hence $\alpha(t'_j) = \alpha(\mathcal{C}_j[t_i]) = \alpha(\mathcal{C}_j)[\alpha(t_i)]$ from the induction hypothesis. By applying the above claim for all p_i's, we have $\alpha(\mathcal{C}[t_1, \ldots, t_n]) = \alpha(\mathcal{C})[\alpha(t_1), \ldots, \alpha(t_n)]$. □

Lemma 2. $||\alpha(t)|| = 5||t||_\Sigma + 3||t||_\mathcal{X}$ *where* $|\alpha(t)|$ *is the length of* $\alpha(t)$*,* $||t||_\Sigma$ *is the number of the occurrence of signature in* Σ*, and* $||t||_\mathcal{X}$ *is the number of the occurence of variables in* $\mathcal{X}$*.*

Proof. We show this lemma by induction on the structure of term.
If $t = x$, then $||\alpha(t)|| = ||$"(x)"$|| = 3 = 5||t||_\Sigma + 3||t||_\mathcal{X}$.
If $t = f(t_1, \ldots, t_n)$, then $||\alpha(t)|| = ||$"$(f($"$\alpha(t_1) \cdots \alpha(t_n)$"$))$"$|| = 5 + ||\alpha(t_1) \cdots \alpha(t_n)||$.
From the induction hypothesis, we have $||\alpha(t_1) \cdots \alpha(t_n)|| = 5\Sigma_{i=1}^n ||t_i||_\Sigma + 3\Sigma_{i=1}^n ||t_i||_\mathcal{X}$.

Thus, we have $||\alpha(t)|| = 5 + 5\Sigma_{i=1}^n ||t_i||_\Sigma + 3\Sigma_{i=1}^n ||t_i||_\mathcal{X} = 5||t||_\Sigma + 3||t||_\mathcal{X}$. □

From Lemma 1, a term $\mathcal{C}[\ell\sigma]$ is transformed to $\alpha(\mathcal{C})[\alpha(\ell)\alpha(\sigma)]$ where $\alpha(\sigma) = \alpha(x\sigma)$ for all domain x of σ. Hence, we can simulate the term rewriting $\mathcal{C}[\ell\sigma] \to \mathcal{C}[r\sigma]$ by the string rewriting $\alpha(\mathcal{C})[\alpha(\ell)\alpha(\sigma)] \to \alpha(\mathcal{C})[\alpha(r)\alpha(\sigma)]$.

Since each production rules of CSTG is monotonic, we can simulate inverse of production rules by LBA.

In the following, we show the simulation of one-step of inverse production of CSTG by LBA. For the production rule $\ell \to r$, let $\alpha(\ell\sigma) = \ell_1(x_1\sigma')\ell_2(x_2\sigma') \cdots (x_m\sigma')\ell_n$, $\alpha(r\sigma') = r_1(x_{m_1}\sigma')r_2(x_{m_2}\sigma') \cdots (x_{m_m}\sigma')r_{n'}$ where $x_{m_j} \in \{x_1, \ldots x_n\}$, and $(x\sigma') = \alpha(x\sigma)$. We simulate $\alpha(\ell\sigma) \to \alpha(r\sigma)$.

In the following, we use different notation for LBA from the one in the proof for Proposition 6 for readability. We denote the configuration such that the head of LBA reads u in $u_1 \cdots u \cdots u_n$ and whose state is q as $\|{:}u_1 \cdots uq \cdots u_n{:}\|$, while we denote $\|{:}u_1 \cdots u^q \cdots u_n{:}\|$ in the proof for Proposition 6.

Simulation of one-step of inverse production. LBA $\mathcal{M}^{\ell}$ can simulate the string rewriting $\alpha(\ell) = \ell_1(x_1\sigma)\ell_2(x_2\sigma)\cdots(x_m\sigma)\ell_n\sigma \to \alpha(r) = r_1(x_{m_1}\sigma)r_2(x_{m_2}\sigma)\cdots(x_{m_m}\sigma)r_{n'}\sigma$ where $\ell \to r \in \mathcal{R}$ for inverse monotonic TRS $\mathcal{R}$, *i.e.* $||\ell|| \geq ||r||$, $l_i, r_j \in (\Sigma \cup \{(,)\})^*$ for $1 \leq i \leq n$ and $1 \leq j \leq n'$, σ is substitution in $\mathcal{X} \to (\Sigma \cup \{(,)\})^*$, $x_i \in X$ for $1 \leq i \leq m$, and $x_{m_j} \in \{x_1, \ldots x_m\}$ and no x_i and x_j for $i \neq j$ are same.

$\mathcal{M}^{\ell}$ is constructed by connecting the following automata.

1. LBA $\mathcal{M}^{\ell}_{\mathtt{in}}$ such that
   Initial state $q^{\ell}_{\mathtt{in}}$
   Final state $q^{\ell_1}_{\mathtt{in}}$
   Behavier Head of the automaton moves right and can change state to $q^{\ell_1}_{\mathtt{in}}$ in any time.
2. LBA $\mathcal{M}^{l_i}$ for $1 \leq i \leq n$ such that
   Initial state $q^{\ell_i}_{\mathtt{in}}$
   Final state $q^{\ell_i}_{f}$
   Behavier $u_1 q^{\ell_i}_{\mathtt{in}} \cdots u_k v \xrightarrow[\mathcal{M}^{\ell_i}]{*} \flat^* v^{q^{\ell_i}_f}$ if $u_1 \cdots u_n = \ell_i$. Otherwise, the automaton terminate and input does not accepted. The final state appears at v *i.e.* the next symbol of rewritten ℓ_i.
3. LBA $\mathcal{M}^{i()}$ for $1 \leq i \leq n-1$ and $n > 1$ such that
   Initial state $q^{\ell_i}_{f}$
   Final state $q^{\ell_{i+1}}_{\mathtt{in}}$
   Behavier If first symbol is "(", then move head to next symbol of corresponding ")" and change state to $q^{\ell_{i+1}}_{\mathtt{in}}$.
4. LBA $\mathcal{M}^{\leftarrow}$ such that
   Initial state $q^{\ell_n}_{f}$
   Final state $q^{m\sigma}_{\mathtt{in}}$
   Behavier Move left until reading ")".
5. LBA $\mathcal{M}^{i\sigma}$ for $1 \leq i \leq m$such that
   Initial state $q^{i\sigma}_{\mathtt{in}}$
   Final state $q^{i\sigma}_{f}$
   Behavier Move "$(x_i\sigma)$" to right until the next symbol become other than $\flat$. Final state appears at the first position of "(" in "$(x_i\sigma)$".
6. LBA $\mathcal{M}^{i\sigma\leftarrow}$ for $2 \leq i \leq m$ such that
   Initial state $q^{i\sigma}_{f}$
   Final state $q^{(i-1)\sigma}_{\mathtt{in}}$
   Behavier Move left until reading ")".
7. LBA $\mathcal{M}^{sh}$ such that
   Initial state $q^{1\sigma}_{f}$
   Final state q^{sh}_{f}
   Behavier Shuffle "$(x_1\sigma)\cdots(x_m\sigma)$" to "$(x_{m_1}\sigma)\cdots(x_{m_m}\sigma)$". Final state appears at first "(" in "$(x_{m_1}\sigma)$".
8. LBA $\mathcal{M}'^{\leftarrow}$ such that
   Initial state q^{sh}_{f}
   Final state $q^{r_1}_{\mathtt{in}}$

Behavior Move left until reading the symbol other than $\flat$.

9. LBA $\mathcal{M}^{r_i}$ for $1 \le i \le n'$ such that
   Initial state $q_{\mathtt{in}}^{r_i}$
   Final state $q_f^{r_i}$
   Behavior Rewrite $\flat^*$ to r_i. The final state appears at the next symbol of r_i.
10. LBA $\mathcal{M}'^{i()}$ for $1 \le i \le n'$ such that
    Initial state $q_f^{r_i}$
    Final state $q_{\mathtt{in}}^{r_{i+1}}$
    Behavior Move "(u)" that is most near and at right side of head to left until no symbol $\flat$ is there at left of "(u)". Final state appears at next symbol of ")" of "(u)".
11. LBA $\mathcal{M}^{\flat}$ such that
    Initial state $q_f^{r_{n'}}$
    Final state q_f^{ℓ}
    Behavior Move all of $\flat$ to right-end and finally head move to left-end.
    Move $\flat$ to right-end.

In the following, we prove that $\mathcal{C}[\ell\sigma] \xrightarrow[R]{} \mathcal{C}[r\sigma]$ iff $\alpha(\mathcal{C})[\alpha(\ell\sigma)] \xrightarrow[\mathcal{A}]{*} \alpha(\mathcal{C})[\alpha(r\sigma)]$ for inverse monotonic TRSs.

Lemma 3. *If $\mathcal{C}[\ell\sigma] \xrightarrow[R]{} \mathcal{C}[r\sigma]$ for inverse monotonic TRS $\mathcal{R}$, then $\| : q_{\mathtt{in}}^{l}\alpha(C)[\alpha(l\sigma)]{:}\| \xrightarrow[\mathcal{A}]{*} \flat_l q_f^{l}\alpha(C)[\alpha(r\sigma)]\flat^* : \|$.*

Proof. From Lemma 1, we can take $\alpha(\mathcal{C}) = u_1 \cdots u_k \Box v_1 \cdots , v_h$, $\alpha(\ell\sigma) = \ell_1(x_1\sigma)\cdots(x_m\sigma)\ell_n$, and $\alpha(r\sigma) = r_1(x_{m_1}\sigma)\cdots(x_{m_m}\sigma)r_n$. Thus, this lemma holds from the following transition sequence.

$$\begin{aligned}
& \|{:}q_{\mathtt{in}}^{\ell}u\ell_1(x_1\sigma)\cdots(x_m\sigma)\ell_n v{:}\| && (1)\\
\xrightarrow[\mathcal{M}^{\ell}]{*}\ & \|{:}uq_{\mathtt{in}}^{\ell_1}\ell_1(x_1\sigma)\cdots(x_m\sigma)\ell_n v{:}\| && (2)\\
\xrightarrow[\mathcal{M}^{\ell_1}]{*}\ & \|{:}u\flat^* q_{\mathtt{in}}^{\ell_1}(x_1\sigma)\cdots(x_m\sigma)\ell_n v{:}\| && (3)\\
\xrightarrow[\mathcal{A}^{1()}]{*}\ & \|{:}u\flat^*(x_1\sigma)q_{\mathtt{in}}^{\ell_2}\cdots(x_m\sigma)\ell_n v{:}\| && (4)\\
\xrightarrow[\mathcal{A}^{\ell_i},\mathcal{A}^{i}()]{*}\ & \|{:}u\flat^*(x_1\sigma)\cdots(x_m\sigma)\flat^* q_f^{\ell_n} v{:}\| && (5)\\
\xrightarrow[\mathcal{A}^{\leftarrow}]{*}\ & \|{:}u\flat^*(x_1\sigma)\cdots(x_m\sigma q_{\mathtt{in}}^{m\sigma})\flat^* v{:}\| && (6)\\
\xrightarrow[\mathcal{A}^{m\sigma}]{*}\ & \|{:}u\flat^*(x_{m_1}\sigma)\sigma)\cdots\flat^* q_f^{m\sigma}(x_m\sigma)v{:}\| && (7)\\
\xrightarrow[\mathcal{A}^{i\sigma},\mathcal{A}^{i\sigma\leftarrow}]{*}\ & \|{:}u\flat^* q_f^{1\sigma}(x_1\sigma)\cdots(x_m\sigma)v{:}\| && (8)\\
\xrightarrow[\mathcal{A}^{sh}]{*}\ & \|{:}u\flat^* q_f^{sh}(x_{m_1}\sigma)\cdots(x_{m_m}\sigma)v{:}\| && (9)\\
\xrightarrow[\mathcal{A}'^{\leftarrow}]{*}\ & \|{:}uq_{\mathtt{in}}^{r_1}\flat^*(x_{m_1}\sigma)\cdots(x_{m_m}\sigma)v{:}\| && (10)\\
\xrightarrow[\mathcal{A}^{r_1}]{*}\ & \|{:}ur_1q_f^{r_1}\flat^*(x_{m_1}\sigma)\cdots(x_{m_m}\sigma)v{:}\| && (11)\\
\xrightarrow[\mathcal{A}'^{i()}]{*}\ & \|{:}ur_1(x_{m_1}\sigma)q_{\mathtt{in}}^{r_2}\flat^*\cdots(x_{m_m}\sigma)v{:}\| && (12)\\
\xrightarrow[\mathcal{A}^{r_i},\mathcal{A}'^{i()}]{*}\ & \|{:}ur_1(x_{m_1}\sigma)\cdots(x_{m_m}\sigma)r_{n'}q_f^{r_{n'}}\flat^* v{:}\| && (13)\\
\xrightarrow[\mathcal{A}^{\flat}]{*}\ & \|{:}q_f^{\ell}ur_1(x_{m_1}\sigma)\cdots(x_{m_m}\sigma)r_{n'}v\flat^*{:}\| && (14)
\end{aligned}$$

Since $||\ell|| \geq ||r||$, we can generate the all r_i's because $\Sigma_{i=1}^{n}||\ell_i|| < \Sigma_{i=1}^{n'}||r_i||$ and there are sufficient $\flat$'s to be rewritten to each r_i. $\square$

Lemma 4. *If* $\|{:}q^{l}_{\mathtt{in}}\alpha(\mathcal{C})[\alpha(\ell'\sigma)]{:}\| \xrightarrow[\mathcal{A}^{l}]{*} \|{:}q^{l}_{f}\alpha(\mathcal{C})[\alpha(r'\sigma)]\flat^{*}{:}\|$, *then* $\mathcal{C}[\ell'\sigma] \xrightarrow[\mathcal{R}]{} \mathcal{C}[r'\sigma]$ *for monotonic TRS* $\mathcal{R}$.

Proof. From Lemma 1, we can represent $\alpha(\mathcal{C}[\ell'\sigma])$ and $\alpha(\mathcal{C}[r'\sigma])$ as $u\ell'_1(x_1\sigma)\cdots(x_m\sigma)\ell'_n v$ and $ur'_1(x_{m_1}\sigma)\cdots(x_{m_m}\sigma)r'_{n'}v$ where $\alpha(\mathcal{C}) = u\Box v$, $\alpha(\ell'\sigma) = \ell'_1(x_1\sigma)\cdots(x_m\sigma)\ell'_n$, and $\alpha(r'\sigma) = r'_1(x_{m_1}\sigma)\cdots(x_{m_m}\sigma)r'_{n'}$. From the construction of $\mathcal{A}^{\ell}$, the transition $\|{:}q^{\ell}_{\mathtt{in}}u\ell'_1(x_1\sigma)\cdots(x_m\sigma)\ell'_n v{:}\| \xrightarrow[\mathcal{A}^{\ell}]{*} \|{:}q^{\ell}_{f}r'_1(x_{m_1}\sigma)\cdots(x_{m_m}\sigma)r'_{n'}\flat^{*}{:}\|$ is as of the form the transition in Lemma 3. Transition $\|{:}uq^{\ell_1}_{\mathtt{in}}\ell_1(x_1\sigma)\cdots(x_m\sigma)\ell_n v{:}\| \xrightarrow[\mathcal{A}^{\ell}]{*} \|{:}u\flat^{*}(x_1\sigma)\cdots(x_m\sigma)\flat^{*}q^{\ell_n}_{f}v{:}\|$ and $\|{:}uq^{r_1}_{\mathtt{in}}\flat^{*}(x_{m_1}\sigma)\cdots(x_{m_m}\sigma)r_{n'}v{:}\| \xrightarrow[\mathcal{A}^{\ell}]{*} \|{:}ur_1(x_{m_1}\sigma)\cdots(x_{m_m}\sigma)r_{n'}q^{r_{n'}}_{f}v{:}\|$ implies that $\ell_1(x_1\sigma)\cdots(x_m\sigma)\ell_n = \alpha(l\sigma)$ and $r_1(x_{m_1}\sigma)\cdots(x_{m_m}\sigma)r_{n'} = \alpha(r\sigma)$ for some $l \to r \in \mathcal{R}$. Thus, we have $\mathcal{C}[\ell'\sigma] \xrightarrow[\mathcal{R}]{} \mathcal{C}[r'\sigma]$. $\square$

We construct the LBA $\mathcal{A} = \langle Q, \Gamma, q_{\mathtt{in}}, \{q_f\}, \Theta\rangle$ that simulates context-sensitive grammar $G = \langle N, T, P, S\rangle$ from $\mathcal{A}^{\ell}$'s for $\ell \to r \in P^{-1}$. Q is disjoint union of all Q^{ℓ}'s. Γ are same for all $\mathcal{A}^{\ell}$'s. Θ is consists of union of all Θ^{ℓ}'s and the rules $\langle q_{\mathtt{in}}, x, q_{\mathtt{in}}, x, \text{left}\rangle$ for $x \neq \|{:}$, $\langle q_{\mathtt{in}}, \|{:}, q^{\ell}_{\mathtt{in}}, \|{:}, \text{right}\rangle$, $\langle q^{\ell}_{f}, \|{:}, q_{\mathtt{in}}, \|{:}, \text{stay}\rangle$, $\langle q_{\mathtt{in}}, \|{:}, q_{S_1}, \|{:}, \text{right}\rangle$, $\langle q_{S_1}, \text{“(”}, q_{S_2}, \text{“(”}, \text{right}\rangle$, $\langle q_{S_2}, S, q_{S_3}, S, \text{right}\rangle$, $\langle q_{S_3}, \text{“)”}, q_{\flat}, \text{“)”}, \text{right}\rangle$, $\langle q_{\flat}, \flat, q_{\flat}, \flat, \text{right}\rangle$, and $\langle q_{\flat}, {:}\|, q_f, {:}\|, \text{stay}\rangle$.

The following lemmas state completeness and soundness of LBA $\mathcal{A}$.

Lemma 5. *If* $t \in \mathcal{L}(G)$, *then* $\alpha(t) \in \mathcal{A}$.

Proof. We prove that $\|{:}q_{\mathtt{in}}\alpha(t){:}\| \xrightarrow[\mathcal{A}]{*} {:}\|(S)\flat^{*}q_f{:}\|$ for the start symbol S of G. If $t = S$, then this lemma holds trivially. Otherwise, let $t \xrightarrow[P]{} t' \xrightarrow[P]{*} S$. We have $\|{:}q_{\mathtt{in}}\alpha(t){:}\| \xrightarrow[\mathcal{A}]{*} q^{\ell}_{\mathtt{in}}\|{:}\alpha(t){:}\| \xrightarrow[\mathcal{A}]{*} q^{\ell}_{f}\|{:}\alpha(t'){:}\| \xrightarrow[\mathcal{A}]{} q_{\mathtt{in}}\|{:}\alpha(t'){:}\|$ from Lemma 3. Thus, we have $q_{\mathtt{in}}\|{:}\alpha(t){:}\| \xrightarrow[\mathcal{A}]{*} q_f\|{:}(S)\flat^{*}{:}\|$ by applying lemma 3 as the above transition. $\square$

Lemma 6. *If* $\alpha(t) \in \mathcal{A}$, *then* $t \in \mathcal{L}(G)$.

Proof. $\alpha(t) \in \mathcal{A}$ implies $\|{:}q_{\mathtt{in}}\alpha(t){:}\| \xrightarrow[\mathcal{A}]{*} \|{:}(S)\flat^{*}q_f{:}\|$.

If $t = S$, then this lemma holds trivially.

Otherwise, we have the transition $\|{:}q_{\mathtt{in}}\alpha(t){:}\| \xrightarrow[\mathcal{A}]{*} q^{\ell}_{\mathtt{in}}\|{:}\alpha(t){:}\| \xrightarrow[\mathcal{A}]{} q^{\ell}_{f}\|{:}\alpha(t'){:}\| \xrightarrow[\mathcal{A}]{*} \|{:}(S)\flat^{*}q_f{:}\|$ for some ℓ and t'. From lemma 4, we have $t \xrightarrow[\mathcal{R}]{} t'$. Thus, we have $t \xrightarrow[\mathcal{R}]{} S$ by applying lemma 4 repeatedly. $\square$

The Proposition 3 follows from Lemmata 5 and 6.

B Proof of Theorem 1

We give below a detailed proof of Theorem 1, which states that given a CSTG $\mathcal{G}$ and a monotonic CntTRS $\mathcal{R}$, one can construct in PTIME a CSTG generating the closure of $L(\mathcal{G})$ by $\mathcal{R}$, and whose size is linear in the size of $\mathcal{G}$ and $\mathcal{R}$.

We denote the ith rewrite rule of CntTRS by $\mathcal{A}_i : \ell_i \to r_i$ where $\ell_i, r_i \in \mathcal{T}(F, X)$ and $\mathcal{A}_i = \langle Q_i, Q_i^f, S_i, \Delta_i \rangle$ is a selection automaton. We assume that Q_i's are disjoint each other. In the sequel, we use large character A for non-terminal, and $\mathcal{C}, \mathcal{C}'$ for contexts that has no terminal, $\langle t \rangle$ for the term obtained by replacing every signature a in t by the non-terminal $\langle a \rangle$, $\langle t \rangle^\lambda$ for the term obtained by replacing root symbol f of $\langle t \rangle$ by $\langle f \rangle^\lambda$, and let $Q := \bigcup_i Q_i$, $F := \bigcup_i F_i$, and $\Delta := \bigcup_i \Delta_i$.

We sometimes denote the sequence of terms $t_1, \ldots, t_n$ by $\overline{t}$ for readability.

Completion procedure.

Input CSTG $\mathcal{G} = \langle \mathcal{N}, S, \Sigma, P \rangle$ such that arbitrary terminal a is only produced from the non-terminal $\langle a \rangle$ by the rule $\langle a \rangle(x_1, \ldots, x_n) \to a(x_1, \ldots, x_n)$, and monotonic CntTRS $\mathcal{R}$

Output Context-sensitive tree grammar $\mathcal{G}_* = \langle \mathcal{N}_*, S^\lambda, \Sigma, P_* \rangle$ that recognizes $R^*(L(G))$.

Step1(initialize) Let $\mathcal{N}_*$ be as follows:

– $\mathcal{N}_* := \mathcal{N} \cup \{A^\lambda, \langle A, q \rangle, \langle A, q \rangle^\lambda, \langle A, q \rangle_{fin}, \langle A, q \rangle^\lambda_{fin} \mid A \in \mathcal{N}, q \in Q, 1 \leq i \leq n\}$ where n is the number of rewrite rules.

Before constructing P_*, we construct P' to mark λ at root symbol.

$$\begin{aligned} P' := {} & P \\ & \cup \{A^\lambda(\mathcal{C}[x_1, \ldots, x_n]) \to A'^\lambda(\mathcal{C}'[x_1, \ldots, x_n]) \mid A(\mathcal{C}[x_1, \ldots, x_n]) \to A'(\mathcal{C}'[x_1, \ldots, x_n]) \in P\} \\ & \cup \{\langle a \rangle^\lambda(x_1, \ldots, x_n) \to a(x_1, \ldots, x_n) \mid \langle a \rangle(x_1, \ldots, x_n) \to a(x_1, \ldots, x_n) \in P\} \end{aligned}$$

P_* is composed by P' and the following sets of production rules. These are used to simulate a selection automaton.

$$P_{\mathcal{A}} := \left\{ \begin{array}{l|l} \langle f \rangle(\langle f_1, q_1 \rangle(\overline{x_1}), \ldots, \langle f_n, q_n \rangle(\overline{x_n})) & \\ \to \langle f, q \rangle(\langle f_1, q_1 \rangle(\overline{x_1}), \ldots, \langle f_n, q_n \rangle(\overline{x_n})) & f, f_1, \ldots, f_n \in \Sigma, \\ \langle f \rangle^\lambda(\langle f_1, q_1 \rangle(\overline{x_1}), \ldots, \langle f_n, q_n \rangle(\overline{x_n})) & f(q_1, \ldots, q_n) \to q \in \Delta \\ \to \langle f, q \rangle^\lambda(\langle f_1, q_1 \rangle(\overline{x_1}), \ldots, \langle f_n, q_n \rangle(\overline{x_n})) & \end{array} \right\}$$

$$P_{fin} := \left\{ \begin{array}{l|l} \langle f, q_f \rangle^\lambda(\overline{x}) \to \langle f, q_f \rangle^\lambda_{fin}(\overline{x}) & \\ \langle f, q \rangle_{fin}(\langle f_1, q_1 \rangle(\overline{x_1}), \ldots, \langle f_n, q_n \rangle(\overline{x_n})) & f, f_1, \ldots, f_n \in \Sigma, \\ \to \langle f \rangle(\langle f_1, q_1 \rangle_{fin}(\overline{x_1}), \ldots, \langle f_n, q_n \rangle_{fin}(\overline{x_n})) & f(q_1, \ldots, q_n) \to q \in \Delta \\ \langle f, q \rangle^\lambda_{fin}(\langle f_1, q_1 \rangle(\overline{x_1}), \ldots, \langle f_n, q_n \rangle(\overline{x_n})) & q \in F \\ \to \langle f \rangle^\lambda(\langle f_1, q_1 \rangle_{fin}(\overline{x_1}), \ldots, \langle f_n, q_n \rangle_{fin}(\overline{x_n})) & \end{array} \right\}$$

$$P_{\mathcal{R}} := \left\{ \begin{array}{l|l} \langle f, q \rangle_{fin}(\langle l_1 \rangle, \ldots, \langle l_n \rangle) \to \langle r_i \rangle & q \in S_i, \mathcal{A}_i : f(l_1, \ldots, l_n) = \ell_i \to r_i \in \mathcal{R} \\ \langle f, q \rangle^\lambda_{fin}(\langle l_1 \rangle, \ldots, \langle l_n \rangle) \to \langle r_i \rangle^\lambda & \end{array} \right\}.$$

$$P_{re} := \left\{ \begin{array}{l|l} \langle f, q \rangle^\lambda(\overline{x}) \to \langle f \rangle^\lambda(\overline{x}) & \\ \langle f, q \rangle(\overline{x}) \to \langle f \rangle(\overline{x}) & f \in \Sigma, \\ \langle f, q \rangle^\lambda_{fin}(\overline{x}) \to \langle f \rangle^\lambda(\overline{x}) & q \in Q \\ \langle f, q \rangle_{fin}(\overline{x}) \to \langle f \rangle(\overline{x}) & \end{array} \right\}$$

$P_{\mathcal{A}}$ is used to simulate transition of SA, P_{fin} is used to propagate information that final state occur at the root position by marking fin, $P_{\mathcal{R}}$ is used to simulate rewriting by CntTRS $\mathcal{R}$, and P_{re} is used to clear states in each non-terminal of the form $\langle f, q\rangle$.

Lemma 7. *$S \xrightarrow[P]{*} A(t_1, \ldots, t_n)$ iff $S^{\lambda} \xrightarrow[P']{*} A^{\lambda}(t_1, \ldots, t_n)$.*

Proof. We can prove this lemma by induction on the length of $\xrightarrow[P]{*}$ for only if part and $\xrightarrow[P']{*}$ for if part. □

Lemma 8. *For all i, $f(t_1, \ldots, t_n) \xrightarrow[\mathcal{A}_i]{*} q$ iff $\langle f\rangle(\langle t_1\rangle, \ldots, \langle t_n\rangle) \xrightarrow[\mathcal{G}_*]{*} \langle f, q\rangle(\langle t_1\rangle, \ldots, \langle t_n\rangle)$ and $\langle f\rangle^{\lambda}(\langle t_1\rangle, \ldots, \langle t_n\rangle) \xrightarrow[\mathcal{G}_*]{*} \langle f, q\rangle^{\lambda}(\langle t_1\rangle, \ldots, \langle t_n\rangle)$.*

Proof. Since Q_i's are disjoint each other, we can easily prove only if part of this Lemma by the rules in $P_{\mathcal{A}}$ and P_{re}. In the following, we prove if part by induction on $||f(t_1, \ldots, t_n)||$.

Let $f(t_1, \ldots, t_n)$ be $f(f_1(\overline{t_1}), \ldots, f_n(\overline{t_n}))$ for $n \geq 0$. Since $\langle f, q\rangle$ is produced by the rule $\langle f\rangle(\langle f_1, q_1\rangle(\overline{x_1}), \ldots, \langle f_n, q_n\rangle(\overline{x_n})) \to \langle f, q\rangle(\langle f_1, q_1\rangle(\overline{x_1}), \ldots, \langle f_n, q_n\rangle(\overline{x_n}))$ where $f(q_1, \ldots, q_n) \to q \in \Delta_i$ for some i, we have $f_i(\overline{t_i}) \xrightarrow[\Delta_i]{*} q_i$ from the induction hypothesis. Thus, $f(f_1(\overline{t_1}), \ldots, f_n(\overline{t_n})) \to q$ follows.

In the case where the root position has mark λ, we can prove similarly. □

Lemma 9. *1. For $q \in Q_i$, $\mathcal{C}[f(t_1, \ldots, t_n)] \xrightarrow[\mathcal{A}_i]{*} \mathcal{C}[q] \xrightarrow[\mathcal{A}_i]{*} q_f$ for some $q_f \in F_i$ iff $\langle\mathcal{C}\rangle^{\lambda}[\langle f\rangle(\langle t_1\rangle, \ldots, \langle t_n\rangle)] \xrightarrow[\mathcal{G}_*]{*} \langle\mathcal{C}\rangle^{\lambda}[\langle f, q\rangle_{fin}(\langle t_1\rangle, \ldots, \langle t_n\rangle)]$ for $||\mathcal{C}|| > 0$, and*
2. $f(t_1, \ldots, t_n) \xrightarrow[\mathcal{A}_i]{*} q_f \in F_i$ iff $\langle f\rangle^{\lambda}(\langle t_1\rangle, \ldots, \langle t_n\rangle) \xrightarrow[\mathcal{G}_*]{*} \langle f, q\rangle^{\lambda}_{fin}(\langle t_1\rangle, \ldots, \langle t_n\rangle)$.

Proof. 2 of this lemma follows from Lemma 8 and rules in P_{fin}.

We prove both direction of 1 of this lemma by induction on the length of the position p that $f(t_1, \ldots, t_n)$ occur in $\mathcal{C}[f(t_1, \ldots, t_n)]$. Let $\mathcal{C}[x] = \mathcal{C}'[f'(t'_1, \ldots, x, \ldots, t'_n)]$ and hence $\mathcal{C}[f(t_1, \ldots, t_n)]$ is represented as $\mathcal{C}'[f'(t'_1, \ldots, f(t_1, \ldots, t_n), \ldots, t'_n)]$.

(only if part) Assume that we have the transition $\mathcal{C}'[f'(f_1(\overline{t'_1}), \ldots, f(t_1, \ldots, t_n), \ldots, f_n(\overline{t'_n}))] \xrightarrow[\mathcal{A}_i]{*} \mathcal{C}'[f'(q'_1, \ldots, q, \ldots, q'_n)] \xrightarrow[\mathcal{A}_i]{} \mathcal{C}'[q'] \xrightarrow[\mathcal{A}_i]{*} q_f \in F_i$. Then, from Lemma 8, we have $\langle f_i\rangle(\overline{\langle t'_i\rangle}) \xrightarrow[\mathcal{G}_*]{*} \langle f_i, q'_i\rangle(\overline{\langle t'_i\rangle})$ and $\langle f\rangle(\langle t_1\rangle, \ldots, \langle t_n\rangle) \xrightarrow[\mathcal{G}_*]{*} \langle f, q\rangle(\langle t_1\rangle, \ldots, \langle t_n\rangle)$, and from the induction hypothesis or 1 of this lemma, we have the production:

$$\begin{aligned}&\langle\mathcal{C}'\rangle[\langle f'\rangle(\langle f_1\rangle(\overline{\langle t'_1\rangle}), \ldots, \langle f\rangle(\langle t_1\rangle, \ldots, \langle t_n\rangle), \ldots, \langle f_n\rangle(\overline{\langle t'_n\rangle}))]\\ \xrightarrow[\mathcal{G}_*]{*}\ &\langle\mathcal{C}'\rangle[\langle f', q'\rangle_{fin}(\langle f_1\rangle(\overline{\langle t'_1\rangle}), \ldots, \langle f\rangle(\langle t_1\rangle, \ldots, \langle t_n\rangle), \ldots, \langle f_n\rangle(\overline{\langle t'_n\rangle}))]\end{aligned}$$

Thus, we obtaine the production

$$\begin{aligned}&\langle\mathcal{C}'\rangle[\langle f'\rangle(\langle f_1\rangle(\overline{\langle t'_1\rangle}), \ldots, \langle f\rangle(\langle t_1\rangle, \ldots, \langle t_n\rangle), \ldots, \langle f_n\rangle(\overline{\langle t'_n\rangle}))]\\ \xrightarrow[\mathcal{G}_*]{*}\ &\langle\mathcal{C}'\rangle[\langle f', q'\rangle_{fin}(\langle f_1, q'_1\rangle(\overline{\langle t'_1\rangle}), \ldots, \langle f, q\rangle(\langle t_1\rangle, \ldots, \langle t_n\rangle), \ldots, \langle f_n, q'_n\rangle(\overline{\langle t'_n\rangle}))]\\ \xrightarrow[\mathcal{G}_*]{}\ &\langle\mathcal{C}'\rangle[\langle f'\rangle(\langle f_1, q'_1\rangle_{fin}(\overline{\langle t'_1\rangle}), \ldots, \langle f, q\rangle_{fin}(\langle t_1\rangle, \ldots, \langle t_n\rangle), \ldots, \langle f_n, q'_n\rangle_{fin}(\overline{\langle t'_n\rangle}))]\\ \xrightarrow[\mathcal{G}_*]{}\ &\langle\mathcal{C}'\rangle[\langle f'\rangle(\langle f_1\rangle(\overline{\langle t'_1\rangle}), \ldots, \langle f, q\rangle_{fin}(\langle t_1\rangle, \ldots, \langle t_n\rangle), \ldots, \langle f_n\rangle(\overline{\langle t'_n\rangle}))]\end{aligned}$$

from the rule in P_{fin} and P_{re}, and $f(q'_1, \ldots, q, \ldots, q'_n) \to q' \in \Delta_i$.

(if part) Since the non-terminal $\langle f, q\rangle_{fin}$ must be produced by the rule $\langle f', q'\rangle_{fin}(\langle f_1, q'_1\rangle(\overline{x_1}), \ldots, \langle f, q\rangle(\overline{x}), \ldots, \langle f_n, q'_n\rangle(\overline{x_n})) \to \langle f'\rangle(\langle f_1, q'_1\rangle_{fin}(\overline{x_1}), \ldots, \langle f, q\rangle_{fin}(\overline{x}), \ldots, \langle f_n, q'_n\rangle_{fin}(\overline{x_n}))$, we have the production $\langle \mathcal{C}'\rangle[\langle f'\rangle(\langle f_1\rangle(\langle\overline{t'_1}\rangle), \ldots, \langle f\rangle(\langle t_1\rangle, \ldots, \langle t_n\rangle), \ldots, \langle f_n\rangle(\langle\overline{t'_n}\rangle))] \xrightarrow[\mathcal{G}_*]{*} \langle \mathcal{C}'\rangle[\langle f', q'\rangle_{fin}(\langle f_1, q'_1\rangle(\langle\overline{t'_1}\rangle), \ldots, \langle f, q\rangle(\langle t_1\rangle, \ldots, \langle t_n\rangle), \ldots, \langle f_n, q'_n\rangle(\langle\overline{t'_n}\rangle))]$. Hence, from the induction hypothesis, we have $\mathcal{C}'[f'(f_1(\overline{t_1}), \ldots, f(t_1, \ldots, t_n), \ldots, f(\overline{t_n}))] \xrightarrow[\Delta_i]{*} \mathcal{C}'[q'] \xrightarrow[\Delta_i]{*} q_f$ for some $q_f \in F_i$. Moreover, we have $f(q_1, \ldots, q, \ldots, q_n) \to q \in \Delta_i$ since states are disjoint for each $\mathcal{A}_i$, and from Lemma 8, $f_i(\overline{t_i} \xrightarrow[\mathcal{A}_i]{*} q_i$ for all i and $f(t_1, \ldots, t_n) \xrightarrow[\mathcal{A}_i]{*} q$. Therefore, we have $\mathcal{C}[f(t_1, \ldots, t_n)] \xrightarrow[\mathcal{A}_i]{*} \mathcal{C}[q] \xrightarrow[\mathcal{A}_i]{*} q_f \in F_i$.

□

Lemma 10. *If $\mathcal{C}[\ell\sigma] \xrightarrow[\mathcal{R}]{} \mathcal{C}[r\sigma]$ and $\mathcal{C}[\ell\sigma] \in L(\mathcal{G}_*)$, then $\mathcal{C}[r\sigma] \in L(\mathcal{G}_*)$.*

Proof. Consider the case where $||\mathcal{C}|| > 0$. We suppose that $\mathcal{C}[\ell\sigma]$ is rewritten to $\mathcal{C}[r\sigma]$ by the rule $\mathcal{A}_i : f(l_1, \ldots, l_n) \to r$. Then, we have the transition $\mathcal{C}[f(l_1, \ldots, l_n)\sigma] \xrightarrow[\mathcal{A}_i]{*} \mathcal{C}[q] \xrightarrow[\mathcal{A}_i]{*} q_f$ where $q \in S_i$ and $q_f \in Q^f_i$. Since $S^\lambda \xrightarrow[\mathcal{G}_*]{*} \langle \mathcal{C}\rangle[\langle f(l_1, \ldots, l_n)\sigma\rangle]$, we have $S^\lambda \xrightarrow[\mathcal{G}_*]{*} \langle \mathcal{C}\rangle^\lambda[\langle f, q\rangle(\langle l_1\rangle, \ldots, \langle l_n\rangle)\langle\sigma\rangle] \xrightarrow[\mathcal{G}_*]{*} \langle \mathcal{C}\rangle^\lambda[\langle f, q\rangle_{fin}(\langle l_1\rangle, \ldots, \langle l_n\rangle)\langle\sigma\rangle]$ from 1 of Lemma 9. From the production rule $P_\mathcal{R}$, we have $\langle \mathcal{C}\rangle[\langle f, q\rangle_{fin}(\langle l_1\rangle, \ldots, \langle l_n\rangle)\langle\sigma\rangle] \xrightarrow[\mathcal{G}_*]{*} \langle \mathcal{C}\rangle[\langle r\rangle\langle\sigma\rangle]$. Thus, we have $S^\lambda \xrightarrow[\mathcal{G}_*]{*} \mathcal{C}[r\sigma]$. In the case of $||\mathcal{C}|| = 0$, we can prove similary to the previous case. □

Lemma 11. *If $s \in L(\mathcal{G})$ and $s \xrightarrow[R]{*} t$, then $t \in L(\mathcal{G}_*)$.*

Proof. From Lemma 7, we have $S^\lambda \xrightarrow[\mathcal{G}_*]{*} \langle s\rangle^\lambda$, and since P_* has the rules in P and the rule $\langle f\rangle^\lambda(x_1, \ldots, x_n) \to f(x_1, \ldots, x_n)$, we have $S^\lambda \xrightarrow[\mathcal{G}_*]{*} s$. By applying Lemma 10 repeatedly, we also have $t \in L(\mathcal{G}_*)$. □

Lemma 12. *If $t \in L(\mathcal{G}_*)$, then there exists s such that $s \xrightarrow[R]{*} t$ and $s \in L(\mathcal{G})$.*

Proof. We show this lemma by induction on the number of the rules in $P_\mathcal{R}$ that occur in the production $S \xrightarrow[\mathcal{G}_*]{*} t$.

Consider the case that there is no rule in $P_\mathcal{R}$ in the production $S^\lambda \xrightarrow[G_*]{*} t$. We have $S^\lambda \xrightarrow[G_*]{*} \langle t\rangle \xrightarrow[G_*]{*} t$. Here we aim at one symbol $\langle a\rangle$ in $\langle t\rangle$. Suppose that $\langle a\rangle$ is produced by the rule in $P_*\backslash(P \cup P_\mathcal{R})$. In this case, we can produce $\langle a\rangle$ by the rule of the form $\langle a, q\rangle \to \langle a\rangle$ or $\langle a, q\rangle_{fin} \to \langle a\rangle$. Since the symbols $\langle a, q\rangle$ or $\langle a, q\rangle_{fin}$ is produced from $\langle a\rangle$ by the rules in $P_\mathcal{A}$ and P_{fin}, the symbol $\langle a\rangle$ must be produced by P. Moreover, this claim also holds for the symbols of the form $\langle a\rangle^\lambda$. By applying this claim to all symbols in $\langle t\rangle$, we have $S^\lambda \xrightarrow[P']{*} \langle t\rangle$, and hence $S \xrightarrow[P]{*} \langle t\rangle$ from Lemma 7.

Otherwise, let $t = \mathcal{C}[\mathcal{C}_1[t_1, \ldots, t_n]]$ where $||\mathcal{C}|| > 1$, and t is produced as $S^\lambda \xrightarrow[\mathcal{G}_*]{*} \langle \mathcal{C}\rangle^\lambda[\langle \mathcal{C}_2\rangle[\langle t_1\rangle, \ldots, \langle t_n\rangle]] \xrightarrow[P_\mathcal{R}]{} \langle \mathcal{C}\rangle^\lambda[\langle C_1\rangle[\langle t_1\rangle, \ldots, \langle t_n\rangle]] \xrightarrow[\mathcal{G}_*]{*} t$. From the construction of $P_\mathcal{R}$, we have the rewrite rule $\mathcal{A}_i : \mathcal{C}_1[x_1, \ldots, x_m] \to \mathcal{C}_2[x_1, \ldots, x_m]$ where $\mathcal{C}_1[x_1, \ldots, x_m]\sigma = \mathcal{C}_1[t_1, \ldots, t_n]$ and $\mathcal{C}_2[x_1, \ldots, x_m]\sigma = \mathcal{C}_2[t_1, \ldots, t_n]$ for some σ. Since root symbol in $\langle C_2\rangle$ is of the form $\langle f, q\rangle_{fin}$, we have the transition $\mathcal{C}[\mathcal{C}_2[t_1, \ldots, t_n]] \xrightarrow[\mathcal{A}_i]{*} \mathcal{C}[q] \xrightarrow[\mathcal{A}_i]{*} q_f$ for some $q \in S_i$ and $q_f \in Q^f_i$ from Lemma 9.

Hence we have $\mathcal{C}[\mathcal{C}_2[t_1,\dots,t_n]] \xrightarrow[R]{} \mathcal{C}[\mathcal{C}_1[t_1,\dots,t_n]]$. Here, we can easily obtain the production $\langle\mathcal{C}\rangle^{\lambda}[\langle\mathcal{C}_2\rangle[\langle t_1\rangle,\dots,\langle t_n\rangle]] \xrightarrow[G_*]{*} \mathcal{C}[\mathcal{C}_2[t_1,\dots,t_n]]$. Thus, there exists s such that $s \xrightarrow[R]{*} \mathcal{C}[\mathcal{C}_2[t_1,\dots,t_n]]$ from the induction hypothesis and we have $s \xrightarrow[R]{*} t$.

If $||C|| = 0$, then root symbol in $\langle C_2\rangle$ is of the form $\langle f,q\rangle^{\lambda}_{f}$. In this case, we can prove similary to the previous case. □

Theorem 1 follows from Lemma 11 and Lemma 12.

C Proof of Proposition 6

We give a detailed proof of Proposition 6, which states that reachability is NLINSPACE-complete and regular model checking is undecidable for monotonic and flat CntTRS over unary signatures.

It works by reducing the acceptance (for reachability) and emptiness (for regular model checking) problems for linear bounded automata (LBA) [20].

Let $\mathcal{M} = \langle P, \Gamma, p_0, G, \Theta\rangle$ be a non-deterministic LBA where P is the set of states, Γ is an input alphabet containing in particular the left and right endmarkers $\|\!:$ and $:\!\|$ are $p_0 \in P$ is the initial state, $G \subseteq P$ is the set of accepting states and Θ is the transition relation in $P \times \Gamma \times P \times \Gamma \times \{\mathsf{left}, \mathsf{right}\}$. The transition relation Θ is such that $\mathcal{M}$ cannot move left from $\|\!:$, cannot move right from $:\!\|$ or print another symbols over $\|\!:$ or $:\!\|$.

Let $\Gamma' = \Gamma \cup \{a^p \mid a \in \Gamma, p \in P\}$ and let us define the unary signature $\Sigma = \Gamma' \cup (\Gamma' \times \Theta) \cup \{\bot\}$, where $\bot$ has arity 0 and every other symbol has arity 1. For the sake of simplicity, a term $a_1(a_2(\dots a_n(\bot)))$ will be denoted by the string $a_1\, a_2 \dots a_n$.

Every configuration of $\mathcal{M}$ will be represented by a term of $\mathcal{T}(\Sigma)$ of the form $\|\!:\!x_1 \dots x_{j-1} x_j^p x_{j+1} \dots x_n\!:\!\|$, where, for each $1 \le i \le n$, $x_i \in \Gamma \setminus \{\|\!:, :\!\|\}$ is the content of the i^{th} cell of the tape of $\mathcal{M}$, j is the current position of the head of $\mathcal{M}$ and $p \in P$ is the current state. In particular, given an input word $x_1 \dots x_n \in \big(\Gamma \setminus \{\|\!:, :\!\|\}\big)^*$, the initial configuration of $\mathcal{M}$ is represented by $\|\!: x_1^{p_0} x_2 \dots x_n\!:\!\|$. We assume indeed *wlog* that the computation space of $\mathcal{M}$ is limited to the exact size of the input word. We moreover assume *wlog* that when $\mathcal{M}$ reaches an accepting state, it writes a special blank symbol $\flat \in \Gamma \setminus \{\|\!:, :\!\|\}$ in all the cells of the tape (except the left and right ends), then moves to the leftmost cell, and finally changes its state to a special state p_1 which does not occur in the left hand of any transition of Θ.

To every transition $\theta = \langle p, a, p', b, \mathsf{left}\rangle$ in Θ, we associate the four following monotonic and flat controlled rules

$$\begin{array}{lcr}
\|\!: \Gamma^* c\, a^p \Gamma^* \!:\!\| & : & a^p(x) \to \langle a^p, \theta\rangle(x) \\
\|\!: \Gamma^* c\, \langle a^p, \theta\rangle \Gamma^* \!:\!\| & : & c(x) \to \langle c, \theta\rangle(x) \\
\|\!: \Gamma^* \langle c, \theta\rangle\, \langle a^p, \theta\rangle \Gamma^* \!:\!\| & : & \langle a^p, \theta\rangle(x) \to b(x) \\
\|\!: \Gamma^* \langle c, \theta\rangle\, b \Gamma^* \!:\!\| & : & \langle c, \theta\rangle(x) \to c^{p'}(x)
\end{array}$$

We use here for SA a simplified notation with regular expressions, assuming that the SA selects the position of the only rewritable letter in the expression.

Similarly, to every transition $\theta = \langle p, a, p', b, \mathsf{right}\rangle$ in Θ, we associate the following controlled rules

$$\begin{array}{lcr}
\|\!:\varGamma^* a^p c\, \varGamma^* :\!\| & : & a^p(x) \to \langle a^p, \theta\rangle(x) \\
\|\!:\varGamma^* \langle a^p, \theta\rangle\, c\, \varGamma^* :\!\| & : & c(x) \to \langle c, \theta\rangle(x) \\
\|\!:\varGamma^* \langle a^p, \theta\rangle\, \langle c, \theta\rangle \varGamma^* :\!\| & : & \langle a^p, \theta\rangle(x) \to b(x) \\
\|\!:\varGamma^*\, b\, \langle c, \theta\rangle \varGamma^* :\!\| & : & \langle c, \theta\rangle(x) \to c^{p'}(x)
\end{array}$$

Let $\mathcal{R}$ be the CntTRS containing all the controlled rewrite rules associated to the transitions of Θ. It is easy to show that $\mathcal{R}$ simulates the moves of $\mathcal{M}$, and that only the correct moves of $\mathcal{M}$ are simulated by $\mathcal{R}$.

Therefore, with the above hypotheses, $\mathcal{M}$ will accept the initial word $x_1 \ldots x_n$ iff $\|\!: x_1^{p_0} x_2 \ldots x_n :\!\| \xrightarrow[\mathcal{R}]{*} \|\!: \flat^{p_1} \underbrace{\flat \ldots \flat}_{n-1} :\!\|$ and the language of $\mathcal{M}$ is empty iff

$\mathcal{R}^*\big(\|\!:((\varGamma \setminus \{\|\!:, :\!\|\}) \times P)\,(\varGamma \setminus \{\|\!:, :\!\|\})^* :\!\|\big) \cap \|\!: \flat^{p_1}\, \flat^* :\!\| = \emptyset.$

D Proof of Theorem 3

We give below a detailed proof of Theorem 3, stating that regular model-checking is decidable in EXPTIME for linear and right-shallow recursive pCntTRS.

Given a right-shallow and linear recursive pCntTRS $\mathcal{R}$ and the language $L \subseteq \mathcal{T}(\varSigma)$ of a TA $\mathcal{A}_L = \langle Q_L, F_L, \varDelta_L\rangle$, the proof is the construction of an alternating tree automata with epsilon-transitions (ε-ATA) $\mathcal{A}'$ recognizing the rewrite closure $\mathcal{R}^*(L)$.

Intuitively, an alternating tree automata $\mathcal{A}$ is a top-down tree automaton that can spawn in several copies during computation on a term t. At every computation step, there are several copies of $\mathcal{A}$ in different positions of t, each copy in its own state. Initially, there is one copy of $\mathcal{A}$ in its initial state at the root of t. Then, the copies can be propagated down to the leaves step by step.

Formally, an *alternating TA* with ε-transitions (ε-ATA) over a signature $\varSigma$ is a tuple $\mathcal{A} = \langle Q, q_0, \delta\rangle$ where Q is a finite set of states, $q_0 \in Q$ is the initial state and δ is a function which associates to every state $q \in Q$ a disjunction of conjunctions of propositional predicates of the following form

– $a \in \varSigma$,
– $\langle q', \varepsilon\rangle$, for $q' \in Q \setminus \{q\}$,
– $\langle q', i\rangle$, for $q' \in Q$ and $1 \leq i \leq m$ where m is the maximal arity of a symbol in $\varSigma$.

We say that a set P of predicates as above satisfies a disjunction of conjunctions D, denoted by $P \models D$, if either D is the empty disjunction or $D = C_1 \vee \ldots \vee C_k$, for $k \geq 1$, and there exists one j, $1 \leq j \leq k$, such that every propositional predicate in the conjunction C_j belongs to P.

A *run* of $\mathcal{A}$ on $t \in \mathcal{T}(\Sigma)$ is a function ρ from $\mathcal{P}os(t)$ into 2^Q such that for all position $p \in \mathcal{P}os(t)$, with $t(p) = a \in \Sigma_n$, $n \geq 0$, and for all state $q \in \rho(p)$, it holds that

$$a, \langle \rho(p.1), 1\rangle, \ldots, \langle \rho(p.n), n\rangle, \langle \rho(p), \varepsilon\rangle \models \delta(q)$$

where $\langle S, p\rangle$ is a notation for all the predicates $\langle q, p\rangle$ with $q \in S$.
The language $L(\mathcal{A})$ of $\mathcal{A}$ is the set of terms $t \in \mathcal{T}(\Sigma)$ on which there exists a run ρ of $\mathcal{A}$ such that $q_0 \in \rho(\varepsilon)$ (terms *recognized* by $\mathcal{A}$).

Given an ε-ATA $\mathcal{A} = \langle Q, q_0, \delta\rangle$, one can build a TA $\mathcal{A}' = \langle 2^Q, \{S \subseteq Q \mid q_0 \in S\}, \Delta\rangle$ where Δ contains all the TA transitions of the form $a(S_1, \ldots, S_n) \to S$, for $a \in \Sigma_n$, $S_1, \ldots, S_n, S \subseteq Q$ such that for all $q \in S$, $a, \langle S_1, 1\rangle, \ldots, \langle S_n, n\rangle, \langle S, \varepsilon\rangle \models \delta(q)$. With these definitions, it is easy to see that every run of $\mathcal{A}$ on t is also a run of $\mathcal{A}'$ on t, and reciprocally, hence $L(\mathcal{A}') = L(\mathcal{A})$.

On the other hand, given a TA $\mathcal{A} = \langle Q, F, \Delta\rangle$, one can construct an ε-ATA $\mathcal{A}' = \langle Q \cup \{q_0\}, q_0, \delta\rangle$, with $q_0 \notin Q$, $\delta(q_0) = \bigvee_{q \in F} \langle q, \varepsilon\rangle$, and for all $q \in Q$,

$$\delta(q) = \bigvee_{a(q_1,\ldots,q_n) \to q \in \Delta} (a \wedge \langle q_1, 1\rangle \wedge \ldots \wedge \langle q_n, n\rangle)$$

We will now construct an ε-ATA $\mathcal{A}'$ recognizing the rewrite closure $\mathcal{R}^*(L)$. Let $sr(\mathcal{R})$ be the set of ground direct subterms of the *rhs* of rules of $\mathcal{R}$. For each $r \in sr(\mathcal{R})$, we construct a TA $\mathcal{A}_r = \langle Q_r, F_r, \Delta_r\rangle$ such that $L(\mathcal{A}_r) = \{r\}$. It is defined by $Q_r = \mathcal{P}os(r)$, $F_r = \{\varepsilon\}$ and $\Delta = \{a(p.1, \ldots, p.n) \to p \mid p \in \mathcal{P}os(r), r(p) = a \in \Sigma_n\}$.

Let $\mathcal{A}'_r$ be the ε-ATA associated to $\mathcal{A}_r$ as above. Let $\mathcal{A}'_L = \langle Q_L, q_0, \delta_L\rangle$ be the ε-ATA associated to $\mathcal{A}_L$ and for each controlled rule $\mathcal{C} : \ell \to r$ of $\mathcal{R}$, let $\mathcal{C}'$ be the ε-ATA associated to $\mathcal{C}$. We assume *wlog* that all the automata defined above have disjoint state sets. We will construct incrementally a finite sequence of ε-ATA $\mathcal{A}'_0, \ldots, \mathcal{A}'_k$ whose last element $\mathcal{A}'_k$ recognizes $\mathcal{R}^*(L)$.

The first ε-ATA of the sequence, $\mathcal{A}'_0 = \langle Q_0, q_0, \delta_0\rangle$ is defined as the disjoint union of the ε-ATAs $\mathcal{A}'_L$, all the $\mathcal{A}'_r$ for $r \in sr(\mathcal{R})$, and all $\mathcal{C}'$, for $\mathcal{C}$ prefix SA controlling a rule of $\mathcal{R}$. Every other $\mathcal{A}'_i$ in this sequence will be $\langle Q_0, q_0, \delta_i\rangle$, where the transition function δ_i is defined as follows.

We introduce a notation that will be useful for the construction of the δ_i. Let $t = a(t_1, \ldots, t_n)$ be a ground term of $\mathcal{T}(\Sigma \cup Q)$, where $n \geq 0$ and the symbols of Q are assumed of arity 0. We write $t \models_i q$ if we are in one of the following cases

$\delta_i(q) = \phi_1 \vee \phi_2$ and $t \models_i \phi_1$ or $t \models_i \phi_2$
$\delta_i(q) = \phi_1 \wedge \phi_2$ and $t \models_i \phi_1$ and $t \models_i \phi_2$
$\delta_i(q) = \langle q', \varepsilon\rangle$ and $t \models_i q'$
$\delta_i(q) = \langle q', j\rangle$, with $1 \leq j \leq n$ and $t_j \models_i q'$
$\delta_i(q) = a$.

Assume that we have constructed all the functions up to δ_i, and let us define δ_{i+1}. If there is a linear controlled rule $\mathcal{C} : \ell \to b(r_1, \ldots, r_m)$ in $\mathcal{R}$, where every

r_i is either a variable or a ground term of $\mathcal{T}(\Sigma)$, and a substitution σ from $\mathcal{X}$ into Q grounding for ℓ, for all state $q \in Q$ such that $\ell\sigma \models_i q$, let

$$\delta_{i+1}(q) = \delta_i(q) \vee \big(b \wedge \bigwedge_{i \in N} \langle q_{r_i}, i\rangle \wedge \bigwedge_{j \in V} \langle r_j\sigma, j\rangle \wedge \langle q_s, \varepsilon\rangle\big),\ \delta_{i+1}(q') = \delta_i(q') \text{ for all } q' \neq q$$

where $N = \{i \leq m \mid r_i \in \mathcal{T}(\Sigma)\}$, $V = \{j \leq m \mid r_j \in \mathcal{X}\}$, and q_s is the unique selection state of the ε-ATA $\mathcal{C}'$ associated to $\mathcal{C}$.

If there is a linear controlled rule $\mathcal{C} : \ell \to x$ in $\mathcal{R}$, where $x \in X$, and a substitution σ from $\mathcal{X}$ into Q grounding for ℓ and such that $\ell\sigma \models_i q$ then let

$$\delta_{i+1}(q) = \delta_i(q) \vee \langle q_s, \varepsilon\rangle, \quad \delta_{i+1}(q') = \delta_i(q') \text{ for all } q' \neq q$$

where q_s is the unique selection state of the ε-ATA $\mathcal{C}'$ associated to $\mathcal{C}$.
The number of conjunctions that can be added to a $\delta_i(q)$ in the above construction is bounded. Assuming that we do not add twice the same conjunction, the process will terminate with a fixpoint $\mathcal{A}'_k = \mathcal{A}'$. The size of $\mathcal{A}'$ is polynomial in the sizes of $\mathcal{A}_L$ and $\mathcal{R}$. It can be shown that $L(\mathcal{A}') \subseteq \mathcal{R}^*(L)$ by induction on the multiset of indexes i of the transition functions δ_i used in a run of $\mathcal{A}'$ on a term, and that $L(\mathcal{A}') \supseteq \mathcal{R}^*(L)$ by induction on the length of a rewrite sequence.

It follows that there exists a TA $\mathcal{A}$ whose size is exponential in the size of $\mathcal{A}'$ and such that $L(\mathcal{A}) = L(\mathcal{A}') = \mathcal{R}^*(L)$, and hence we can decide the problem of regular model checking for $\mathcal{R}$ in exponential time.

Example 6. Let us come back the recursive pCntTRS $\mathcal{R}$ of Example 5. The transitions of the prefix SA $\mathcal{C}_1$ and $\mathcal{C}_2$ for control in R are explicitly the following (their final states are respectively q^1_2 and q^2_4)

$$\mathcal{C}_1 : \bot \to q^1_s \mid q^1_0, f(q^1_0) \to q^1_s \mid q^1_0, a(q^1_s) \to q^1_1, a(q^1_1) \to q^1_2 \quad (f \in \{a,b,c,d\})$$
$$\mathcal{C}_2 : \bot \to q^2_s \mid q^1_0, f(q^2_0) \to q^2_s \mid q^2_0, a(q^2_s) \to q^2_1, a(q^1_1) \to q^2_2, a(q^2_2) \to q^2_3, a(q^2_3) \to q^2_4$$

The associated ε-ATA $\mathcal{C}'_1$ and $\mathcal{C}'_2$ have the following transition functions

$$\begin{aligned}
\delta^1 :\ & q^1_2 \mapsto a \wedge \langle q^1_1, 1\rangle, q^1_1 \mapsto a \wedge \langle q^1_s, 1\rangle, q^1_s \mapsto \bot \vee \bigvee_{f=a,b,c,d} (f \wedge \langle q^1_0, 1\rangle),\\
& q^1_0 \mapsto \bot \vee \bigvee_{f=a,b,c,d} (f \wedge \langle q^1_0, 1\rangle)\\
\delta^2 :\ & q^2_4 \mapsto a \wedge \langle q^2_3, 1\rangle, q^2_3 \mapsto a \wedge \langle q^2_2, 1\rangle, q^2_2 \mapsto a \wedge \langle q^2_1, 1\rangle, q^2_1 \mapsto a \wedge \langle q^2_s, 1\rangle,\\
& q^2_s \mapsto \bot \vee \bigvee_{f=a,b,c,d} (f \wedge \langle q^2_0, 1\rangle),\ q^2_0 \mapsto \bot \vee \bigvee_{f=a,b,c,d} (f \wedge \langle q^2_0, 1\rangle)
\end{aligned}$$

Let us consider the initial language $L = \{aaaac\}$, recognized by the ε-ATA $\mathcal{A}'_L$ with the following transition function

$$\delta_L : q_5 \mapsto a \wedge \langle q_4, 1\rangle, q_4 \mapsto a \wedge \langle q_3, 1\rangle, q_3 \mapsto a \wedge \langle q_2, 1\rangle, q_2 \mapsto a \wedge \langle q_1, 1\rangle, q_1 \mapsto c \wedge \langle q_0, 1\rangle, q_0 \mapsto \bot.$$

The transition function δ_0 in the above construction is the union of δ^1, δ^2 and δ_L. It holds that $a(a(q_1)) \models_0 q_3$. Hence, with the first rule of $\mathcal{R}$, we can let

$$\delta_1(q_3) = \delta_0(q_3) \vee \big(b \wedge \langle q_1, 1\rangle \wedge \langle q_s^1, \varepsilon\rangle\big)$$

Similarly, $a(a(q_s^2)) \models_1 q_2^2$, and we can let $\delta_2(q_2^2) = \delta_0(q_2^2) \vee \big(b \wedge \langle q_s^2, 1\rangle \wedge \langle q_s^1, \varepsilon\rangle\big)$. Moreover, $c(q_0) \models_2 q_1$, and with the second rule of $\mathcal{R}$, we can let

$$\delta_3(q_1) = \delta_0(q_1) \vee \big(d \wedge \langle q_0, 1\rangle \wedge \langle q_s^2, \varepsilon\rangle\big)$$

With these transitions functions, we have the following runs for 2 descendants of L

$$\begin{array}{ccccc} a & a & b & c & \perp \\ \begin{Bmatrix} q_5 \\ q_2^1 \end{Bmatrix} & \begin{Bmatrix} q_4 \\ q_1^1 \end{Bmatrix} & \begin{Bmatrix} q_3 \\ q_s^1 \end{Bmatrix} & \begin{Bmatrix} q_1 \\ q_0^1 \end{Bmatrix} & \begin{Bmatrix} q_0 \\ q_0^1 \end{Bmatrix} \end{array} \qquad \begin{array}{ccccc} a & a & b & d & \perp \\ \begin{Bmatrix} q_5 \\ q_4^2 \\ q_2^1 \end{Bmatrix} & \begin{Bmatrix} q_4 \\ q_3^2 \\ q_1^1 \end{Bmatrix} & \begin{Bmatrix} q_3 \\ q_2^2 \\ q_s^1 \end{Bmatrix} & \begin{Bmatrix} q_1 \\ q_s^2 \\ q_0^1 \end{Bmatrix} & \begin{Bmatrix} q_0 \\ q_0^2 \\ q_0^1 \end{Bmatrix} \end{array}$$

automaton for the initial term $aaaac$

$$q_5 \xleftarrow{a} q_4 \xleftarrow{a} q_3 \xleftarrow{a} q_2 \xleftarrow{a} q_1 \xleftarrow{c} q_0 \xleftarrow{\perp}$$

selection automaton $\mathcal{A}_1$

$$u_2 \xleftarrow{a} u_1 \xleftarrow{a} u_s \xleftarrow{\perp, \Sigma} u_0 \xleftarrow{\Sigma} u_0$$

(last transition is a loop)

the sequence of transitions $q_3 \xleftarrow{a} q_2 \xleftarrow{a} q_1$ matches the lhs of the first rewrite rule $\mathcal{A}_1 : aa \to b$, and we add the alternating transition: $q_3 \wedge u_s \xleftarrow{b} q_1$. (the conjunction with u_s simulates the recursive control)

selection automaton $\mathcal{A}_2$

$$v_4 \xleftarrow{a} v_3 \xleftarrow{a} v_2 \xleftarrow{a} v_1 \xleftarrow{a} v_s \xleftarrow{\perp, \Sigma} v_0 \xleftarrow{\Sigma} u_0$$

the sequence of transitions $v_2 \xleftarrow{a} v_1 \xleftarrow{a} v_s$ matches the lhs of the first rewrite rule $\mathcal{A}_1 : aa \to b$, and we add the alternating transition: $v_2 \wedge u_s \xleftarrow{b} v_s$.

the transition $q_1 \xleftarrow{c} q_0$ matches the lhs of the second rewrite rule $\mathcal{A}_2 : c \to d$, and we add the alternating transition: $q_1 \wedge v_s \xleftarrow{d} q_0$.

with these transitions, we have the following runs for 2 descendants of $aaaac$

$$\begin{array}{ccccc} a & a & b & c & \perp \\ \begin{Bmatrix} q_5 \\ u_2 \end{Bmatrix} & \begin{Bmatrix} q_4 \\ u_1 \end{Bmatrix} & \begin{Bmatrix} q_3 \\ u_s \end{Bmatrix} & \begin{Bmatrix} q_1 \\ u_0 \end{Bmatrix} & \begin{Bmatrix} q_0 \\ u_0 \end{Bmatrix} \end{array} \qquad \begin{array}{ccccc} a & a & b & d & \perp \\ \begin{Bmatrix} q_5 \\ v_4 \\ u_2 \end{Bmatrix} & \begin{Bmatrix} q_4 \\ v_3 \\ u_1 \end{Bmatrix} & \begin{Bmatrix} q_3 \\ v_2 \\ u_s \end{Bmatrix} & \begin{Bmatrix} q_1 \\ v_s \\ u_0 \end{Bmatrix} & \begin{Bmatrix} q_0 \\ v_0 \\ u_0 \end{Bmatrix} \end{array}$$