Derivation Trees of Ground Term Rewriting Systems

Joost Engelfriet

Department of Computer Science, Leiden University
P.O.Box 9512, 2300 RA Leiden, The Netherlands
e-mail: engelfri@wi.leidenuniv.nl

Abstract. A notion of derivation tree is introduced for ground term rewriting systems and new proofs are given for some old results.

1 Introduction

A ground term rewriting system is a term rewriting system of which the rules do not contain variables. We will show that a natural concept of derivation tree can be defined for these rewriting systems, in such a way that a tree t_1 can be (iteratively) rewritten to a tree t_2 iff there is a derivation tree of which the "yield" is the pair (t_1, t_2), with an appropriate definition of 'yield'. Derivations that differ only in the order of independent rule applications, correspond to the same derivation tree. Moreover, the set of derivation trees forms a regular tree language. Thus, the situation is analogous to (and, in fact, generalizes) the situation for context-free grammars. Using this concept of derivation tree, and the well-known closure properties of the regular tree languages, we give a new proof for (a slight extension of) the main result of [Bra]: the set of trees that can be obtained by (iterated) rewriting of the trees of a regular tree language (using the rules of a ground term rewriting system), is again a regular tree language. Viewing strings as monadic trees in the usual way, the result of [Bra] generalizes the original result of [Büc]: every regular canonical system generates a regular string language (effectively). Thus, we provide in particular a tree language theoretic proof of Büchi's result on strings. Based on the result of [Bra] we also give a new proof of the following result of [DauTis1, DHLT]. For every ground term rewriting system there exist regular tree languages $L_1, R_1, \ldots, L_n, R_n$ such that a tree t_1 can be (iteratively) rewritten to a tree t_2 iff t_2 can be obtained (in one stroke) from t_1 by replacing independent subtrees $u_1, \ldots, u_k$ of t_1 by subtrees $v_1, \ldots, v_k$, respectively, where for every i there exists j such that $(u_i, v_i) \in (L_j, R_j)$. In the terminology of [DauTis1, DHLT], every ground term rewriting system can be simulated by a ground tree transducer. This result was used in [DauTis1, DHLT, DauTis2] to give an elegant proof of the decidability of confluence of a ground term rewriting system (also proved in [Oya]), and, more generally, of the decidability of the first-order theory of ground term rewriting. At the end of the paper we discuss this decidability result, together with the decidability of termination of a ground term rewriting system (shown in [HueLan]).

Derivation trees of ground term rewriting sytems were considered before in [Oya, CoqGil], but they seem to be less natural than the ones introduced here, which were inspired by [DauTis1, DHLT].

2 Ground Term Rewriting Systems

We assume the reader to be familiar with tree language theory (see, e.g., [GécSte1, GécSte2]), in particular with the notion of a regular (or recognizable) tree language, i.e., a tree language generated by a regular tree grammar (or accepted by a finite tree automaton). For a ranked alphabet Σ, the class of regular tree languages over Σ is denoted REGT_Σ. The class of all regular tree languages is denoted REGT. We will make extensive use of well-known (effective) closure properties of REGT, such as closure under union, intersection, and complementation (see, e.g., Theorem II.4.2 of [GécSte1]).

For a ranked alphabet Σ, the set of all trees (or ground terms) over Σ is denoted T_Σ. Trees with variables are not allowed in ground rewriting systems. However, they will be used as a technical tool, in particular to define the context in which ground terms are replaced by other ground terms. Trees with variables are trees over $\Sigma \cup X$, where $X = \{x_1, x_2, x_3, \dots\}$ and each variable x_i is of rank 0. For a tree $t \in T_{\Sigma \cup X}$ and trees $t_1, \dots, t_k$ ($k \in \mathbb{N} = \{0, 1, 2, \dots\}$), $t[t_1, \dots, t_k]$ denotes the tree obtained from t by substituting t_i for every occurrence of x_i, for $1 \leq i \leq k$. For $k \in \mathbb{N}$, a *k-place context* is a tree c over $\Sigma \cup \{x_1, \dots, x_k\}$ such that every variable from $\{x_1, \dots, x_k\}$ occurs in c exactly once. As usual, a tree u is a subtree of a tree t if $t = c[u]$ for some 1-place context c. Intuitively, such a decomposition $c[u]$ of t is uniquely determined by a node of t, viz. the root of the (occurrence of the) subtree u in t. For an example see Fig. 1.

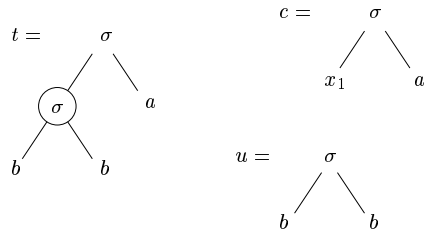

Fig. 1. A node of t determines a decomposition $c[u]$ of t; for $c = \sigma(x_1, a)$ and $u = \sigma(b, b)$, $t = c[u] = \sigma(\sigma(b, b), a)$.

Let Σ be a ranked alphabet. A *ground rewrite system* over Σ is a finite subset P of $T_\Sigma \times T_\Sigma$. An element (u, v) of P is called a rule (or production) of P, and is also written $u \to v$. The *rewrite relation* $\to_P \subseteq T_\Sigma \times T_\Sigma$ is defined as follows: for $s, t \in T_\Sigma$, $s \to_P t$ iff there are a rule $u \to v$ of P and a 1-place

context c such that $s = c[u]$ and $t = c[v]$. As usual, $\rightarrow_P^*$ denotes the reflexive, transitive closure of $\rightarrow_P$. The *parallel rewrite relation* $\Rightarrow_P \subseteq T_\Sigma \times T_\Sigma$ is defined in the following way: for $s, t \in T_\Sigma$, $s \Rightarrow_P t$ iff there are a $k \in \mathbb{N}$, a k-place context c, and rules $u_1 \rightarrow v_1, \ldots, u_k \rightarrow v_k$ in P, such that $s = c[u_1, \ldots, u_k]$ and $t = c[v_1, \ldots, v_k]$. Thus, in a parallel rewrite step any number of rules can be applied, to independent subtrees. Note that $\Rightarrow_P^* = \rightarrow_P^*$.

Example 1. Let $\Sigma = \{\sigma, a, b, p, q, r, s\}$, where σ has rank 2 and all other symbols have rank 0. As an example, consider the ground rewrite system $P = P_1 \cup P_2$ over Σ, where P_1 consists of the rules $a \rightarrow p$, $\sigma(p,p) \rightarrow p$, $\sigma(p,p) \rightarrow q$, $q \rightarrow \sigma(q,b)$, and $q \rightarrow b$, and P_2 consists of the rules $\sigma(b,b) \rightarrow r$, $\sigma(r,b) \rightarrow r$, $r \rightarrow s$, $s \rightarrow \sigma(a,s)$, and $s \rightarrow a$. Then, for instance, $\sigma(\sigma(a,a),a) \rightarrow_P^* \sigma(\sigma(b,b),a)$, as a result of the following rewrite steps: $\sigma(\sigma(a,a),a) \rightarrow_P \sigma(\sigma(p,a),a) \rightarrow_P \sigma(\sigma(p,p),a) \rightarrow_P \sigma(q,a) \rightarrow_P \sigma(\sigma(q,b),a) \rightarrow_P \sigma(\sigma(b,b),a)$. And, for instance, $\sigma(\sigma(b,b),s) \Rightarrow_P \sigma(r,\sigma(a,s))$. □

Apart from the usual ground rewrite systems we will also be interested in ground rewrite systems with infinitely many rules that can be represented by regular tree languages. For ground rewrite systems with infinitely many rules the above definitions are valid too. An *extended ground rewrite system* over Σ is a finite subset P of $\mathrm{REGT}_\Sigma \times \mathrm{REGT}_\Sigma$. Let $P' \subseteq T_\Sigma \times T_\Sigma$ be the (ordinary) ground rewrite system consisting of all rules $u \rightarrow v$ such that $u \in L$ and $v \in R$ for some $(L,R) \in P$. Then, by definition, $\rightarrow_P = \rightarrow_{P'}$, $\Rightarrow_P = \Rightarrow_{P'}$, and the rules of P are those of P'. Thus, each "regular rule" (L,R), where L and R are regular tree languages, abbreviates all rules $u \rightarrow v$ with $u \in L$ and $v \in R$. Note that the rules that are used in a parallel rewrite step of P, are derived from possibly different regular rules. For algorithmic purposes, an extended ground rewrite system is specified by giving regular tree grammars (or finite tree automata) for the regular tree languages involved. Obviously every ground rewrite system is also an extended ground rewrite system.

Example 2. Let $\Delta = \{\sigma, a, b\}$, where σ has rank 2 and a, b have rank 0. Consider the extended ground rewrite system Q over Δ containing the two regular rules (A, C_b) and (C_b', C_a), where A is the set of all trees over $\{\sigma, a\}$ that contain at least one σ, i.e., $A = T_{\{\sigma,a\}} - \{a\}$, C_b is the set of all trees $\sigma(\sigma(\cdots\sigma(b,b)\cdots,b),b)$ with $n \geq 0$ symbols σ, C_b' is the same as C_b except that $n \geq 1$, and C_a is the set of all trees $\sigma(a,\sigma(a,\cdots\sigma(a,a)\cdots))$ with $n \geq 0$ symbols σ. It is not difficult to see that for all trees $t_1, t_2 \in T_\Delta$, $t_1 \rightarrow_Q^* t_2$ if and only if $t_1 \rightarrow_P^* t_2$, where P is the ground rewrite system of Example 1. □

The relation of interest for an extended ground rewrite system P is the relation $\rightarrow_P^*$. Whenever we are mainly interested in the parallel rewrite relation $\Rightarrow_P$, an extended ground rewrite system P will also be called a *ground tree transducer*. Ground tree transducers were introduced in [DauTis1, DHLT], in a different, but obviously equivalent, way (cf. II.3 of [DauTis1]). It is shown in II.5 of [DauTis1] and Proposition 2 of [DHLT] that for every extended ground rewrite system P there is a ground tree transducer Q such that $\rightarrow_P^* = \Rightarrow_Q$. Thus, any sequence of

rewrite steps in P is simulated by one parallel rewrite step in Q, and vice versa. In Section 4 we will give a new proof of this result.

A *ground tree grammar*, introduced in [Bra] where it is called a regular system, is a tuple $G = (\Delta, \Sigma, P, S)$ where P is a ground rewrite system over Σ, $\Delta \subseteq \Sigma$, and S is a finite subset of T_Σ. The language generated by G is $L(G) = \rightarrow_P^*(S) \cap T_\Delta$, i.e., the set of all trees $t \in T_\Delta$ such that $s \rightarrow_P^* t$ for some $s \in S$. A *regular tree grammar* is a ground tree grammar $G = (\Delta, \Sigma, P, S)$ such that (1) all elements of $\Sigma - \Delta$ have rank 0 (and are called nonterminals), (2) the left-hand side of each rule of P is in $\Sigma - \Delta$, and (3) S is a singleton containing one element of $\Sigma - \Delta$. This is the usual notion of regular tree grammar (see Section II.3 of [GécSte1]). The main result of [Bra] is that for every ground tree grammar an equivalent regular tree grammar can effectively be constructed. In Section 4 we will give a new proof of this result, and show, as a slight generalization, that it also holds for every *extended ground tree grammar*, which is defined as above, except that P is an extended ground rewrite system over Σ.

Example 3. Consider the ground tree grammar $G = (\Delta, \Sigma, P, S)$ where P is the ground rewrite system over Σ of Example 1, $\Delta = \{\sigma, a, b\}$, and $S = \{s\}$. It can be shown that $L(G) = L(G')$ where $G' = (\Delta, \Sigma', P', S)$ is the regular tree grammar with $\Sigma' = \{\sigma, a, b, s\}$ and P' consists of all rules $s \rightarrow \sigma(a, s)$, $s \rightarrow \sigma(s, b)$, $s \rightarrow a$, and $s \rightarrow b$. An example of an extended ground tree grammar is $G'' = (\Delta, \Delta, Q, \{\sigma(a, a)\})$ where Q is the extended ground rewrite system over Δ of Example 2. It can be shown that also $L(G'') = L(G')$. □

The main result of [Bra] is a generalization of the following result of [Büc] for strings: every regular canonical system generates a regular string language (effectively), see, e.g., Section 2.3 of [Sal]. In fact, it is well known that strings correspond to trees over a monadic ranked alphabet. A ranked alphabet Σ is *monadic* if it is of the form $\Sigma = A \cup \{e\}$ where e is a fixed symbol of rank 0 (standing for the empty string) and every element of A has rank 1. The string $a_1 a_2 \cdots a_n$ over the alphabet A will be identified with the tree $a_n(\cdots a_2(a_1(e))\cdots)$ over Σ. A *regular canonical system* is a ground tree grammar $G = (\Delta, \Sigma, P, S)$ with monadic ranked alphabets Σ and Δ. Thus, if $\Delta = A \cup \{e\}$, then $L(G) \subseteq A^*$. Note that, on strings, the rules of P are Chomsky type 0 rules that are applied to prefixes of the sentential forms only (because the subtrees of a monadic tree are the prefixes of the corresponding string). Since, in the monadic case, a regular tree grammar is the same as a left-linear grammar (with productions of the form $X \rightarrow Yw$ or $X \rightarrow w$ where X and Y are nonterminals and w is a terminal string), it should be clear that the result of [Büc] is the monadic case of the result of [Bra].

3 Derivation Trees

Let P be an extended ground rewrite system over Σ. A *derivation* of P is a sequence of trees $t_1 \rightarrow_P t_2 \rightarrow_P \cdots \rightarrow_P t_n$. The basic idea of this paper is that the derivations of P can be represented by derivation trees (modulo the interchange

of independent derivation steps), and that the derivation trees of P form a regular tree language. This is similar to the situation for context-free grammars. If t is a derivation tree of the above derivation, then the "transduction" of t is the pair of trees (t_1, t_n); hence, the set of transductions of all derivation trees of P is the relation $\rightarrow_P^*$. This is similar to the fact that the set of yields of derivation trees of a context-free grammar G is the language generated by G. Thus, in our setting, 'transduction' plays the role of 'yield'. Also similar to yield, the transduction of a tree t can be defined in a straightforward way, for arbitrary trees rather than just derivation trees. We will use a special symbol # which, for derivation trees, indicates the application of a rule.

For a ranked alphabet Σ we denote by $\Sigma\#$ the ranked alphabet $\Sigma \cup \{\#\}$, where # is a new symbol of rank 2. For a tree $t \in T_{\Sigma\#}$, the trees $\mathrm{left}(t)$ and $\mathrm{right}(t)$ in T_Σ are defined recursively as follows, where σ is an element of Σ of rank $k \geq 0$, and the t_i are trees in $T_{\Sigma\#}$:

$$\begin{aligned}
&\mathrm{left}(\sigma(t_1, \ldots, t_k)) = \sigma(\mathrm{left}(t_1), \ldots, \mathrm{left}(t_k)),\\
&\mathrm{left}(\#(t_1, t_2)) = \mathrm{left}(t_1),\\
&\mathrm{right}(\sigma(t_1, \ldots, t_k)) = \sigma(\mathrm{right}(t_1), \ldots, \mathrm{right}(t_k)), \text{ and}\\
&\mathrm{right}(\#(t_1, t_2)) = \mathrm{right}(t_2).
\end{aligned}$$

For a tree $t \in T_{\Sigma\#}$, the *transduction of* t is defined as $\mathrm{trans}(t) = (\mathrm{left}(t), \mathrm{right}(t))$. For a tree language $L \subseteq T_{\Sigma\#}$, the *transduction of* L is defined as $\mathrm{trans}(L) = \{\mathrm{trans}(t) \mid t \in L\}$. Note that $\mathrm{trans}(L) \subseteq T_\Sigma \times T_\Sigma$.

Thus, for a tree $t \in T_{\Sigma\#}$, $\mathrm{left}(t)$ ($\mathrm{right}(t)$) is obtained from t by choosing the left (right) subtree of every occurrence of #. Clearly, both 'left' and 'right' are linear tree homomorphisms from $T_{\Sigma\#}$ to T_Σ (see, e.g., Section II.4 of [GécSte1] for the concept of a linear tree homomorphism). Intuitively, $\mathrm{left}(t)$ can be seen as a part of t, in the sense that the nodes of $\mathrm{left}(t)$ are a subset of the nodes of t and the edges of $\mathrm{left}(t)$ are paths in t, as follows. A node x of t is a node of $\mathrm{left}(t)$ if its label is not # and, walking from the root of t to x, at each #-labeled node the left child is chosen. For two nodes x and y of t that are also nodes of $\mathrm{left}(t)$, y is the left (right) child of x in $\mathrm{left}(t)$ if y is a descendant of the left (right) child of x in t and, walking from x to y in t, all intermediate nodes have label #. In the same way $\mathrm{right}(t)$ can be viewed as a part of t, see Fig. 2 for an example.

Let P be an extended ground rewrite system over Σ. A *derivation tree* of P is a tree $t \in T_{\Sigma\#}$ such that for every subtree $\#(t_1, t_2)$ of t, $\mathrm{right}(t_1) \rightarrow \mathrm{left}(t_2)$ is a rule of P. The set of all derivation trees of P is denoted D_P.

Example 4. Figure 3 shows a derivation tree t of the ground rewrite system P of Example 1. Considering the five nodes with label # in infix order, the rules $\mathrm{right}(t_1) \rightarrow \mathrm{left}(t_2)$ in P corresponding to these nodes are $a \rightarrow p$, $a \rightarrow p$, $\sigma(p,p) \rightarrow q$, $q \rightarrow \sigma(q,b)$, and $q \rightarrow b$, respectively. Since, as shown in Fig. 2, $\mathrm{left}(t) = \sigma(\sigma(a,a),a)$ and $\mathrm{right}(t) = \sigma(\sigma(b,b),a)$, the transduction of t is the pair $\mathrm{trans}(t) = (\sigma(\sigma(a,a),a), \sigma(\sigma(b,b),a))$. In fact, as will be clear from the proof of

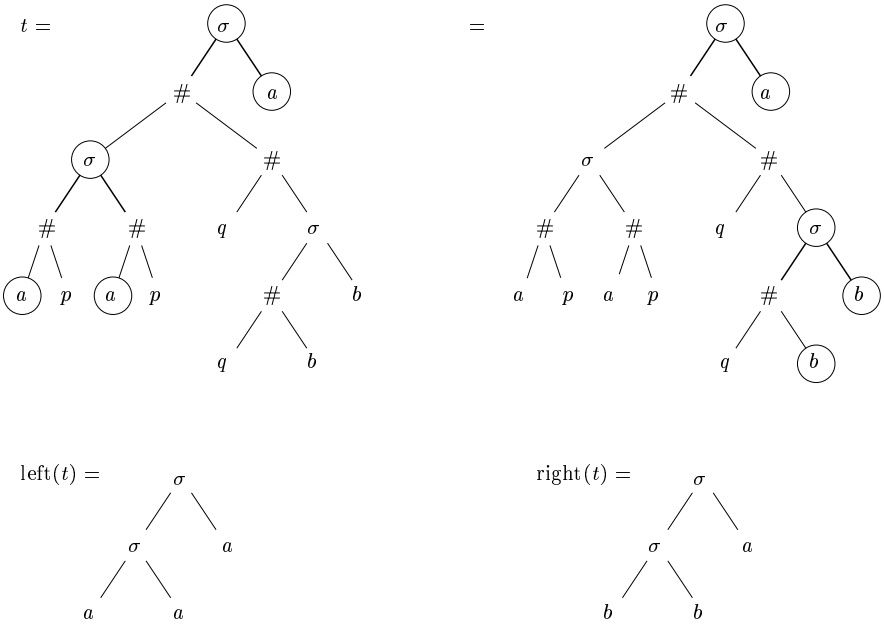

Fig. 2. The left and right of a tree $t \in T_{\Sigma\#}$.

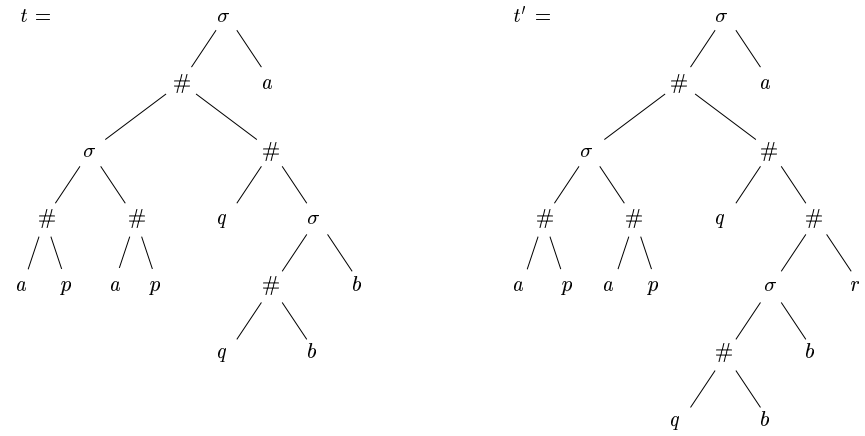

Fig. 3. Derivation trees t and t' of the ground rewrite system P of Example 1.

the next theorem, t corresponds to the derivation $\sigma(\sigma(a,a),a) \to_P^* \sigma(\sigma(b,b),a)$ given in Example 1.

Figure 3 also shows another derivation tree t' of P, closely related to t. The (infix order) sequence of rules $\mathrm{right}(t_1) \to \mathrm{left}(t_2)$ of t' is the same as that of t, followed by the rule $\sigma(b,b) \to r$. □

The main properties of D_P are that $\mathrm{trans}(D_P) = \to_P^*$ and that D_P is a regular tree language.

Theorem 1. *For every extended ground rewrite system P, $\mathrm{trans}(D_P) = \to_P^*$.*

Proof. Let P be an extended ground rewrite system over Σ. As in the case of context-free grammars, we will associate derivations with derivation trees, and derivation trees with derivations. We start with the former.

To prove the inclusion $\mathrm{trans}(D_P) \subseteq \to_P^*$ we show the following by structural induction on t: if $t \in D_P$, then $\mathrm{left}(t) \to_P^* \mathrm{right}(t)$. First, let $t = \sigma(t_1, \ldots, t_k)$ with $\sigma \in \Sigma$. Then, by definition, $\mathrm{left}(t) = \sigma(\mathrm{left}(t_1), \ldots, \mathrm{left}(t_k))$ and $\mathrm{right}(t) = \sigma(\mathrm{right}(t_1), \ldots, \mathrm{right}(t_k))$. Note that every subtree of t is in D_P. Hence, by induction, $\mathrm{left}(t_i) \to_P^* \mathrm{right}(t_i)$ for every $1 \leq i \leq k$. This implies that $\mathrm{left}(t) \to_P^* \mathrm{right}(t)$. Second, let $t = \#(t_1, t_2)$. Then $\mathrm{left}(t) = \mathrm{left}(t_1)$ and $\mathrm{right}(t) = \mathrm{right}(t_2)$. Since t is a derivation tree of P, $\mathrm{right}(t_1) \to \mathrm{left}(t_2)$ is a rule of P and hence $\mathrm{right}(t_1) \to_P \mathrm{left}(t_2)$. Thus, by induction, $\mathrm{left}(t_1) \to_P^* \mathrm{right}(t_1) \to_P \mathrm{left}(t_2) \to_P^* \mathrm{right}(t_2)$. This shows that $\mathrm{left}(t) \to_P^* \mathrm{right}(t)$.

Next we show the inclusion $\to_P^* \subseteq \mathrm{trans}(D_P)$. For this purpose we prove the following: for trees $s_1, \ldots, s_n \in T_\Sigma$ $(n \geq 1)$, if $s_1 \to_P s_2 \to_P \cdots \to_P s_n$ then there exists $t \in D_P$ such that $s_1 = \mathrm{left}(t)$ and $s_n = \mathrm{right}(t)$. We prove this by induction on the sum of the sizes of $s_1, \ldots, s_n$, distinguishing two cases. In the first case there exists a derivation step $s_i \to_P s_{i+1}$ such that $s_i \to s_{i+1}$ is in P. By induction there are derivation trees t_1 and t_2 such that $\mathrm{left}(t_1) = s_1$, $\mathrm{right}(t_1) = s_i$, $\mathrm{left}(t_2) = s_{i+1}$, and $\mathrm{right}(t_2) = s_n$. Hence $t = \#(t_1, t_2)$ satisfies the requirements. In the second case *no* $s_i \to s_{i+1}$ is in P. This means intuitively that the roots of the s_i remain unchanged. Formally it is straightforward to show that there exist $k \geq 0$, $\sigma \in \Sigma$ of rank k, and trees $r_{i,j}$ $(1 \leq i \leq n, 1 \leq j \leq k)$ such that $s_i = \sigma(r_{i,1}, \ldots, r_{i,k})$, and $r_{i,j} \to_P r_{i+1,j}$ or $r_{i,j} = r_{i+1,j}$. Thus $r_{1,j} \to_P^* r_{n,j}$ by a smaller derivation, and so, by induction, there are derivation trees $t_1, \ldots, t_k$ such that $\mathrm{left}(t_j) = r_{1,j}$ and $\mathrm{right}(t_j) = r_{n,j}$. Hence $t = \sigma(t_1, \ldots, t_k)$ satisfies the requirements. □

It is easy to see that the above inductive proofs describe a constructive way of associating derivations with derivation trees, and vice versa. As in the case of context-free grammars, the derivations associated with derivation trees are left-most derivations (where 'left-most' is defined in the obvious way). In fact, every node with label # of a derivation tree t corresponds to the application of a rule, and in the corresponding left-most derivation the rules are applied according to the infix order of these nodes in t, cf. Example 4. In the other direction, the proof does not produce a unique derivation tree; it is, however, unique modulo associativity of #. This is due to the fact that in a derivation there may be

several derivation steps that are rules in P. If systematically the left-most such derivation step is always taken, then the constructed derivation trees t have the following property: the left child of a node with label # does not have label # (cf. Fig. 3). It can be shown (but we will not do this here) that, analogous to the case of context-free grammars, there is a one-to-one correspondence between derivation trees with the above property and left-most derivations. As an example, derivation tree t of Fig. 3 corresponds in this way to the derivation given in Example 1, and derivation tree t' corresponds to that same derivation extended by $\sigma(\sigma(b,b),a) \to_P \sigma(r,a)$. Note that there is a one-to-one correspondence between left-most derivations and equivalence classes of derivations with respect to the interchange of independent derivation steps. Thus, as for context-free grammars, derivation trees (with the above property) faithfully represent the "parallelism" in derivations. This does not hold for the derivation trees in [CoqGil]. As a simple example, if P has two rules $a \to p$ and $a \to q$, then both derivations $\sigma(a,a) \to_P \sigma(p,a) \to_P \sigma(p,q)$ and $\sigma(a,a) \to_P \sigma(a,q) \to_P \sigma(p,q)$ have derivation tree $\sigma(\#(a,p),\#(a,q))$.

Derivation trees can also be constructed incrementally: if t is a derivation tree for a derivation $s_1 \to_P^* s_2$, and $s_2 \to_P s_3$ is another derivation step, then it is straightforward to construct a derivation tree t' for $s_1 \to_P^* s_2 \to_P s_3$, as follows. Suppose that $s_2 = c[u]$ and $s_3 = c[v]$ with $u \to v$ in P; thus $\mathrm{right}(t) = c[u]$. Now it can be shown that $t = c'[u']$ with $\mathrm{right}(c') = c$ and $\mathrm{right}(u') = u$ (and the root label of u' is not #), and it can be shown that $t' = c'[\#(u',v)]$ satisfies the requirements, cf. Fig. 3. In fact, if the decomposition $c[u]$ of $\mathrm{right}(t)$ is determined by node x of $\mathrm{right}(t)$, i.e., x is the root of (the occurrence of) u in $\mathrm{right}(t)$, then the decomposition $c'[u']$ of t is also determined by x, viewed as a node of t. Recall that, intuitively, $\mathrm{right}(t)$ can be viewed as a part of t, as shown in Fig. 2. Thus, if (in that figure) x is the lowest node of $\mathrm{right}(t)$ with label σ, then the corresponding node x in t is also the lowest (encircled) node with label σ. The decomposition $c[u]$ is shown in Fig. 1, and $t = c'[u']$ with $c' = \sigma(\#(\sigma(\#(a,p),\#(a,p)),\#(q,x_1)),a)$ and $u' = \sigma(\#(q,b),b)$.

Up to now we did not use the regularity of the tree languages L and R in a "regular rule" (L,R) of an extended ground rewrite system. Thus, Theorem 1 holds in fact for arbitrary term rewriting systems (with variables), viewed as abbreviations of ground term rewriting systems with infinitely many rules, in the obvious way.

The regularity of the set of derivation trees D_P of an extended ground rewrite system P is an easy exercise in tree language theory.

Theorem 2. *For every extended ground rewrite system P, D_P is a regular tree language (effectively).*

Proof. To prove this we use (effective) closure properties of the class of regular tree languages. For any tree language L, let $\mathrm{allsub}(L)$ denote the set of all trees t such that every subtree of t is in L. It is easy to see that REGT is effectively closed under the allsub operation (see, e.g., Section II.8 of [GécSte1], where 'allsub' is denoted 'rest'). For a symbol σ of rank k and tree languages $L_1, \ldots, L_k$,

$\sigma(L_1, \ldots, L_k)$ denotes the set of all trees $\sigma(t_1, \ldots, t_k)$ such that $t_i \in L_i$ for every $1 \leq i \leq k$. It is well known that REGT is effectively closed under these operations (see, e.g., Corollary II.4.12 of [GécSte1]).

Let P be an extended ground rewrite system over Σ. Define D'_P to be the set of all trees $t \in T_{\Sigma\#}$ such that either the root label of t is in Σ or $t = \#(t_1, t_2)$ and $\mathrm{right}(t_1) \to \mathrm{left}(t_2)$ is a rule of P. Then $D_P = \mathrm{allsub}(D'_P)$. Clearly, D'_P is the (finite) union of all tree languages $\sigma(T_{\Sigma\#}, \ldots, T_{\Sigma\#})$, for $\sigma \in \Sigma$, and all tree languages $\#(\mathrm{right}^{-1}(L), \mathrm{left}^{-1}(R))$, for $(L, R) \in P$. The result now follows from the fact that $T_{\Sigma\#}$ and all L and R are regular, from the above closure properties and closure under union, and from the (effective) closure of REGT under inverse tree homomorphisms (see, e.g., Theorem II.4.18 of [GécSte1]). Recall that both 'left' and 'right' are tree homomorphisms. □

These results show that the derivation trees of extended ground rewrite systems have properties similar to those of context-free grammars. In fact, in a sense to be explained now (informally), the former can be viewed as a proper generalization of the latter. With every context-free grammar G one can associate a ground rewrite system G' in a natural (and well-known) way. In fact, G' is a regular tree grammar that has the same nonterminals as G (with the same initial nonterminal), and for every production $A \to \alpha_1 \cdots \alpha_k$ of G (where A is a nonterminal and each α_i is either a nonterminal or a terminal) G' has a rule $A \to c_k(\alpha_1, \ldots, \alpha_k)$ where c_k is a (new) terminal symbol of rank k (intuitively standing for the concatenation of k strings), and each α_i has rank 0. It should now be clear that there is a natural one-to-one correspondence between the derivations of G and G', and thus a very close one-to-one relationship between the (usual) derivation trees of G and the derivation trees of G', see the following example. Thus, the derivation trees of ground rewrite systems model the parallelism in derivations in the same way as those of context-free grammars.

Example 5. Consider the context-free grammar G with productions $A \to aAB$, $A \to adB$, $B \to bB$, and $B \to bb$, generating all strings $a^n db^m$ with $n \geq 1$ and $m \geq 2n$ (assuming that A is the initial nonterminal). Then the ground rewrite system (regular tree grammar) G' has rules $A \to c_3(a, A, B)$, $A \to c_3(a, d, B)$, $B \to c_2(b, B)$, and $B \to c_2(b, b)$. Figure 4 shows derivation trees of corresponding derivations of G and G'. Clearly, for a derivation tree t of G, the corresponding derivation tree $\mathrm{der}(t)$ of G' can be obtained recursively as follows:

$$\begin{aligned}
&\mathrm{der}(A(t_1, t_2, t_3)) = \#(A, c_3(\mathrm{der}(t_1), \mathrm{der}(t_2), \mathrm{der}(t_3))) \\
&\mathrm{der}(B(t_1, t_2)) = \#(B, c_2(\mathrm{der}(t_1), \mathrm{der}(t_2))), \text{ and} \\
&\mathrm{der}(x) = x \text{ for } x \in \{a, b, d\}.
\end{aligned}$$

Thus, 'der' is a straightforward linear tree homomorphism. Note that the infix order of the #-labeled nodes of $\mathrm{der}(t)$ corresponds to the (usual) prefix order of the nonterminal nodes of t. Thus, the association between left-most derivations and derivation trees is the same in G' and G. □

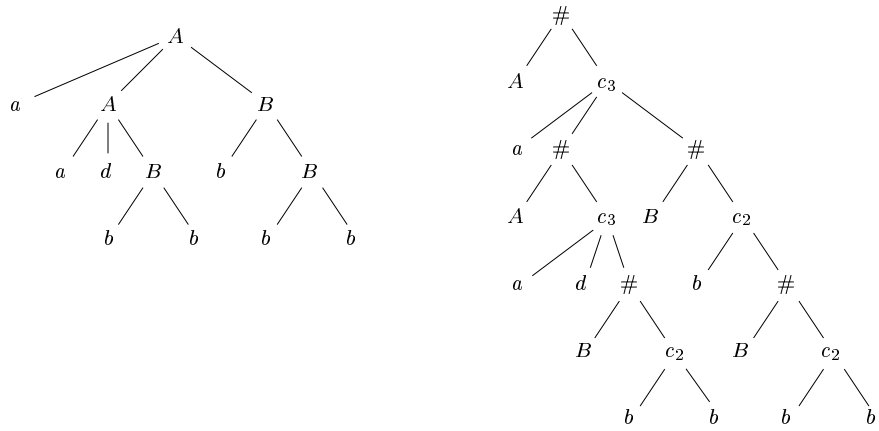

Fig. 4. Derivation trees of a context-free grammar and the corresponding ground rewrite system.

4 New Proofs of Old Results

Using the known closure properties of REGT (as in the proof of Theorem 2), it is now easy to show that the relation $\rightarrow_P^*$ preserves regular tree languages.

Theorem 3. *For every extended ground rewrite system P and every regular tree language R, $\rightarrow_P^*(R)$ is a regular tree language (effectively).*

Proof. By Theorem 1, $\rightarrow_P^*(R) = \{s_2 \mid s_1 \rightarrow_P^* s_2 \text{ for some } s_1 \in R\} = \{s_2 \mid (s_1, s_2) \in \text{trans}(D_P) \text{ for some } s_1 \in R\} = \{\text{right}(t) \mid t \in D_P, \text{left}(t) \in R\}$. Hence $\rightarrow_P^*(R) = \text{right}(D_P \cap \text{left}^{-1}(R))$. Since D_P is regular by Theorem 2, and since REGT is effectively closed under inverse tree homomorphisms, intersection, and linear tree homomorphisms (for the latter, see, e.g., Theorem II.4.16 of [GécSte1]), the result follows. □

The language generated by an extended ground tree grammar $G = (\Delta, \Sigma, P, S)$ is $L(G) = \rightarrow_P^*(S) \cap T_\Delta$. Since every finite tree language S is regular and REGT is closed under intersection with T_Δ, the (slight extension of the) main result of [Bra] follows immediately from Theorem 3.

Theorem 4. *For every extended ground tree grammar G a regular tree grammar G' with $L(G') = L(G)$ can effectively be constructed.*

It was shown in Theorem 3.21 of [Bra] that in a ground tree grammar $G = (\Delta, \Sigma, P, S)$ one can also allow the set S to be a regular tree language. This means that, in fact, Theorem 3 was also proved in [Bra] (for ordinary ground rewrite systems).

Using Theorem 3 (and Theorem 1) it is now straightforward to prove result II.5 of [DauTis1] (see also Proposition 2 of [DHLT], which, however, does not show effectivity).

Theorem 5. *For every extended ground rewrite system P a ground tree transducer Q such that $\rightarrow_P^* = \Rightarrow_Q$, can effectively be constructed.*

Proof. Let P be an extended ground rewrite system over Σ. By $\leftarrow_P^*$ we denote the relation $(\rightarrow_P^*)^{-1}$. It is easy to see that this is the relation $\rightarrow_{P^{-1}}^*$, where P^{-1} is the extended ground rewrite system $\{(R, L) \mid (L, R) \in P\}$. This shows, by Theorem 3, that $\leftarrow_P^* (L)$ is (effectively) regular for every regular tree language L.

Define $Q = \{(\leftarrow_P^* (L), \rightarrow_P^* (R)) \mid (L, R) \in P\}$. Then Q is a ground tree transducer (effectively), by Theorem 3. We first show that $\rightarrow_P^* \subseteq \Rightarrow_Q$. Let $s \rightarrow_P^* s'$. By Theorem 1 there is a tree $t \in D_P$ such that $\mathrm{left}(t) = s$ and $\mathrm{right}(t) = s'$. The derivation tree $t \in T_{\Sigma\#}$ can be decomposed (in a unique way) as $t = c[\#(t_1, t_1'), \ldots, \#(t_k, t_k')]$ where c is a k-place context (for some $k \geq 0$) that does not contain $\#$, and the t_i, t_i' are derivation trees of P such that $\mathrm{right}(t_i) \rightarrow \mathrm{left}(t_i')$ is a rule of P. Define, for $1 \leq i \leq k$, the trees $p_i = \mathrm{left}(t_i)$ and $p_i' = \mathrm{right}(t_i')$ over Σ. Since $t_i, t_i' \in D_P$, it follows from Theorem 1 that $p_i \in \leftarrow_P^* (\mathrm{right}(t_i))$ and $p_i' \in \rightarrow_P^* (\mathrm{left}(t_i'))$. Hence, since $\mathrm{right}(t_i) \rightarrow \mathrm{left}(t_i')$ is a rule of P, $p_i \in \leftarrow_P^* (L)$ and $p_i' \in \rightarrow_P^* (R)$ for some $(L, R) \in P$. Thus, $p_i \rightarrow p_i'$ is a rule of Q. Since clearly $s = \mathrm{left}(t) = c[p_1, \ldots, p_k]$ and $s' = \mathrm{right}(t) = c[p_1', \ldots, p_k']$, this shows that $s \Rightarrow_Q s'$.

We note here that an experienced reader can easily give the above proof without the use of derivation trees, i.e., without the use of Theorem 1: if $s \rightarrow_P^* s'$ then, obviously, there exist a k-place context c and trees p_i, p_i', u_i, u_i' such that $s = c[p_1, \ldots, p_k]$, $s' = c[p_1', \ldots, p_k']$, $p_i \rightarrow_P^* u_i \rightarrow_P u_i' \rightarrow_P^* p_i'$, and $u_i \rightarrow u_i'$ in P (which shows $s \Rightarrow_Q s'$). However, the above proof illustrates that derivation trees can be used to give precise formal proofs of such obvious statements.

The proof of the inclusion $\Rightarrow_Q \subseteq \rightarrow_P^*$ is even easier. Given a context c and trees $p_i \in \leftarrow_P^* (L)$ and $p_i' \in \rightarrow_P^* (R)$ (for some $(L, R) \in P$ depending on i) such that $s = c[p_1, \ldots, p_k]$ and $s' = c[p_1', \ldots, p_k']$, there are derivations $p_i \rightarrow_P^* u_i$ and $u_i' \rightarrow_P^* p_i'$ such that $u_i \rightarrow u_i'$ is a rule of P. Hence $p_i \rightarrow_P^* p_i'$ and so $s = c[p_1, \ldots, p_k] \rightarrow_P^* c[p_1', \ldots, p_k'] = s'$. Note that if t_i and t_i' are derivation trees of $p_i \rightarrow_P^* u_i$ and $u_i' \rightarrow_P^* p_i'$, respectively, then $c[\#(t_1, t_1'), \ldots, \#(t_k, t_k')]$ is a derivation tree of $s \rightarrow_P^* s'$. □

Theorem 5 is equivalent to saying that the class of ground tree transductions is closed under star (as shown in [DauTis1, DHLT]), because for every ground tree transducer P, $\Rightarrow_P^* = \rightarrow_P^*$.

It is rather obvious that ground tree transducers also have "derivation trees".

Lemma 6. *For every ground tree transducer P over Σ there is a regular tree language D over $\Sigma\#$ such that $\mathrm{trans}(D) = \Rightarrow_P$ (effectively).*

Proof. Define D to be the set of all trees $t \in T_{\Sigma\#}$ such that for every subtree $\#(t_1, t_2)$ of t, $(t_1, t_2) \in (L, R)$ for some $(L, R) \in P$. Note that in such a tree there are no nested occurrences of $\#$. It should be clear that $\mathrm{trans}(D) = \Rightarrow_P$. To show that D is (effectively) regular, let D' be the regular tree language that is the union of all $\#(L, R)$, for $(L, R) \in P$. Then D is the set of all trees

$c[t_1, \ldots, t_k]$ where c is a k-place context (for some $k \geq 0$) and $t_i \in D'$. From this it easily follows that D is regular (to be precise, $D = T_{\Sigma \cup \{x_1\}} \cdot_{x_1} D'$, see, e.g., Theorem II.4.6 of [GécSte1]). □

This lemma immediately gives us the following result, by exactly the same proof as the one of Theorem 3.

Theorem 7. *For every ground tree transducer P and every regular tree language R, $\Rightarrow_P(R)$ is a regular tree language (effectively).*

Note that Theorem 3 follows from Theorems 5 and 7, as shown in Proposition 3.2 of [CoqGil], and in Lemmas 3.6 and 3.7 of [FülVág] for the case that $\rightarrow_P^*$ is a congruence. Thus, Theorems 3 and 5 are quite closely related.

As observed in Section 2, the above results hold in particular for monadic ranked alphabets, in which case they concern strings rather than trees: ground rewrite systems correspond to the regular canonical systems of [Büc] (which are Chomsky type 0 grammars of which the productions are applied to prefixes of the sentential forms only), and regular tree languages correspond to regular string languages. Thus, Theorem 4 expresses the (extended version of the) main result of [Büc]: for every regular canonical system an equivalent left-linear grammar (or right-linear grammar, or finite automaton) can effectively be constructed (for other proofs see, e.g., Section 2.3 of [Sal], or see [FraPag]). This result is equivalent with the well-known fact that the possible contents of a pushdown automaton form a regular language (see [Gre] and, e.g., page 335 of [Har]). The results of [Büc] and [Bra] were rediscovered in [DauTis1, DHLT] (in the sense of the above-mentioned close relationship between Theorems 3 and 5), in [FülVág], and in [FraPag]. Complexity issues are considered in [CoqGil, FraPag, Vág]. Theorem 3 is generalized to linear semi-monadic term rewriting systems in Theorem 5.1 of [CDGV]; it is not clear whether the notion of derivation tree is relevant to this generalization.

In the remainder of this section we discuss confluence and termination of ground rewrite systems.

An extended ground rewrite system P over Σ is *confluent* if for all trees $t, u, v \in T_\Sigma$ with $t \rightarrow_P^* u$ and $t \rightarrow_P^* v$, there is a tree $w \in T_\Sigma$ such that $u \rightarrow_P^* w$ and $v \rightarrow_P^* w$. In [DauTis1, DHLT, DauTis2] it is shown on the basis of Theorem 5 that confluence is decidable for extended ground rewrite systems. The nicest proof is the one in [DauTis2], where it is even shown that the first-order theory of extended ground rewrite systems is decidable. This first-order theory includes properties such as confluence (as should be clear from the above standard definition) and unique normalization (i.e., for every $s \in T_\Sigma$ there is a unique $t \in T_\Sigma$ such that $s \rightarrow_P^* t$ and there is no $u \in T_\Sigma$ with $t \rightarrow_P u$). The essence of the proof in [DauTis2] is that for every extended ground rewrite system P, $\rightarrow_P^*$ is a so-called binary RR relation (introduced in [DauTis2] and shown to have a decidable first-order theory). In view of Theorem 5, it is in fact proved that every ground tree transduction is a binary RR relation. We now wish to convince the reader who is familiar with [DauTis2], that the proof can as well be based on Theorems 1 and 2, instead of on Theorem 5.

Let Non denote the set of all trees $t \in T_{\Sigma\#}$ that do not have nested occurrences of # (i.e., for every subtree $\#(t_1, t_2)$ of t, t_1 and t_2 are in T_Σ).

Lemma 8. *For every regular tree language R over $\Sigma\#$ there is (effectively) a regular tree language R' over $\Sigma\#$ such that* $\mathrm{trans}(R') = \mathrm{trans}(R)$ *and* $R' \subseteq \mathrm{Non}$.

Proof. Let 'prune' be the mapping from $T_{\Sigma\#}$ to $T_{\Sigma\#}$, defined recursively as follows, where σ is an element of Σ of rank k, and $t_i \in T_{\Sigma\#}$:

$$\mathrm{prune}(\sigma(t_1, \ldots, t_k)) = \sigma(\mathrm{prune}(t_1), \ldots, \mathrm{prune}(t_k)), \text{ and}$$
$$\mathrm{prune}(\#(t_1, t_2)) = \#(\mathrm{left}(t_1), \mathrm{right}(t_2)).$$

Clearly, for every $t \in T_{\Sigma\#}$, $\mathrm{trans}(\mathrm{prune}(t)) = \mathrm{trans}(t)$, and # is not nested in $\mathrm{prune}(t)$. Thus, $\mathrm{trans}(\mathrm{prune}(R)) = \mathrm{trans}(R)$ and $\mathrm{prune}(R) \subseteq \mathrm{Non}$. From the recursive definition of 'prune' (and 'left' and 'right') it is immediate that 'prune' is a linear top-down tree transduction (see, e.g., Chapter IV of [GécSte1] where top-down tree transducers are called root-to-frontier tree transducers). Since REGT is effectively closed under linear top-down tree transductions (see Corollary IV.6.6 of [GécSte1]), $\mathrm{prune}(R)$ is regular. Thus, $R' = \mathrm{prune}(R)$ satisfies the requirements.

We observe here that for every extended ground rewrite system P, $\mathrm{prune}(D_P)$ is the set of derivation trees (as defined in the proof of Lemma 6) of the ground tree transducer Q defined in the proof of Theorem 5. □

Tree transductions of the form $\mathrm{trans}(R)$, where R is a regular tree language consisting of trees that do not have nested occurrences of #, are just a slight generalization of ground tree transductions (cf. the proof of Lemma 6). Thus, it is straightforward to generalize the proof of the Lemma in Section 5 of [DauTis2], which shows that every ground tree transduction is a binary RR relation, to a proof that every transduction $\mathrm{trans}(R)$ with $R \subseteq \mathrm{Non}$ is a binary RR relation. Together with Lemma 8, this gives the following proposition.

Proposition 9. *For every regular tree language R over $\Sigma\#$,* $\mathrm{trans}(R)$ *is a binary RR relation (effectively).*

Clearly, Proposition 9 and Theorems 1 and 2 imply that $\to_P^*$ is a binary RR relation for every extended ground rewrite system P.

Finally we discuss the termination problem of extended ground rewrite systems. Termination does not seem to be expressible in the first-order theory of ground rewriting (cf. [DauTis2]). However, its decidability is much easier to show than that of confluence. An extended ground rewrite system P is *finitely terminating* or *noetherian* if there does not exist an infinite derivation $t_1 \to_P t_2 \to_P t_3 \to_P \cdots$. As mentioned in [HueOpp], decidability of the noetherian property for (ordinary) ground rewrite systems is shown in [HueLan]; a proof that uses ground tree transducers is given in V.3 of [DauTis1].

Theorem 10. *It is decidable for an extended ground rewrite system P whether or not P is finitely terminating.*

Proof. Let P be an extended ground rewrite system over Σ. We say that a tree $t \in T_\Sigma$ is nonterminating if there is an infinite derivation $t_1 \to_P t_2 \to_P t_3 \to_P \cdots$ that starts with t, i.e., $t = t_1$. We first prove (by structural induction on t) that if t is nonterminating then there exist a 1-place context c and a rule $u \to v$ of P such that $t \to_P^* c[u]$ and v is nonterminating. If, in the above infinite derivation, $t_i \to t_{i+1}$ is a rule of P, for some $i \geq 1$, then the statement obviously holds (with the empty context x_1). Otherwise, the roots of $t_1, t_2, \ldots$ remain unchanged, which implies that there must be an infinite derivation that starts with a proper subtree s_j of $t = \sigma(s_1, \ldots, s_k)$. Then, by induction, $s_j \to_P^* c'[u]$ and v is nonterminating, for some context c' and rule $u \to v$. Clearly, for this rule and context $c = \sigma(s_1, \ldots, c', \ldots, s_k)$ the statement holds.

Suppose that P is not finitely terminating. Then, by repeated application of the above argument, there is an infinite sequence $u_1 \to v_1, u_2 \to v_2, u_3 \to v_3, \ldots$ of rules of P such that for every i, $v_i \to_P^* c[u_{i+1}]$ for some 1-place context c. This implies that for all i, j with $i < j$ there is a 1-place context c such that $v_i \to_P^* c[u_j]$. Now let $(L_i, R_i) \in P$ with $u_i \in L_i$ and $v_i \in R_i$, for all i. Since P is finite, there exist $i < j$ such that $(L_i, R_i) = (L_j, R_j)$. This shows the existence of a rule $u \to v$ (viz. $u_j \to v_i$) such that $v \to_P^* c[u]$ for some context c. In the other direction, the existence of such a rule clearly implies that P is not finitely terminating: $v \to_P^* c[u] \to_P c[v] \to_P^* c[c[u]] \to_P \cdots$.

Thus, P is finitely terminating if and only if $\to_P^*(R) \cap \mathrm{exsub}(L) = \emptyset$ for all $(L, R) \in P$, where $\mathrm{exsub}(L)$ is the set of all trees that have at least one subtree in L (i.e., all $c[u]$ with $u \in L$). By Theorem 3, $\to_P^*(R)$ is a regular tree language. Clearly, $\mathrm{exsub}(L)$ is a regular tree language (e.g., $\mathrm{exsub}(L) = T_\Sigma - \mathrm{allsub}(T_\Sigma - L)$ and REGT is closed under complementation; for 'allsub' see the proof of Theorem 2). Hence, by the closure of REGT under intersection, $\to_P^*(R) \cap \mathrm{exsub}(L)$ is a regular tree language. Thus, since the emptiness problem is decidable for regular tree languages (see, e.g., Theorem II.10.2 of [GécSte1]), the above property can be decided. □

Acknowledgement

I thank Jurriaan Hage for his comments on a previous version of this paper. I thank the referee for several constructive remarks. I am grateful to Sophie Tison for Theorem 10 and its proof.

References

[Bra] W.S.Brainerd; Tree generating regular systems, Inform. and Control 14 (1969), 217–231

[Büc] J.R.Büchi; Regular canonical systems, Archiv für Math. Logik und Grundlagenforschung 6 (1964), 91–111

[CDGV] J.L.Coquidé, M.Dauchet, R.Gilleron, S.Vágvölgyi; Bottom-up tree pushdown automata: classification and connection with rewrite systems, Theor. Comput. Sci. 127 (1994), 69–98

[CoqGil] J.L.Coquidé, R.Gilleron; Proofs and reachability problem for ground rewrite systems, in *Aspects and Prospects of Theoretical Computer Science* (J.Dassow, J.Kelemen, eds.), Lecture Notes in Computer Science 464, Springer-Verlag, Berlin, 1990, pp.120–129

[DauTis1] M.Dauchet, S.Tison; Decidability of confluence for ground term rewriting systems, Proc. FCT'85 (L.Budach, ed.), Lecture Notes in Computer Science 199, Springer-Verlag, Berlin, 1985, pp.80–89

[DauTis2] M.Dauchet, S.Tison; The theory of ground rewrite systems is decidable, Proc. 5th Ann. IEEE Symp. on Logic in Computer Science (LICS), Philadelphia, 1990, pp.242–248

[DHLT] M.Dauchet, T.Heuillard, P.Lescanne, S.Tison; Decidability of the confluence of finite ground term rewrite systems and of other related term rewrite systems, Inform. and Comput. 88 (1990), 187–201

[FraPag] M.Frazier, C.D.Page; Prefix grammars: an alternative characterization of the regular languages, Inf. Proc. Letters 51 (1994), 67–71

[FülVág] Z.Fülöp, S.Vágvölgyi; Congruential tree languages are the same as recognizable tree languages - a proof for a theorem of D.Kozen, Bulletin of the EATCS 39 (1989), 175–185

[GécSte1] F.Gécseg, M.Steinby; *Tree Automata*, Akadémiai Kiadó, Budapest, 1984

[GécSte2] F.Gécseg, M.Steinby; Tree Languages, Chapter 1 of the *Handbook of Formal Languages*, Volume 3: *Beyond Words*, G.Rozenberg and A.Salomaa, eds., Springer-Verlag, Berlin, 1997

[Gre] S.A.Greibach; A note on pushdown store automata and regular systems, Proc. Amer. Math. Soc. 18 (1967), 263–268

[Har] M.A.Harrison; *Introduction to Formal Language Theory*, Addison-Wesley, Reading, Mass., 1978

[HueLan] G.Huet, D.S.Lankford; On the uniform halting problem for term rewriting systems, Rapport Laboria 283, IRIA, 1978

[HueOpp] G.Huet, D.C.Oppen; Equations and rewrite rules - a survey, in *Formal Language Theory: Perspectives and Open Problems* (R.V.Book, ed.), Academic Press, New York, 1980, pp.349–405

[Oya] M.Oyamaguchi; The Church-Rosser property for ground term-rewriting systems is decidable, Theor. Comput. Sci. 49 (1987), 43–79

[Sal] A.Salomaa; *Computation and Automata*, Cambridge University Press, Cambridge, 1985

[Vág] S.Vágvölgyi; A fast algorithm for constructing a tree automaton recognizing a congruential tree language, Theor. Comput. Sci. 115 (1993), 391–399

This article was processed using the LaTeX macro package with LLNCS style