Transition Graphs of Rewriting Systems over Unranked Trees

Christof Löding and Alex Spelten

RWTH Aachen, Germany

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 (binary) 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 reachability problem remains decidable.

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. [22]).

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. [12]) and thus allows to express reachability properties. A well-known representative of a class of graphs with decidable MSO theory is the class of transition systems generated by pushdown automata (cf. [18]). Furthermore, Caucal showed in [5] that applying the formalisms of interpretation and unfolding in alternation, one obtains a "pushdown hierarchy" of graph classes, throughout which the decidability of MSO is preserved. 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 the main point of 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. [17]). "Ground rewriting" means that no variables occur in the rules, thus in ground tree (or term) rewriting systems, only explicitly specified subtrees can be replaced by other explicitly specified subtrees. Though in general MSO is undecidable for transition graphs of ground tree rewriting systems, there is a decidable logic that allows to express reachability problems: first-order logic with the reachability relation [11]. In [15, 17] the structure of the transition graphs of ground tree rewriting systems and their relation to other classes of infinite graphs, in particular to pushdown graphs was studied. Furthermore, in [16, 17] 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 propagation 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 adaption 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 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, by a flat front we indicate a sequence of children of a node, all of which are leaves. With this approach related to ground tree rewriting, we obtain a class of infinite graphs which has a decidable reachability problem. Furthermore, analogous to regular ground tree rewriting systems over ranked trees, a regular variant of these rewriting systems is considered.

On introducing a constraint relying on the inner structure of the vertices of the transition graphs (i.e. the trees) to the reachability problem, we discover that the decidability of this problem fails for the introduced rewriting formalisms while it remains decidable for ground rewriting on ranked trees. 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. 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. [14], 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 a 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 an extended model (denoted by ε-N↑TA) with ε-transitions from the set $Q \times Q$ 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 $t' = t[x|s]$. In this case, t' is *derived* from t by the rule $s \overset{\sigma}{\hookrightarrow} s'$ and we write $t \to^\sigma_{\mathcal{R}} t'$. The tree language that is generated by $\mathcal{R}$ is denoted $T(\mathcal{R}) = \{t \in T_\Sigma \mid t_{in} \to^*_{\mathcal{R}} t\}$; 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. [17].

One way of dealing with unranked trees is to encode them by ranked trees. We use here a formalism proposed in [21], 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: $a(t_1, \ldots, t_n) @ t' = a(t_1, \ldots, t_n, t')$, respectively for case $n = 0 : \ a @ t' = a(t')$. Furthermore, this formalism can be extended to hedges in the intuitive way. Note that every unranked tree can be generated uniquely from trees of height 0 using the extension operator: $a(t_1, \ldots, t_n) = [(\cdots (a @ t_1) @ t_2) \cdots @ t_n]$, 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 including the root in the corresponding unranked tree. The rewriting system therefore is defined such that exactly those partial subtrees are replaced.

Definition 3.1. 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. Thus, any tree $t \in T_{\Sigma,\xi}$ can be written as $t = \bar{t} @ \xi$ with $\bar{t} \in T_\Sigma$.

Definition 3.2. 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 are 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).

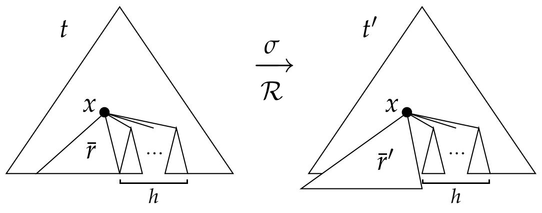
The class of transition graphs of PSRSs is denoted by PSRG.

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

Theorem 3.3. *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 [20].

Figure 1 Application of rewrite rule $r \overset{\sigma}{\hookrightarrow} r'$ according to Definition 3.2.



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. ■

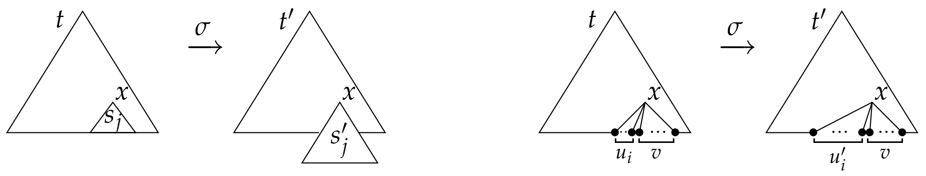
This class equivalence of GTRG and PSRG shows that by disregarding the inner structure of the vertices (unranked vs. ranked trees), the transition graphs are of identical structure.

Corollary 3.4. *The first-order theory with reachability is decidable for PSRSs.*

Additionally, several other decidability and undecidability results for GTRG can be transferred to PSRG (cf. [2, 17]).

However, since an encoding of unranked trees as ranked ones does lose some structural information, we want to mention a difference between the graph classes GTRG and PSRG. Therefore, we consider *deterministic top down tree automata (D↓TA)*. For the ranked case, this model decides which states to send to the successors of a node depending on the current state, current label, and number of successors of the current node. Although for unranked trees we allow a D↓TA to additionally read the successor sequence of the current node before proceeding and assigning states to the successors, analogous to the ranked case, it is not strong enough to capture the full class of regular unranked tree languages (cf. [10]). Thus, restricting the transition graph of a rewriting system to the tree language $T(\mathcal{A})$ of a D↓TA $\mathcal{A}$ (i.e. each vertex of the graph is in $T(\mathcal{A})$) results in a variant of the classical constraint reachability problem "Given a rewriting system $\mathcal{R}$, vertex t, and regular sets T_1, T_2 of trees, is there a path starting in t that only visits vertices of T_1 until eventually reaching a vertex of T_2?". The target set T_2 remains regular, but with the restriction of the transition graph, the set T_1 is now the tree language recognized by a D↓TA. This problem yields different decidability results for the ranked and unranked setting. While from [6] it follows that the reachability problem over the restricted graph remains decidable for the ranked case (and GTRSs), it can be shown via a reduction to the halting problem of Turing machines that this decidability result fails for the case of unranked trees and PSRSs (cf. [20]).

Figure 2 Application of rewrite rules of according to Definition 4.1.



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

4 Subtree and Flat Prefix Rewriting Systems

In order to exploit the advantage of an arbitrary number of successors of each node, subtree substitution is combined 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. It will be shown that this rewriting principle yields a new class of transition graphs.

Definition 4.1. 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.

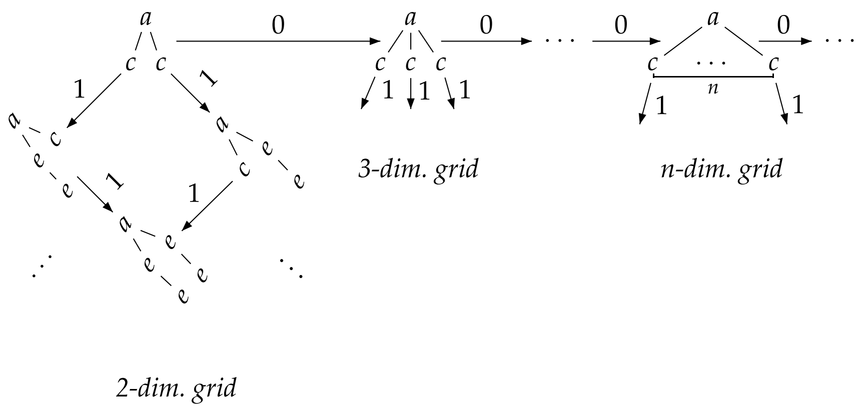
A tree t' is derived from t ($t \rightarrow^{\sigma}_{\mathcal{R}} t'$) by applying a subtree rewrite rule r_j, if there is a node $x \in dom_t$ with $t_{\downarrow x} = s_j$ such that $t[x|s'_j] = t'$ (cf. Figure 2(a)).

A tree t' is derived from t by applying a prefix rewrite rule r_i, if there are a node $x \in dom_t$ with $ht(t_{\downarrow x}) = 1$ and $flatfront(t_{\downarrow x}) = u_i v$, a tree $s \in T_\Sigma$ with $ht(s) = 1$ and $s(\varepsilon) = t(x)$, $flatfront(s) = u'_i v$, 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. Conversely, SFPRSs can be regarded as the special case of singular sets in the rules of regular SFPRSs.

Note that the newly introduced 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 control is not available for the previously considered rewriting systems.

Figure 3 Transition graph of SFPRS R_0.



4.1 Classification of Transition Graph Classes

Towards a classification of the transition graph classes PSRG and SFPRG, consider the SFPRS $\mathcal{R}_0 = (\Sigma, \Gamma, R, t_{in})$ with $\Sigma = \{a, c, e\}$, $\Gamma = \{0, 1\}$,

$R = \{r_1 : c \overset{1}{\hookrightarrow} \begin{smallmatrix} e \\ | \\ e \end{smallmatrix}, r_2 : e \overset{1}{\hookrightarrow} \begin{smallmatrix} e \\ | \\ e \end{smallmatrix}, r_3 : c \overset{0}{\hookrightarrow} cc\}$ $(I = \{3\}, J = \{1, 2\})$, and $t_{in} = \begin{smallmatrix} & a & \\ / & & \backslash \\ c & & c \end{smallmatrix}$,

whose transition graph is depicted in Figure 3.

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

PROOF (SKETCH). Towards a contradiction: consider the 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. ■

Thus, the SFPRS $\mathcal{R}_0$ over unranked trees has a transition graph which cannot be generated by a ground tree rewriting system over ranked trees. On the other hand, 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 4.3. *The class PSRG of transition graphs of PSRSs is strictly included in the class SFPRG of transition graphs of SFPRSs: PSRG $\subsetneqq$ SFPRG.*

Thus the undecidability results for PSRSs carry over to SFPRSs. These include the reachability problems: constrained reachability, universal reachability, and universal recurrence (cf. [17]).

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 [19]).

4.2 Reachability with Saturation

The main contribution of this paper is the decidability of the reachability problem for transition graphs of 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 $\text{pre}^*_{\mathcal{R}}(T) = \{t \in T_\Sigma \mid \exists t' \in T : t \to^*_{\mathcal{R}} t'\}$ of trees from which the set T can be reached. Thereby, the rewrite rules of a 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 [17]. In the very same manner, the set $\text{post}^*_{\mathcal{R}}(T) = \{t \in T_\Sigma \mid \exists t' \in T : t' \to^*_{\mathcal{R}} t\}$ 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.

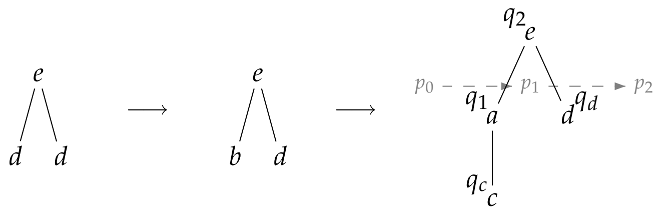
However, due to the different natures of the two types of rules of SFPRSs, and the employment of word automata in ε-N↑TAs over unranked trees, the saturation is based on an interleaving of two saturation algorithms on different levels of automata. In detail, for a prefix rewrite rule the saturation is 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.

As the technical presentation of the algorithm and the formal correctness proof are rather cumbersome, we prefer to give an informal description of how the algorithm works. As mentioned above, it is a combination of the saturation algorithms for ground rewrite systems on trees and for prefix rewrite systems on words. We first briefly describe these two parts and then explain how the interleaving is implemented to ensure that the prefix rewriting can only be applied to flat fronts.

Assume that we are given an N↑TA $\mathcal{A}$ accepting the target set. For the simulation of a subtree rewrite rule $t \hookrightarrow t'$ we consider a state q that can be reached by $\mathcal{A}$ on reading the right hand side t' of the rule, i.e. with $\mathcal{A} : t' \to^* q$. We introduce an ε-transition from q_t to q, where q_t is a special state that is (at the beginning) only reachable via t. These states are added to $\mathcal{A}$ for all left hand sides of the subtree rewrite rules before the saturation starts. Now assume $\mathcal{A}$ accepts a tree with t' as subtree, using q at the root of t' in the accepting run. After the saturation step described above, $\mathcal{A}$ can accept the same tree with t' replaced by t by first moving to q_t on reading the subtree t, then using the new ε-transition, and continuing as in the run on the original tree.

For the prefix rewrite rules we proceed similarly but now saturating the word automata that recognize the sequence of labels of the successors of a node, called horizontal automata in the following. Before the saturation, special states q_a for each letter $a \in \Sigma$ are added to the tree automaton that can only be reached via reading

Figure 4 Derivation to the target set $e(a(c)d)$ and accepting run.



an a at a leaf. In the horizontal automata we introduce for each u on the left hand side of a prefix rewrite rule a new state p_u that can be reached only on reading the sequence $q_{a_1} \cdots q_{a_n}$, where $u = a_1 \cdots a_n$. In the following we denote this sequence of states by $\underline{q}_u$.

Consider a prefix rewrite rule $u \hookrightarrow u'$ and assume that in some horizontal automaton a state p can be reached on reading $\underline{q}_{u'}$, i.e. the state sequence corresponding to the right hand side of the rule. Then we add an ε-transition from p_u to p in this automaton. Assume that $\mathcal{A}$ accepts a tree with a flat front having u' as prefix. Then an accepting run labels the corresponding sequence of leafs by $\underline{q}_{u'}$ and the horizontal automaton reaches a state p on reading $\underline{q}_{u'}$. Using the new ε-transition it is now also possible to accept the tree where the prefix u' is replaced by u at the flat front. Hence we have simulated one step of prefix rewriting.

However, the previous description is not accurate due to the following. Assume that the sequence of successors of some node starts with some leaves forming the word u' but that it is not a flat front, i.e. there are also some successors that are not leaves. The saturation described above does not distinguish these cases and hence the saturation also simulates prefix rewritings that are not applied to a flat front. We have to make sure that after using an ε-transition from the saturation on the horizontal level, the run can only continue using states of the form q_a, indicating that the automaton is really processing a flat front.

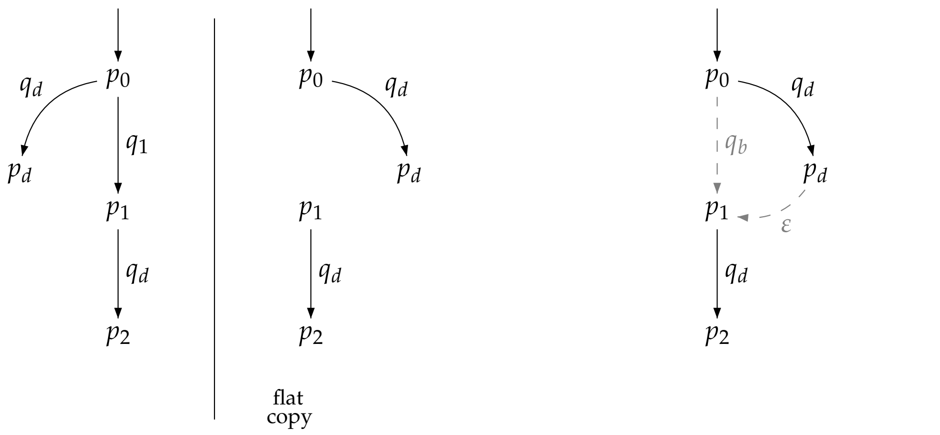
For this purpose, we introduce for each horizontal automaton a second copy in which we erase all transitions that are labeled with states not of the form q_a (the resulting automaton has two initial states, one for each copy). We refer to this copy as the flat copy of the automaton because it can only process states that are assumed at a flat front.

We restrict the saturation process to the flat copy of the automaton. But now we are facing a new problem: The (backward) application of the subtree rewrite rules as simulated by the saturation on the level of the tree automaton might introduce new flat fronts. This has to be reflected in the flat copies of the horizontal automata. We illustrate this with the following small example.

Consider the subtree rewrite rule $b \hookrightarrow a(c)$ and the prefix rewrite rule $d \hookrightarrow b$, and assume that the tree $e(a(c)d)$ is in the target set. Figure 4 shows a derivation with the application of the two rules and the relevant parts of the accepting run of $\mathcal{A}$ on the tree in the goal set.

Figure 5(a) shows (the relevant part of) the horizontal automaton whose run is indicated in Figure 4. The saturation for the subtree rewrite rule gives an ε-transition from q_b (the state for the left hand side of the rule) to q_1 (the state that is reached via the right hand side of the rule). After this saturation the automaton also accepts the

Figure 5 Horizontal automaton used in Figure 4.



(a) Before the saturation step.

(b) Flat copy with the two inserted edges.

tree $e(bd)$. But in the flat copy of the horizontal automaton there is no state that can be reached via q_b, which is the state sequence (of length 1) corresponding to the right hand side of the prefix rewrite rule. The algorithm now reflects the fact that q_1 can be replaced by q_b everywhere because there is an ε-transition from q_b to q_1 by copying the transition $p_0 \xrightarrow{q_1} p_1$ from the non-flat copy to the flat copy of the horizontal as a transition $p_0 \xrightarrow{q_b} p_1$.

Now the saturation is possible in the flat copy, and an ε-transition from p_d is added to p_1 as depicted in Figure 5(b). After this step also the rightmost tree in Figure 4 is accepted.

This completes the description of the algorithm. The main steps are summarized as follows.

- On the level of the tree automaton we apply the usual saturation.
- For each horizontal automaton we introduce a "flat copy" that can only read the states of the form q_a assumed at the flat front of a tree.
- The saturation of the horizontal level is applied to the flat copies only.
- For each ε-transition of the form (q_a, q) we transfer all q-transitions from the non-flat copy as q_a-transitions to the flat copy.

For an elaborate example, the full construction (also for regular SFPRSs), and the correctness proof, we refer the reader to [20]. The automaton resulting from the saturation accepts exactly those trees from which the target set is reachable. As emptiness for unranked tree automata is decidable, we obtain the following theorem.

Theorem 4.4. *For every SFPRS $\mathcal{R}$ over unranked trees, the reachability problem: "Given $\mathcal{R}$, vertex t, and regular set T of vertices, is there a path from t to a vertex in T?" is decidable.*

Similar to the case of PSRSs, it is shown in [20] via a reduction to the halting problem of Turing machines that the decidability of the reachability problem fails when restricting the transition graphs of (regular) SFPRSs to the tree language of a deterministic top down tree automaton (cf. Section 3).

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; nevertheless the encoding fails to capture the complete structural information and thus leads to different decidability results when taking the inner structure of the vertices into account.

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 SFPRG. Similar to PSRSs, the decidability of the reachability problem fails when taking the inner structure of the vertices into account.

As a class strictly subsuming the transition graphs of ground rewriting systems over ranked trees, the class of (regular) SFPRSs deserves further study. Recently we have shown that the first-order theory with reachability relation is decidable for these graphs by proving that a transition graph enriched with reachability edges 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]). Nevertheless, the decidability of other reachability problems, such as the recurrence problem, is unsettled.

In general, other rewriting principles over unranked trees have yet to be investigated. One aspect could be to use other word rewriting techniques than prefix rewriting 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 [13].

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 infinite terms having a decidable theory. In *Proc. MFCS*, volume 2420 of *LNCS*, pages 165–176. Springer, 2002.

[6] T. Colcombet. On families of graphs having a decidable first order theory with reachability. In *Proc. ICALP 2002*, volume 2380 of *LNCS*, pages 98–109. 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] J. Cristau, C. Löding, and W. Thomas. Deterministic automata on unranked trees. In *Proc. FCT 2005*, volume 3623 of *LNCS*, pages 68–79. Springer, 2005.

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

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

[13] P. Gyenizse and S. Vágvölgyi. Linear generalized semi-monadic rewrite systems effectively preserve recognizability. *TCS*, 194(1–2):87–122, 1998.

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

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

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

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

[18] D. Muller and P. Schupp. The theory of ends, pushdown automata, and second-order logic. *TCS*, 37:51–75, 1985.

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

[20] 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`.

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

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