Transition Graphs of Rewriting Systems over Unranked Trees

Christof Löding and Alex Spelten

RWTH Aachen, Germany
{loeding,spelten}@informatik.rwth-aachen.de

Abstract. We investigate algorithmic properties of infinite transition graphs that are generated by rewriting systems over unranked trees. Two kinds of such rewriting systems are studied. For the first, we construct a reduction to ranked trees via an encoding and to standard ground tree rewriting, thus showing that the generated classes of transition graphs coincide. In the second rewriting formalism, we use subtree rewriting combined with a new operation called flat prefix rewriting and show that strictly more transition graphs are obtained while the first-order theory with reachability relation remains decidable.

Keywords: infinite graphs, reachability, rewriting, unranked trees.

1 Introduction

One of the main trends in verification is the field of infinite state model checking, in which procedures (and limits to their applicability) are developed to check systems with infinite state spaces against formal specifications (for a survey on infinite graphs cf. [19]).

In automatic verification, checking whether a system can reach an undesirable state or configuration translates to the reachability problem "Given a finite representation of an infinite graph G and two vertices u, u' of G, is there a path from u to u'?". From this point of view, an important task in the development of a theory of infinite graphs is to identify classes of infinite graphs where such elementary problems like reachability are decidable.

The strong formalism of monadic second-order logic (MSO) subsumes temporal logic (cf. [11]) and thus allows to express reachability properties. A well-known representative of graph classes with decidable MSO theory is the "pushdown hierarchy" introduced by Caucal [6]. Although this hierarchy is very rich and contains a lot of graphs, grid-like structures are not captured.

In order to compensate this weakness, a different approach of generating transition systems is to employ ranked trees (or terms) as the basic objects of the rewriting formalism, as already considered in [2]. Thereby, the internal structure of the trees is not of primary interest, but the different rewriting operations that can be applied on trees. Consequently, the vertices of the generated infinite graphs are represented by ranked trees, while the edge relation is induced by (simple) tree operations.

Among attractive subclasses of rewriting systems, an interesting and practical subclass is made up by the ground tree rewriting systems (which contain the infinite grid as transition graph; for an extensive analysis cf. [15]). "Ground rewriting" means that no variables occur in the rules, thus in ground tree (or term) rewriting systems (GTRSs), only explicitly specified subtrees can be replaced by other explicitly specified subtrees. Though in general MSO is undecidable for transition graphs of GTRSs, there is a decidable logic that allows to express reachability problems: first-order logic with the reachability relation [10]. In [13, 15] the structure of the transition graphs of GTRSs and their relation to other classes of infinite graphs, in particular to pushdown graphs, was studied. Furthermore, in [14, 15] several variants of the reachability problem for this class of graphs were investigated and a decidable logic was defined for the class of (regular) ground tree rewriting graphs.

For many applications however, the modelling of system states, messages, or data by ranked trees is not the most intuitive approach (if not impossible as e.g. the modelling of associative operations), since every symbol is of a fixed arity. Thus, our aim is to investigate a possible generalization of the idea of ground tree rewriting systems to the case of unranked trees. Briefly, unranked trees are finite labeled trees where nodes can have an arbitrary but finite number of children, and no fixed rank is associated to any label.

In this paper, we investigate to which extent results for ground tree rewriting systems are transferable to the unranked case. Note however, that the direct adaptation of this rewriting principle is not of interest: When starting from a fixed initial tree and applying a finite set of rewrite rules with constant trees, the resulting trees are of bounded branching and hence can be traced to the case of ground tree rewriting over ranked trees. Another natural approach to handle unbounded branching of unranked trees is to encode unranked trees as binary trees. Using this formalism, we show that there is a class of rewriting systems over unranked trees, which will be called *partial subtree rewriting systems*, that generates the same class of infinite graphs as ground tree rewriting systems over ranked trees.

However, encodings are problematic as they alter locality and path properties. This means that this approach of compensating unbounded branching via a dispersal into subtrees blurs a decisive point, namely the separation of two types of unboundedness: one is derived from the arbitrariness of "hierarchy levels" (represented by the height of the tree) while the other unboundedness refers to the number of data on the same hierarchy level. Pursuing the latter aspect, we define a new class of rewriting systems over unranked trees, the *subtree and flat prefix rewriting systems*, which combine ground tree rewriting with prefix word rewriting on the "flat front" of a tree. Here, a flat front indicates a successor sequence wherein all nodes are leaves. With this approach related to ground tree rewriting, we obtain a class of infinite graphs which has a decidable first-order theory with reachability predicate. Furthermore, analogous to regular ground tree rewriting systems over ranked trees, a regular variant of these rewriting systems is considered.

After introducing the basic terminology in Section 2, the class of transition graphs of partial subtree rewriting systems is treated in Section 3. We show that this class coincides with the class of transition graphs of ground tree rewriting systems over ranked trees. Section 4 introduces (regular) subtree and flat prefix rewriting systems, relates the classes of transition graphs to the previous ones, and investigates the decidability of the reachability problem over the transition graphs of (regular) subtree and flat prefix rewriting systems. Furthermore, it is shown that the structure consisting of the set of unranked trees and the relations reachability and one-step reachability is automatic for a suitable definition over unranked trees and thus has a decidable first-order theory (cf. [1]). Section 5 concludes with a short summary and points to further aspects of interest.

2 Preliminaries

It is assumed that the reader is familiar with the basic notions of automata theory and regular languages (for an introduction cf. [12], for automata on ranked trees cf. [7], and on unranked trees cf. [3]).

An *unranked tree* over an alphabet Σ is a mapping from a nonempty finite domain $dom_t \subseteq \mathbb{N}^*$ to Σ, where dom_t is prefix closed and it holds that if $xi \in dom_t$ then $xj \in dom_t$ for $x \in \mathbb{N}^*$, $i \in \mathbb{N}$, and $j \leq i$. In an unranked tree, each node may have an arbitrary but finite number of successors. If the root of a finite tree t is labeled by $a \in \Sigma$ and has k successors at which the subtrees $t_1, \ldots, t_k$ are rooted, then t can be written as the term $a(t_1, \ldots, t_k)$. The set of unranked trees over an alphabet Σ is denoted by T_Σ.

The *subtree* $t_{\downarrow x}$ of t is the tree rooted at node $x \in dom_t$ (i.e. $t_{\downarrow x}(u) = t(xu)$ for $xu \in dom_t$). The *height* of a tree is defined as $ht(t) := max\{|x| \mid x \in dom_t\}$; if a tree t is of height 1, the word derived from the front (i.e. the sequence of leaves read from left to right) of t is called the *flat front*.

A *hedge* as introduced by Courcelle [9] is a (possibly empty) finite ordered sequence of trees. The width of a hedge is defined as the number of trees that are contained in the sequence; consequently, a tree is a hedge of width 1.

A *nondeterministic bottom up tree automaton (N↑TA)* on unranked trees is of the form $\mathcal{A} = (Q, \Sigma, \Delta, F)$ over an unranked alphabet Σ, with a finite set Q of states, a set $F \subseteq Q$ of final states, and a finite set of transitions $\Delta \subseteq REG(Q) \times \Sigma \times Q$, where $REG(Q)$ denotes the class of regular word languages over Q, which are given for single transitions e.g. by a nondeterministic finite (word) automaton (NFA). A *run* of $\mathcal{A}$ on t is a mapping $\rho : dom_t \to Q$ such that for each node $x \in dom_t$ there is a transition $(L, t(x), \rho(x)) \in \Delta$ such that the sequence $q_1 \cdots q_n$ of states formed by the run at the successors of x is a word in L. Thus, an N↑TA employs NFAs that read the successor sequence of a node, and decide with this word and the label of the current node which state to assign to the current node.

As usual, a run is *accepting* if the root is labeled with a final state, and the accepted language $T(\mathcal{A})$ contains all trees for which there is an accepting run. If there is a run labeling the root with state q then we write $\mathcal{A} : t \to^* q$.

We also use the equivalent model N↓TA as well as an extended model (denoted by ε-N↑TA) with ε-transitions from the set $Q \times Q$, each with the standard semantics.

A tree is called *ranked* if every symbol $a \in \Sigma$ is assigned a unique arity $rk(a) \in \mathbb{N}$, and each node labeled with a has exactly $rk(a)$ successors.

A *ground tree rewriting system (GTRS)* over ranked trees is defined as a tuple $\mathcal{R} = (\Sigma, \Gamma, R, t_{in})$, with ranked alphabet Σ, transition alphabet Γ, finite set R of rules of the form $s \overset{\sigma}{\hookrightarrow} s'$ with $s, s' \in T_\Sigma$, $\sigma \in \Gamma$, and initial tree $t_{in} \in T_\Sigma$. A rule $s \overset{\sigma}{\hookrightarrow} s' \in R$ is applicable to a tree t if there is a node $x \in dom_t$ with $s = t_{\downarrow x}$, and the resulting tree is $t' = t[x|s']$, where the subtree rooted at node x is replaced by the tree s'. In this case, t' is *derived* from t by the rule $s \overset{\sigma}{\hookrightarrow} s'$ and we write $t \rightarrow^\sigma_\mathcal{R} t'$. The tree language that is generated by $\mathcal{R}$ is denoted $T(\mathcal{R}) = \{t \in T_\Sigma \mid t_{in} \rightarrow^*_\mathcal{R} t\}$; the focus of this paper will be the structure induced by the rewriting system with respect to the tree language. This is a directed edge labeled *transition graph* $G_\mathcal{R} = (V_\mathcal{R}, E_\mathcal{R}, \Gamma)$, of $\mathcal{R}$ with $V_\mathcal{R} = T(\mathcal{R})$, and $(t, \sigma, t') \in E_\mathcal{R}$ iff $t \rightarrow^\sigma_\mathcal{R} t'$. Note that the vertex set $V_\mathcal{R}$ is defined as the set of trees that are reachable from t_{in} by repeated application of the rewrite rules. The class of transition graphs of GTRSs is denoted by GTRG. For an extensive survey on GTRG cf. [15].

One way of dealing with unranked trees is to encode them by ranked trees. We use here a formalism proposed in [18], and employed as an encoding in [4], that uses only one binary symbol corresponding to an operation for constructing unranked trees. The *extension operator* $@ : T_\Sigma \times T_\Sigma \rightarrow T_\Sigma$ extends a given tree t by t' by adjoining t' as the next sibling of the last child of t: $a(t_1, \dots, t_n) @ t' = a(t_1, \dots, t_n, t')$, respectively for case $n = 0:\ \ a @ t' = a(t')$. Furthermore, we can also adjoin a hedge instead of a single tree t' in the intuitive way. Note that every unranked tree can be generated uniquely from trees of height 0 using the extension operator: $a(t_1, \dots, t_n) = [(\cdots (a @ t_1) @ t_2) \cdots @ t_n]$, and thus, this formalism can be used as an encoding of unranked trees into binary ones (by assigning rank 0 to each symbol of the unranked alphabet and rank 2 to the extension operator @).

3 Partial Subtree Rewriting Systems

As mentioned in the Introduction, the direct transfer of the ground tree rewriting principle to unranked trees would result in bounded branching, therefore new rewriting principles have to be considered. The first rewriting principle considered aims at an easy transfer of nice properties of GTRSs. Therefore, unranked trees are encoded into ranked ones via the extension operator encoding as introduced in Section 2. Subtrees of the tree obtained after the encoding are hereby mapped to *partial subtrees* in the corresponding unranked tree; where if $a(t_1, \dots, t_n)$ is a subtree, then $a(t_1, \dots, t_i)$ is a partial subtree for each $0 \le i \le n$. The rewriting system therefore is defined such that exactly those partial subtrees are replaced.

The set $T_{\Sigma,\xi}$ is the set of all unranked trees over Σ with one occurrence of the variable ξ as leaf and rightmost child of the root, i.e. the set of trees of the form $\bar{t} @ \xi$ with $\bar{t} \in T_\Sigma$.

A *partial subtree rewriting system (PSRS)* over unranked trees in T_Σ is of the form $\mathcal{R} = (\Sigma, \Gamma, R, t_{in})$, with an unranked alphabet Σ, a transition alphabet Γ, a finite set of rules R, and an initial tree t_{in}. The set R consists of subtree rewrite rules over trees of $T_{\Sigma,\xi}$ of the form: $r \overset{\sigma}{\hookrightarrow} r'$ with $r, r' \in T_{\Sigma,\xi}$ and $\sigma \in \Gamma$.

A tree t' is derived from t ($t \rightarrow^{\sigma}_{\mathcal{R}} t'$), if there is a node $x \in dom_t$, a hedge h over Σ, and a rule $r \overset{\sigma}{\hookrightarrow} r' \in R$, such that $r[\xi|h] = t_{\downarrow x}$, and $t[x|r'[\xi|h]] = t'$ (cf. Figure 1). The class of transition graphs of PSRSs is denoted by PSRG.

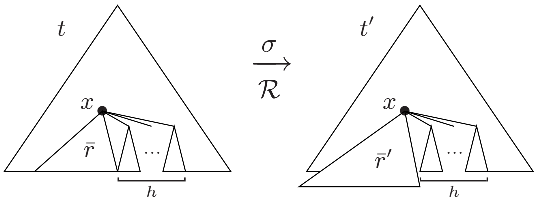


Fig. 1. Application of rewrite rule $r \overset{\sigma}{\hookrightarrow} r'$ according to the definition of PSRSs.

With these definitions it can be shown that PSRSs over unranked trees and GTRSs over ranked trees generate the same transition graphs up to isomorphism.

Theorem 1. *Partial subtree rewriting systems generate the same class of transition graphs as ground tree rewriting systems (PSRG = GTRG).*

Proof (Sketch). When unranked trees are encoded into ranked ones by the extension operator encoding, subtrees of the ranked encoding correspond exactly to the partial trees of $T_{\Sigma,\xi}$ by construction. Thus applying a rule of a PSRS corresponds to rewriting an entire subtree in the ranked tree obtained after the encoding. The technical details of the construction of a GTRS for a given PSRS can be found in [17].

Since ranked trees can be viewed as unranked trees, and since each symbol has a unique rank, the construction of a PSRS $\mathcal{R}$ for a GTRS $\mathcal{S} = (\Sigma_r, \Gamma, S, t_{in})$ over ranked alphabet Σ_r is straightforward. The ranks of the symbols are simply omitted, the initial tree is kept, and the given rules of the GTRS are endorsed by extending the trees in the rules with the variable ξ to obtain trees in $T_{\Sigma,\xi}$. Consequently, with the same initial tree for both rewriting systems, the variable ξ can only be substituted by the empty hedge, thus resulting in isomorphic transition graphs. $\square$

This class equivalence of GTRG and PSRG induces that by disregarding the inner structure of the vertices (unranked vs. ranked trees), the transition graphs are of identical structure.

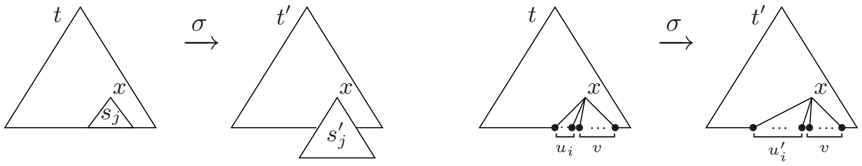
Corollary 2. *The first-order theory with reachability is decidable for PSRG.*

Additionally, several other decidability and undecidability results for GTRG can be transferred to PSRG (cf. [2, 15]).

4 Subtree and Flat Prefix Rewriting Systems

Previously, unbounded branching was coped with via a dispersal into the unboundedness of depth of a tree. In order to respect the nature of these two different types of unboundedness, we now consider a new rewriting formalism. Towards a compromise between known principles and meeting this requirement, we combine standard subtree substitution with flat prefix substitution. That means, for subtrees of height 1, a prefix of the successor sequence of this subtree can be replaced by another sequence, enabling us to exploit properties of prefix rewriting over words.

Note that these prefix rewrite rules can be regarded as a kind of synchronization: they can only be applied at a node x if all subtrees rooted at its successors have a certain property, namely are of height 0. This kind of structural control is not available for the previously considered rewriting systems, and thus yields a new class of transition graphs.



(a) Subtree substitution. (b) Flat prefix substitution.

Fig. 2. Application of rewrite rules of according to the definition of SFPRSs.

A *subtree and flat prefix rewriting system (SFPRS)* over unranked trees in T_Σ is of the form $\mathcal{R} = (\Sigma, \Gamma, R, t_{in})$, with a finite unranked alphabet Σ, a finite transition alphabet Γ, an initial tree t_{in}, and a finite set R of rules of two types:

1. subtree substitution (cf. Figure 2(a)) with rules of the form $r_j : s_j \overset{\sigma}{\hookrightarrow} s'_j$ for $j \in J$, $s_j, s'_j \in T_\Sigma$, $\sigma \in \Gamma$, and
2. flat prefix substitution at the flat front of the tree (cf. Figure 2(b)) with rules of the form $r_i : u_i \overset{\sigma}{\hookrightarrow} u'_i$ for $i \in I$, $u_i, u'_i \in \Sigma^+$, $\sigma \in \Gamma$,

with $I \cup J = \{1, \ldots, |R|\}$ and $I \cap J = \emptyset$. The class of transition graphs of SFPRSs is denoted by SFPRG.

A tree t' is derived from t ($t \to^\sigma_\mathcal{R} t'$) by applying a subtree rewrite rule r_j, if there is a node $x \in dom_t$ with $t_{\downarrow x} = s_j$ such that $t[x|s'_j] = t'$ (cf. Figure 2(a)).

A tree t' is derived from t by applying a prefix rewrite rule r_i, if there is a node $x \in dom_t$ with $ht(t_{\downarrow x}) = 1$ and $\mathit{flatfront}(t_{\downarrow x}) = u_i v$, a tree $s \in T_\Sigma$ with $ht(s) = 1$ and $s(\varepsilon) = t(x)$, $\mathit{flatfront}(s) = u_i' v$, such that $t[x|s] = t'$ for some $v \in \Sigma^*$ (cf. Figure 2(b)).

Naturally, the definition of SFPRSs can be extended to *regular* SFPRSs by introducing regular sets of trees resp. words in the rules to obtain an even larger class of transition graphs. Conversely, SFPRSs can be regarded as the special case of singletons in the rules of regular SFPRSs. Note that all negative results in this paper are shown for SFPRSs while correspondingly, all positive results are shown for regular SFPRSs and thus hold for both classes of transition graphs.

4.1 Classification of Transition Graph Classes

Towards a classification of the transition graph classes PSRG and (regular) SF-PRG, consider the SFPRS $\mathcal{R}_0 = (\Sigma, \Gamma, R, t_{in})$ with $\Sigma = \{a, c, e\}$, $\Gamma = \{0, 1\}$, $R = \{r_1 : c \xhookrightarrow{1} \begin{array}{c} e \\ | \\ e \end{array}, r_2 : e \xhookrightarrow{1} \begin{array}{c} e \\ | \\ e \end{array}, r_3 : c \xhookrightarrow{0} cc\}$ $(I = \{3\}, J = \{1, 2\})$, and $t_{in} = \begin{array}{c} a \\ / \ \backslash \\ c \quad c \end{array}$, whose transition graph is depicted in Figure 3.

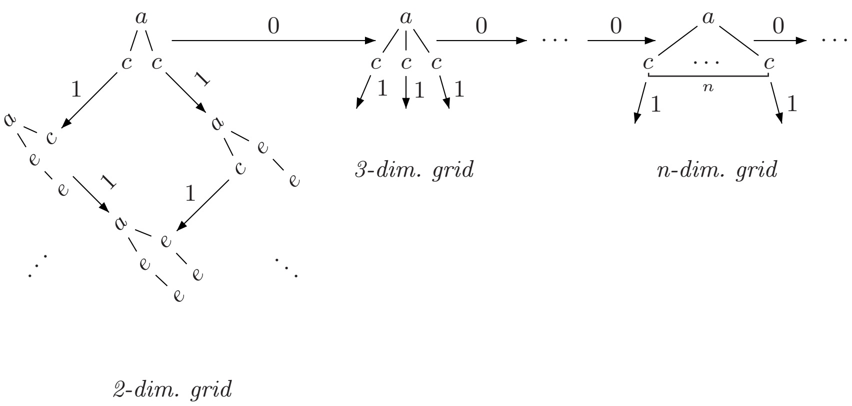


Fig. 3. Transition graph of SFPRS $\mathcal{R}_0$.

Note that the 0-transition r_3 can only be applied at the trees of the vertices on the top line in Figure 3, since these are the only trees that have a subtree of height 1 with flat front cw for $w \in \Sigma^+$. For the transition graph this means that after traversing a 1-edge, no 0-edges are available any more.

Lemma 3. *The transition graph of SFPRS $\mathcal{R}_0$ cannot be generated by a GTRS.*

Proof (Sketch). It can be shown that using a GTRS, an enabled 0-transition cannot be disabled by an aritrary number of 1-transitions leading to different nodes.

Towards a contradiction: consider a vertex of the top row of Figure 3 with n out edges with label 1 and one out edge with label 0. For the tree at this vertex in a corresponding transition graph of a GTRS $\mathcal{S}$, there have to be n different 1-transitions which rewrite the subtree available for the applicable 0-transition in order to prevent a 0-transition afterwards. However, the number of nodes in the tree where these 1-transitions have to be applied in order to fulfill this requirement is bounded by the number of rewrite rules of $\mathcal{S}$ and the height of the trees of the left hand sides of the rewrite rules of $\mathcal{S}$. For n large enough this is a contradiction (for details, we refer the reader to [17]). □

Note that Lemma 3 is already true if we omit rule r_2 from SFPRS $\mathcal{R}_0$. Conversely, every ground tree rewriting system can always be conceived as a SFPRS with subtree rewrite rules only. With the same initial tree and the same subtree rewrite rules, omitting the ranks of the symbols does not provide more substitution possibilities. Since the classes of transition graphs GTRG and PSRG are equivalent, one obtains the following.

Proposition 4. *The class of transition graphs of PSRSs is strictly included in the class of transition graphs of SFPRSs: PSRG $\subsetneq$ SFPRG.*

Thus, undecidability results for PSRSs carry over to (regular) SFPRSs. These include the reachability problems: constrained reachability, universal reachability, and universal recurrence (cf. [15]).

Additionally, since this is the case for ground tree rewriting systems, the monadic second-order logic of SFPRSs is undecidable. This can also be observed directly from the transition graph of $\mathcal{R}_0$, since it includes the two-dimensional grid, whose monadic second-order logic is undecidable (as proven e.g. in [16]).

We would like to point out that the increase of expressive power of SFPRSs over PSRSs results from the fact that prefix rewrite rules can only be applied to flat fronts. Due to this restriction it is not possible to transfer these rules to standard rewriting rules over encodings.

4.2 Reachability via Saturation

The main contribution of this paper is the decidability of the reachability problem for transition graphs of (regular) SFPRSs. This is done by an adaption of the well-known saturation algorithm which e.g. solves the reachability problem for semi-monadic linear rewriting systems over ranked trees (cf. [8]) by calculating the set $\mathrm{pre}^*_{\mathcal{R}}(T) = \{t \in T_\Sigma \mid \exists t' \in T : t \to^*_{\mathcal{R}} t'\}$ of trees from which the set T can be reached. Thereby, the rewrite rules of a (regular) SFPRS are simulated by adding transitions to an ε-N↑TA that recognizes the union of the target set T and all trees that correspond to a left hand side of the rewrite rules similar to the construction in [15]. In the very same manner, the set $\mathrm{post}^*_{\mathcal{R}}(T) = \{t \in T_\Sigma \mid \exists t' \in T : t' \to^*_{\mathcal{R}} t\}$ of trees which are reachable from the set T can be obtained by pursuing the same strategy for the reversed rewriting system, i.e. the left and right hand sides of the rules are simply swapped.

However, due to the different natures of the two types of rules of (regular) SFPRSs, and the employment of word automata in ε-N↑TAs over unranked trees, the saturation is based on an interleaving of two saturation algorithms on different levels of automata. In detail, for a prefix rewrite rule the saturation is basically realized by adding ε-transitions on the level of word automata that recognize the sequence of labels of the successors of a node, while for subtree rewrite rules, the saturation is realized by adding ε-transitions on the level of tree automata. The crucial interleaving aspects include that by the application of subtree rewrite rules new flat fronts may be introduced, which also need to be saturable.

For an elaborate example, the full construction, and the formal correctness proof, we refer the reader to [17]. The automaton resulting from the saturation accepts exactly those trees from which the target set is reachable and thus we obtain the following theorem.

Theorem 5. *Given a (regular) SFPRS $\mathcal{R}$, and a regular set T of unranked trees, the sets $pre^*_{\mathcal{R}}(T)$ and $post^*_{\mathcal{R}}(T)$ are again regular.*

As emptiness for unranked tree automata is decidable, we obtain the following corollary.

Corollary 6. *The reachability problem for (regular) SFPRSs: "Given a (regular) SFPRS $\mathcal{R}$, vertex t, and regular set T of vertices, is there a path from t to a vertex in T?" is decidable.*

4.3 First-Order Theory via Automatic Structures

In addition to the decidability of the reachability problem for (regular) SFPRSs, we now address the first-order theory for these rewriting systems. We will show that the first-order theory enriched with the predicates reachability and one-step reachability remains decidable and thus obtain a proper superclass of (regular) GTRG with the same decidability properties.

We show that the structure consisting of the universe T_Σ, the one-step reachability relation, and the reachability relation is tree-automatic for a suitable definition over unranked trees, thus exploiting the feature that any (tree-) automatic structure has a decidable first-order theory (cf. [1]). Due to space restrictions, we stick to an informal description of the automaton that works on the convolution of two trees.

Briefly, the convolution $t = \langle t_1, t_2 \rangle$ encodes two trees $t_1, t_2 \in T_\Sigma$ such that the automaton reading the new tree t has access to both original ones. This is realized by labeling the node of t with pairs of symbols from t_1 and t_2 such that the successor sequences of the nodes line up on the right and are padded with a filling symbol $\Box$ on the left where necessary. For example, the trees $t_1 = a(bc)$ and $t_2 = d(efg)$ are convolved into $t = \langle t_1, t_2 \rangle = [a, d]([\Box, e], [b, f], [c, g])$.

In the construction of the automaton recognizing the reachability relation $\rightarrow^*$, we embark on a strategy similar to one for pushdown systems (cf. [5]): Given two trees $t_1, t_2 \in T_\Sigma$, we guess the set of "minimal" points of the rewriting steps in $t_1 \rightarrow_{\mathcal{R}}^* t_2$ and then check whether the first component of the convolution can be rewritten into the left side of the applied rule while the second component can be rewritten from the right side of the rule.

For (regular) SFPRSs this means that we start with an automaton $\mathcal{B}$ working on the convolution $t = \langle t_1, t_2 \rangle$ of two trees, which recognizes the identity function, i.e. the set $\{\langle t_1, t_2 \rangle \mid t_1 = t_2\}$. The automaton then nondeterministically guesses the set of minimal (w.r.t. the prefix ordering) nodes at which a rewrite rule was applied. Furthermore, $\mathcal{B}$ also guesses which rule was applied for each of these nodes.

For a subtree rewrite rule, $\mathcal{B}$ checks whether the projections of the subtree $t_{\downarrow x}$ belong to the regular sets pre^* resp. $post^*$ of the applied rule. Theorem 5 yields automata $\mathcal{A}_{pre^*}$, $\mathcal{A}_{post^*}$ over Σ for these sets, and with a straightforward automaton construction, we can add transitions to $\mathcal{B}$ such that if the projections of $t_{\downarrow x}$ to the components belong to the corresponding sets, $\mathcal{B}$ accepts subtree $t_{\downarrow x}$ of the convolution. A similar but slightly more involved strategy works for flat prefix rewrite rules.

The construction of an automaton for the one-step reachability relation $\rightarrow$ is straightforward. The automaton works in a similar way but ensures that exactly one rule was applied.

Since both relations are automatic, we obtain the following theorem.

Theorem 7. *The first-order theory enriched by the relations reachability and one-step reachability is decidable for (regular) SFPRSs.*

5 Summary and Outlook

We showed that using rewriting systems over unranked trees one can generate a class of infinite graphs that coincides with the class of transition graphs of ground tree rewriting systems over ranked trees. The rewriting principle of these PSRSs consists of substituting unranked trees partially, which corresponds to ground tree rewriting over an encoding of unranked trees as ranked ones. Due to the class equivalence, several decidability results over the transition graphs of GTRSs over ranked trees can be transferred to those of PSRSs.

Furthermore, (regular) SFPRSs over unranked trees were introduced, which add flat prefix rewriting to the known paradigm of subtree substitution. The class of transition graphs of SFPRSs was shown to strictly include the class of transition graphs of PSRSs, which allows to transfer several undecidability results. Additionally, we described a saturation algorithm which yields the decidability of the reachability problem over (regular) SFPRG. We have also shown that the structure consisting of the set T_Σ of unranked trees and the relations reachability and one-step reachability is automatic for a suitable definition over unranked trees, and thus we can conclude that the first-order theory with these reachability relations is decidable for (regular) SFPRSs. Thus, the class of (regular) SFPRSs contains strictly more graphs than GTRG, but has the same decidable properties.

In general, other rewriting principles over unranked trees have yet to be investigated. One aspect could be to use other word rewriting techniques in combination with subtree substitution. Another interesting point of application is to define and investigate an adaption of (semi) monadic rewriting systems to unranked trees, which were introduced for ranked trees in [8].

Finally, we would like to thank Arnaud Carayol for his useful comments on the decidability proof for the first-order theory.

References

[1] A. Blumensath. *Automatic Structures*. Diploma thesis, RWTH Aachen, Germany, 1999. `http://www-mgi.informatik.rwth-aachen.de/Publications/pub/blume/AutStr.ps.gz`.

[2] W. Brainerd. Tree generating regular systems. *Inf. and Contr.*, 14(2):217–231, 1969.

[3] A. Brüggemann-Klein, M. Murata, and D. Wood. Regular tree and regular hedge languages over unranked alphabets. Unfinished technical report, Hongkong University, April 2001. `http://citeseer.ist.psu.edu/451005.html`.

[4] J. Carme, J. Nieren, and M. Tommasi. Querying unranked trees with stepwise tree automata. In *Proc. RTA 2004*, volume 3091 of *LNCS*, pages 105–118. Springer, 2004.

[5] D. Caucal. On the regular structure of prefix rewriting. *TCS*, 106(1):61–86, 1992.

[6] D. Caucal. On infinite terms having a decidable theory. In *Proc. MFCS*, volume 2420 of *LNCS*, pages 165–176. Springer, 2002.

[7] H. Comon, M. Dauchet, R. Gilleron, F. Jacquemard, D. Lugiez, S. Tison, and M. Tommasi. *Tree Automata Techniques and Applications*. Unpublished electronic book, 1997. `http://www.grappa.univ-lille3.fr/tata`.

[8] J.-L. Coquidé, M. Dauchet, R. Gilleron, and S. Vágvölgyi. Bottom-up tree pushdown automata: classification and connection with rewrite systems. *TCS*, 127(1):69–98, 1994.

[9] B. Courcelle. A representation of trees by languages. *TCS*, 7:25–55, 1978.

[10] M. Dauchet and S. Tison. The theory of ground rewrite systems is decidable. In *Proc. LICS 1990*, pages 242–248. IEEE CSP, 1990.

[11] E. A. Emerson. Temporal and modal logic. In J. van Leeuwen, editor, *Handbook of TCS*, volume B, pages 995–1072. Elsevier, 1990.

[12] J. Hopcroft, R. Motwani, and J. Ullman. *Introduction to Automata Theory, Languages, and Computation*. Addison-Wesley, Boston, 2 edition, 2001.

[13] C. Löding. Ground tree rewriting graphs of bounded tree width. In *Proc. STACS 2002*, volume 2285 of *LNCS*, pages 559–570. Springer, 2002.

[14] C. Löding. Model-checking infinite systems generated by ground tree rewriting. In *Proc. FoSSaCS 2002*, volume 2303 of *LNCS*, pages 280–294. Springer, 2002.

[15] C. Löding. *Infinite Graphs Generated by Tree Rewriting*. PhD thesis, RWTH Aachen, Germany, 2003.

[16] D. Seese. Entscheidbarkeits- und Definierbarkeitsfragen der Theorie ,,netzartiger" Graphen-I. *Wiss. Zeitschrift HU Berlin*, XXI(5):513–517, 1972.

[17] A. Spelten. *Rewriting Systems over Unranked Trees*. Diploma thesis, RWTH Aachen, Germany, 2006. `http://www-i7.informatik.rwth-aachen.de/download/papers/spelten/sp06.pdf`.

[18] M. Takahashi. Generalizations of regular sets and their application to a study of context-free languages. *Inf. and Contr.*, 27(1):1–36, 1975.

[19] W. Thomas. A short introduction to infinite automata. In *Proc. DLT 2001*, volume 2295 of *LNCS*, pages 130–144. Springer, 2002.

Appendix

A Proof of Theorem 7

The intuitive way to define convolution for unranked trees is to align successor sequences on the left and insert filling symbols on the right where appropriate: let $\Sigma_{\Box}^{n} = (\Sigma \cup \{\Box\})^n \setminus \{\Box^n\}$ and $t^{\Box}(x) := \begin{cases} t(x) & \text{if } x \in dom_t, \\ \Box & \text{otherwise.} \end{cases}$

For $t_1, \ldots, t_n \in T_\Sigma$ define the left convolution as $t = \langle t_1, \ldots, t_n \rangle_{left} \in T_{\Sigma_{\Box}^{n}}$ with $dom_t = dom_{t_1} \cup \cdots \cup dom_{t_n}$, and $t(x) = (t_1^{\Box}(x), \ldots, t_n^{\Box}(x))$.

For prefix rewriting though, it is more convenient to define the convolution such that successor sequences line up on the right, while filling symbols appear on the left. This can be derived from the left convolution with the function *flip* defined by $\mathit{flip}(a(t_1, \ldots, t_m)) := a(\mathit{flip}(t_m), \ldots, \mathit{flip}(t_1))$. Thus, the right convolution is defined as $t = \langle t_1, \ldots, t_n \rangle_{right} = \mathit{flip}(\langle \mathit{flip}(t_1), \ldots, \mathit{flip}(t_n) \rangle_{left})$ with $dom_t = dom_{t_1} \cup \cdots \cup dom_{t_n}$ (cf. Figure 4). In the following, if a convolution $\langle t_1, \ldots, t_n \rangle$ is denoted without a subscript "left" or "right", we are always referring to the right convolution $\langle t_1, \ldots, t_n \rangle_{right}$.

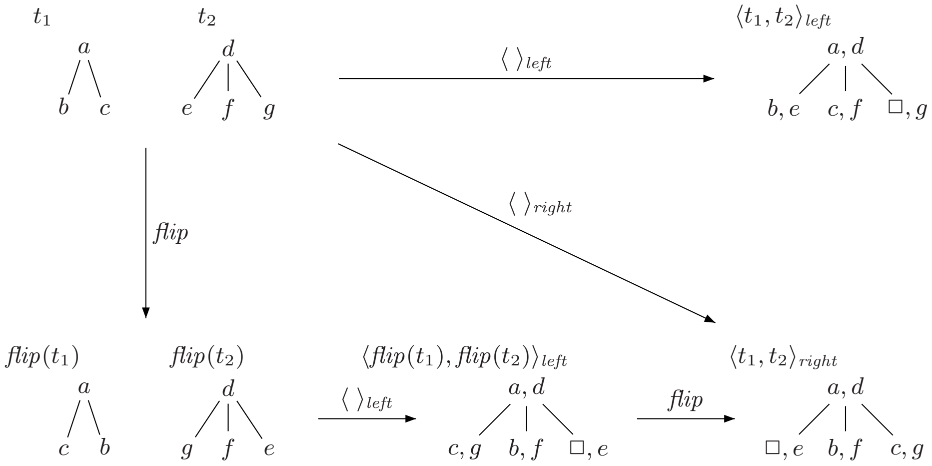


Fig. 4. The left and right convolution of the trees $t_1 = a(bc)$ and $t_2 = d(efg)$.

The i-th *projection* π_i for $1 \le i \le n$ yields the i-th entry of the convolution, i.e. the tree t_i. Thereby, the $\Box$es are dropped. For $t = \langle t_1, \ldots, t_n \rangle$ that is: $\pi_i(t) = t_i$; $\pi_i(t(x)) = \begin{cases} t_i^{\Box}(x) & \text{if } t_i(x) \neq \Box \\ \text{undefined} & \text{otherwise} \end{cases}$, i.e. $dom_{\pi_i(t)} = \{x \mid x \in dom_t \wedge \pi_i(t(x)) \in \Sigma\} = dom_{t_i}$.

For an induction on the construction of first-order formulae, we need several automata constructions. Boolean operations such as complementation, union, and projection for unranked tree automata can be obtained by a straightforward adaption from the ranked case (as it can be found e.g. in [7], Chapter 3).

The crucial construction is needed for the atomic formulae, i.e. the reachability relation and the one-step reachability relation. Here, two additional constructions are needed, which will be defined for automata over $\Sigma^2_\square$.

With two N↓TAs $\mathcal{A}_1, \mathcal{A}_2$ over Σ, we construct the N↓TA $\mathcal{A} = \mathcal{A}_1 \otimes \mathcal{A}_2$ over $\Sigma^2_\square$ such that $\mathcal{A}$ checks whether the i-th projection of a tree $t = \langle t_1, t_2 \rangle \in T_{\Sigma^2_\square}$ belongs to the regular set $T(\mathcal{A}_i)$ for $1 \leq i \leq 2$. Thus, with $\mathcal{A}_i = (Q_i, \Sigma, \Delta_i, Q_0^i)$, we construct $\mathcal{A} = (Q_1 \times Q_2, \Sigma^2_\square, \Delta, Q_0^1 \times Q_0^2)$ with

$$\begin{aligned}\Delta =& \{(\{\square\}^* \cdot L_{a_1q_1} \times \{\square\}^* \cdot L_{a_2q_2}, (a_1, a_2), (q_1, q_2)) \mid (L_{a_iq_i}, a_i, q_i) \in \Delta_i, i = 1, 2\} \\ &\cup \{(\{\square\}^* \times L_{a_2q_2}, (\square, a_2), (q_1, q_2)) \mid (L_{a_2q_2}, a_2, q_2) \in \Delta_2, q_1 \in Q_1\} \\ &\cup \{(L_{a_1q_1} \times \{\square\}^*, (a_1, \square), (q_1, q_2)) \mid (L_{a_1q_1}, a_1, q_1) \in \Delta_1, q_2 \in Q_2\}.\end{aligned}$$

It is not difficult to see that the words of the horizontal automata are padded with the $\square$ symbol when necessary such that $t \in T(\mathcal{A})$ iff $\pi_i(t) = t_i \in T(\mathcal{A}_i)$ for $1 \leq i \leq 2$.

The second construction checks for a tree t whether the partial tree ("left" side of tree including the root) belongs to a regular set, while each tree of the remaining hedge (where each tree of the hedge is rooted one level below the original root) belongs to a different regular set (cf. Figure 5). With N↓TA $\mathcal{A} = (Q_\mathcal{A}, \Sigma, \Delta_\mathcal{A}, Q_0^\mathcal{A})$ and N↓TA $\mathcal{B} = (Q_\mathcal{B}, \Sigma, \Delta_\mathcal{B}, Q_0^\mathcal{B})$, we construct a new N↓TA $\mathcal{A}@\mathcal{B}^\star = (Q, \Sigma, \Delta, Q_0)$ with

- $Q := Q_\mathcal{A} \cup Q_\mathcal{B} \cup \{q_0\}$
- $\Delta := \{(L_{aq} \cdot (Q_0^\mathcal{B})^*, a, q_0) \mid (L_{aq}, a, q) \in \Delta_\mathcal{A}, q \in Q_0^\mathcal{A}\} \cup \Delta_\mathcal{A} \cup \Delta_\mathcal{B}$
- $Q_0 := \{q_0\}$

We use $\star$ in the name of this construction to clearly separate it from $*$, making complex expressions involving this construction more readable.

It holds that $t \in T(\mathcal{A}@\mathcal{B}^\star)$ iff $t \in T(\mathcal{A})@\underbrace{T(\mathcal{B})@\cdots@T(\mathcal{B})}_{n}$ for some $n \geq 0$, i.e. if the root has $m + n$ successors, the "left part" of the tree consisting of the root and the subtrees $t_{\downarrow 1}, \ldots, t_{\downarrow m}$ belong to $T(\mathcal{A})$, while each subtree $t_{\downarrow m+i} \in T(\mathcal{B})$ for $1 \leq i \leq n$ (cf. Figure 5).

We construct N↓TA $\mathcal{B}_{id}$ over $\Sigma^2_\square$ recognizing the identity function as follows: $\mathcal{B}_{id} = (Q, \Sigma^2_\square, \Delta, Q_0)$ with $\Delta = \{(\{\varepsilon\}, (a, a), q_0) \mid a \in \Sigma\} \cup \{(L(\mathcal{C}_{id}), (a, a), q_0) \mid a \in \Sigma\}$ and horizontal NFA $\mathcal{C}_{id} = (Q_{id}, \Sigma^2, q_{id}, \Delta_{id}, F_{id})$, where $Q_{id} = F_{id} = \{q_{id}\}$ and $\Delta_{id} = \{(q_{id}, (a, a), q_{id}) \mid a \in \Sigma\}$.

Reachability Relation $\rightarrow^*$

This proof is provided for a *regular* SFPRS $\mathcal{R} = (\Sigma, \Gamma, R, t_{in})$; i.e. both types of rewrite rules of R are given over regular sets:

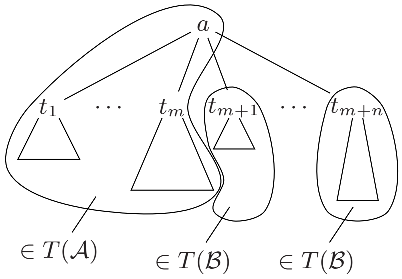


Fig. 5. A tree $t \in T(\mathcal{A}@\mathcal{B}^\star)$.

- subtree rewrite rules of the form $j : S_j \overset{\sigma}{\hookrightarrow} S'_j$ with $S_j, S'_j \subseteq T_\Sigma$ regular sets of trees, and
- flat prefix rewrite rules of the form $i : L_i \overset{\sigma}{\hookrightarrow} L'_i$ with $L_i, L'_i \subseteq \Sigma^*$ regular sets of words.

Thus for the construction of the N↓TA $\mathcal{B}$ over $\Sigma^2_\square$ that recognizes the reachability relation, we employ several automata:

- For each letter $a \in \Sigma$, define two N↓TAs with $T(\mathcal{A}^a_{pre^*}) = pre^*(\{a\})$ and $T(\mathcal{A}^a_{post^*}) = post^*(\{a\})$ (i.e. the sets *pre** resp. *post** of trees of height 0).
- For each subtree rewrite rule $j : S_j \overset{\sigma}{\hookrightarrow} S'_j$, define two N↓TAs with $T(\mathcal{A}^j_{pre^*}) = pre^*(S_j)$ and $T(\mathcal{A}^j_{post^*}) = post^*(S'_j)$ using the constructions described in Section 4.2. The detailed construction can be found in [17].
- Since more than one flat prefix rewrite rule can be applied on the same level, yielding an arbitrary number of successors, we need to calculate the sets *pre** and *post** of a hedge (consisting of the successor sequence) rather than calculating them of a subtree of height 1 where the rule would be applied. In order to trace this back to the known case over trees, we simply add an artificial root labeled by a new symbol $\Diamond$ to the hedge such that the root cannot be rewritten with the present rules.
Therefore, for each flat prefix rewrite rule $i : L_i \overset{\sigma}{\hookrightarrow} L'_i$, define two N↓TAs over $\Sigma \cup \{\Diamond\}$ with $T(\mathcal{A}^{i\Diamond}_{pre^*}) = pre^*(\{\Diamond(w_1, \ldots, w_{|w|}) \mid w \in L_i\})$ and $T(\mathcal{A}^{i\Diamond}_{post^*}) = post^*(\{\Diamond(w'_1, \ldots, w'_{|w'|}) \mid w' \in L'_i\})$.
After calculating these sets, we need to be able to reintegrate our results in the automata recognizing the convolution of two trees, thus we define the *renaming* $\mathcal{A}[\Diamond|b]$ for $b \in \Sigma$ and an N↓TA $\mathcal{A}$ over $\Sigma_\square \cup \{\Diamond\}$ by assigning a for every occurrence of $\Diamond$ in $\mathcal{A}$. This results in an N↓TA $\mathcal{A}'$ over $\Sigma_\square$ which can again be used in the above mentioned automata constructions to obtain tree automata working on convolutions.
A sample derivation is depicted in Figure 6. Here, we consider the projections of a subtree $t_{\downarrow x} \in T_{\Sigma^2_\square}$ when flat prefix rewrite rule $i : L_i \overset{\sigma}{\hookrightarrow} L'_i$ was applied at node x with $\pi_1(t(x)) = \pi_2(t(x)) = b$. Disregarding automata $\mathcal{F}$ and $\mathcal{H}$

which will be described in detail later on, automata $\mathcal{E}$ and $\mathcal{G}$ are of the mentioned form to recognize a prefix rewriting step: $\mathcal{E} = \mathcal{A}^{i\diamond}_{pre^*}[\diamond|b]$ and $\mathcal{G} = \mathcal{A}^{i\diamond}_{post^*}[\diamond|b]$. Thus, with the described $\otimes$ operator, these automata can be combined to recognize the convolution of $\pi_1(t_{\downarrow x})$ and $\pi_2(t_{\downarrow x})$, that is, $t_{\downarrow x} \in T_{\Sigma^2_\Box}$.

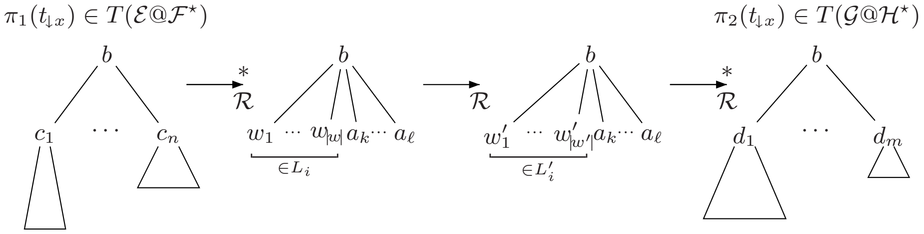


Fig. 6. The projections of subtree $t_{\downarrow x} \in T_{\Sigma^2_\Box}$ when flat prefix rewrite rule $i : L_i \overset{\sigma}{\hookrightarrow} L'_i$ was applied at minimal point x.

With these preliminary remarks, we now describe the automaton $\mathcal{B}$ over $\Sigma^2_\Box$ as an extension of $\mathcal{B}_{id}$:

N↓TA $\mathcal{B}$ nondeterministically guesses for each path from the root to the leaves of $t \in T_{\Sigma^2_\Box}$ the upmost node x on which a rewrite rule was applied, and guesses the applied rewrite rule.

Case A. The applied rewrite rule was subtree rewrite rule j: $S_j \overset{\sigma}{\hookrightarrow} S'_j$.
Then $\mathcal{B}$ works like $(\mathcal{A}^j_{pre^*} \otimes \mathcal{A}^j_{post^*})$ on $t_{\downarrow x}$.

Case B. The applied rewrite rule was flat prefix rewrite rule i: $L_i \overset{\sigma}{\hookrightarrow} L'_i$.
With $\pi_1(t(x)) = \pi_2(t(x)) = b$, $\mathcal{B}$ works like $\mathcal{C} @ \mathcal{D}^\star$ on $t_{\downarrow x}$, where
$\mathcal{C} = (\mathcal{A}^{i\diamond}_{pre^*}[\diamond|b] \otimes \mathcal{A}^{i\diamond}_{post^*}[\diamond|b])$ recognizing the partial subtree and
$\mathcal{D} = \big(\biguplus_{a\in\Sigma}(\mathcal{A}^a_{pre^*} \otimes \mathcal{A}^a_{post^*})\big)$ recognizing each tree of the remaining hedge.

Proof of Correctness Claim: For the automaton $\mathcal{B}$ described above and $t \in T_{\Sigma^2_\Box}$ it holds that

$$t \in T(\mathcal{B}) \Leftrightarrow \pi_1(t) \xrightarrow[\mathcal{R}]{*} \pi_2(t).$$

$\Rightarrow$ Let $t \in T(\mathcal{B})$.
That means there is an accepting run ρ of $\mathcal{B}$ on $t = \langle t_1, t_2\rangle$.
Three cases are to be distinguished:

Case 0. No transition from *Case* A nor *Case* B were traversed in ρ.
Thus it holds that $t_1 = t_2$ and therefore $t_1 \rightarrow^*_{\mathcal{R}} t_2$.

Case 1. Transitions from *Case* A were traversed in ρ on $t_{\downarrow x}$.
This means that $t_{\downarrow x} \in T(\mathcal{A}^j_{pre^*} \otimes \mathcal{A}^j_{post^*})$ for a subtree rewrite rule j: $S_j \overset{\sigma}{\hookrightarrow} S'_j$, and therefore it holds that $\pi_1(t_{\downarrow x}) \in \mathit{pre}^*(S_j)$ and $\pi_2(t_{\downarrow x}) \in \mathit{post}^*(S'_j)$. Thus, there are trees $s \in S_j$, $s' \in S'_j$ with $\pi_1(t_{\downarrow x}) \rightarrow^*_{\mathcal{R}} s$ and $s' \rightarrow^*_{\mathcal{R}} \pi_2(t_{\downarrow x})$ and therefore $\pi_1(t_{\downarrow x}) \rightarrow^*_{\mathcal{R}} \pi_2(t_{\downarrow x})$.
Case 2. Transitions from *Case* B were traversed in ρ on $t_{\downarrow x}$.
This means that $\pi_1(t(x)) = \pi_2(t(x)) = b \in \Sigma$ and there is a flat prefix rewrite rule $i : L_i \overset{\sigma}{\hookrightarrow} L'_i$ such that $t_{\downarrow x} \in T(\mathcal{C} \mathbin{@} \mathcal{D}^\star)$ where $\mathcal{C} = (\mathcal{A}^{i\diamond}_{pre^*}[\diamond|b] \otimes \mathcal{A}^{i\diamond}_{post^*}[\diamond|b])$ and $\mathcal{D} = \left(\dot{\bigcup}_{a\in\Sigma}(\mathcal{A}^a_{pre^*} \otimes \mathcal{A}^a_{post^*})\right)$ (cf. Figure 6). That means that the projection to the first component of the subtree $t_{\downarrow x}$ is in $T(\mathcal{E} \mathbin{@} \mathcal{F}^\star)$ with $\mathcal{E} = \mathcal{A}^{i\diamond}_{pre^*}[\diamond|b]$ and $\mathcal{F} = \dot{\bigcup}_{a\in\Sigma}(\mathcal{A}^a_{pre^*})$ and thus can be rewritten to a successor sequence $wa_k \cdots a_\ell$ with $w = w_1 \cdots w_{|w|} \in L_i$ and $a_k, \ldots, a_\ell$ trees of height 0, while the projection to the second component of $t_{\downarrow x}$ is in $T(\mathcal{G} \mathbin{@} \mathcal{H}^\star)$ with $\mathcal{G} = \mathcal{A}^{i\diamond}_{post^*}[\diamond|b]$ and $\mathcal{H} = \dot{\bigcup}_{a\in\Sigma}(\mathcal{A}^a_{post^*})$ and thus can be rewritten *from* a successor sequence $w'a_k \cdots a_\ell$ with $w' = w'_1 \cdots w'_{|w'|} \in L'_i$, for the same $a_k, \ldots, a_\ell \in \Sigma$. Thus, we obtain that $\pi_1(t_{\downarrow x}) \rightarrow^*_{\mathcal{R}} b(w_1, \ldots, w_{|w|}, a_k, \ldots, a_\ell)$ and $b(w'_1, \ldots, w'_{|w'|}, a_k, \ldots, a_\ell) \rightarrow^*_{\mathcal{R}} \pi_2(t_{\downarrow x})$, and therefore $\pi_1(t_{\downarrow x}) \rightarrow^*_{\mathcal{R}} \pi_2(t_{\downarrow x})$.

Since in ρ for each node x, all predecessors $y \prec x$ were traversed with transitions of the identity function and thus the context of all $t_{\downarrow x}$ was not changed, it follows that $\pi_1(t) \rightarrow^*_{\mathcal{R}} \pi_2(t)$.

$\Leftarrow$ Let $\pi_1(t) \rightarrow^*_{\mathcal{R}} \pi_2(t)$.
That means for each path from the root to the leaves of t there is a minimal (w.r.t. the prefix ordering) node $x \in \mathit{dom}_t$ such that a rewrite rule was applied at x or it holds that $\pi_1(t(x)) = \pi_2(t(x))$ for all x on this path. We consider the set of all such minimal x. At all nodes y that are either predecessors of such an x or not in the subtree $t_{\downarrow x}$ of any such x, we apply transitions of the identity function. Otherwise, for each x two cases are to be considered.

Case 1. The rule applied at node x was subtree rewrite rule j. That means that there are trees s, s' such that $\pi_1(t_{\downarrow x}) \rightarrow^*_{\mathcal{R}} s \in S_j$ and $s' \rightarrow^*_{\mathcal{R}} \pi_2(t_{\downarrow x})$ where $s' \in S'_j$. Therefore, it holds that $\pi_1(t_{\downarrow x}) \in \mathit{pre}^*(S_j)$ and $\pi_2(t_{\downarrow x}) \in \mathit{post}^*(S'_j)$, thus $\pi_1(t_{\downarrow x}) \in T(\mathcal{A}^j_{pre^*})$ and $\pi_2(t_{\downarrow x}) \in T(\mathcal{A}^j_{post^*})$. Consequently, it holds that $t_{\downarrow x} \in T(\mathcal{A}^j_{pre^*} \otimes \mathcal{A}^j_{post^*})$ and thus $t_{\downarrow x} \in T(\mathcal{B})$.
Case 2. The highest and rightmost (i.e. the flat prefix rewrite rule on this level that rewrote the longest prefix) applied rule was flat prefix rewrite rule i on the successor sequence of node x, and let xk be the first node not to be altered by flat prefix rewriting (if there is no such son xk of x, then the complete successor sequence is involved in the flat prefix rewriting steps). Let $\pi_1(t(x)) = \pi_2(t(x)) = b$. There is a tree $b(w_1, \ldots, w_{|w|}, a_k, \ldots, a_\ell)$ of height 1 with $w = w_1 \cdots w_{|w|} \in L_i$ such that the projection to the first component of $t_{\downarrow x}$ can be rewritten to this tree: $\pi_1(t_{\downarrow x}) \rightarrow^*_{\mathcal{R}} b(w_1, \ldots, w_{|w|}, a_k, \ldots, a_\ell)$. Note that the length $|w| + \ell - k$ of this successor sequence does not necessarily need to be equal to the length n of the successor sequence of x since

other flat prefix rewrite rules might have been applied on this level. Since $(\mathcal{A}^{i\diamond}_{pre^*}[\diamond|b])@\big(\dot{\bigcup}_{a\in\Sigma}(\mathcal{A}^{a}_{pre^*})\big)^{\star}$ recognizes all subtrees with root $b = \pi_1(t(x))$ that can be rewritten to a tree of height 1 with the flat front of the desired form, it follows that $\pi_1(t_{\downarrow x}) \in T\big((\mathcal{A}^{i\diamond}_{pre^*}[\diamond|b])@\big(\dot{\bigcup}_{a\in\Sigma}(\mathcal{A}^{a}_{pre^*})\big)^{\star}\big)$.

For the second component, since flat prefix rewrite rule $i : L_i \overset{\sigma}{\hookrightarrow} L'_i$ was applied and $a_k, \ldots, a_\ell$ remained unchanged by any flat prefix rewrite rule, there is a tree $b(w'_1, \ldots, w'_{|w'|}, a_k, \ldots, a_\ell)$ with $w' = w'_1, \ldots, w'_{|w'|} \in L'_i$, such that $b(w'_1, \ldots, w'_{|w'|}, a_k, \ldots, a_\ell) \rightarrow^*_{\mathcal{R}} \pi_2(t_{\downarrow x})$. As again more flat prefix rewrite rules might have been applied on this level, we need to calculate the set *post** of the hedge $w'_1, \ldots, w'_{|w'|}$ and the set *post** of each unaltered a_λ for $k \leq \lambda \leq \ell$. Thus, $\pi_2(t_{\downarrow x}) \in T\big((\mathcal{A}^{i\diamond}_{post^*}[\diamond|b])@\big(\dot{\bigcup}_{a\in\Sigma}(\mathcal{A}^{a}_{post^*})\big)^{\star}\big)$.

Combined, this yields for the convolution $t = \langle t_1, t_2\rangle$ that $t_{\downarrow x} \in T\big((\mathcal{A}^{i\diamond}_{pre^*}[\diamond|b] \otimes\mathcal{A}^{i\diamond}_{post^*}[\diamond|b])@\big(\dot{\bigcup}_{a\in\Sigma}(\mathcal{A}^{a}_{pre^*} \otimes \mathcal{A}^{a}_{post^*})\big)^{\star}\big)$.

Since each node x was chosen to be the highest node on a path from the root to the leaves of t where a rewrite rule was applied, the context of these subtrees was not rewritten and thus is identical. As $\mathcal{B}$ contains all transitions of the identity function, we obtain that $t = \langle t_1, t_2\rangle \in T(\mathcal{B})$.

One-Step Reachability Relation $\rightarrow$

The construction of an N↓TA for this relation is straightforward. We give an informal description: the automaton nondeterministically guesses the one node x where a rewrite rule was applied, and guesses which rewrite rule was applied at x. Then it checks whether the projection to the first component makes up the left hand side of the rule while the projection to the second component makes up the right hand side of the rule. For a subtree rewrite rule $j : S_j \overset{\sigma}{\hookrightarrow} S'_j$ this simply means working like $\mathcal{A}_j \otimes \mathcal{A}'_j$ with $T(\mathcal{A}_j) = S_j$ and $T(\mathcal{A}'_j) = S'_j$ on $t_{\downarrow x}$. For flat prefix rewrite rule $i : L_i \overset{\sigma}{\hookrightarrow} L'_i$, some more work is required (checking that $t_{\downarrow x}$ is of height 1, and the prefix of the successor sequence is made up from a convolution of words from L_i and L'_i while the suffix is a convolution of identical words), but it is not difficult to see that this can be done with the above introduced or mentioned constructions.
Thus, the one-step reachability relation is also automatic.