Decidability Questions for Term Rewriting Systems and Tree Transducers

Ph.D. Dissertation

by

Pál Gyenizse

József Attila University
Department of Computer Science
Szeged, Árpád tér 2
H-6720 Hungary

1998

Contents

List of Figures

List of Tables

Introduction

Term rewriting systems are finite devices processing terms over ranked alphabets. They play a very important role in theoretical computer science. They are attractive tools because of their simple syntax and semantics. This simplicity facilitates a satisfactory mathematical analysis. On the other hand, term rewriting systems have the full power of Turing machines.

The notion of term rewriting systems is paradigmatic for study of many theoretical areas. For instance, it is very useful and fruitful in investigation of the λ-calculus, denotational semantics, mechanical theorem-proving, and symbolic algebraic computation. Many other applications of term rewriting systems can be found in N. Dershowitz and J.P Jouannaud's work ([13]), J.W. Klop's work ([47]), and R.V. Book and F. Otto's book ([4]).

In this dissertation, we study decidability questions concerning term rewriting systems. In general, most of the important properties of term rewriting systems are undecidable, for example such properties are the termination and the confluence. Many other properties of term rewriting systems are shown to be undecidable in the above mentioned works ([13], [47], [4]) and G. Huet's fundamental paper ([46]). In spite of these undecidability results, several interesting results on decidable properties were obtained for special kinds of term rewriting systems. For instance, for ground term rewriting systems, most of the significant properties are decidable (termination, confluence and so on), see [11], [12], [16], [47], [52]. A term rewriting system is called ground if its rules consist of ground terms.

In a part of this work, we shall investigate a special kind of term rewriting systems: term rewriting systems which preserve recognizability. A term rewriting system preserves recognizability if, for any recognizable tree language L, the set of descendants of trees being in L is also recognizable. A descendant of a tree is obtained from the tree by applying the rules of the term rewriting system to the tree successively. The term rewriting systems which preserve recognizability are very attractive tools because the use of an algebraic tool (regular tree languages) can clarify and simplify some proofs concerning such systems. For example, regular tree languages are suitable when we want to avoid the non-linearity. Several types of term rewriting systems preserving recognizability were defined by J.H. Gallier and R.V. Book in [38], J.L. Coquidé et al in [7], and P. Gyenizse and S. Vágvölgyi in [45]. Moreover, see K. Salomaa's paper ([54]), R. Gilleron and S. Tison's survey ([43]), and F. Otto's work ([51]) for relevant results. Similar results were considered in F. Gécseg's work ([39]) and Z. Ésik's works ([21], [22]) for tree transducers, which are also special term rewriting systems.

A string rewriting system can also be considered as a special term rewriting system. This is a device that processes strings like a term rewriting system processes terms. Many interesting results for string rewriting systems preserving recognizability were explored and carried over from term rewriting systems preserving recognizability. A good survey of these results can be found in R.V. Book and F. Otto's book ([4]), F. Otto's work ([51]), and P. Gyenizse and S. Vágvölgyi's work ([45]).

In the other part of this dissertation, we are also going to consider top-down tree transducers and bottom-up tree transducers. Top-down and bottom-up tree transducers have been studied since the early seventies. First, W.C. Rounds and J.W. Thatcher introduced the notion of a top-down tree transducer in [53], [56]. Then, J.W. Thatcher defined the concept of a bottom-up tree transducer in [57]. Later on, J. Engelfriet introduced the notion of a top-down tree transducer with regular look-ahead in [15] in order to increase the transformational capacity.

There are still other tree transducers, e.g., macro tree transducers ([17], [37]), attributed tree transducers ([23], [37]), macro attributed tree transducers ([37], [49]), high level tree transducers ([18]), modular tree transducers ([19]), high level modular tree transducers ([59]), however, we will not deal with them in this work.

Using top-down and bottom-up transducers, abstract and formal models of the syntax-directed translation method can be given, which is a widespread way of specifying the semantics of high level programming languages.

Some restricted types of top-down and bottom-up tree transducers (such as deterministic, total, linear, nondeleting etc.) were defined and compared with each other with respect to transformational capacity in the works of

W.C. Rounds ([53]), J.W. Thatcher ([56]), J. Engelfriet ([14], [15]), and B.S. Baker ([1], [2], [3]). Moreover, F. Gécseg and M. Steinby gave a survey of tree languages and tree transducers in [40], and [41]. Recently, G. Dányi and Z. Fülöp defined and investigated superlinear deterministic top-down tree transducers in [8].

Tree transducers induce tree transformations, which are binary relations over trees. Moreover, a tree transformation class is a class consisting of tree transformations. Since tree transformations are binary relations, the operation composition is defined for them as for binary relations. In this dissertation, we shall work only with deterministic tree transducers, which induce partial functions.

The compositions and decompositions of tree transformation classes are fundamental in the theory of tree transducers and tree transformations. Many proofs became simpler and more obvious by using results on compositions and decompositions. Therefore, they were investigated in a large number of papers. For example, B.S. Baker ([3]), J. Engelfriet ([14], [15]), Fülöp ([24]), Fülöp and Vágvölgyi ([28], [29], [30], [32], [33], [34], [35], [36]), Gécseg and Steinby ([40]), Gyenizse and Vágvölgyi ([44]), Slutzki and Vágvölgyi ([55]), Dányi and Fülöp ([8], [9]), and Fülöp and Vogler ([37]) studied the compositions and decompositions of different types of tree transformations.

The subject of this dissertation is to study some decidability questions of term rewriting systems and tree transducers. Our results can be summarized as follows.

(1) We introduce the concept of a generalized semi-monadic term rewriting system. We show that linear generalized semi-monadic term rewriting systems effectively preserve recognizability and we give several decidability and undecidability results on term rewriting systems effectively preserving recognizability.

(2) We prove that the injectivity problem of linear deterministic top-down tree transducers is decidable and that the same problem is undecidable for (nonlinear) homomorphism tree transducers.

(3) We give a linear time algorithm to determine the correct inclusion relationship between two tree transformation classes which are compositions of some "fundamental" tree transformation classes taken from the set $\{DT^R, LDT^R, DT, LDT, DB, LDB, H, LH\}$.

Here DB, DT, H stand for the class of deterministic bottom-up tree transformations, the class of deterministic top-down tree transformations, and the class of homomorphism tree transformations, respectively. Moreover, the prefix L stands for the restriction linear, and the superscript R stands for the regular look-ahead.

The dissertation consists of five chapters, of which the contents are the following.

The opening chapter presents the common preparatory notion, notation and terminology, focusing our attention on trees. It also surveys those types of rewriting systems and tree transducers which are used in this work.

Then we present our results in detail as follows.

In Chapter 2, we introduce further notions and notation, which are used only in this chapter. Then, we define the notion of the generalized semi-monadic term rewriting system which is a generalization of well-known term rewriting systems: the ground term rewriting system, the monadic term rewriting system, and the semi-monadic term rewriting system. As a main result, we show that linear generalized semi-monadic term rewriting systems effectively preserve recognizability. (We note, this is the largest known class of term rewriting systems that preserves recognizability.) Furthermore, we prove that a tree language L is recognizable if and only if there exists a term rewriting system R such that $R \cup R^{-1}$ is a linear generalized semi-monadic term rewriting system and that L is the union of finitely many $\leftrightarrow_R^*$-classes. We also show several decidability and undecidability results on term rewriting systems effectively preserving recognizability and on generalized semi-monadic term rewriting systems. Namely, we show that for a term rewriting system R effectively preserving recognizability, it is decidable if R is locally confluent. Moreover, we show that preserving recognizability and effectively preserving recognizability are modular properties of linear collapse-free term rewriting systems. (A property $\mathcal{P}$ is modular for a class of term rewriting systems [which is closed under disjoint union] if the disjoint union of two term rewriting systems R and S from this class has the property $\mathcal{P}$ if and only if both R and S have the property $\mathcal{P}$.) Finally, as a consequence, we obtain that restricted right-left overlapping string rewriting systems effectively preserve recognizability.

In Chapter 3, we give a simple proof for the decidability of injectivity of linear deterministic top-down tree transducers. Moreover, we show that injectivity is undecidable even for homomorphism tree transducers.

In Chapter 4, we show that $DT^R = DT \circ LDB$, that $LDT^R \not\subseteq LDT \circ DB$, and that $DT \not\subseteq LDT^R \circ H$, where the composition of tree transformation classes X and Y is denoted by $X \circ Y$. Using these results and the composition and inclusion results of Engelfriet ([14], [15]), Fülöp ([24]), and Fülöp and Vágvölgyi ([28], [29], [30], [32], [33], [34], [35], [36]) we show that the problem of determining the correct inclusion relationship between two tree transformation classes which are compositions of some "fundamental" tree transformation classes taken from the set $\{ DT^R, LDT^R, DT, LDT, DB, LDB, H, LH \}$ can be solved in linear time.

Finally, we summarize the results of the whole dissertation and mention some open problems regarding term rewriting systems preserving recognizability.

This dissertation is strongly based on the papers [26], [44], and [45]. All results presented here appear in these works.

Chapter 1

General notions and notation

1.1 Sets and relations

In this section we recall the necessary notions and notation concerning sets, relations, and functions.

The set of *nonnegative integers* is denoted by N.

For a set A, we write $Pow(A)$ and $|A|$ for the *power set* and the *cardinality* of A, respectively.

Given two sets A and B, $A \subseteq B$ means that A is a *subset* of B, $A \subset B$ stands for that A is a *proper subset* of B, and $A \not\subseteq B$ denotes that A is *not a subset* of B. Moreover, we denote by $A \bowtie B$ that A and B are *incomparable* with respect to inclusion. We write $A \times B$ and $A - B$ for the *Cartesian product* of A and B and the *difference* of A and B, respectively.

Any subset ρ of the Cartesian product $A \times B$ is called a (binary) *relation* from A to B. We also write $a\rho b$ instead of $(a, b) \in \rho$. The set $dom(\rho) = \{a \mid a\rho b \text{ for some } b \in B\}$ is called the *domain* of ρ, and the set $ran(\rho) = \{b \mid a\rho b \text{ for some } a \in A\}$ is the *range* of ρ.

The relation $\rho^{-1} = \{(b, a) \mid a\rho b\}$ is the *inverse* of ρ.

Let ρ be a relation from A to B, and let τ be a relation from B to C. Then the relation $\rho \circ \tau$ from A to C is defined by $\rho \circ \tau = \{(a, c) \mid a\rho b \text{ and } b\tau c \text{ for some } b \in B\}$ and is called the *composition* of ρ and τ. If Y, Z are classes of relations, then $Y \circ Z = \{\rho \circ \tau \mid \rho \in Y \text{ and } \tau \in Z\}$.

A relation from A to A is called a *relation on A* or *over A*. The *identity relation* over A is $Id(A) = \{(a, a) \mid a \in A\}$. Let ρ be a relation on A. The *n-fold compositions* of ρ is defined by the following induction: $\rho^0 = Id(A)$ and $\rho^n = \rho \circ \rho^{n-1}$, for $n > 0$. Moreover, let Z be a class of relations on A. Then, $Z^1 = Z$ and for any $n > 1$, $Z^n = Z \circ Z^{n-1}$.

The *transitive closure* and the *reflexive, transitive closure* of a relation ρ over A are the relations $\rho^+ = \bigcup_{n\geq 1} \rho^n$ and $\rho^* = \bigcup_{n\geq 0} \rho^n$, respectively. Moreover, the *reflexive, symmetric and transitive closure* of ρ is $\bigcup_{n\geq 0}(\rho \cup \rho^{-1})^n$. Clearly, the reflexive, symmetric and transitive closure of ρ is an equivalence relation.

A *partial function* ρ from A to B is a relation from A to B such that for every $a \in A$ there exists at most one $b \in B$ such that $a\rho b$. When this b exists, we denote it by $\rho(a)$. For a subset $A' \subseteq A$, we put $\rho(A') = \{b \in B \mid b = \rho(a)$ for some $a \in A'\}$. A partial function ρ is *total* if $dom(\rho) = A$. Moreover, a total function is called a *function* or a *mapping*. Finally, a partial function ρ from A to B is called *injective* if, for every $a, b \in A$, such that $\rho(a)$ and $\rho(b)$ exist the condition $\rho(a) = \rho(b)$ implies $a = b$.

1.2 Strings and trees

This section contains our notions and notation for strings and trees. First, we introduce the basic concepts of strings.

An *alphabet* Σ is a finite, nonempty set of symbols. A *string* or *word* over Σ is a finite sequence of elements of Σ. The *empty string* is denoted by λ. For strings u and v over Σ, we denote the *concatenation* of u and v by uv or $u \cdot v$. The set of all strings over Σ is denoted by Σ^*. It is well-known that Σ^*, equipped with the operation concatenation, is a monoid of which the unit element is λ. An equivalence relation ρ over Σ^* is called a *congruence* over Σ^* if, for any $u_1, u_2, v_1, v_2 \in \Sigma^*$, $u_1 \rho u_2$ and $v_1 \rho v_2$ imply $u_1 v_1 \rho u_2 v_2$. Any subset of Σ^* is called a *language*. Suppose that $w_1, w_2, w_3 \in \Sigma^*$ are such that $w_1 = w_2 w_3$. Then we say that w_1 is an *extension* of w_2, that w_2 is a *prefix* of w_1, and that w_3 is a *suffix* of w_1. Moreover, if $w_1 \neq w_2$, then w_2 is *proper prefix* of w_1. The *length* of a string $w \in \Sigma^*$ is denoted by $|w|$ and is defined by the following induction:

(i) if $w = \lambda$, then $|w| = 0$;

(ii) if $w = va$ for some $v \in \Sigma^*$ and $a \in \Sigma$, then $|w| = |v| + 1$.

We now collect the concepts for terms or trees which will be used. A *ranked alphabet* Σ is an alphabet in which every symbol has a unique rank in N. For $m \geq 0$, Σ_m denotes the set of all elements of Σ which have rank m. Let Σ and Δ be ranked alphabets. We say that $\Sigma \subseteq \Delta$ if and only if $\Sigma_m \subseteq \Delta_m$ for each $m \geq 0$. For $f \in \Sigma$ we write shortly $f^{(m)}$ to mean that $f \in \Sigma_m$.

For a ranked alphabet Σ and a set H, disjoint with Σ, the set $T_\Sigma(H)$ of *trees* or *terms* over Σ indexed by H is the smallest set U satisfying the following two conditions:

(i) $\Sigma_0 \cup H \subseteq U$,

(ii) $f(t_1, \ldots, t_m) \in U$ whenever $m \geq 1$, $f \in \Sigma_m$ and $t_1, \ldots, t_m \in U$.

The set $T_\Sigma(\emptyset)$ is written as T_Σ and its elements are called *ground terms.*

Let Σ and Δ be two ranked alphabets. Any subset of T_Σ is called a *tree language* and any relation from T_Σ to T_Δ is a *tree transformation* from T_Σ to T_Δ. We denote by I the tree transformation class consisting of all identity tree transformations (i.e., $I = \{Id(T_\Sigma) \mid \Sigma \text{ is a ranked alphabet }\}$).

We need a countable set $X = \{x_1, x_2, \ldots\}$ of *variables* which will be kept fixed in this dissertation. We suppose that $\Sigma \cap X = \emptyset$ for each ranked alphabet Σ. Moreover, we put $X_m = \{x_1, \ldots, x_m\}$, for $m \geq 0$. Hence $X_0 = \emptyset$.

A tree $t \in T_\Sigma(X)$ is *linear* if each variable of X occurs at most once in t. Moreover, $\hat{T}_\Sigma(X_m)$ is the set of linear trees in $T_\Sigma(X_m)$.

For every $m \geq 1$, we distinguish a subset $\bar{T}_\Sigma(X_m)$ of $\hat{T}_\Sigma(X_m)$: a tree $t \in \hat{T}_\Sigma(X_m)$ is in $\bar{T}_\Sigma(X_m)$ if and only if each variable in X_m appears exactly once in t and the order of the variables from left to right in t is $x_1, \ldots, x_m$. For example, if $\Sigma = \Sigma_0 \cup \Sigma_2$ with $\Sigma_0 = \{\sharp\}$ and $\Sigma_2 = \{f\}$, then $f(x_1, f(\sharp, x_1)) \in T_\Sigma(X_1)$ but $f(x_1, f(\sharp, x_1)) \notin \bar{T}_\Sigma(X_1)$ and thus $f(x_1, f(\sharp, x_1)) \notin \hat{T}_\Sigma(X_1)$. Furthermore, $f(x_2, f(\sharp, x_1)) \in \hat{T}_\Sigma(X_2)$ but $f(x_2, f(\sharp, x_1)) \notin \bar{T}_\Sigma(X_2)$. However, $f(x_1, f(\sharp, x_2)) \in \bar{T}_\Sigma(X_2)$.

Let Σ be a ranked alphabet. Let $f \in \Sigma_1$, $t \in T_\Sigma$ be arbitrary. The tree $f^k(t) \in T_\Sigma$, $k \geq 0$, is defined by induction: $f^0(t) = t$, and $f^{k+1}(t) = f(f^k(t))$ for $k \geq 0$.

For a tree $t \in T_\Sigma(X)$, we define the *root* of t, the *height* of t, the set of *subtrees* of t, the set of *paths* of t, the *longest leftmost path* of t, the *longest rightmost path* of t, and the *set of variables* of t. They are defined by the functions

$$\begin{array}{lcl} root & : & T_\Sigma(X) \rightarrow \Sigma \cup X, \\ height & : & T_\Sigma(X) \rightarrow N, \\ sub & : & T_\Sigma(X) \rightarrow Pow(T_\Sigma(X)), \\ path & : & T_\Sigma(X) \rightarrow Pow(N^*), \\ llp & : & T_\Sigma(X) \rightarrow N^*, \\ lrp & : & T_\Sigma(X) \rightarrow N^*, \\ var & : & T_\Sigma(X) \rightarrow Pow(X), \end{array}$$

respectively, where, for every $t \in T_\Sigma(X)$, their values are defined by induction as follows:

(i) if $t \in \Sigma_0 \cup X$, then

$$\begin{array}{lcl} root(t) & = & t, \\ height(t) & = & 0, \\ sub(t) & = & \{t\}, \\ path(t) & = & \{\lambda\}, \\ llp(t) & = & \lambda, \\ lrp(t) & = & \lambda, \\ var(t) & = & \emptyset, \text{ if } t \in \Sigma_0 \text{ and } var(t) = \{t\}, \text{ if } t \in X; \end{array}$$

(ii) if $t = f(t_1, \ldots, t_m)$ with $m \geq 1$ and $f \in \Sigma_m$, then

$$\begin{array}{lcl} root(t) & = & f, \\ height(t) & = & 1 + max\{height(t_i) \mid 1 \leq i \leq m\}, \\ sub(t) & = & \{t\} \cup (\cup_{i=1}^m sub(t_i)), \\ path(t) & = & \{\lambda\} \cup \{i\alpha \mid 1 \leq i \leq m \text{ and } \alpha \in path(t_i)\}, \\ llp(t) & = & 1 \cdot llp(t_1), \\ lrp(t) & = & m \cdot lrp(t_m), \\ var(t) & = & \cup_{i=1}^m var(t_i). \end{array}$$

We note that $height(t) = max\{|\alpha| \mid \alpha \in path(t)\}$.

For each $t \in T_\Sigma(X)$ and $\alpha \in path(t)$, we introduce the *subtree* $t/\alpha \in sub(t)$ of t at α as follows:

(i) for $t \in \Sigma_0 \cup X$, $t/\lambda = t$;

(ii) for $t = f(t_1, \ldots, t_m)$ with $m \geq 1$ and $f \in \Sigma_m$, if $\alpha = \lambda$ then $t/\alpha = t$, otherwise, if $\alpha = i\beta$ for some $1 \leq i \leq m$ and $\beta \in N^*$, then $t/\alpha = t_i/\beta$.

Obviously, $sub(t) = \{t/\alpha \mid \alpha \in path(t)\}$.

For each $t \in T_\Sigma(X)$ and $\alpha \in path(t)$, if $t/\alpha = f(t_1, \ldots, t_m)$ with $m \geq 1$ and $f \in \Sigma_m$, then $t_1, \ldots, t_m$ are called the *direct subtrees* at α.

In term rewriting system theory, the substitution of the subtree t/α of a tree t at a path α by a tree r is a fundamental operation. This is formalized as follows.

For $t \in T_\Sigma(X)$, $\alpha \in path(t)$, and $r \in T_\Sigma(X)$, we define $t[\alpha \leftarrow r] \in T_\Sigma(X)$ by induction on the length of α:

(i) if $\alpha = \lambda$, then $t[\alpha \leftarrow r] = r$;

(ii) if $\alpha = i\beta$, for some $i \in N$ and $\beta \in N^*$, then necessarily $t = f(t_1, \ldots, t_m)$ for some $m \geq 1$, $f \in \Sigma_m$ and $t_1, \ldots, t_m \in T_\Sigma(X)$ such that $1 \leq i \leq m$. Then $t[\alpha \leftarrow r] = f(t_1, \ldots, t_{i-1}, t_i[\beta \leftarrow r], t_{i+1}, \ldots, t_m)$.

There is another kind of substitution, called *substitution*, where we substitute trees for variables in a tree. Such a substitution is a mapping $\theta : X \to T_\Sigma(X)$ which is different from the identity only on a finite subset of X. For a substitution θ, the term $\theta(t)$ is produced from t by replacing each occurrence of x_i with $\theta(x_i)$.

For any $k, m \in N$ with $1 \leq m \leq k$, for every tree $t \in T_\Sigma(\{x_m, \ldots, x_k\})$ and for every substitution θ with $\theta(x_m) = t_m, \ldots, \theta(x_k) = t_k$, we denote $\theta(t)$ also by $t[x_m \leftarrow t_m, \ldots, x_k \leftarrow t_k]$. Moreover, in case $m = 1$, the denotation $t[x_1 \leftarrow t_1, \ldots, x_k \leftarrow t_k]$ is abbreviated as $t[t_1, \ldots, t_k]$.

Let Σ be a ranked alphabet and $s, t \in T_\Sigma(X)$. A *unifier* of s and t is a substitution θ such that $\theta(s) = \theta(t)$. A *most general unifier* of s and t is a unifier θ of s and t such that for each unifier η of s and t, there is a substitution η' satisfying that $\eta'(\theta(s)) = \eta(s)$ and $\eta'(\theta(t)) = \eta(t)$. It is decidable if s and t are unifiable ([50]). Moreover, if s and t are unifiable, then one can effectively construct a most general unifier of s and t, see Theorem 4.3 in [50].

Throughout the dissertation we shall consider the most general unifiers of unifiable and linear terms $s, t \in T_\Sigma(X)$ satisfying $var(s) \cap var(t) = \emptyset$. For such s and t, we can construct a most general unifier $\eta : X \to T_\Sigma(X)$ as follows.

1. For every $\alpha \in path(s)$, if $s/\alpha = x \in X$ and $\alpha \in path(t)$, then let $\eta(x) = t/\alpha$, otherwise let $\eta(x) = x$.

2. For every $\alpha \in path(t)$, if $t/\alpha = x \in X$, $\alpha \in path(s)$, and $s/\alpha \notin X$, then let $\eta(x) = s/\alpha$, otherwise let $\eta(x) = x$.

3. For every $x \in X$ not defined in 1. or 2., let $\eta(x) = x$.

It should be clear that η is a most general unifier of s and t. It is well known that a most general unifier of s and t is unique up to renaming of variables. Hence for each most general unifier η_1 of s and t and for every variable $x \in var(s) \cup var(t)$, if $\eta(x) \in T_\Sigma$ then $\eta(x) = \eta_1(x)$.

Let Σ be a ranked alphabet and let $u, v \in T_\Sigma(X)$. The tree u is a *supertree* of v if u is linear and there is a substitution θ such that $v = \theta(u)$. We illustrate the concept of a supertree by an example. Let $\Sigma = \Sigma_0 \cup \Sigma_1 \cup \Sigma_2$, $\Sigma_0 = \{\sharp\}$, $\Sigma_1 = \{f\}$, $\Sigma_2 = \{g\}$. The trees $f(x_2)$, $f(g(x_2, x_1))$, $f(g(\sharp, x_2))$ are supertrees of $f(g(\sharp, \sharp))$. On the other hand, $f(f(x_1))$ is not a supertree of $f(g(\sharp, \sharp))$, because there is no substitution θ such that $\theta(f(f(x_1))) = f(g(\sharp, \sharp))$. Moreover, $f(g(x_1, x_1))$ is not a supertree of $f(g(\sharp, \sharp))$ because $f(g(x_1, x_1))$ is not linear.

1.3 Rewriting systems and their properties

In this section we recall and introduce some notation, basic definitions and terminology for rewriting systems. Nevertheless the reader is assumed to be familiar with basic concepts of rewriting systems (see, e.g. [4], [7], [13], [46]).

1.3.1 Abstract rewriting systems

An *abstract rewriting system* is a structure $\mathcal{R} = (U, \rightarrow)$ consisting of a set U and a binary relation $\rightarrow$ on U.

We denote the transitive closure, the reflexive, transitive closure, and the reflexive, symmetric and transitive closure of $\rightarrow$ by $\rightarrow^+$, $\rightarrow^*$, and $\leftrightarrow^*$, respectively. We recall that $\leftrightarrow^*$ is an equivalence relation.

We say that $\mathcal{R}$ is

(i) *locally confluent* if for any $u, u_1, u_2 \in U$, if $u \rightarrow u_1$ and $u \rightarrow u_2$, then a $u_3 \in U$ exists such that $u_1 \rightarrow^* u_3$ and $u_2 \rightarrow^* u_3$;

(ii) *confluent* if for any $u, u_1, u_2 \in U$, if $u \rightarrow^* u_1$ and $u \rightarrow^* u_2$, then a $u_3 \in U$ exists such that $u_1 \rightarrow^* u_3$ and $u_2 \rightarrow^* u_3$;

(iii) *noetherian* if there is no infinite sequence of the form

$$u_1 \to u_2 \to u_3 \to \dots$$

with $u_i \in U$, $i = 1, 2, \dots$;

(iv) *convergent* if $\mathcal{R}$ is confluent and noetherian.

Proposition 1.3.1 [46] *A locally confluent and noetherian abstract rewriting system is a confluent and thus it is a convergent.*

An element $u \in U$ is called *irreducible* with respect to $\mathcal{R}$ if there does not exist $u' \in U$ with $u \to u'$. Moreover, if $u, u' \in U$ such that $u \to^* u'$ and u' is irreducible with respect to $\mathcal{R}$ then we call u' a *normal form* of u. It is well-known that if $\mathcal{R}$ is convergent then, for every $u \in U$, there exists exactly one $u' \in U$ such that u' is a normal form of u (see, e.g. [46]).

1.3.2 Term rewriting systems

Let Σ be a ranked alphabet. A *term rewriting system* (or shortly a rewriting system) over Σ is a finite subset R of $T_\Sigma(X) \times T_\Sigma(X)$ such that for every $(l, r) \in R$, each variable of r also occurs in l (i.e., $var(r) \subseteq var(l)$). The elements (l, r) of R are called rules and are denoted also by $l \to r$. We denote by $sign(R)$ ($\subseteq \Sigma$) the ranked alphabet consisting of all symbols appearing in the rules of R.

Note, that $R \cup R^{-1}$ is also a rewriting system if and only if for each $l \to r$ in R, each variable of l also occurs in r.

We now define the relation $\to_R$ over $T_\Sigma(X)$. For two terms $s, t \in T_\Sigma(X)$, $s \to_R t$ if there are a path $\alpha \in path(s)$, a rule $l \to r \in R$, and a substitution $\theta : X \to T_\Sigma(X)$ such that $s/\alpha = \theta(l)$ and $t = s[\alpha \leftarrow \theta(r)]$. If $s \to_R t$, then we say that R rewrites s to t or, if the details are also necessary, that R rewrites s to t applying the rule $l \to r$ at α.

Note that R rewrites such terms in which not only symbols being in $sign(R)$ appear. So, it is important to specify the ranked alphabet Σ over which R is considered. If we do not specify it, then we consider R over $sign(R)$.

To every rewriting system R over Σ, we can associate the abstract rewriting system $\mathcal{R} = (T_\Sigma(X), \to_R)$. Thus, we can easily adopt the notions and notation from abstract rewriting systems to term rewriting systems as follows.

A rewriting system R over Σ is locally confluent (confluent, noetherian, convergent) if the abstract rewriting system $\mathcal{R}$ associated to R is locally confluent (confluent, noetherian, convergent).

Analogously, a term $t \in T_\Sigma(X)$ is called irreducible with respect to R if t is irreducible with respect to $\mathcal{R}$. Moreover, for two terms $s, t \in T_\Sigma(X)$, we call t an R-normal form of s if t is a normal form of s with respect to $\mathcal{R}$.

The set of all irreducible terms with respect to R is denoted by $IRR(R)$. Moreover, we put $IRR_g(R) = IRR(R) \cap T_\Sigma$, hence $IRR_g(R)$ is the set of all ground terms which are irreducible with respect to R.

For a term $s \in T_\Sigma(X)$, we denote the set of normal forms of s with respect to R by $NF(s, R)$. We extend this notation for a tree language $L \subseteq T_\Sigma(X)$, by letting

$$NF(L, R) = \cup_{s \in L} NF(s, R).$$

It should be clear that if $L \subseteq T_\Sigma$, then $NF(L, R) \subseteq T_\Sigma$.

We denote by $[t]_R$ the $\leftrightarrow_R^*$-class of a tree $t \in T_\Sigma(X)$. Note that if $t \in T_\Sigma$ and $R \cup R^{-1}$ is also rewriting system, then $[t]_R \subseteq T_\Sigma$.

We say that the pair $(l_1, r_1) \in T_\Sigma(X) \times T_\Sigma(X)$ is a *variant* of the pair $(l_2, r_2) \in T_\Sigma(X) \times T_\Sigma(X)$ if there is an injective substitution $\theta : X \to X$ such that $\theta(l_2) = l_1$ and $\theta(r_2) = r_1$.

We adopt the concept of a *critical pair* ([46], [47]). Let R be a rewriting system over Σ and assume that the rules $l_1 \to r_1$, $l_2 \to r_2$ are in R. Let us take a variant $l_2' \to r_2'$ of $l_2 \to r_2$ such that $var(l_1) \cap var(l_2') = \emptyset$. Let us assume that there is a tree $t = l_1/\alpha$, where $\alpha \in path(l_1)$, such that $t \notin X$, t and l_2' are unifiable. Let θ be a most general unifier of t and l_2'. Let $v_1 = \theta(r_1)$ and define $v_2 = \theta(l_1)\ [\alpha \leftarrow \theta(r_2')]$. Then we call (v_1, v_2) a critical pair of R. Huet [46] showed the following result.

Proposition 1.3.2 *Let R be a rewriting system over Σ. Then R is locally confluent if and only if, for every critical pair (v_1, v_2) of R, there exists a tree $v \in T_\Sigma(X)$ such that $v_1 \to_R^* v$ and $v_2 \to_R^* v$.*

1.3.3 String rewriting systems

Let Σ be an alphabet. A *string rewriting system* S over Σ is a finite subset of $\Sigma^* \times \Sigma^*$ and each element (u, v) of S is called rule. We also write $u \to v \in S$

meaning that $(u, v) \in S$.

For each $w, z \in \Sigma^*$, $w \rightarrow_S z$ if and only if there exists $x, y \in \Sigma^*$ and $u \rightarrow v \in S$ such that $w = xuy$ and $z = xvy$.

It should be clear that, $\leftrightarrow_S^*$ i.e., the reflexive, symmetric and transitive closure of $\rightarrow_S$, is a congruence over Σ^*.

It is well-known that the symbols of an alphabet Σ can be considered as unary function symbols and hence words over Σ can be considered as unary trees over the ranked alphabet $\Sigma \cup \{\sharp^{(0)}\}$, where $\sharp \notin \Sigma$. For example, the word *apple* can be considered as the term $a(p(p(l(e(\sharp)))))$.

Let S be a string rewriting system over Σ. The term rewriting system associated to S is the term rewriting system R over Δ, where $\Delta = \{a^{(1)} \mid a \in \Sigma\} \cup \{\sharp^{(0)}\}$ ($\sharp \notin \Sigma$) and the rules of R are obtained from that of S such that x_1 is put to the right end of both sides of the rules of S. That is $R = \{u(x_1) \rightarrow v(x_1) \mid u \rightarrow v \in S\}$. (For $u = a_1 a_2 \ldots a_n$, $n \geq 0$, $u(x_1) = a_1(a_2(\ldots a_n(x_1)\ldots))$, thus $\lambda(x_1) = x_1$.) Hence, our notions and results on term rewriting systems can be carried over to string rewriting systems.

1.4 Deterministic tree recognizers, tree transducers and tree transformations

In this section, we define the concept of a deterministic bottom-up tree transducer, of a deterministic bottom-up tree recognizer, of a deterministic top-down tree transducer, of a deterministic top-down tree recognizer, and of a deterministic top-down tree transducer with regular look-ahead. We define some restricted versions of these devices and introduce other necessary notions and notation concerning them. The readers, who are not familiar enough with these concepts, can consult with [40] and [41] for more details.

1.4.1 Bottom-up tree transducers

A *deterministic bottom-up tree transducer* (db for short) is a system $\mathcal{A} = (\Sigma, A, \Delta, A', R)$, where

(1) Σ is a ranked alphabet, called the *input* alphabet;

(2) A is a ranked alphabet, called the *state* alphabet, such that $A = A_1$ and $A \cap (\Sigma \cup \Delta \cup X) = \emptyset$;

(3) Δ is a ranked alphabet, called the *output* alphabet;

(4) $A'(\subseteq A)$ is the set of *final states*;

(5) R is a finite set of *rules* of the form

$$f(a_1(x_1), \ldots, a_m(x_m)) \to a(r)$$

where $m \geq 0$, $f \in \Sigma_m$, $a, a_1, \ldots a_m \in A$, and $r \in T_\Delta(X_m)$. Moreover, there are no two different rules in R with the same left-hand side.

The tree transformation induced by a db is formalized as follows. Define the binary relation $\Rightarrow_\mathcal{A}$ on the set $T_{\Sigma \cup A \cup \Delta}(X)$ so that for any $t, s \in T_{\Sigma \cup A \cup \Delta}(X)$, $t \Rightarrow_\mathcal{A} s$ if and only if the following condition holds: there is a rule $f(a_1(x_1), \ldots, a_m(x_m)) \to a(r)$ in R such that s can be obtained from t by replacing an occurrence of a subtree $f(a_1(t_1), \ldots, a_m(t_m))$ of t by $a(r[t_1, \ldots, t_m])$, where $t_1, \ldots, t_m \in T_{\Sigma \cup A \cup \Delta}(X)$. The reflexive, transitive closure of $\Rightarrow_\mathcal{A}$ is denoted by $\Rightarrow_\mathcal{A}^*$. The tree transformation induced by $\mathcal{A}$ is the relation

$$\tau_\mathcal{A} = \{\, (t, s) \in T_\Sigma \times T_\Delta \mid t \underset{\mathcal{A}}{\overset{*}{\Rightarrow}} a(s) \text{ for some } a \in A' \,\}.$$

Clearly, the relation $\Rightarrow_\mathcal{A}$ is interpreted as a method of rewriting terms into terms. Hence the db $\mathcal{A}$ can also be considered as a term rewriting system P over $\Sigma \cup A \cup \Delta$, where $P = R$. Moreover, P is locally confluent due to the shape of rules in R and the fact that there are no different rules in R with the same left-hand side: these conditions exclude any overlapping of left-hand sides. On the other hand, it is not hard to see that P is also noetherian. Hence, by Proposition 1.3.1, it is convergent and thus $\tau_\mathcal{A}$ is a partial function.

Let $\mathcal{A} = (\Sigma, A, \Delta, A', R)$ be a db and let $B \subseteq A$. Then we denote by $\mathcal{A}(B)$ the db $(\Sigma, A, \Delta, B, R)$.

We now introduce four special types of db's. Let $\mathcal{A} = (\Sigma, A, \Delta, A', R)$ be a db. We say that $\mathcal{A}$ is

(a) a *deterministic bottom-up tree recognizer* (dbr) if $\Sigma = \Delta$ (hence we denote it by $\mathcal{A} = (\Sigma, A, A', R)$) and each rule in R is of the form

$$f(a_1(x_1), \ldots, a_m(x_m)) \to a(f(x_1, \ldots, x_m))$$

where $a, a_1, \ldots, a_m \in A$. In that case, the tree transformation $\tau_\mathcal{A}$ is a partial identity on T_Σ;

(b) a *linear deterministic bottom-up tree transducer* (ldb) if, for each rule $f(a_1(x_1), \ldots, a_m(x_m)) \to a(r)$ in R, r is linear;

(c) a *total deterministic bottom-up tree transducer* , if, for any states $a_1, \ldots, a_m \in A$ and symbol $f \in \Sigma_m$ $(m \geq 0)$ there is a rule (hence exactly one) in R with left-hand side $f(a_1(x_1), \ldots, a_m(x_m))$;

(d) a *bottom-up homomorphism tree transducer* (bh) if A is a singleton set, $A = A'$, and $\mathcal{A}$ is total.

Let $\mathcal{A} = (\Sigma, A, A', R)$ be a dbr. Then we also say that $\mathcal{A}$ is a dbr over Σ.

The class of tree transformations induced by all db's (respectively, ldb's) is denoted by DB (respectively, LDB). The *tree language recognized* by a dbr $\mathcal{A}$ is $L(\mathcal{A}) = dom(\tau_{\mathcal{A}})$. The class of tree languages recognized by dbr's is denoted by REC. A tree language is called *recognizable* if it is in REC.

1.4.2 Top-down tree transducers

We need the following notation. Let Σ, A be ranked alphabets, where $A = A_1$ i.e., A consists only of unary symbols. Then the set $T_\Sigma(A(X))$ of trees consists of all trees $t \in T_{\Sigma \cup A}(X)$ of the form $t = s[a_1(x_{i_1}), \ldots, a_n(x_{i_n})]$, where $n \geq 0$, $s \in \bar{T}_\Sigma(X_n)$, and $a_i \in A$ for $1 \leq i \leq n$.

Now, a *deterministic top-down tree transducer* (dt for short) is a system $\mathcal{A} = (\Sigma, A, \Delta, a_0, R)$, where

(1) Σ, Δ, A are the same as for bottom-up tree transducers;

(2) a_0 is an element of A, the *initial state*;

(3) R is a finite set of *rules* of the form

$$a(f(x_1, \ldots, x_m)) \to r$$

where $m \geq 0$, $f \in \Sigma_m$, $a \in A$, and $r \in T_\Delta(A(X_m))$. Moreover, there are no two different rules in R with the same left-hand side.

The tree transformation induced by a dt is formalized as follows. Define the binary relation $\Rightarrow_{\mathcal{A}}$ on the set $T_{\Sigma \cup A \cup \Delta}(X)$ so that for any $t, s \in T_{\Sigma \cup A \cup \Delta}(X)$, $t \Rightarrow_{\mathcal{A}} s$ if and only if the following condition holds: there is a rule $a(f(x_1,\ldots,x_m)) \to r$ in R such that s can be obtained from t by replacing an occurrence of a subtree $a(f(t_1,\ldots,t_m))$ of t by $r[t_1,\ldots,t_m]$, where $t_1,\ldots,t_m \in T_{\Sigma\cup A\cup\Delta}(X)$.

The reflexive, transitive closure of $\Rightarrow_{\mathcal{A}}$ is denoted by $\Rightarrow^*_{\mathcal{A}}$. The tree transformation induced by $\mathcal{A}$ is the relation

$$\tau_{\mathcal{A}} = \{\, (t,s) \in T_\Sigma \times T_\Delta \mid a_0(t) \overset{*}{\underset{\mathcal{A}}{\Rightarrow}} s \,\}.$$

Clearly, the relation $\Rightarrow_{\mathcal{A}}$ is interpreted as a method of rewriting terms into terms. Again, $\mathcal{A}$ can be considered as a term rewriting system P over $\Sigma \cup A \cup \Delta$, where $P = R$. Moreover, P is again locally confluent and terminating. Hence, it is convergent, see Proposition 1.3.1, and thus $\tau_{\mathcal{A}}$ is a partial function, cf. [27].

We say that, a deterministic top-down tree transducer $\mathcal{A}$ is *injective* if $\tau_{\mathcal{A}}$ is injective i.e., for any $t, s \in T_\Sigma$ such that $t \neq s$ we have $\tau_{\mathcal{A}}(t) \neq \tau_{\mathcal{A}}(s)$.

For each $a \in A$, $\mathcal{A}(a) = (\Sigma, A, \Delta, a, R)$ is the dt $\mathcal{A}$ with initial state a instead of a_0.

Let $\mathcal{A} = (\Sigma, A, \Delta, a_0, R)$ be a dt. We say that $\mathcal{A}$ is

(a) a *deterministic top-down tree recognizer* (dtr) if $\Sigma = \Delta$ (hence we denote it by $\mathcal{A} = (\Sigma, A, a_0, R)$) and each rule in R is of the form

$$a(f(x_1,\ldots,x_m)) \to f(a_1(x_1),\ldots,a_m(x_m))$$

where $a, a_1, \ldots, a_m \in A$. In that case the tree transformation $\tau_{\mathcal{A}}$ is a partial identity on T_Σ;

(b) a *linear deterministic top-down tree transducer* (ldt) if for each rule $a(f(x_1,\ldots,x_m)) \to r$ in R, r is linear;

(c) a *nondeleting deterministic top-down tree transducer* (ndt) if for each rule $a(f(x_1,\ldots,x_m)) \to r$ in R, each of the variables $x_1,\ldots,x_m$ appears at least once in r;

(d) a *linear nondeleting deterministic top-down tree transducer* (lndt) if it is a linear and nondeleting top-down tree transducer;

(e) a *total deterministic top-down tree transducer*, if for any state $a \in A$ and symbol $f \in \Sigma_m$ $(m \geq 0)$ there is a rule (and hence exactly one) in R with left-hand side $a(f(x_1,\ldots,x_m))$;

(f) a *top-down homomorphism tree transducer* (th) if $A = \{a_0\}$, and $\mathcal{A}$ is total.

The class of tree transformations induced by all dt's (respectively, ldt's) is denoted by DT (respectively, LDT). The *tree language recognized* by the dtr $\mathcal{A}$ is $L(\mathcal{A}) = dom(\tau_{\mathcal{A}})$. The class of tree languages recognized by dtr's is denoted by $DREC$. It is well known that $DREC \subset REC$.

Consider the class of bh and th tree transducers. By Theorem 1.9 in Chapter IV of [40] the class of all tree transformations induced by bh transducers coincides with the class of all tree transformations induced by th transducers. We denote this tree transformation class by H. The proof carries over to the linear case as well, hence the class of all tree transformations induced by linear bh transducers coincides with the class of all tree transformations induced by linear th transducers. We denote this tree transformation class by LH.

1.4.3 Top-down tree transducers with regular look-ahead

Top-down tree transducers with regular look-ahead were defined in [15]. It transpired that they have a number of nice properties, especially in the deterministic case. For example, the class of deterministic top-down tree transformations with regular look-ahead is closed under composition.

A *deterministic top-down tree transducer with regular look-ahead* (dt^R) is a system $\mathcal{A} = (\Sigma, A, \Delta, a_0, R)$, where the first four components are defined exactly as in the previous subsection. Here R is a finite set of rules of the form

$$\langle a(f(x_1, \ldots, x_m)) \to r;\ L_1, \ldots, L_m \rangle$$

where $a(f(x_1, \ldots, x_m)) \to r$ is an ordinary dt-rule, see the previous subsection, and for each $1 \le i \le m$, $L_i \subseteq T_\Sigma$ is a tree language in REC. Moreover, $L_i \cap L'_i = \emptyset$ holds for some $1 \le i \le m$, whenever $\langle a(f(x_1, \ldots, x_m)) \to r_1;\ L_1, \ldots, L_m \rangle$ and $\langle a(f(x_1, \ldots, x_m)) \to r_2;\ L'_1, \ldots, L'_m \rangle$ are different rules in R.

One step in the transformation of $\mathcal{A}$ is represented by the binary relation $\Rightarrow_{\mathcal{A}}$ on $T_{\Sigma \cup A \cup \Delta}(X)$ defined as follows: $t \Rightarrow_{\mathcal{A}} s$ if and only if the following condition holds: there is a rule $\langle a(f(x_1, \ldots, x_m)) \to r;\ L_1, \ldots, L_m \rangle$ in R

such that s can be obtained from t by replacing an occurrence of a subtree $a(f(t_1, \ldots, t_m))$ of t by $r[t_1, \ldots, t_m]$ where $t_i \in L_i$ for $1 \leq i \leq m$.

It can be seen from the definition of $\Rightarrow_{\mathcal{A}}$ what the notion look-ahead means: a rule can be applied at a node of a tree only if the direct subtrees of that node are in the tree languages, respectively, given in the rule. Note that $\mathcal{A}$ can apply at most one rule at any given node. This is because for any two different rules in R with the same left-hand side there exists a variable x_i such that the ith look-ahead sets are disjoint. As usual, $\Rightarrow_{\mathcal{A}}^*$ is the reflexive, transitive closure of $\Rightarrow_{\mathcal{A}}$ and the partial function

$$\tau_{\mathcal{A}} = \{\, (t, s) \in T_\Sigma \times T_\Delta \mid a_0(t) \underset{\mathcal{A}}{\overset{*}{\Rightarrow}} s \,\}$$

is the tree transformation induced by $\mathcal{A}$.

Let $\mathcal{A} = (\Sigma, A, \Delta, a_0, R)$ be a dt^R. We say that $\mathcal{A}$ is a *linear deterministic top-down tree transducer with regular look-ahead* (ldt^R) if for each rule

$$\langle a(f(x_1, \ldots, x_m)) \to r;\ L_1, \ldots, L_m \rangle$$

in R, r is linear.

The class of all tree transformations induced by all dt^R's (respectively ldt^R's) is denoted by DT^R (respectively LDT^R).

Chapter 2

On term rewriting systems preserving recognizability

This chapter is divided into four sections. Section 2.1 consists of the necessary notions and notation. We summarize the results of this chapter in Section 2.2. Moreover, in Section 2.3, we show that lgsm rewriting systems effectively preserve recognizability and we illustrate our constructions by an example. Finally, in Section 2.4, we study rewriting systems preserving recognizability and gsm rewriting systems.

2.1 Further notions and notation

In this chapter, we need the concept of a (nondeterministic) bottom-up tree automaton.

Let Σ be a ranked alphabet. A *bottom-up tree automaton* over Σ is a quadruple $\mathcal{A} = (\Sigma, A, A', R)$, where A is a finite set of states of rank 0, $\Sigma \cap A = \emptyset$, $A'(\subseteq A)$ is the set of final states, R is a finite set of rules of the following two types:

(i) $f(a_1, \ldots, a_n) \to a$ with $n \geq 0$, $f \in \Sigma_n$, $a_1, \ldots, a_n, a \in A$.

(ii) $a \to a'$ with $a, a' \in A$ (called λ-rules).

We consider R as a ground rewriting system over $\Sigma \cup A$. The tree language recognized by a bottom-up tree automaton $\mathcal{A}$ is

$$L(\mathcal{A}) = \{t \in T_\Sigma \mid \text{ there exists } a \in A' \text{ such that } t \xrightarrow[R]{*} a\}.$$

It is well known that the class of tree languages recognized by bottom-up tree automata is REC (see [40]). The bottom-up tree automaton $\mathcal{A} = (\Sigma, A, A', R)$ is deterministic if R has no λ-rules and R has no two rules with the same left-hand side. It should be clear that a deterministic bottom-up tree automaton is equivalent with a dbr defined in Subsection 1.4.1 with respect to recognizing capacity and vice versa.

We say that the bottom-up tree automaton $\mathcal{A}$ is connected if for every $a \in A$ there exists $t \in T_\Sigma$ such that $t \Rightarrow^*_{\mathcal{A}} a$. Every recognizable tree language can be recognized by a deterministic connected bottom-up tree automaton (see [40]).

We need the following result, which was shown by Brainerd [5], Kozen [48], and Fülöp and Vágvölgyi [31].

Proposition 2.1.1 *A tree language L is recognizable if and only if there exists a ground rewriting system R such that L is the union of finitely many $\leftrightarrow^*_R$-classes.*

Next we define some restricted versions of rewriting systems. Therefore, let R be a rewriting system over Σ. We say that R is

(i) *left-linear* (*right-linear*) if, for every rule $l \to r$ in R, l (r) is a linear tree;

(ii) *linear* if it is both left-linear and right-linear;

(iii) *ground* if, for every rule $l \to r$ in R, both l and r are ground trees;

(iv) *monadic* if, for every rule $l \to r$ in R, $height(l) \geq 1$ and $height(r) \leq 1$;

(v) *semi-monadic* if, for every rule $l \to r$ in R, $height(l) \geq 1$ and either $height(r) = 0$ or $r = f(y_1, \ldots, y_k)$, where $f \in \Sigma_k$, $k \geq 1$, and for each $i \in \{1, \ldots, k\}$, either y_i is a variable (i.e., $y_i \in X$) or y_i is a ground term (i.e., $y_i \in T_\Sigma$);

(vi) *collapse-free* if there is no rule $l \to r$ in R such that $l \in X$ or $r \in X$;

(vii) *left-to-right minimal* if, for every rule $l \to r$ in R, $\to^*_{R-\{l\to r\}} \subset \to^*_R$;

(viii) *left-to-right ground minimal* if, for each rule $l \to r$ in R, $\to^*_{R-\{l\to r\}} \cap (T_\Sigma \times T_\Sigma) \subset \to^*_R \cap (T_\Sigma \times T_\Sigma)$;

(ix) *two-way minimal* if, for every rule $l \to r$ in R, $\leftrightarrow^*_{R-\{l\to r\}} \subset \leftrightarrow^*_R$;

(x) *two-way ground minimal* if, for each rule $l \to r$ in R, $\leftrightarrow^*_{R-\{l\to r\}} \cap (T_\Sigma \times T_\Sigma) \subset \leftrightarrow^*_R \cap (T_\Sigma \times T_\Sigma)$.

It is immediate that the concept of a semi-monadic rewriting system generalizes the notion of a monadic rewriting system. Moreover, properties (vii)-(x) are not static in the sense that we cannot decide by direct inspection whether a term rewriting system R has these properties or not. In fact, we will show later that (vii)-(x) are decidable properties of linear generalized semi-monadic rewriting systems.

Next we introduce the concept of a modular property for a class of rewriting systems. A class $\mathbf{C}$ of rewriting systems is *closed under disjoint union* if for any rewriting systems R, $S \in \mathbf{C}$ over Σ and Δ, respectively, such that $\Sigma \cap \Delta = \emptyset$, the rewrite system $R \cup S$ over $\Sigma \cup \Delta$ also belongs to $\mathbf{C}$.

Let $\mathbf{C}$ be a class of rewriting systems, closed under disjoint union. A property $\mathcal{P}$ over $\mathbf{C}$ is *modular* for $\mathbf{C}$ if for any $R, S \in \mathbf{C}$ over Σ and Δ, respectively, such that $\Sigma \cap \Delta = \emptyset$, $R \cup S$ over $\Sigma \cup \Delta$ has the property $\mathcal{P}$ if and only if both R over Σ and S over Δ have the property $\mathcal{P}$. For a short survey on the disjoint union of rewriting systems, see the introduction of [6]. Moreover, see [6] also for recent results in this area.

We introduce some further notation. Let Σ be a ranked alphabet and let R be a rewriting system over Σ. Moreover, let Δ be a ranked alphabet such that $\Sigma \subseteq \Delta$ and let $L \subseteq T_\Delta$. Then we define $R^*_\Delta(L) = \{p \mid q \to^*_R p \text{ for some } q \in L\}$. We call $R^*_\Delta(L)$ the set of descendants of elements of L and if Δ is clear from the context, we write $R^*(L)$ rather than $R^*_\Delta(L)$. We say that R *preserves Δ-recognizability*, if for every recognizable $L \subseteq T_\Delta$, $R^*_\Delta(L)$ is also recognizable.

A rewriting system R over Σ *preserves recognizability* if, for every ranked alphabet Δ with $\Sigma \subseteq \Delta$, R preserves Δ-recognizability.

Again, let R be a rewriting system over Σ and let $\Sigma \subseteq \Delta$. We say that R *effectively preserves Δ-recognizability* if, for every bottom-up tree automaton

$\mathcal{B}$ over Δ, we can effectively construct a bottom-up tree automaton $\mathcal{C}$ over Δ such that $L(\mathcal{C}) = R^*_\Delta(L(\mathcal{B}))$.

Finally, a rewriting system R over Σ *effectively preserves recognizability* if, for every ranked alphabet Δ with $\Sigma \subseteq \Delta$, R effectively preserves Δ-recognizability.

It is easy to see that a rewriting system over a ranked alphabet effectively preserves recognizability then it preserves recognizability.

We shall need the following concepts concerning string rewriting systems. Let S be a string rewriting system. We say that S is

(i) λ-*free* if there is no rule $u \to v$ in S such that $u = \lambda$ or $v = \lambda$;

(ii) *monadic* if $(u, v) \in S$ implies that $|u| > |v|$ and $(|v| = 1$ or $|v| = 0)$.

We say that a string rewriting system S over Σ (effectively) preserves (Σ-) recognizability if the term rewriting system R over Δ associated to S (effectively) preserves (Δ-) recognizability. It is well known that monadic string rewriting systems effectively preserve recognizability, see Theorem 4.1.2 in [4].

2.2 Summary of results

In [42] Gilleron showed that for a rewriting system R it is undecidable if R preserves $sign(R)$-recognizability. Moreover, in [51] F. Otto showed that it is undecidable in general whether a rewriting system preserves recognizability. We obtain the following results.

- There is a ranked alphabet Σ and there is a linear rewriting system R over Σ such that R preserves Σ-recognizability but does not preserve recognizability. (Theorem 2.4.1)

- Let R be a rewriting system, and let $\Sigma = \{ f, \sharp \} \cup sign(R)$, where $f \in \Sigma_2 - sign(R)$ and $\sharp \in \Sigma_0 - sign(R)$. Then, R preserves Σ-recognizability if and only if R preserves recognizability. (Theorem 2.4.3)

- Let R be a rewriting system, and let $\Sigma = \{ f, \sharp \} \cup sign(R)$, where $f \in \Sigma_2 - sign(R)$ and $\sharp \in \Sigma_0 - sign(R)$. Then, R effectively preserves Σ-recognizability if and only if R effectively preserves recognizability. (Theorem 2.4.6)

In spite of Gilleron's undecidability results, we know several rewriting systems which preserve recognizability. Brainerd [5] showed that ground rewriting systems over any ranked alphabet Σ effectively preserve Σ-recognizability, see also [16]. Gallier and Book [38] introduced the notion of a monadic rewriting system, and Salomaa [54] showed that linear monadic rewriting systems over any ranked alphabet Σ effectively preserve Σ-recognizability. Coquidé et al [7] defined the concept of a semi-monadic rewriting system generalizing the notion of a monadic rewriting system. Coquidé et al [7] showed that linear semi-monadic rewriting systems over any ranked alphabet Σ effectively preserve Σ-recognizability.

We generalize the concept of a semi-monadic rewriting system and of a ground rewriting system by introducing the concept of a generalized semi-monadic rewriting system (gsm rewriting system for short). We obtain the following main result.

- For every ranked alphabet Δ and linear gsm (lgsm) rewriting system R over Δ, R effectively preserves recognizability. (Theorem 2.3.19)

The proof of this statement can be sketched in the following way. Let $\Delta \subseteq \Sigma$, L be a recognizable tree language over Σ, and let $\mathcal{B} = (\Sigma, B, B', R_{\mathcal{B}})$ be a tree automaton recognizing L. Similarly to the constructions of Salomaa [54] and Coquidé et al [7], we construct a sequence of bottom-up tree automata $\mathcal{C}_i = (\Sigma, C, B', R_i)$, $i \geq 0$ having the same ranked alphabet, state set, and final state set. The rule set R_0 contains $R_{\mathcal{B}}$. Moreover, R_0 contains rules which enable R_0 to recognize the right-hand sides of rules in R. For each $i \geq 0$, R_{i+1} contains R_i, and for each rule $l \to r$ in R, $\mathcal{C}_{i+1}$ simulates, on the right-hand side r, the computation of $\mathcal{C}_i$ on the left-hand side l. There is a least integer $M \geq 0$ such that $R_M = R_{M+1}$. Hence $\mathcal{C}_M = \mathcal{C}_{M+1}$. We show that $L(\mathcal{C}_M) = R^*(L)$.

Brainerd [5], Kozen [48], and Fülöp and Vágvölgyi [31] showed that a tree language L is recognizable if and only if there exists a ground rewriting system R such that L is the union of finitely many $\leftrightarrow_R^*$-classes. We obtain a similar characterization for recognizable tree languages by proving the following.

- A tree language L is recognizable if and only if there exists a rewriting system R such that $R \cup R^{-1}$ is an lgsm rewriting system and that L is the union of finitely many $\leftrightarrow_R^*$-classes. (Theorem 2.3.21)

Our concepts and results carry over to strings as well. We generalize the concept of monadic string rewriting systems by introducing the concept of restricted right-left overlapping string rewriting systems.

We show the following two statements.

- Restricted right-left overlapping string rewriting systems effectively preserve recognizability. (Theorem 2.3.23)

- A string language L is recognizable if and only if there exists a string rewriting system S such that $S \cup S^{-1}$ is a restricted right-left overlapping string rewriting system and that L is the union of finitely many $\leftrightarrow_S^*$-classes. (Theorem 2.3.23)

We also show the following sequence of decidability results for rewriting systems (effectively) preserving recognizability.

- Let R_1, R_2 be rewriting systems. Let R_1 effectively preserve recognizability. Then it is decidable if $\rightarrow_{R_2}^* \subseteq \rightarrow_{R_1}^*$. (Theorem 2.4.8)

- For an lgsm rewriting system R, it is decidable whether R is left-to-right minimal. (Consequence 2.4.12)

- Let R_1 and R_2 be rewriting systems such that $R_1 \cup R_1^{-1}$ and $R_2 \cup R_2^{-1}$ are rewriting systems and effectively preserve recognizability. Then it is decidable if $\leftrightarrow_{R_1}^* \subseteq \leftrightarrow_{R_2}^*$. (Consequence 2.4.13)

- Let R be a rewriting system such that $R \cup R^{-1}$ is an lgsm rewriting system. Then it is decidable whether R is two-way minimal. (Consequence 2.4.14)

- Let R_1, R_2 be rewriting systems over a ranked alphabet Σ. Suppose that R_1 effectively preserves recognizability. Let $g \in \Sigma - \Sigma_0$ be such that g does not occur on the left-hand side of any rule in R_1, and let $\sharp \in \Sigma_0$ be irreducible with respect to R_1. Then it is decidable if $\rightarrow_{R_2}^* \cap (T_\Sigma \times T_\Sigma) \subseteq \rightarrow_{R_1}^* \cap (T_\Sigma \times T_\Sigma)$. (Theorem 2.4.15)

- Let R be an lgsm rewriting system over Σ. Moreover, let $g \in \Sigma - \Sigma_0$ be such that g does not occur on the left-hand side of any rule in R, and let $\sharp \in \Sigma_0$ be irreducible with respect to R. Then it is decidable whether R is left-to-right ground minimal. (Consequence 2.4.18)

- Let R_1 and R_2 be rewriting systems over Σ such that $R_1 \cup R_1^{-1}$ and $R_2 \cup R_2^{-1}$ are rewriting systems and effectively preserve recognizability. Moreover, let $g_1, g_2 \in \Sigma - \Sigma_0$ be such that for each $i \in \{1, 2\}$, g_i does not occur in R_i. Let $\sharp_1, \sharp_2 \in \Sigma_0$ be such that for each $i \in \{1, 2\}$, $\sharp_i$ is irreducible with respect to $R_i \cup R_i^{-1}$. Then it is decidable if $\leftrightarrow^*_{R_1} \cap (T_\Sigma \times T_\Sigma) \subseteq \leftrightarrow^*_{R_2} \cap (T_\Sigma \times T_\Sigma)$. (Consequence 2.4.19)

- Let R be a rewriting system over Σ such that $R \cup R^{-1}$ is an lgsm rewriting system. Moreover, let $g \in \Sigma - \Sigma_0$ be such that g does not occur in any rule of R, and let $\sharp \in \Sigma_0$ be irreducible with respect to $R \cup R^{-1}$. Then it is decidable whether R is two-way ground minimal. (Consequence 2.4.20)

- Let R be a rewriting system over Σ effectively preserving recognizability, and let $p, q \in T_\Sigma(X)$. Then it is decidable if there exists a tree $r \in T_\Sigma(X)$ such that $p \rightarrow^*_R r$ and $q \rightarrow^*_R r$. (Lemma 2.4.21)

- Let R be a rewriting system over Σ effectively preserving recognizability. Then it is decidable if R is locally confluent. (Theorem 2.4.22)

By direct inspection we obtain that for any dt $\mathcal{A} = (\Sigma, A, \Delta, a_0, R)$ with $\Sigma \cap \Delta = \emptyset$, R is a convergent left-linear gsm rewriting system over the ranked alphabet $A \cup \Sigma \cup \Delta$. Hence Fülöp's [25] undecidability results on deterministic top-down tree transducers imply the following.

- Each of the following questions is undecidable for any convergent left-linear gsm rewriting systems R_1 and R_2 over a ranked alphabet Ω, for any recognizable tree language $L \subseteq T_\Omega$ given by a tree automaton over Ω recognizing L, where Γ is the smallest ranked alphabet for which $NF(L, R_1) \subseteq T_\Gamma$.

  (i) Is $NF(L, R_1) \cap NF(L, R_2)$ empty?

  (ii) Is $NF(L, R_1) \cap NF(L, R_2)$ infinite?

  (iii) Is $NF(L, R_1) \cap NF(L, R_2)$ recognizable?

  (iv) Is $T_\Gamma - NF(L, R_1)$ empty?

  (v) Is $T_\Gamma - NF(L, R_1)$ infinite?

  (vi) Is $T_\Gamma - NF(L, R_1)$ recognizable?

(vii) Is $NF(L, R_1)$ recognizable?

(viii) Is $NF(L, R_1) = NF(L, R_2)$?

(ix) Is $NF(L, R_1) \subseteq NF(L, R_2)$?

(Proved in Theorem 2.4.27)

Finally, we show that preserving recognizability and effectively preserving recognizability are modular properties of linear collapse-free rewriting systems. That is, the following statement holds.

- Let R and S be linear collapse-free rewriting systems over disjoint ranked alphabets Σ and Δ, respectively. Then R and S (effectively) preserve recognizability if and only if $R \cup S$ over $\Sigma \cup \Delta$ also (effectively) preserves recognizability. (Consequence 2.4.31 and Theorem 2.4.32)

This result implies the following.

- Preserving recognizability and effectively preserving recognizability are modular properties also of λ-free string rewriting systems. (Theorem 2.4.33)

2.3 Generalized semi-monadic rewriting systems

2.3.1 Linear generalized semi-monadic rewriting systems preserve recognizability

In this subsection we introduce the notion of a gsm rewriting system and show that linear gsm rewriting systems effectively preserve recognizability.

Definition 2.3.1 Let R be a rewriting system over Σ. We say that R is a *generalized semi-monadic rewriting system* (gsm rewriting system for short) if there is no rule $l \to r$ in R with $l \in X$ and the following holds. For any rules $l_1 \to r_1$ and $l_2 \to r_2$ in R, for any occurrences $\alpha \in path(r_1)$ and $\beta \in path(l_2)$, and for any supertree $l_3 \in T_\Sigma(X)$ of l_2/β with $var(l_3) \cap var(l_1) = \emptyset$, if

(i) $\alpha = \lambda$ or $\beta = \lambda$,

(ii) r_1/α and l_3 are unifiable, and

(iii) θ is a most general unifier of r_1/α and l_3,

then

(a) $l_2/\beta \in X$ or

(b) for each $\gamma \in path(l_3)$, if $l_2/\beta\gamma \in X$, then $\theta(l_3/\gamma) \in X \cup T_\Sigma$.

Notice that Condition (a) implies that $l_3 \in X$. We abbreviate the expression linear gsm to lgsm.

Example 2.3.2 Let $\Sigma = \Sigma_0 \cup \Sigma_1 \cup \Sigma_2$, $\Sigma_0 = \{\sharp\}$, $\Sigma_1 = \{f\}$, and $\Sigma_2 = \{g\}$. Let the rewriting system R over Σ consist of the rule

$$g(x_1, x_2) \rightarrow f(g(x_1, \sharp)) \ .$$

We obtain by direct inspection that R is lgsm.

Definition 2.3.3 A rewriting system R over Σ is *restricted right-left overlapping* if there is no rule $l \rightarrow r$ in R with $l \in X$ and the following holds. For any rules $l_1 \rightarrow r_1$ and $l_2 \rightarrow r_2$ in R, for any occurrences $\alpha \in path(r_1)$ and $\beta \in path(l_2)$, and for any supertree $l_3 \in T_\Sigma(X)$ of l_2/β with $var(l_3) \cap var(l_1) = \emptyset$, if (i), (ii), and (iii) in Definition 2.3.1 hold, then (a'), (b'), or (c') hold.

(a') $\alpha = \lambda$, $l_2/\beta \in X$.

(b') $\alpha = \lambda$ and for each $\gamma \in path(l_3)$, if $l_2/\beta\gamma \in X$, then $\theta(l_3/\gamma) \in X \cup T_\Sigma$.

(c') $\beta = \lambda$ and for each $\gamma \in path(l_3)$, if $l_2/\gamma \in X$, then $\theta(l_3/\gamma) \in X \cup T_\Sigma$.

Note that Condition (a') implies that $l_3 \in X$. We visualize the unification of r_1/α and the supertree l_3 of l_2/β by the most general unifier θ, when Condition (a') (Condition (b'), Condition (c'), respectively) holds on Figure 2.1 (Figure 2.2, Figure 2.3, respectively).

The proofs of the following two results are straightforward.

Observation 2.3.4 *A rewriting system R is gsm if and only if R is restricted right-left overlapping.*

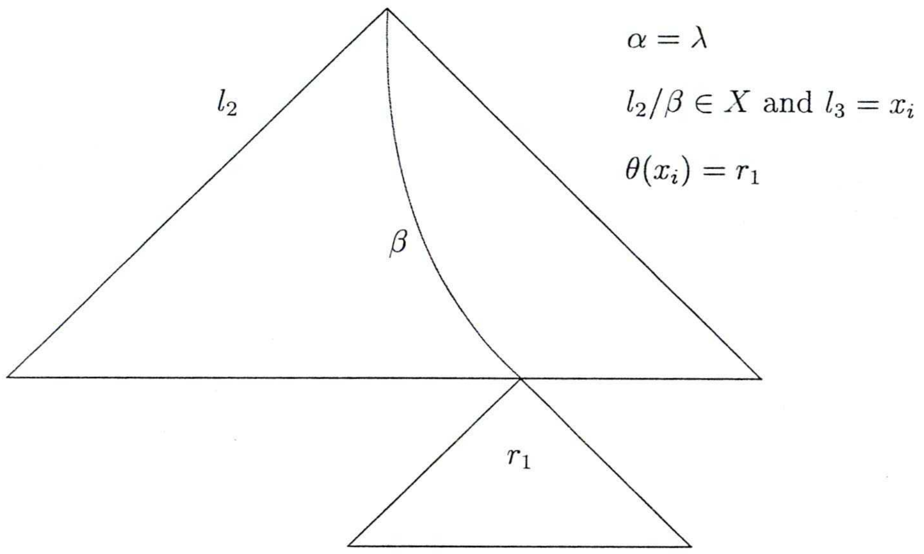

Figure 2.1: The unification of r_1/α and the supertree l_3 of l_2/β by the most general unifier θ, when Condition (a') holds.

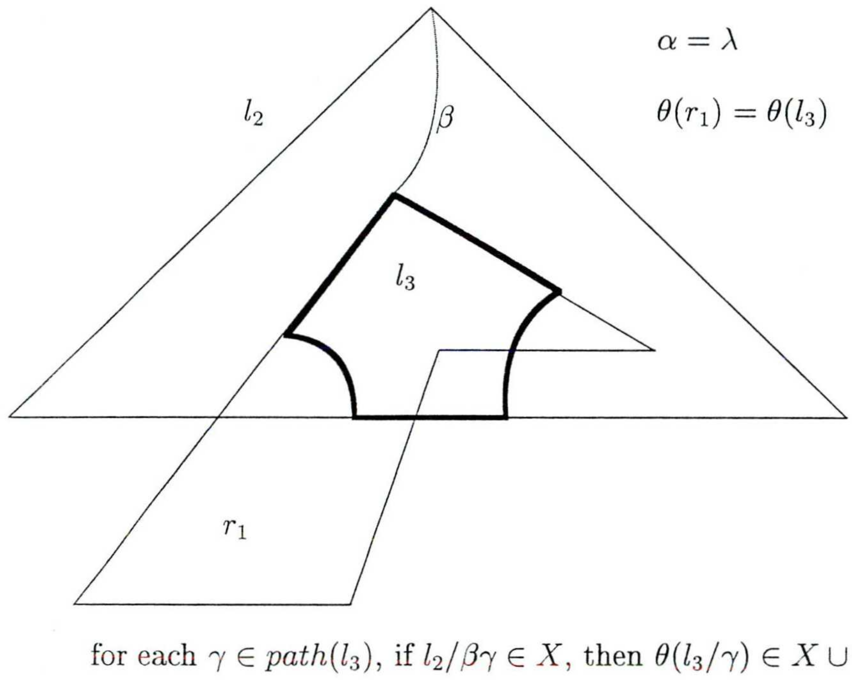
$\alpha = \lambda$
$\theta(r_1) = \theta(l_3)$
l_2
β
l_3
r_1
for each $\gamma \in path(l_3)$, if $l_2/\beta\gamma \in X$, then $\theta(l_3/\gamma) \in X \cup T_\Sigma$

Figure 2.2: The unification of r_1/α and the supertree l_3 of l_2/β by the most general unifier θ, when Condition (b') holds.

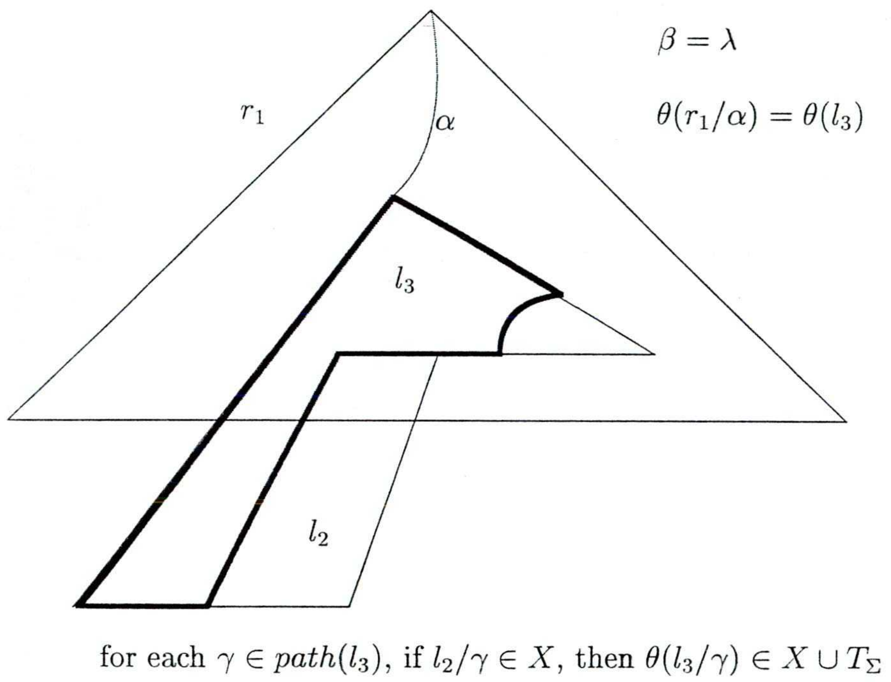
$\beta = \lambda$
$\theta(r_1/\alpha) = \theta(l_3)$
r_1
α
l_3
l_2
for each $\gamma \in path(l_3)$, if $l_2/\gamma \in X$, then $\theta(l_3/\gamma) \in X \cup T_\Sigma$

Figure 2.3: The unification of r_1/α and the supertree l_3 of l_2 by the most general unifier θ, when Condition (c') holds.

Observation 2.3.5 *Each semi-monadic rewriting system is gsm as well.*

We now show that lgsm rewriting systems effectively preserve recognizability. Throughout this subsection, R is an lgsm rewriting system over some ranked alphabet Δ, and Σ is an arbitrary ranked alphabet such that $\Delta \subseteq \Sigma$. Moreover, let $L = L(\mathcal{B})$ be a recognizable tree language over Σ, where $\mathcal{B} = (\Sigma, B, B', R_B)$ is a deterministic connected bottom-up tree automaton over Σ. Via a series of theorems and lemmas we show that $R^*_\Sigma(L)$ is recognizable. In fact we construct a tree automaton $\mathcal{C}$ over Σ such that $L(\mathcal{C}) = R^*_\Sigma(L)$. Our construction is illustrated by an example in Subsection 2.3.2. As we are interested in the tree language $R^*_\Sigma(L)$ rather than in $R^*_\Delta(L)$, by $R^*(L)$ we always mean $R^*_\Sigma(L)$.

Let E be the set of all ground terms u over Σ such that there are rules $l_1 \to r_1$ and $l_2 \to r_2$ in R, and there are occurrences $\alpha \in path(r_1)$ and $\beta \in path(l_2)$, and there is a supertree $l_3 \in T_\Sigma(X) - X$ of l_2/β with $var(l_3) \cap var(l_1) = \emptyset$ such that

(i) $\alpha = \lambda$ or $\beta = \lambda$,

(ii) r_1/α and l_3 are unifiable, and

(iii) θ is a most general unifier of r_1/α and l_3, and

(iv) there is an occurrence $\gamma \in path(l_3)$ such that $l_2/\beta\gamma \in X$ and $\theta(l_3/\gamma) \in T_\Sigma$, and that u is a subterm of $\theta(l_3/\gamma)$.

It should be clear that E is finite and effectively constructable.

Recall that $\mathcal{B} = (\Sigma, B, B', R_B)$ is a deterministic connected bottom-up tree automaton such that $L(\mathcal{B}) = L$. We lose no generality by assuming that $B \cap N = \emptyset$. Moreover, without loss of generality we may assume that for each rule $l \to r$ in R, $l \in \bar{T}_\Sigma(X_n)$ for some $n \geq 0$. Let

$D = B \cup \{ p[a_1, \ldots, a_n] \mid n \geq 0, p \in T_\Sigma(X_n), a_1, \ldots, a_n \in B \cup E, p$ is a subtree of the right-hand side r of some rule $l \to r$ in $R\}$.

It should be clear that $B \cup E \subseteq D$. Let

$$C = B \cup \{ 1, \ldots, |D - B| \} .$$

We consider C as a ranked alphabet, for each $c \in C$ the rank of c is 0. Let $\langle\ \rangle : D \to C$ be a bijection such that $\langle b \rangle = b$ for each $b \in B$.

For each $i \geq 1$, consider the bottom-up tree automaton $\mathcal{C}_i = (\Sigma, C, B', R_i)$, where R_i is defined by recursion on i (for an example see Subsection 2.3.2).

We define R_0 as follows.

(i) $R_{\mathcal{B}} \subseteq R_0$.

(ii) For all $n \geq 0$, $f \in \Sigma_n$, $t_1, \ldots, t_n \in D$, if $f(t_1, \ldots, t_n) \in D$, then we put the rule $f(\langle t_1 \rangle, \ldots, \langle t_n \rangle) \to \langle f(t_1, \ldots, t_n) \rangle$ in R_0.

We shall refer to a rule appearing in (ii) as a (ii)-type rule of R_0.

Let us assume that $i \geq 1$ and we have defined the set R_{i-1}. Then we define R_i as follows.

(a) $R_{i-1} \subseteq R_i$.

(b) For any rule $l \to r$ in R with $n \geq 0$, $l \in \bar{T}_\Sigma(X_n)$, for all $a_1, \ldots, a_n \in B \cup E$, if $l[\langle a_1 \rangle, \ldots, \langle a_n \rangle] \to^*_{R_{i-1}} c$ for some $c \in C$, then we put the rule $\langle r[a_1, \ldots, a_n] \rangle \to c$ in R_i.

As $\mathcal{B}$ is connected, all states in B are reachable in $\mathcal{C}_0$. By (ii) in the definition of R_0, all states in $\{1, \ldots, |D - B|\}$ are reachable in R_0. Hence $\mathcal{C}_0$ is connected. As $R_i \subseteq R_{i+1}$ for $i \geq 0$, $\mathcal{C}_i$ is connected for $i \geq 1$.

It should be clear that there is an integer $M \geq 0$ such that $R_M = R_{M+1}$. Let M be the least integer such that $R_M = R_{M+1}$. Let $\mathcal{C} = \mathcal{C}_M$. Let $S = R_M$, and from now on we write $\mathcal{C} = (\Sigma, C, B', S)$, rather than $\mathcal{C}_M = (\Sigma, C, B', R_M)$.

Our aim is to show that $R^*(L) = L(\mathcal{C})$. To this end, first we show five preparatory lemmas, then the inclusion $L(\mathcal{C}) \subseteq R^*(L)$, then again five preparatory lemmas, and finally the inclusion $R^*(L) \subseteq L(\mathcal{C})$.

Lemma 2.3.6 $L = L(\mathcal{C}_0)$.

Proof. By direct inspection of the set R_0 of rules. □

Lemma 2.3.7 *For any $p \in T_\Sigma$, if $p \to^*_{R_0} \langle r[a_1, \ldots, a_n] \rangle$ for some $r \in \bar{T}_\Sigma(X_n)$, $n \geq 0$, and $a_1, \ldots, a_n \in B \cup E$, then $p = r[p_1, \ldots, p_n]$, where $p_i \in T_\Sigma$ and $p_i \to^*_{R_0} \langle a_i \rangle$ for $1 \leq i \leq n$.*

Proof. By direct inspection of the rules of R_0. □

The following statement is a simple consequence of Lemma 2.3.7.

Lemma 2.3.8 *For any $p \in T_\Sigma$, if $p \rightarrow^*_{R_0} \langle r[a_1, \ldots, a_n]\rangle$ for some $r \in \hat{T}_\Sigma(X_n)$, $n \geq 0$, and $a_1, \ldots, a_n \in B \cup E$, then $p = r[p_1, \ldots, p_n]$, where for each $1 \leq i \leq n$, if the variable x_i appears in the tree r, then $p_i \in T_\Sigma$ and $p_i \rightarrow^*_{R_0} \langle a_i \rangle$.*

Lemma 2.3.9 *For any $i \geq 1$, $p \in T_\Sigma$, $q, t \in T_{\Sigma \cup C}$, $k \geq 1$, and $v_1, \ldots, v_k \in T_{\Sigma \cup C}$, if*

$$p = v_1 \underset{R_0}{\rightarrow} v_2 \underset{R_0}{\rightarrow} \ldots \underset{R_0}{\rightarrow} v_k = q \underset{R_i}{\rightarrow} t \, , \tag{2.1}$$

and C_i applies an $(R_i - R_{i-1})$-rule in the last step $q \rightarrow_{R_i} t$ of (2.1), then there exists an $s \in T_\Sigma$ such that

$$s \underset{R}{\rightarrow} p \text{ and } s \overset{*}{\underset{R_{i-1}}{\rightarrow}} t \, . \tag{2.2}$$

Proof. Let α be the occurrence where C_i applies an $(R_i - R_{i-1})$-rule

$$\langle r[a_1, \ldots, a_n]\rangle \rightarrow c$$

in the last step $q \rightarrow_{R_i} t$ of (2.1). Then

$$q = u[\langle r[a_1, \ldots, a_n]\rangle] \, ,$$

where $u \in \bar{T}_\Sigma(X_1)$, $u/\alpha = x_1$, $r \in \hat{T}_\Sigma(X_n)$, $n \geq 0$, and $a_1, \ldots, a_n \in B \cup E$. By Lemma 2.3.8,

$$p = u[r[p_1, \ldots, p_n]] \, ,$$

for each $1 \leq i \leq n$, if the variable x_i appears in the tree r, then $p_i \in T_\Sigma$ and $p_i \rightarrow^*_{R_0} \langle a_i \rangle$. Finally, $t = u[c]$. By (b) of the definition of rules of R_i, $i \geq 1$, there is a rule $l \rightarrow r$ in R with $l \in \bar{T}_\Sigma(X_n)$, $n \geq 0$, and there are states and trees $a'_i \in B \cup E$ for $1 \leq i \leq n$ such that for each $1 \leq i \leq n$, $a'_i = a_i$ if x_i appears in the tree r, and that

$$l[\langle a'_1 \rangle, \ldots, \langle a'_n \rangle] \overset{*}{\underset{R_{i-1}}{\rightarrow}} c \, .$$

As C_{i-1} is connected, there are trees $q_1, \ldots, q_n \in T_\Sigma$ such that for each $1 \leq i \leq n$, if x_i appears in the tree r, then $q_i = p_i$, and that $q_i \rightarrow^*_{R_{i-1}} a'_i$. Let

$$s = u[l[q_1, \ldots, q_n]] \, .$$

Then

$$s \underset{R}{\rightarrow} p$$

and

$$s = u[l[q_1, \ldots, q_n]] \xrightarrow[R_0]{*} u[l[a_i', \ldots, a_n']] \xrightarrow[R_{i-1}]{*} u[c] = t\ .$$

Hence (2.2) holds. □

Lemma 2.3.10 *For every $i \geq 0$, $p \in T_\Sigma$, $q \in T_{\Sigma \cup C}$, if $p \to_{R_i}^* q$, then there is an $s \in T_\Sigma$ such that*

$$s \xrightarrow[R]{*} p \text{ and } s \xrightarrow[R_0]{*} q\ .$$

Proof. We proceed by induction on i. For $i = 0$ the statement is trivial. Let us suppose that $i \geq 1$ and that we have shown the statement for $1, 2, \ldots, i-1$. Let

$$p \xrightarrow[R_i]{*} q\ , \tag{2.3}$$

and let m be the number of $(R_i - R_{i-1})$-rules applied by C_i along (2.3). We show by induction on m that

$$\text{there is an } s \in T_\Sigma \text{ such that } s \xrightarrow[R]{*} p \text{ and } s \xrightarrow[R_0]{*} q\ . \tag{2.4}$$

If $m = 0$, then $p \to_{R_{i-1}}^* q$ and hence by the induction hypothesis on i, (2.4) holds.

Let us suppose that $m \geq 1$ and that for $0, 1, \ldots, m-1$, we have shown (2.4). Let $p \to_{R_i}^* q$ where $\mathcal{C}$ applies m $(R_i - R_{i-1})$-rules. Then there are integers n, k, $1 \leq k \leq n$, and there are trees $t_1, t_2, u_1, u_2, \ldots, u_n \in T_{\Sigma \cup C}$ such that (I), (II), (III), and (IV) hold.

(I) $p = u_1 \to_{R_i} \ldots \to_{R_i} u_k = t_1 \to_{R_i} u_{k+1} = t_2 \to_{R_i} \ldots \to_{R_i} u_n = q$.

(II) along the reduction subsequence $p = u_1 \to_{R_i} \ldots \to_{R_i} u_k = t_1$ of (I), $\mathcal{C}_i$ applies no $(R_i - R_{i-1})$-rule.

(III) in the rewriting step $u_k \to_{R_i} u_{k+1}$ $\mathcal{C}_i$ applies an $(R_i - R_{i-1})$-rule.

(IV) along the reduction subsequence $t_2 = u_{k+1} \to_{R_i} \ldots \to_{R_i} u_n = q$ of (I), $\mathcal{C}_i$ applies $m-1$ $(R_i - R_{i-1})$-rules.

By the induction hypothesis on i, there is a tree $s_1 \in T_\Sigma$ such that

$$s_1 \xrightarrow[R]{*} p \text{ and } s_1 \xrightarrow[R_0]{*} t_1\ . \tag{2.5}$$

Hence

$$s_1 \xrightarrow[R_0]{*} t_1 \xrightarrow[R_i]{} t_2 .$$

By Lemma 2.3.9, there is a tree $s_2 \in T_\Sigma$ such that

$$s_2 \xrightarrow[R]{} s_1 \text{ and } s_2 \xrightarrow[R_{i-1}]{*} t_2 . \tag{2.6}$$

Hence there is $j \geq 0$ and there are $w_1, \dots, w_j \in T_{\Sigma \cup C}$ such that

$$s_2 = w_1 \xrightarrow[R_{i-1}]{} w_2 \xrightarrow[R_{i-1}]{} \dots \xrightarrow[R_{i-1}]{} w_j = t_2 = u_{k+1} \xrightarrow[R_i]{} \dots \xrightarrow[R_i]{} u_n = q , \tag{2.7}$$

and along (2.7), $\mathcal{C}_i$ applies $m-1$ $(R_i - R_{i-1})$-rules. By the induction hypothesis on m, there is a tree $s_3 \in T_\Sigma$ such that

$$s_3 \xrightarrow[R]{*} s_2 \text{ and } s_3 \xrightarrow[R_0]{*} q .$$

Hence by (2.5) and (2.6),

$$s_3 \xrightarrow[R]{*} s_2 \xrightarrow[R]{} s_1 \xrightarrow[R]{*} p .$$

Thus (2.4) holds. □

Theorem 2.3.11 $L(\mathcal{C}) \subseteq R^*(L)$.

Proof. Let $p \in L(\mathcal{C})$. Then $p \to_S^* b$ for some $b \in B'$. Hence by Lemma 2.3.10, there is an $s \in T_\Sigma$ such that

$$s \xrightarrow[R]{*} p \text{ and } s \xrightarrow[R_0]{*} b . \tag{2.8}$$

Hence $s \in L(\mathcal{C}_0)$. By Lemma 2.3.6, $s \in L$. Thus by (2.8), $p \in R^*(L)$. □

Now we show the inclusion $R^*(L) \subseteq L(\mathcal{C})$. First we prove five lemmas.

Lemma 2.3.12 *Let $l_1 \to r_1$ and $l_2 \to r_2$ be rules in R. Let $\alpha \in path(r_1)$, where $r_1/\alpha \in T_\Sigma(X_j)$, $j \geq 0$. Let $\beta \in path(l_2)$, where $l_2/\beta \in T_\Sigma(X) - X$, and let $s \in \bar{T}_\Sigma(X_k) - X$, $k \geq 1$, be a supertree of l_2/β. Let $\alpha = \lambda$ or $\beta = \lambda$. Let*

$$(r_1/\alpha)[a_1, \dots, a_j] = s[z_1, \dots, z_k] , \tag{2.9}$$

where $a_1, \dots, a_j \in B \cup E$, $z_1, \dots, z_k \in T_{\Sigma \cup B}$. Let $\gamma \in path(s)$ be such that $l_2/\beta\gamma \in X$, and $s/\gamma = x_\nu$, for some $1 \leq \nu \leq k$. Then $z_\nu \in B \cup E$.

Proof. Let $l_1 \in T_\Sigma(X_m)$ for some $m \geq 0$. Let $l_3 = s[x_{m+1}, \ldots, x_{m+k}]$. Then $l_3 \in T_\Sigma(\{ x_{m+1}, \ldots, x_{m+k} \})$ is a supertree of l_2/β, for each $m+1 \leq i \leq m+k$, x_i appears exactly once in l_3. Moreover, $var(l_1) \cap var(l_3) = \emptyset$, and by (2.9),

$$(r_1/\alpha)[a_1, \ldots, a_j] = l_3[x_{m+1} \leftarrow z_1, \ldots, x_{m+k} \leftarrow z_k] \,. \qquad (2.10)$$

Let $\theta_1 : X \to T_\Sigma(X)$ be a most general unifier of r_1/α and l_3. By (2.10), there is a substitution $\theta_2 : X \to T_{\Sigma \cup B}(X)$ such that

$$\theta_2(\theta_1(r_1/\alpha)) = (r_1/\alpha)[a_1, \ldots, a_j] = l_3[x_{m+1} \leftarrow z_1, \ldots, x_{m+k} \leftarrow z_k] =$$

$$\theta_2(\theta_1(l_3)),$$

where $\theta_2(\theta_1(x_i)) = a_i$ for $1 \leq i \leq j$ and $\theta_2(\theta_1(x_{m+i})) = z_i$ for $1 \leq i \leq k$. By Definition 2.3.1 and by the definition of E, $\theta_1(x_{m+\nu}) \in X \cup E$. If $\theta_1(x_{m+\nu}) \in X$, then $\theta_2(\theta_1(x_{m+\nu}))$ is a subtree of a_μ for some $\mu \in \{1, \ldots, j\}$. Hence by the definition of E, $z_\nu = \theta_2(\theta_1(x_{m+\nu})) \in B \cup E$. If $\theta_1(x_{m+\nu}) \in E$, then $z_\nu = \theta_2(\theta_1(x_{m+\nu})) = \theta_1(x_{m+\nu}) \in E$. □

Intuitively, the following lemma states that along a reduction sequence of S we can reverse the order of the consecutive application of a (ii)-type rule of R_0 at $\alpha \in N^*$ and the application of an $(S - R_0)$-rule at $\beta \in N^*$ if α is not a prefix of β and β is not a prefix of α.

Lemma 2.3.13 *Let*

$$u_1 \underset{S}{\to} u_2 \underset{S}{\to} u_3$$

be a reduction sequence of $\mathcal{C}$. Let $\alpha \in path(u_1)$, and $\beta \in path(u_2)$ be such that $u_1 \to_S u_2$ applying a (ii)-type rule **rule₁** *of R_0 at α, and that $u_2 \to_S u_3$ applying an $(S - R_0)$-rule* **rule₂** *at β. If α is not a prefix of β and β is not a prefix of α, then there is a tree $v \in T_{\Sigma \cup C}$ such that $u_1 \to_S v$ applying* **rule₂** *at β, and $v \to_S u_3$ applying* **rule₁** *at α.*

Proof. Straightforward. □

Lemma 2.3.14 *Let $i \geq 0$, $t \in \bar{T}_{\Sigma \cup C}(X_1)$, $\alpha \in path(t)$, $t/\alpha = x_1$, $c \in \{ 1, \ldots, |D - B| \}$, and $b \in B$. Let*

$$t[c] = u_1 \underset{R_i}{\to} u_2 \underset{R_i}{\to} \ldots \underset{R_i}{\to} u_n = b \qquad (2.11)$$

with $n \geq 1$, $u_1, \ldots, u_n \in T_{\Sigma \cup C}$. Then along (2.11), $\mathcal{C}_i$ applies a rule in $R_i - R_0$ at some prefix β of α.

Proof. By direct inspection of the construction of the $\mathcal{C}_i$'s. □

Lemma 2.3.15 *For any $n \geq 0$, $u \in \bar{T}_\Sigma(X_n)$, $v_1, \ldots, v_n, v \in D$, $m \geq 1$, and $w_1, \ldots, w_m \in T_{\Sigma \cup C}$, if*

$$u[\langle v_1 \rangle, \ldots, \langle v_n \rangle] = w_1 \xrightarrow[S]{} w_2 \xrightarrow[S]{} \ldots \xrightarrow[S]{} w_m = \langle v \rangle, \tag{2.12}$$

and $\mathcal{C}$ applies only (ii)-type rules of R_0 along (2.12), then $u[v_1, \ldots, v_n] = v$.

Proof. We proceed by induction on $height(u)$. The basis $height(u) = 0$ of the induction is trivial. The induction step is a simple consequence of (ii) in the definition of R_0 and of the inclusion $R_0 \subseteq S$. □

Lemma 2.3.16 *Let $t \in L(\mathcal{C})$, $m \geq 1$, $t_1, \ldots, t_m \in T_{\Sigma \cup C}$, $b \in B'$, and let*

$$t = t_1 \xrightarrow[S]{} t_2 \xrightarrow[S]{} t_3 \xrightarrow[S]{} \ldots \xrightarrow[S]{} t_m = b\ . \tag{2.13}$$

Let $l \to r$ be a rule in R, where $l \in \bar{T}_\Sigma(X_n)$ and $n \geq 1$. Moreover, let $1 \leq j \leq m$, and let

$$t_j/\alpha = l[\langle v_1 \rangle, \ldots, \langle v_n \rangle]\ , \tag{2.14}$$

where $n \geq 1$, $v_1, \ldots, v_n \in D$, $\alpha \in path(t_j)$. Let $\alpha_1, \ldots, \alpha_n \in path(l)$ such that

$$l/\alpha_i = x_i \text{ for } 1 \leq i \leq n\ . \tag{2.15}$$

Consider the reduction subsequence

$$t_j \xrightarrow[S]{} t_{j+1} \xrightarrow[S]{} \ldots \xrightarrow[S]{} t_m = b \tag{2.16}$$

of (2.13). If $\mathcal{C}$ does not apply any rules at the occurrences $\alpha\alpha_1, \ldots, \alpha\alpha_n$ along (2.16), then $v_1, \ldots, v_n \in B \cup E$.

Proof. Let $1 \leq i \leq n$, and let us assume that $v_i \in D - B$. By (2.14) and (2.15),

$$t_j/\alpha\alpha_i = \langle v_i \rangle\ . \tag{2.17}$$

By Lemma 2.3.14, $\mathcal{C}$ applies a rule in $S - R_0$ at some prefix of $\alpha\alpha_i$ along (2.16). Let $\beta \in path(t_j)$ be the longest prefix of $\alpha\alpha_i$ such that $\mathcal{C}$ applies a rule **rule** in $S - R_0$ at β along (2.16). Then **rule** is of the form $\langle r_1[a_1, \ldots, a_\kappa] \rangle \to c$, where $\kappa \geq 0$, $r_1 \in \bar{T}_\Sigma(X_\kappa)$, $a_1, \ldots, a_\kappa \in B \cup E$, and there is a rule $l_1 \to r_1$ in R. Moreover there exists ξ, $j < \xi \leq m$, such that

$$t_j/\beta \underset{S}{\overset{*}{\to}} t_{j+1}/\beta \underset{S}{\overset{*}{\to}} \cdots \underset{S}{\overset{*}{\to}} t_\xi/\beta = \langle r_1[a_1, \ldots, a_\kappa]\rangle \ ,$$

where for each π, $j \leq \pi \leq \xi - 1$, $t_\pi/\beta = t_{\pi+1}/\beta$ or $t_\pi/\beta \to_S t_{\pi+1}/\beta$. We lose no generality by assuming that

$$t_j/\beta \underset{S}{\to} t_{j+1}/\beta \underset{S}{\to} \cdots \underset{S}{\to} t_\xi/\beta = \langle r_1[a_1, \ldots, a_\kappa]\rangle \ . \tag{2.18}$$

By Lemma 2.3.13 we may assume that there exists ν, $j \leq \nu \leq \xi$ such that

(a) along the reduction subsequence

$$t_j/\beta \underset{S}{\to} \cdots \underset{S}{\to} t_\nu/\beta \tag{2.19}$$

of (2.18) no rule is applied at any prefix of $\alpha\alpha_i$, that

(b) along (2.19) each application of a (ii)-type rule of R_0 at some $\delta \in N^*$ is followed somewhere later by an application of an $S - R_0$-rule of S at a prefix ϵ of δ, and that

(c) along the reduction subsequence

$$t_\nu/\beta \underset{S}{\to} \cdots \underset{S}{\to} t_\xi/\beta = \langle r_1[a_1, \ldots, a_\kappa]\rangle$$

of (2.18), S applies only (ii)-type rules of R_0.

Then

$$t_\nu/\beta = s[\langle z_1\rangle, \ldots, \langle z_k\rangle] \tag{2.20}$$

for some $k \geq 1$, $s \in \bar{T}_\Sigma(X_k)$, and $\langle z_1\rangle, \ldots, \langle z_k\rangle \in C$. By (2.20), (c) of the definition of ν, and Lemma 2.3.15,

$$s[z_1, \ldots, z_k] = r_1[a_1, \ldots, a_\kappa] \ . \tag{2.21}$$

The word α is a prefix of β or β is a prefix of α. Hence we can distinguish two cases.

Case 1 α is a prefix of β, see Figure 2.4. In this case,

$$\beta = \alpha\gamma \tag{2.22}$$

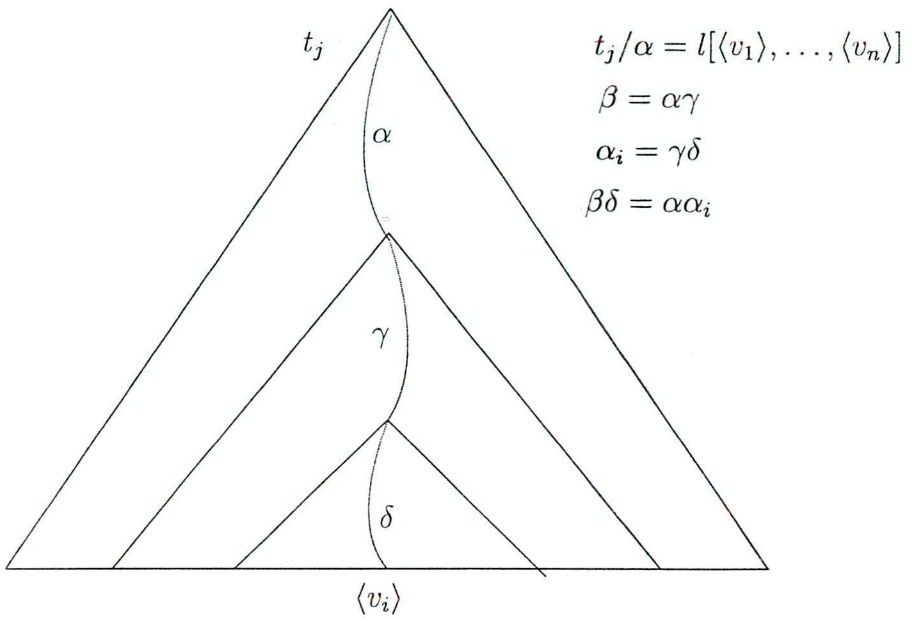
t_j
$t_j/\alpha = l[\langle v_1\rangle, \ldots, \langle v_n\rangle]$
$\beta = \alpha\gamma$
$\alpha_i = \gamma\delta$
$\beta\delta = \alpha\alpha_i$
α
γ
δ
$\langle v_i\rangle$

Figure 2.4: Case 1. of Lemma 2.3.16

for some $\gamma \in N^*$, and hence t_ν/β is a subtree of t_ν/α. Now by (2.14), the definition of ν, and (2.20),

$$s \text{ is a supertree of } l/\gamma\ . \tag{2.23}$$

Let ω be the prefix of $\alpha\alpha_i$ with $length(\omega) = length(\alpha\alpha_i) - 1$. Observe that C applies a (ii)-type rule of R_0 at the occurrence ω along (2.16). Hence

$$s \notin X\ . \tag{2.24}$$

Let $\delta \in N^*$ be defined by the equation $\gamma\delta = \alpha_i$. Then

$$\beta\delta = \alpha\alpha_i\ , \tag{2.25}$$

and by (a) of the definition of ν,

$$\delta \in path(s), \delta \in path(l/\gamma), \text{ and } (l/\gamma)/\delta = x_i\ . \tag{2.26}$$

By (2.25) and by (a) of the definition of ν,

$$\beta\delta \in path(t_\nu)\ .$$

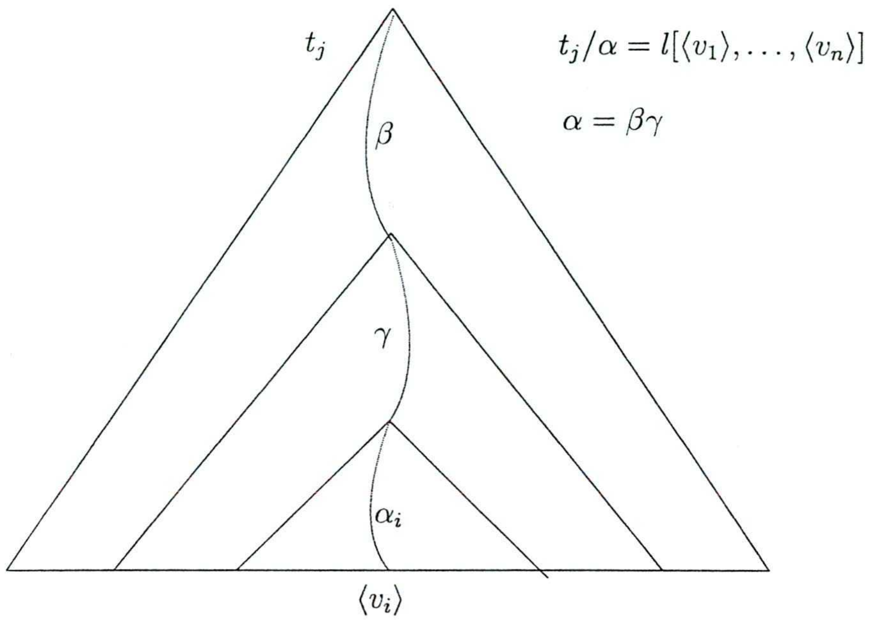

Figure 2.5: Case 2. of Lemma 2.3.16

By (2.17), (2.25), (a) of the definition of ν, and (2.20),

$$\langle v_i\rangle = (t_j/\beta)/\delta = (t_\nu/\beta)/\delta = s[\langle z_1\rangle, \dots, \langle z_k\rangle]/\delta = \langle z_\mu\rangle \tag{2.27}$$

for some $1 \le \mu \le k$. As R is gsm, by (2.23), (2.24), (2.26), (2.21), and Lemma 2.3.12, $z_\mu \in B \cup E$. By (2.27), $v_i \in B \cup E$.

Case 2 β is a prefix of α, see Figure 2.5. In this case

$$\alpha = \beta\gamma \tag{2.28}$$

for some $\gamma \in N^*$, and hence t_j/α is a subtree of t_j/β. Now by (2.14), the definition of ν, and (2.20),

$$s/\gamma \text{ is a supertree of } l\ . \tag{2.29}$$

Moreover, by (a) of the definition of ν,

$$\alpha_i \in path(s/\gamma), l/\alpha_i \in X, \text{ and } (s/\gamma)/\alpha_i \in X\ . \tag{2.30}$$

Let ω be the prefix of $\alpha\alpha_i$ with $length(\omega) = length(\alpha\alpha_i) - 1$. Observe that $\mathcal{C}$ applies a (ii)-type rule of R_0 at the occurrence ω along (2.16). Hence

$$s/\gamma \notin X . \tag{2.31}$$

By (2.28) and by (a) of the definition of ν,

$$\beta\gamma\alpha_i = \alpha\alpha_i \in path(t_\nu) . \tag{2.32}$$

Then by (2.17), (2.32), (a) of the definition of ν, and (2.20),

$$\langle v_i \rangle = (t_j/\beta)/\gamma\alpha_i = (t_\nu/\beta)/\gamma\alpha_i = s[\langle z_1 \rangle, \ldots, \langle z_k \rangle]/\gamma\alpha_i = \langle z_\mu \rangle \tag{2.33}$$

for some $1 \leq \mu \leq k$. By (2.21),

$$(s/\gamma)[z_1, \ldots, z_k] = s[z_1, \ldots, z_k]/\gamma = r_1[a_1, \ldots, a_\kappa]/\gamma . \tag{2.34}$$

As R is gsm, by (2.29), (2.31), (2.30), (2.33), (2.34), and Lemma 2.3.12, $z_\mu \in B \cup E$. By (2.33), $v_i \in B \cup E$. □

Theorem 2.3.17 $R^*(L) \subseteq L(\mathcal{C})$.

Proof. By (i) in the definition of R_0, $R_\mathcal{B} \subseteq R_0$. Hence $L \subseteq L(\mathcal{C}_0)$. As $R_{i-1} \subseteq R_i$ for $i \geq 1$, we have $L \subseteq L(\mathcal{C}_i)$ for $i \geq 0$. Hence $L \subseteq L(\mathcal{C})$. Thus it is sufficient to show that for each $t \in L(\mathcal{C})$, if $t \rightarrow_R t'$, then $t' \in L(\mathcal{C})$. To this end, let us suppose that $t \rightarrow_R t'$, applying the rule $l \rightarrow r$ in R at $\alpha \in path(t)$. Here $l \in \bar{T}_\Sigma(X_n)$ for some $n \geq 0$. Let $\alpha_1, \ldots, \alpha_n \in path(l)$ be such that

$$l/\alpha_i = x_i \text{ for } 1 \leq i \leq n .$$

Then

$$t = s[l[u_1, \ldots, u_n]] ,$$

where $s \in \bar{T}_\Sigma(X_1)$, $\alpha \in path(s)$, $s/\alpha = x_1$, and $u_1, \ldots, u_n \in T_\Sigma$. Moreover,

$$t' = t[\alpha \leftarrow r[u_1, \ldots, u_n]] = s[r[u_1, \ldots, u_n]] .$$

As $t \in L(\mathcal{C})$, there is a reduction sequence

$$t = t_1 \underset{S}{\rightarrow} t_2 \underset{S}{\rightarrow} t_3 \underset{S}{\rightarrow} \ldots \underset{S}{\rightarrow} t_m = b, \tag{2.35}$$

where $m \geq 1$, $b \in B'$, $t_1, \ldots, t_m \in T_{\Sigma \cup C}$, and there are integers j, k with $1 \leq j, k \leq m$ such that

(i) $t_j = s[l[\langle v_1\rangle, \ldots, \langle v_n\rangle]]$, where $v_i \in D$ and $u_i \rightarrow_S^* \langle v_i\rangle$ for $1 \leq i \leq n$,

(ii) $t_k = s[c_0]$, for some $c_0 \in C$, where $l[\langle v_1\rangle, \ldots, \langle v_n\rangle] \rightarrow_S^* c_0$, and that

(iii) along the reduction subsequence $t_j \rightarrow_S t_{j+1} \rightarrow_S \ldots \rightarrow_S t_k$ of (2.35), $\mathcal{C}$ does not apply any rules at the occurrences $\alpha\alpha_1, \ldots, \alpha\alpha_n$.

By Lemma 2.3.16, $v_1, \ldots, v_n \in B \cup E$. Hence by Condition (ii) in the definition of R_i, $i \geq 1$, and by the definition of $\mathcal{C}$, the rule $r[\langle v_1\rangle, \ldots, \langle v_n\rangle] \rightarrow c_0$ is in S. Thus we get

$$t' = s[r[u_1, \ldots, u_n]] \xrightarrow[S]{*} s[r[\langle v_1\rangle, \ldots, \langle v_n\rangle]] \xrightarrow[S]{} s[c_0] \xrightarrow[S]{*} b\,.$$

As $b \in B'$, $t' \in L(\mathcal{C})$. □

By Theorems 2.3.11 and 2.3.17, we get the following.

Theorem 2.3.18 $R^*(L) = L(\mathcal{C})$.

As Δ, R, Σ ($\Delta \subseteq \Sigma$), and $\mathcal{B}$ are arbitrary, we have the following result.

Theorem 2.3.19 *Linear generalized semi-monadic rewriting systems effectively preserve recognizability.*

Theorem 2.3.20 *A tree language L is recognizable if and only if there exists a rewriting system R such that $R \cup R^{-1}$ is a rewriting system preserving recognizability and that L is the union of finitely many $\leftrightarrow_R^*$-classes.*

Proof. Let us assume that L is recognizable. Then by Proposition 2.1.1 there is a ground rewriting system R such that L is the union of finitely many $\leftrightarrow_R^*$-classes. Clearly, $R \cup R^{-1}$ is an lgsm rewriting system and hence, by Theorem 2.3.19, preserves recognizability.

Let us assume that there exists a rewriting system R such that $R \cup R^{-1}$ is a rewriting system preserving recognizability and that L is the union of finitely many $\leftrightarrow_R^*$-classes. That is to say, $L = [t_1]_R \cup [t_2]_R \cup \ldots \cup [t_k]_R$ for some $k \geq 0$. As $\rightarrow_{R \cup R^{-1}}^* = \leftrightarrow_R^*$, $L = (R \cup R^{-1})^*(\{t_1, \ldots, t_k\})$. It should be clear that the tree language $\{t_1, \ldots, t_k\}$ is recognizable. Since $R \cup R^{-1}$ preserves recognizability, L is also recognizable. □

The following theorem is a simple consequence of our results.

Theorem 2.3.21 *A tree language L is recognizable if and only if there exists a rewriting system R such that $R \cup R^{-1}$ is an lgsm rewriting system and that L is the union of finitely many $\leftrightarrow_R^*$-classes.*

Example 2.3.22 Let $\Sigma = \Sigma_0 \cup \Sigma_1 \cup \Sigma_2$, $\Sigma_0 = \{ \sharp, \$ \}$, $\Sigma_1 = \{ f \}$, $\Sigma_2 = \{ g \}$. Let R consist of the rules

$$g(g(x_1, \$), x_2) \to f(g(g(\$, x_1), x_2)) ,$$

$$g(g(\$, x_2), x_1) \to f(g(g(\$, x_1), x_2)) .$$

Then $R \cup R^{-1}$ is an lgsm rewriting system. Hence, by Theorem 2.3.21, the union of finitely many arbitrary $\leftrightarrow_R^*$-classes is recognizable.

Let S be a string rewriting system over Σ. We say that S is restricted right-left overlapping if there is no rule $\lambda \to r$ in S, and the following holds. For any rules $l_1 \to r_1$ and $l_2 \to r_2$ in S, for any nonempty suffix $u \in \Sigma^+$ of r_1 and any nonempty suffix $v \in \Sigma^+$ of l_2, if $u = r_1$ or $v = l_2$, then v cannot be a proper prefix of u. For example the string rewriting system

$$\{ \textit{apple} \to \textit{peach} \}$$

is restricted right-left overlapping.

It is not hard to see that each monadic string rewriting system is restricted right-left overlapping as well.

The following theorem is an interesting consequence of our results on rewriting systems.

Theorem 2.3.23 *Restricted right-left overlapping string rewriting systems effectively preserve recognizability. Moreover, a string language L is recognizable if and only if there exists a string rewriting system S such that $S \cup S^{-1}$ is a restricted right-left overlapping string rewriting system and that L is the union of finitely many $\leftrightarrow_S^*$-classes.*

2.3.2 An example

In this subsection we illustrate the construction of C_j, $j \geq 0$ by an example. Let $\Sigma = \Sigma_0 \cup \Sigma_1 \cup \Sigma_2$, $\Sigma_0 = \{ \sharp \}$, $\Sigma_1 = \{ f \}$, $\Sigma_2 = \{ g \}$. Let the rewriting system R over Σ consist of the following two rules.

$$\begin{aligned} f(f(g(x_1,\sharp)) &\to f(f(x_1))\,, \\ g(x_1,x_2) &\to f(g(x_1,\sharp))\,. \end{aligned}$$

By direct inspection we obtain that R is an lgsm rewriting system. Here $E = \{\, \sharp \,\}$. Let $L = \{\, g(\sharp, \sharp) \,\}$. It is not hard to see that

$$R^*(L) = \{\, f^n(g(\sharp,\sharp)) \mid n \geq 0 \,\} \cup \{\, f^n(\sharp) \mid n \geq 2 \,\}\,.$$

Consider the bottom-up tree automaton $\mathcal{B} = (\Sigma, B, B', R_{\mathcal{B}})$, where $B = \{\, b_1, b_2 \,\}$, $B' = \{\, b_2 \,\}$, and $R_{\mathcal{B}}$ consists of the following two rules: $\sharp \to b_1$, $g(b_1, b_1) \to b_2$. It is not hard to show that $L = L(\mathcal{B})$. By direct inspection we obtain that the set of subterms of the right-hand sides of the rules of R is

$$\{\, x_1, f(x_1), f(f(x_1)), \sharp, g(x_1,\sharp), f(g(x_1,\sharp)) \,\}\,.$$

Then

$$\begin{aligned} D = \{\, & b_1, b_2, \sharp, f(b_1), f(b_2), f(\sharp), f(f(b_1)), f(f(b_2)), f(f(\sharp)), g(b_1,\sharp), \\ & g(b_2,\sharp), g(\sharp,\sharp), f(g(b_1,\sharp)), f(g(b_2,\sharp)), f(g(\sharp,\sharp)) \,\}\,. \end{aligned}$$

Moreover, $C = \{\, b_1, b_2, 1, \ldots, 13 \,\}$. Let $\langle\ \rangle : D \to \{\, 1, \ldots, |D - B| \,\}$ be defined by

$$\begin{array}{lclllclllcl} \langle b_1 \rangle &=& b_1, & \langle b_2 \rangle &=& b_2, & \langle \sharp \rangle &=& 1, \\ \langle f(b_1) \rangle &=& 2, & \langle f(b_2) \rangle &=& 3, & \langle f(\sharp) \rangle &=& 4, \\ \langle f(f(b_1)) \rangle &=& 5, & \langle f(f(b_2)) \rangle &=& 6, & \langle f(f(\sharp)) \rangle &=& 7, \\ \langle g(b_1,\sharp) \rangle &=& 8, & \langle g(b_2,\sharp) \rangle &=& 9, & \langle g(\sharp,\sharp) \rangle &=& 10, \\ \langle f(g(b_1,\sharp)) \rangle &=& 11, & \langle f(g(b_2,\sharp)) \rangle &=& 12, & \langle f(g(\sharp,\sharp)) \rangle &=& 13. \end{array}$$

Then $\mathcal{C}_0 = (\Sigma, C, B', R_0)$ is determined by the set R_0 of rules. R_0 consists of the following fifteen rules.

$$\begin{array}{lcllcl} \sharp &\to& b_1, & g(b_1,b_1) &\to& b_2, \\ \sharp &\to& \langle \sharp \rangle, & f(b_1) &\to& \langle f(b_1) \rangle, \\ f(b_2) &\to& \langle f(b_2) \rangle, & f(\langle \sharp \rangle) &\to& \langle f(\sharp) \rangle, \\ f(\langle f(b_1) \rangle) &\to& \langle f(f(b_1)) \rangle, & f(\langle f(b_2) \rangle) &\to& \langle f(f(b_2)) \rangle, \\ f(\langle f(\sharp) \rangle) &\to& \langle f(f(\sharp)) \rangle, & g(b_1, \langle \sharp \rangle) &\to& \langle g(b_1,\sharp) \rangle, \\ g(b_2, \langle \sharp \rangle) &\to& \langle g(b_2,\sharp) \rangle, & g(\langle \sharp \rangle, \langle \sharp \rangle) &\to& \langle g(\sharp,\sharp) \rangle, \\ f(\langle g(b_1,\sharp) \rangle) &\to& \langle f(g(b_1,\sharp)) \rangle, & f(\langle g(b_2,\sharp) \rangle) &\to& \langle f(g(b_2,\sharp)) \rangle, \\ f(\langle g(\sharp,\sharp) \rangle) &\to& \langle f(g(\sharp,\sharp)) \rangle. & & & \end{array}$$

That is, R_0 consists of the following fifteen rules.

$$\begin{array}{llllll}
\sharp & \to b_1, & g(b_1,b_1) & \to b_2, & \sharp & \to 1, \\
f(b_1) & \to 2, & f(b_2) & \to 3, & f(1) & \to 4, \\
f(2) & \to 5, & f(3) & \to 6, & f(4) & \to 7, \\
g(b_1,1) & \to 8, & g(b_2,1) & \to 9, & g(1,1) & \to 10, \\
f(8) & \to 11, & f(9) & \to 12, & f(10) & \to 13.
\end{array}$$

The bottom-up tree automaton $\mathcal{C}_1 = (\Sigma, C, B', R_1)$ is determined by the set R_1 of rules. R_1 contains all rules of R_0 and the following five rules.

$$\begin{array}{llll}
\langle f(f(b_1))\rangle & \to \langle f(f(b_2))\rangle, & \langle f(g(b_1,\sharp))\rangle & \to b_2, \\
\langle f(g(b_1,\sharp))\rangle & \to \langle g(b_1,\sharp)\rangle, & \langle f(g(b_2,\sharp))\rangle & \to \langle g(b_2,\sharp)\rangle, \\
\langle f(g(\sharp,\sharp))\rangle & \to \langle g(\sharp,\sharp)\rangle. & &
\end{array}$$

That is, R_1 contains all rules of R_0 and the following five rules.

$$5 \to 6, 11 \to b_2, 11 \to 8, 12 \to 9, 13 \to 10.$$

The bottom-up tree automaton $\mathcal{C}_2 = (\Sigma, C, B', R_2)$ is determined by the set R_2 of rules. R_2 contains all rules of R_1 and the following seven rules.

$$\begin{array}{llll}
\langle f(f(b_1))\rangle & \to \langle f(g(b_1,\sharp))\rangle, & \langle f(f(b_1))\rangle & \to \langle g(b_1,\sharp)\rangle, \\
\langle f(f(b_1))\rangle & \to b_2, & \langle f(f(b_2))\rangle & \to \langle f(g(b_2,\sharp))\rangle, \\
\langle f(f(b_2))\rangle & \to \langle g(b_2,\sharp)\rangle, & \langle f(f(\sharp))\rangle & \to \langle f(g(\sharp,\sharp))\rangle, \\
\langle f(f(\sharp))\rangle & \to \langle g(\sharp,\sharp)\rangle. & &
\end{array}$$

That is, R_2 contains all rules of R_1 and the following seven rules.

$$5 \to 11, 5 \to 8, 5 \to b_2, 6 \to 12, 6 \to 9, 7 \to 13, 7 \to 10.$$

The bottom-up tree automaton $\mathcal{C}_3 = (\Sigma, C, B', R_3)$ is determined by the set R_3 of rules. R_3 contains all rules of R_2 and the following two rules.

$$\langle f(f(b_1))\rangle \to \langle f(g(b_2,\sharp))\rangle, \quad \langle f(f(b_1))\rangle \to \langle g(b_2,\sharp)\rangle.$$

That is, R_3 contains all rules of R_2 and the following two rules.

$$5 \to 12, \quad 5 \to 9.$$

Since $R_4 = R_3$, the bottom-up tree automaton $\mathcal{C}_4 = (\Sigma, C, B', R_4)$ is equal to $\mathcal{C}_3 = (\Sigma, C, B', R_3)$. Let $S = R_4$ and let us write $\mathcal{C} = (\Sigma, C, B', S)$ for $\mathcal{C}_4 = (\Sigma, C, B', R_4)$. Hence S consists of the following twenty-nine rules.

$$
\begin{array}{llllllll}
\sharp & \rightarrow b_1, & g(b_1,b_1) & \rightarrow b_2, & \sharp & \rightarrow 1, \\
f(b_1) & \rightarrow 2, & f(b_2) & \rightarrow 3, & f(1) & \rightarrow 4, \\
f(2) & \rightarrow 5, & f(3) & \rightarrow 6, & f(4) & \rightarrow 7, \\
g(b_1,1) & \rightarrow 8, & g(b_2,1) & \rightarrow 9, & g(1,1) & \rightarrow 10, \\
f(8) & \rightarrow 11, & f(9) & \rightarrow 12, & f(10) & \rightarrow 13, \\
5 & \rightarrow 6, & 11 & \rightarrow b_2, & 11 & \rightarrow 8, \\
12 & \rightarrow 9, & 13 & \rightarrow 10, & 5 & \rightarrow 11, \\
5 & \rightarrow 8, & 5 & \rightarrow b_2, & 6 & \rightarrow 12, \\
6 & \rightarrow 9, & 7 & \rightarrow 13, & 7 & \rightarrow 10, \\
5 & \rightarrow 12, & 5 & \rightarrow 9. & &
\end{array}
$$

By direct inspection we obtain that the states 3, 4, 6, 7, 9, 10, 12, and 13 are superfluous as no final state can be reached from any of them. Hence we drop all of them and also omit all rules in which they appear. In this way we obtain the bottom-up tree automaton $\mathcal{A}_1 = (\Sigma, C, B', S_1)$, where S_1 consists of the following twelve rules.

$$
\begin{array}{llllll}
\sharp & \rightarrow b_1, & g(b_1,b_1) & \rightarrow b_2, & \sharp & \rightarrow 1, \\
f(b_1) & \rightarrow 2, & f(2) & \rightarrow 5, & g(b_1,1) & \rightarrow 8, \\
f(8) & \rightarrow 11, & 11 & \rightarrow b_2, & 11 & \rightarrow 8, \\
5 & \rightarrow 11, & 5 & \rightarrow 8, & 5 & \rightarrow b_2.
\end{array}
$$

It is not hard to see that the rule $5 \rightarrow 11$ is superfluous. We obtain the bottom-up tree automaton $\mathcal{A}_2 = (\Sigma, C, B', S_2)$, from $\mathcal{A}_1$ by dropping the rule $5 \rightarrow 11$. Thus S_2 consists of the following eleven rules.

$$
\begin{array}{llllll}
\sharp & \rightarrow b_1, & g(b_1,b_1) & \rightarrow b_2, & \sharp & \rightarrow 1, \\
f(b_1) & \rightarrow 2, & f(2) & \rightarrow 5, & g(b_1,1) & \rightarrow 8, \\
f(8) & \rightarrow 11, & 11 & \rightarrow b_2, & 11 & \rightarrow 8, \\
5 & \rightarrow 8, & 5 & \rightarrow b_2. & &
\end{array}
$$

We define the deterministic bottom-up tree automaton $\mathcal{A}_3 = (\Sigma, C, A', S_3)$, from $\mathcal{A}_2$ by applying the subset construction. Then S_3 consists of the seven following rules.

$$\begin{array}{llll}
\sharp & \to \{b_1, 1\}, & g(\{b_1,1\},\{b_1,1\}) & \to \{b_2, 8\}, \\
f(\{b_1,1\}) & \to \{2\}, & f(\{b_2,8\}) & \to \{8, 11, b_2\}, \\
f(\{2\}) & \to \{5, 8, b_2\}, & f(\{5,8,b_2\}) & \to \{8, 11, b_2\}, \\
f(\{8,11,b_2\}) & \to \{8, 11, b_2\}. & &
\end{array}$$

Moreover, A' consists of the three states $\{b_2, 8\}, \{8, 11, b_2\}, \{5, 8, b_2\}$. Let us redenote the states of $\mathcal{A}_3$ as follows. Let

$$a_1 = \{b_1, 1\}, a_2 = \{b_2, 8\}, a_3 = \{2\}, a_4 = \{8, 11, b_2\}, a_5 = \{5, 8, b_2\}.$$

Hence S_3 consists of the following seven rules:

$$\begin{array}{llllll}
\sharp & \to a_1, & g(a_1, a_1) & \to a_2, & f(a_1) & \to a_3, \\
f(a_2) & \to a_4, & f(a_3) & \to a_5, & f(a_5) & \to a_4, \\
f(a_4) & \to a_4. & & & &
\end{array}$$

Moreover, $A' = \{a_2, a_4, a_5\}$.

It should be clear that the states a_2, a_4, a_5 are equivalent. Finally we construct a minimal deterministic bottom-up tree automaton $\mathcal{A}_4 = (\Sigma, C, A'', S_3)$, from $\mathcal{A}_3$ by merging the equivalent states a_2, a_4, a_5. Hence S_4 consists of the following five rules.

$$\sharp \to a_1, g(a_1, a_1) \to a_2, f(a_1) \to a_3, f(a_2) \to a_2, f(a_3) \to a_2.$$

Moreover, $A'' = \{a_2\}$. We obtain by direct inspection that $L(\mathcal{A}_4) = R^*(L)$.

2.4 Results on term and string rewriting systems preserving recognizability

In this section we study rewriting systems preserving recognizability and gsm rewriting systems. First we present a ranked alphabet Σ and a linear rewriting system R over Σ such that R preserves Σ-recognizability but does not preserve recognizability.

Theorem 2.4.1 *There is a ranked alphabet Σ and there is a linear rewriting system R over Σ such that R preserves Σ-recognizability but does not preserve recognizability.*

Proof. Let $\Sigma = \Sigma_1 \cup \Sigma_0$, $\Sigma_1 = \{ f, g \}$, $\Sigma_0 = \{ \sharp \}$. Let R consist of the following five rules.

$f(g(x_1)) \to f(f(g(g(x_1))))$, $f(\sharp) \to \sharp$, $g(\sharp) \to \sharp$, $\sharp \to f(\sharp)$, $\sharp \to g(\sharp)$.
It should be clear that for each tree $t \in T_\Sigma$, $t \to_R^* \sharp$, and $\sharp \to_R^* t$. Hence for each nonempty tree language $L \subseteq T_\Sigma$, $R^*(L) = T_\Sigma$. Thus R preserves Σ-recognizability.

Let $\Delta = \Sigma \cup \{ h \}$, where $h \in \Delta_1$. Let $L = \{ f(g(h(\sharp))) \}$. Since L is finite, L is recognizable. However, $R^*(L) = \{ f^n(g^n(h(t))) \mid n \geq 0, t \in T_\Sigma \}$ is not recognizable. $\square$

We shall need the following two statements, of which the proof are straightforward.

Lemma 2.4.2 *Let R be a rewriting system over Σ. Then the following statements are equivalent.*

(i) *R (effectively) preserves recognizability.*

(ii) *For each ranked alphabet Δ with $sign(R) \subseteq \Delta$, R (effectively) preserves Δ-recognizability.*

Theorem 2.4.3 *Let R be a rewriting system over $sign(R)$, and let $\Sigma = \{ f, \sharp \} \cup sign(R)$, where $f \in \Sigma_2 - sign(R)$ and $\sharp \in \Sigma_0 - sign(R)$. Then, R preserves Σ-recognizability if and only if R preserves recognizability.*

Proof. ($\Leftarrow$) Trivial.

($\Rightarrow$) Let Δ be an arbitrary ranked alphabet with $sign(R) \subseteq \Delta$. To each symbol $g \in \Delta_k - sign(R)$, $k \geq 0$, we assign a tree $t_g \in T_\Sigma(X_k)$. To this end, we number the symbols in $\Delta - sign(R)$ from 1 to $|\Delta - sign(R)|$. Then we define the nth left comb $left_n$ and the nth right comb $right_n$ as follows.

(i) $left_0 = f(\sharp, \sharp)$ and $right_0 = \sharp$,

(ii) for each $n \geq 0$, $left_{n+1} = f(left_n, x_{n+1})$, $right_{n+1} = f(\sharp, right_n)$.

Finally, to a symbol $g \in \Delta_k - sign(R)$, $k \geq 0$, with number l, we assign the tree $t_g = f(left_k, right_l)$.

Consider the rewriting system

$$S = \{ g(x_1, \ldots, x_k) \to t_g \mid k \geq 0, g \in \Delta_k - sign(R), t_g \text{ is assigned to } g \} .$$

It should be clear that S is a convergent rewriting system. For each tree $p \in T_\Delta$, we denote by p', the S-normal form of p. For a tree language $L \subseteq T_\Delta$, let $L' = \{ p' \mid p \in L \}$. It is not hard to show the following two statements.

Claim 2.4.4 *For any $r, s \in T_\Delta$,*

$$r \underset{R}{\to} s \text{ if and only if } r' \underset{R}{\to} s' .$$

Claim 2.4.5 *A tree language L over Δ is recognizable if and only if L' is recognizable over Σ.*

Finishing proof of Theorem 2.4.3:

Let L be any recognizable tree language over Δ. By Claim 2.4.5, L' is a recognizable tree language over Σ. By Claim 2.4.4, $(R^*_\Delta(L))' = R^*_\Sigma(L')$. By Claim 2.4.5, $R^*_\Delta(L)$ is recognizable if and only if $R^*_\Sigma(L')$ is recognizable. Hence if R preserves recognizability over Σ, then R preserves recognizability over Δ. As Δ is an arbitrary ranked alphabet with $sign(R) \subseteq \Delta$, by Lemma 2.4.2, R preserves recognizability. □

The proof of the following result is similar to the proof of Theorem 2.4.3.

Theorem 2.4.6 *Let R be a rewriting system over $sign(R)$, and let $\Sigma = \{ f, \sharp \} \cup sign(R)$, where $f \in \Sigma_2 - sign(R)$ and $\sharp \in \Sigma_0 - sign(R)$. Then, R effectively preserves Σ-recognizability if and only if R effectively preserves recognizability.*

Consequence 2.4.7 *Let R be a rewriting system over Σ such that there is a symbol $f \in \Sigma_2 - sign(R)$ and there is a constant $\sharp \in \Sigma_0 - sign(R)$. Then R preserves recognizability if and only if R preserves Σ-recognizability. Moreover, R effectively preserves recognizability if and only if R effectively preserves Σ-recognizability.*

Theorem 2.4.8 *Let R, S be rewriting systems over a ranked alphabet Σ. Let R effectively preserve recognizability. Then it is decidable if $\to^*_S \subseteq \to^*_R$.*

Proof. Let $m \geq 0$ be such that for all variables x_i occurring on the left-hand side of some rule in S, $x_i \in X_m$, that is, $i \leq m$. Let us introduce new constant symbols $Z = \{ z_1, \ldots, z_m \}$ with $Z \cap \Sigma = \emptyset$. For each $t \in T_\Sigma(X)$, let $t_z \in T_{\Sigma \cup Z}(X)$ be defined by $t_z = t[z_1, \ldots, z_m]$. By direct inspection we obtain that for all $u, v \in T_\Sigma(X)$,

$$u \xrightarrow[R]{} v \text{ if and only if } u_z \xrightarrow[R]{} v_z \ ,$$

hence

$$u \xrightarrow[R]{*} v \text{ if and only if } u_z \xrightarrow[R]{*} v_z \ .$$

Claim 2.4.9 $\rightarrow_S^* \subseteq \rightarrow_R^*$ *if and only if for each rule* $l \rightarrow r$ *in* S, $r_z \in R^*_{\Sigma \cup Z}(\{\, l_z \,\})$.

Proof. ($\Rightarrow$) Let $l \rightarrow r$ be an arbitrary rule in S. Clearly, $l \rightarrow_R^* r$. Thus $r_z \in R^*_{\Sigma \cup Z}(\{\, l_z \,\})$.

($\Leftarrow$) Let us suppose that $t_1, t_2 \in T_\Sigma(X)$, and that $t_1 \rightarrow_S t_2$ applying the rule $l \rightarrow r$. As $r_z \in R^*_{\Sigma \cup Z}(\{\, l_z \,\})$, $l_z \rightarrow_R^* r_z$ holds. Hence $l \rightarrow_R^* r$ implying that $t_1 \rightarrow_R^* t_2$ as well. □

Finishing proof of Theorem 2.4.8:

Let $l \rightarrow r$ be an arbitrary rule in S. We can construct a tree automaton over $\Sigma \cup Z$ recognizing the singleton set $\{\, l_z \,\}$. As R effectively preserves recognizability, $R^*_{\Sigma \cup Z}(\{\, l_z \,\})$ is recognizable, and we can construct a tree automaton over $\Sigma \cup Z$ recognizing $R^*_{\Sigma \cup Z}(\{\, l_z \,\})$. Hence we can decide if $r_z \in R^*_{\Sigma \cup Z}(\{\, l_z \,\})$. Thus by Claim 2.4.9, we can decide if $\rightarrow_S^* \subseteq \rightarrow_R^*$. □

Consequence 2.4.10 *Let* R_1 *and* R_2 *be rewriting systems effectively preserving recognizability. Then it is decidable if* $\rightarrow_{R_1}^* = \rightarrow_{R_2}^*$.

Observation 2.4.11 *If one omits a rule from an lgsm rewriting system, then the resulting rewriting system still remains lgsm.*

One can easily show the following result applying Theorem 2.3.19, Consequence 2.4.8, and Observation 2.4.11.

Consequence 2.4.12 *For an lgsm rewriting system* R, *it is decidable whether* R *is left-to-right minimal.*

Consequence 2.4.8 also implies the following.

Consequence 2.4.13 *Let R_1 and R_2 be rewriting systems such that $R_1 \cup R_1^{-1}$ and $R_2 \cup R_2^{-1}$ are rewriting systems and effectively preserve recognizability. Then it is decidable if $\leftrightarrow^*_{R_1} \subseteq \leftrightarrow^*_{R_2}$.*

Theorem 2.3.19, Observation 2.4.11, and Consequence 2.4.13 imply the following.

Consequence 2.4.14 *Let R be a rewriting system such that $R \cup R^{-1}$ is an lgsm rewriting system. Then it is decidable whether R is two-way minimal.*

Theorem 2.4.15 *Let R_1, R_2 be rewriting systems over a ranked alphabet Σ. Let R_1 effectively preserve recognizability. Let $g \in \Sigma - \Sigma_0$ be such that g does not occur on the left-hand side of any rule in R_1, and let $\sharp \in \Sigma_0$ be irreducible with respect to R_1. Then it is decidable if $\rightarrow^*_{R_2} \cap (T_\Sigma \times T_\Sigma) \subseteq \rightarrow^*_{R_1} \cap (T_\Sigma \times T_\Sigma)$.*

Proof. We assume that $g \in \Sigma_1$. One can easily modify the proof of this case when proving the more general case $g \in \Sigma_k$, $k \geq 1$. For each $t \in T_\Sigma(X)$, let $t_g \in T_\Sigma$ be defined from t by substituting $g^i(\sharp)$ for all occurrences of the variable x_i for $i \geq 1$.

Claim 2.4.16 *$\rightarrow^*_{R_2} \cap (T_\Sigma \times T_\Sigma) \subseteq \rightarrow^*_{R_1} \cap (T_\Sigma \times T_\Sigma)$ if and only if for each rule $l \rightarrow r$ in R_2, $r_g \in R_1^*(\{ l_g \})$.*

Proof. ($\Rightarrow$) Let $l \rightarrow r$ be an arbitrary rule in R_2. Clearly, $l_g \rightarrow_{R_2} r_g$. Thus by our assumption $l_g \rightarrow^*_{R_1} r_g$.

($\Leftarrow$) Let us suppose that $t_1, t_2 \in T_\Sigma$, and that $t_1 \rightarrow_{R_2} t_2$ applying the rule $l \rightarrow r$. As $r_g \in R_1^*(\{ l_g \})$, $l_g \rightarrow^*_{R_1} r_g$ holds. Hence $l \rightarrow^*_{R_1} r$ implying that $t_1 \rightarrow^*_{R_1} t_2$ as well. □

Finishing proof of Theorem 2.4.15:

For each rule $l \rightarrow r$ in R_2, the tree language $\{ l_g \}$ is recognizable, and we can construct a tree automaton over Σ recognizing $\{ l_g \}$. As R_1 effectively preserves recognizability, $R_1^*(\{ l_g \})$ is also recognizable, and we can construct a tree automaton over Σ recognizing $R_1^*(\{ l_g \})$. Hence for each rule $l \rightarrow r$ in R_2, we can decide whether or not $r_g \in R_1^*(\{ l_g \})$. Thus by Claim 2.4.16, we can decide if $\rightarrow^*_{R_2} \cap (T_\Sigma \times T_\Sigma) \subseteq \rightarrow^*_{R_1} \cap (T_\Sigma \times T_\Sigma)$. □

Consequence 2.4.17 *Let R_1 and R_2 be rewriting systems over Σ effectively preserving recognizability. Moreover, let $g_1, g_2 \in \Sigma - \Sigma_0$ be such that for each $i \in \{1,2\}$, g_i does not occur on the left-hand side of any rule in R_i. Let $\sharp_1, \sharp_2 \in \Sigma_0$ be such that for each $i \in \{1,2\}$, $\sharp_i$ is irreducible with respect to R_i. Then it is decidable if $\rightarrow^*_{R_1} \cap (T_\Sigma \times T_\Sigma) = \rightarrow^*_{R_2} \cap (T_\Sigma \times T_\Sigma)$.*

One can easily show the following result applying Theorem 2.3.19, Observation 2.4.11, and Consequence 2.4.15.

Consequence 2.4.18 *Let R be an lgsm rewriting system over Σ. Moreover, let $g \in \Sigma - \Sigma_0$ such that g does not occur on the left-hand side of any rule in R, and let $\sharp \in \Sigma_0$ be irreducible with respect to R. Then it is decidable whether R is left-to-right ground minimal.*

Consequence 2.4.15 also implies the following.

Consequence 2.4.19 *Let R_1 and R_2 be rewriting systems over Σ such that $R_1 \cup R_1^{-1}$ and $R_2 \cup R_2^{-1}$ are rewriting systems and effectively preserve recognizability. Moreover, let $g_1, g_2 \in \Sigma - \Sigma_0$ be such that for each $i \in \{1,2\}$, g_i does not occur in R_i. Let $\sharp_1, \sharp_2 \in \Sigma_0$ be such that for each $i \in \{1,2\}$, $\sharp_i$ is irreducible with respect to $R_i \cup R_i^{-1}$. Then it is decidable if $\leftrightarrow^*_{R_1} \cap (T_\Sigma \times T_\Sigma) \subseteq \leftrightarrow^*_{R_2} \cap (T_\Sigma \times T_\Sigma)$.*

Theorem 2.3.19, Observation 2.4.11, and Consequence 2.4.19 imply the following.

Consequence 2.4.20 *Let R be a rewriting system over Σ such that $R \cup R^{-1}$ is an lgsm rewriting system. Moreover, let $g \in \Sigma - \Sigma_0$ be such that g does not occur in any rule of R, and let $\sharp \in \Sigma_0$ be irreducible with respect to $R \cup R^{-1}$. Then it is decidable whether R is two-way ground minimal.*

Lemma 2.4.21 *Let R be a rewriting system over Σ effectively preserving recognizability, and let $p, q \in T_\Sigma(X)$. Then it is decidable if there exists a tree $r \in T_\Sigma(X)$ such that $p \rightarrow^*_R r$ and $q \rightarrow^*_R r$.*

Proof. Let $m \geq 0$ be such that $var(p) \subseteq X_m$, $var(q) \subseteq X_m$. Let us introduce new constant symbols $Z = \{z_1, \ldots, z_m\}$ with $Z \cap \Sigma = \emptyset$. For each $t \in T_\Sigma(X_m)$, let $t_z \in T_{\Sigma \cup Z}$ be defined by $t_z = t[z_1, \ldots, z_m]$.

The singleton sets $\{ p_z \}$, $\{ q_z \}$ are recognizable, and we can construct two tree automata over $\Sigma \cup Z$ which recognize $\{ p_z \}$ and $\{ q_z \}$, respectively. As R preserves recognizability, $R^*_{\Sigma \cup Z}(\{ p_z \})$ and $R^*_{\Sigma \cup Z}(\{ q_z \})$ are recognizable, and we can construct two tree automata over $\Sigma \cup Z$ which recognize $R^*_{\Sigma \cup Z}(\{ p_z \})$ and $R^*_{\Sigma \cup Z}(\{ q_z \})$, respectively. Hence we can decide if $R^*_{\Sigma \cup Z}(\{ p_z \}) \cap R^*_{\Sigma \cup Z}(\{ q_z \}) = \emptyset$, see [40]. Clearly, $R^*_{\Sigma \cup Z}(\{ p_z \}) \cap R^*_{\Sigma \cup Z}(\{ q_z \}) \neq \emptyset$ if and only if there exists a tree $r \in T_\Sigma(X)$ such that $p \rightarrow^*_R r$ and $q \rightarrow^*_R r$. □

Theorem 2.4.22 *Let R be a rewriting system over Σ effectively preserving recognizability. Then it is decidable if R is locally confluent.*

Proof. By Proposition 1.3.2, R is locally confluent if and only if for every critical pair (v_1, v_2) of R there exists a tree $v \in T_\Sigma(X)$ such that $v_1 \rightarrow^*_R v$ and $v_2 \rightarrow^*_R v$. It is well known that all critical pairs of R are variants of finitely many critical pairs of R. Hence it is sufficient to inspect finitely many critical pairs. Thus the theorem follows from Lemma 2.4.21. □

Proposition 2.4.23 [25] *Let $\mathcal{A} = (\Sigma, A, \Delta, a_0, R)$ be a dt. Then R is a convergent rewriting system over the ranked alphabet $A \cup \Sigma \cup \Delta$. Moreover, $ran(\tau_{\mathcal{A}}) = NF(L, R)$, where $L = \{ a_0(s) \mid s \in dom(\tau_{\mathcal{A}}) \}$.*

Fülöp [25] has obtained the following undecidability results.

Proposition 2.4.24 *Each of the following problems is undecidable for arbitrary dt's $\mathcal{A}_1 = (\Sigma, A_1, \Delta, a_1, R_1)$ and $\mathcal{A}_2 = (\Sigma, A_2, \Delta, a_2, R_2)$, where we denote $L_1 = ran(\tau_{\mathcal{A}_1})$ and $L_2 = ran(\tau_{\mathcal{A}_2})$.*

(i) *Is $L_1 \cap L_2$ empty?*

(ii) *Is $L_1 \cap L_2$ infinite?*

(iii) *Is $L_1 \cap L_2$ recognizable?*

(iv) *Is $T_\Delta - L_1$ empty?*

(v) *Is $T_\Delta - L_1$ infinite?*

(vi) *Is $T_\Delta - L_1$ recognizable?*

(vii) *Is* L_1 *recognizable?*

(viii) *Is* $L_1 = L_2$*?*

(ix) *Is* $L_1 \subseteq L_2$*?*

Applying the results of Proposition 2.4.24, Fülöp [25] has also shown the following undecidability results.

Proposition 2.4.25 *Each of the following questions is undecidable for any convergent left-linear rewriting systems* R_1 *and* R_2 *over a ranked alphabet* Ω*, for any recognizable tree language* $L \subseteq T_\Omega$ *given by a tree automaton over* Ω*, where* Γ *is the smallest ranked alphabet for which* $NF(L, R_1) \subseteq T_\Gamma$*.*

(i) *Is* $NF(L, R_1) \cap NF(L, R_2)$ *empty?*

(ii) *Is* $NF(L, R_1) \cap NF(L, R_2)$ *infinite?*

(iii) *Is* $NF(L, R_1) \cap NF(L, R_2)$ *recognizable?*

(iv) *Is* $T_\Gamma - NF(L, R_1)$ *empty?*

(v) *Is* $T_\Gamma - NF(L, R_1)$ *infinite?*

(vi) *Is* $T_\Gamma - NF(L, R_1)$ *recognizable?*

(vii) *Is* $NF(L, R_1)$ *recognizable?*

(viii) *Is* $NF(L, R_1) = NF(L, R_2)$*?*

(ix) *Is* $NF(L, R_1) \subseteq NF(L, R_2)$*?*

We obtain the following result by direct inspection.

Lemma 2.4.26 *For each dt* $\mathcal{A} = (\Sigma, A, \Delta, a_0, R)$*, there exists a dt* $\mathcal{B} = (\Sigma', A, \Delta, a_0, R')$ *such that* $\Sigma' \cap \Delta = \emptyset$ *and that* $ran(\tau_\mathcal{A}) = ran(\tau_\mathcal{B})$*. Moreover, let* $\mathcal{A} = (\Sigma, A, \Delta, a_0, R)$ *be a dt with* $\Sigma \cap \Delta = \emptyset$*. Then* R *is a left-linear gsm rewriting system.*

Theorem 2.4.27 *Each of the following questions is undecidable for any convergent left-linear gsm rewriting systems R_1 and R_2 over a ranked alphabet Ω, for any recognizable tree language $L \subseteq T_\Omega$ given by a tree automaton over Ω recognizing L, where Γ is the smallest ranked alphabet for which $NF(L, R_1) \subseteq T_\Gamma$.*

(i) *Is $NF(L, R_1) \cap NF(L, R_2)$ empty?*

(ii) *Is $NF(L, R_1) \cap NF(L, R_2)$ infinite?*

(iii) *Is $NF(L, R_1) \cap NF(L, R_2)$ recognizable?*

(iv) *Is $T_\Gamma - NF(L, R_1)$ empty?*

(v) *Is $T_\Gamma - NF(L, R_1)$ infinite?*

(vi) *Is $T_\Gamma - NF(L, R_1)$ recognizable?*

(vii) *Is $NF(L, R_1)$ recognizable?*

(viii) *Is $NF(L, R_1) = NF(L, R_2)$?*

(ix) *Is $NF(L, R_1) \subseteq NF(L, R_2)$?*

Proof. Proposition 2.4.25 appeared as Theorem 5.2 in [25]. We can apply the proof of Theorem 5.2 in [25] with slight modifications. By Lemma 2.4.26, the proofs of (i)-(vii) and of (ix) carry over.

To adopt the proof of (viii), we observe the following. Let $\mathcal{A} = (\Sigma, A, \Delta, a_0, R)$ be a deterministic top-down tree transducer. Then by Lemma 2.4.26, we may assume that $\Sigma \cap \Delta = \emptyset$. Hence, by Lemma 2.4.26, R is a left-linear gsm rewriting system. Let $\star$ be a new symbol with rank 0, such that $\star \notin \Sigma \cup \Delta \cup A$. If we add a rule $a(x_1) \to \star$ (with $a \in A$) to R, then R remains a left-linear gsm rewriting system. $\square$

Lemma 2.4.28 *Let R and S be linear collapse-free rewriting systems over the disjoint ranked alphabets Σ and Δ, respectively. Let Γ be a ranked alphabet with $\Sigma \cup \Delta \subseteq \Gamma$. Consider R and S as rewriting systems over Γ. Then*

(i) $\to_S \circ \to_R \subseteq \to_R \cup (\to_R \circ \to_S)$, *and*

(ii) $\to^*_{R \cup S} = \to^*_R \circ \to^*_S$.

Proof. The proof of (i) is straightforward. Condition (ii) is a simple consequence of (i). □

Lemma 2.4.29 *Let R and S be linear collapse-free rewriting systems over the disjoint ranked alphabets Σ and Δ, respectively. Let R and S preserve recognizability. Then $R \cup S$ over $\Sigma \cup \Delta$ also preserves recognizability.*

Proof. Let L be a recognizable tree language over some ranked alphabet Γ, where $\Sigma \cup \Delta \subseteq \Gamma$. By Lemma 2.4.28, $(R \cup S)^*_\Gamma(L) = S^*_\Gamma(R^*_\Gamma(L))$. As R preserves recognizability, $R^*_\Gamma(L)$ is recognizable. Moreover, since S preserves recognizability, $S^*_\Gamma(R^*_\Gamma(L))$ is also recognizable. □

Lemma 2.4.30 *Let R and S be linear collapse-free rewriting systems over the disjoint ranked alphabets Σ and Δ, respectively. Let $R \cup S$ over $\Sigma \cup \Delta$ preserve recognizability. Then R and S also preserve recognizability.*

Proof. Let L be a recognizable tree language over some ranked alphabet Γ, where $\Sigma \subseteq \Gamma$. It is sufficient to show that $R^*_\Gamma(L)$ is recognizable. Without loss of generality we may rename the symbols of Γ such that $\Gamma \cap \Delta = \emptyset$. Thus $R^*_\Gamma(L) = (R \cup S)^*_{\Gamma\cup\Delta}(L)$. Since $\Sigma \cup \Delta \subseteq \Gamma \cup \Delta$ and $R \cup S$ preserves recognizability, $R^*_\Gamma(L)$ is recognizable. □

Since linear collapse-free rewriting systems are closed under disjoint union, we have obtained the following results.

Theorem 2.4.31 *For the class of linear collapse-free rewriting systems, the property of preserving recognizability is modular.*

The proof of the following result is similar to the proof of Theorem 2.4.31.

Theorem 2.4.32 *For the class of linear collapse-free rewriting systems, the property of effectively preserving recognizability is modular.*

Since λ-free string rewriting system correspond to linear collapse-free rewriting system, our results on linear collapse-free rewriting systems imply that preserving recognizability and effectively preserving recognizability are modular properties of λ-free string rewriting systems.

Theorem 2.4.33 *For the class of λ-free string rewriting systems, the property of (effectively) preserving recognizability is modular.*

Recently, F. Otto [51] has proved the following result which appeared as a conjecture in a previous version of the paper [45].

Theorem 2.4.34 [51] *A string rewriting system S over Σ preserves Σ-recognizability if and only if S preserves recognizability.*

Chapter 3

Decidability of the injectivity of deterministic top-down tree transducers

This chapter is divided into two sections. In Section 3.1, we summarize the results of this chapter. Section 3.2 contains the proof of the decidability of the injectivity problem of linear deterministic top-down tree transducers and the proof of the undecidability of the injectivity problem of homomorphism tree transducers.

3.1 Summary of results

The injectivity problem of deterministic top-down tree transducers sounds as follows.

> Does there exist an algorithm which decides, for every dt $\mathcal{A} = (\Sigma, A, \Delta, a_0, R)$, whether the partial function $\tau_\mathcal{A}$ is injective or not ?

In [20], Z. Ésik studied, among others, the decidability of this injectivity problem. He showed that the injectivity problem of linear deterministic top-down tree transducers is decidable i.e., that for ldt's such an algorithm exists. He gave a rather involved proof. As the main result of this chapter,

- we will give a simpler proof for the decidability of the injectivity problem of linear top-down tree transducers. (Theorem 3.2.4)

Our proof is based on the two well-known facts that linear top-down tree transducers preserve recognizability of tree languages and the emptiness problem is decidable for recognizable tree languages.

Z. Ésik also showed in [20] that the injectivity problem is undecidable for (general) dt's. He reduced this problem to the Post Correspondence Problem (PCP) by showing that an instance of the PCP can be encoded in a deterministic top-down tree transducer such that the deterministic top-down tree transducer is not injective if and only if the instance of PCP has a solution.

We sharpen this negative result in this chapter by showing that the injectivity problem is undecidable already for homomorphism tree transducers.

- Namely, we show that there is no algorithm for deciding whether an arbitrary homomorphism tree transducer is injective or not. (Theorem 3.2.7)

We prove in the way that we reduce the problem to an undecidability result of Dauchet [10] concerning tree codes.

3.2 On the injectivity problem of deterministic top-down tree transducers

3.2.1 The injectivity problem of linear deterministic top-down tree transducers

In this subsection we show that the injectivity problem of linear deterministic top-down tree transducers is decidable. First we recall a result.

Proposition 3.2.1 (Corollary IV.6.6 of [40]) *Let $\mathcal{A} = (\Sigma, A, \Delta, a_0, R)$ be a ldt. Then, for each recognizable subset L of T_Σ, $\tau_{\mathcal{A}}(L)$ is also recognizable. Moreover, given $\mathcal{A}$ and L, $\tau_{\mathcal{A}}(L)$ can effectively be constructed.*

We need the following concept. Let $\mathcal{A} = (\Sigma, A, \Delta, a_0, R)$ be a dt. A state $a \in A$ is *useful* if there exist $t \in \bar{T}_\Sigma(X_1)$, $s \in T_\Sigma$, $t' \in \bar{T}_\Delta(X_1)$ and $s' \in T_\Delta$ such that

$$a_0(t[s]) \overset{*}{\underset{T}{\Rightarrow}} t'[a(s)] \overset{*}{\underset{T}{\Rightarrow}} t'[s'].$$

The main result is based on the following lemma.

Lemma 3.2.2 *Let $\mathcal{A} = (\Sigma, A, \Delta, a_0, R)$ be a lndt. Then, $\mathcal{A}$ is not injective if and only if there exist a useful state $a \in A$ and trees $t,\ t' \in T_\Sigma$ with $root(t) \neq root(t')$ such that $\tau_{\mathcal{A}(a)}(t) = \tau_{\mathcal{A}(a)}(t')$.*

Proof. First suppose that $\mathcal{A}$ is not injective. Then, there are $t,\ t' \in T_\Sigma$ with $t \neq t'$ and $s \in T_\Delta$ such that $\tau_{\mathcal{A}}(t) = \tau_{\mathcal{A}}(t') = s$.

Let us observe that $t \neq t'$ if and only if there exist an integer $k \geq 1$, $u \in \bar{T}_\Sigma(X_k)$ and trees $t_1, \ldots, t_k, t'_1, \ldots, t'_k \in T_\Sigma$ with $root(t_i) \neq root(t'_i)$, for $1 \leq i \leq k$, such that $t = u[t_1, \ldots, t_k]$ and $t' = u[t'_1, \ldots, t'_k]$. Note that the condition $k \geq 1$ is important and that in the special case when $root(t) \neq root(t')$ we have $k = 1,\ u = x_1,\ t_1 = t$ and $t'_1 = t'$.

Since $\mathcal{A}$ is linear and nondeleting, the derivations $a_0(t) \Rightarrow^*_{\mathcal{A}} s$ and $a_0(t') \Rightarrow^*_{\mathcal{A}} s$ can be written in the form

$$a_0(t) = a_0(u[t_1, \ldots, t_k]) \underset{\mathcal{A}}{\overset{*}{\Rightarrow}} r[a_1(t_1), \ldots, a_k(t_k)] \underset{\mathcal{A}}{\overset{*}{\Rightarrow}} r[r_1, \ldots, r_k] = s$$

and

$$a_0(t') = a_0(u[t'_1, \ldots, t'_k]) \underset{\mathcal{A}}{\overset{*}{\Rightarrow}} r[a_1(t'_1), \ldots, a_k(t'_k)] \underset{\mathcal{A}}{\overset{*}{\Rightarrow}} r[r'_1, \ldots, r'_k] = s,$$

respectively, where $r \in \hat{T}_\Delta(X_k)$, $a_1, \ldots, a_k \in A$ and $r_1, \ldots, r_k, r'_1, \ldots, r'_k \in T_\Delta$. Moreover, each variable of X_k appears exactly once in r.

Since $r[r_1, \ldots, r_k] = s = r[r'_1, \ldots, r'_k]$, we get that, for every $1 \leq i \leq k$, $r_i = r'_i$ holds. Therefore, for every $1 \leq i \leq k$, $\tau_{\mathcal{A}(a_i)}(t_i) = \tau_{\mathcal{A}(a_i)}(t'_i)$, what we wanted to prove. (Note that $a_1, \ldots, a_k$ are useful.)

Conversely, assume that there exists a useful state $a \in A$ and trees $t,\ t' \in T_\Sigma$ with $root(t) \neq root(t')$ such that $\tau_{\mathcal{A}(a)}(t) = \tau_{\mathcal{A}(a)}(t') = v$. The state a is useful so there is a context $u \in \bar{T}_\Sigma(X_1)$ which "reaches" a, that is to say, $a_0(u) \Rightarrow^*_{\mathcal{A}} u'[a(x_1)]$, for some $u' \in \bar{T}_\Delta(X_1)$. (Note that the condition $u' \in \bar{T}_\Delta(X_1)$ follows from the fact that $\mathcal{A}$ is both linear and nondeleting.)

Let $r = u[t]$ and $r' = u[t']$. Then, we have $r \neq r'$ because $root(t) \neq root(t')$. Moreover we have

$$a_0(r) = a_0(u[t]) \underset{\mathcal{A}}{\overset{*}{\Rightarrow}} u'[a(t)] \underset{\mathcal{A}}{\overset{*}{\Rightarrow}} u'[v] \quad \text{and} \quad a_0(r') = a_0(u[t']) \underset{\mathcal{A}}{\overset{*}{\Rightarrow}} u'[a(t')] \underset{\mathcal{A}}{\overset{*}{\Rightarrow}} u'[v].$$

Consequently $\mathcal{A}$ is not injective. □

For a ranked alphabet Σ and $f \in \Sigma_m$ with $m \geq 0$, we denote by T_f the set of trees having root f i.e., we put $T_f = \{f(t_1, \ldots, t_m) \mid t_i \in T_\Sigma$, for $1 \leq i \leq m\}$.

It is an obvious fact that T_f is recognizable.

The above lemma can be stated equivalently as follows.

Lemma 3.2.3 *Let $\mathcal{A} = (\Sigma, A, \Delta, a_0, R)$ be a lndt. Then $\mathcal{A}$ is not injective if and only if there exist a useful state $a \in A$ and $f,\ g \in \Sigma$ with $f \neq g$ and*

$$\tau_{\mathcal{A}(a)}(T_f) \cap \tau_{\mathcal{A}(a)}(T_g) \neq \emptyset.$$

Next we note that, by standard construction, for every dt $\mathcal{A}$, there exists a dt $\mathcal{A}'$ such that $\tau_\mathcal{A} = \tau_{\mathcal{A}'}$ and all states of $\mathcal{A}'$ are useful. We shall use this fact in the proof of our main theorem which is as follows.

Theorem 3.2.4 *The injectivity for an arbitrary ldt $\mathcal{A}$ is decidable.*

Proof. Let $\mathcal{A} = (\Sigma, A, \Delta, a_0, R)$ be an arbitrary ldt. If $\mathcal{A}$ has no a useful state, then certainly $\tau_\mathcal{A} = \emptyset$ and hence $\mathcal{A}$ is injective.

Otherwise, let $\mathcal{A}' = (\Sigma, A', \Delta, a_0, R')$ be a ldt such that $\tau_\mathcal{A} = \tau_{\mathcal{A}'}$ and all states of $\mathcal{A}'$ are useful. Now $\mathcal{A}$ is injective if and only if $\mathcal{A}'$ is injective. We distinguish two cases.

Case 1 : $\mathcal{A}'$ is deleting (i.e., is not nondeleting). Then, it is an exercise to show that $\mathcal{A}'$ is certainly not injective.

Case 2 : $\mathcal{A}'$ is nondeleting. Then, by Lemma 3.2.3, it is not injective if and only if

$$\tau_{\mathcal{A}'(a)}(T_f) \cap \tau_{\mathcal{A}'(a)}(T_g) \neq \emptyset,$$

for some state $a \in A'$ and $f,\ g \in \Sigma$ with $f \neq g$. Since T_f and T_g are obviously recognizable, by Proposition 3.2.1, both $\tau_{\mathcal{A}'(a)}(T_f)$ and $\tau_{\mathcal{A}'(a)}(T_g)$ are recognizable and can effectively be constructed. Finally, we note that (1) recognizable tree languages are effectively closed under intersection and that (2) the emptiness problem for recognizable tree languages is known to be decidable. (For proofs of (1) and (2), see Theorems II.4.2 and II.10.2 in [40], for example.)

Hence we can decide whether $\mathcal{A}'$ is injective in Case 2, too. □

3.2.2 The injectivity problem of homomorphism tree transducers

In this subsection we prove that the injectivity problem is undecidable even for homomorphism tree transducers. We show that the so called code problem, which was introduced and shown to be undecidable in [10], is reducible to the injectivity problem.

A *generator set* is a nonempty, finite subset E of $T_\Sigma(X)$ with the properties that

(i) E has at least one ground element and that

(ii) for every $e \in E$, if x_i appears in e, then x_j also appears in e, for every $1 \leq j < i$.

For example, $f(x_2, f(x_2, \#))$ and $f(f(\#, x_3), x_1)$ cannot be elements of any generator set, however $f(x_1, f(x_2, x_1))$ can, where f and $\#$ are symbols of rank 2 and 0, respectively.

Given a generator set E, for $k \geq 0$ we define $E_k = \{e \in E \mid k \text{ is the}$ smallest integer such that $e \in T_\Sigma(X_k)\}$. Then $E_0 \neq \emptyset$ and $E_i \cap E_j = \emptyset$, whenever $i \neq j$.

Any generator set E generates a tree language $T_E \subseteq T_\Sigma$, which is the smallest set H satisfying the following conditions:

(i) $E_0 \subseteq H$,

(ii) if $e \in E_k$ and $t_1, \ldots, t_k \in H$, for some $k \geq 1$, then $e[t_1, \ldots, t_k] \in H$.

For each $e \in E_k$, with $k \geq 0$ we put

$$eT_E = \{e[t_1, \ldots, t_k] \mid t_1, \ldots, t_k \in T_E\}.$$

As an example, let $\Sigma = \{f^{(2)}, g^{(1)}, \#^{(0)}\}$ be a ranked alphabet. Then $E = \{f(x_1, g(x_2)), f(x_1, x_1), g(x_1), \#\}$ is a generator set.

The following definition of the code problem is adopted from [10] .

Definition 3.2.5 *A generator set $E \subseteq T_\Sigma(X_n)$ is called a code if, for every e_1, $e_2 \in E$ with $e_1 \neq e_2$, the condition $e_1T_E \cap e_2T_E = \emptyset$ holds.*

Informally speaking, E is a code if and only if each element of T_E can be constructed unambiguously with tree substitution from elements of E. Thus the example appearing at the end of Subsection 3.2.1 is not a code, since for $e_1 = f(x_1, g(x_2))$ and $e_2 = f(x_1, x_1)$, we have $f(g(\#), g(\#)) \in e_1T_E \cap e_2T_E$.

In [10] it was shown that it is undecidable whether an arbitrary generator set $E \subseteq T_\Sigma(X_n)$ is a code in the following way. For an arbitrary instance of the PCP, a generator set E can be constructed such that E is not a code if and only if that instance has a solution. On the other hand, as was noted in [10], it is decidable, if a linear generator set E is a code. (A generator set E is linear if each $e \in E$ is linear i.e., every variable occurs at most once in e. Hence, if this is the case, then eT_E is recognizable, for every $e \in E$.)

We explicitly recall the main result of [10].

Proposition 3.2.6 ([10]) *There is no algorithm for deciding if an arbitrary generator set $E \subseteq T_\Sigma(X_n)$ is a code.*

Next we introduce the concept of a tree homomorphism . Let Σ and Δ be ranked alphabets. Moreover, let $\overline{th} : \Sigma \to T_\Delta(X)$ be a mapping with the property that if $f \in \Sigma_k$ for some $k \geq 0$, then we have $\overline{th}(f) \in T_\Delta(X_k)$. The tree homomorphism induced by $\overline{th}$ is the mapping $th : T_\Sigma \to T_\Delta$ defined by induction as follows :

(i) If $f \in \Sigma_0$, then $th(f) = \overline{th}(f)$.

(ii) If $f \in \Sigma_k$ for some $k \geq 1$ and $t_1, \ldots, t_k \in T_\Sigma$, then

$$th(f(t_1, \ldots, t_k)) = \overline{th}(f)[th(t_1), \ldots, th(t_k)].$$

(Note that [ ] is tree substitution.)

It is an easy exercise to show that for any tree homomorphism th, a homomorphism tree transducer $\mathcal{A}$ can be constructed effectively such that $th = \tau_\mathcal{A}$. Conversely, for any homomorphism tree transducer $\mathcal{A}$, the mapping τ_T is a tree homomorphism, which can be given effectively from $\mathcal{A}$.

Now we prove our other result.

Theorem 3.2.7 *There is no algorithm for deciding whether an arbitrary homomorphism tree transducer is injective or not.*

Proof. We perform the proof in two main steps.

In the first step we show that for an arbitrary generator set E a tree homomorphism th can be constructed such that E is a code if and only if th is an injective mapping.

To this end, let $E \subseteq T_\Sigma(X_n)$ with $E = E_0 \cup \ldots \cup E_n$ be an arbitrary generator set. For $0 \le k \le n$, we define $\Sigma'_k = \{f_e \mid e \in E_k\}$ and we put $\Sigma' = \cup_{i=0}^{n} \Sigma'_i$.

Let $\overline{th} : \Sigma' \to E$ be the bijection defined by $\overline{th}(f_e) = e$, for each $f_e \in \Sigma'$. Let $th : T_{\Sigma'} \to T_\Sigma$ be the tree homomorphism induced by $\overline{th}$. Then the following two statements hold.

Statement 3.2.8 *Let e be an arbitrary tree in E_k for some $k \ge 0$. Then $th(T_{f_e}) = eT_E$. (For the definition of T_{f_e}, see Subsection 3.2.2.)*

Proof. It is an easy exercise to show that $th(T_{\Sigma'}) = T_E$. Then the proof of our Statement follows immediately. □

Statement 3.2.9 *The generator set $E \subseteq T_\Sigma(X_n)$ is a code if and only if th is injective.*

Proof. First suppose that E is not a code. Then, there exist $e_1 \in E_k$ and $e_2 \in E_l$ with $e_1 \ne e_2$ and $e_1T_E \cap e_2T_E \ne \emptyset$.

Let t be a tree in $e_1T_E \cap e_2T_E$. By Statement 3.2.8 there exist $t'_1, \ldots, t'_k \in T_{\Sigma'}$ and $u'_1, \ldots, u'_l \in T_{\Sigma'}$ such that $th(f_{e_1}(t'_1, \ldots, t'_k)) = th(f_{e_2}(u'_1, \ldots, u'_l)) = t$.

Consequently, th is not injective, since $f_{e_1} \ne f_{e_2}$.

Conversely, assume that th is not injective. Then, there exist t', $u' \in T_{\Sigma'}$ such that $t' \ne u'$ and $th(t') = th(u')$.

Choose a pair t', u' with the above property, such that the sum $height(t') + height(u')$ is minimal. Then $root(t') \ne root(u')$ holds.

For, if $root(t') = root(u')$, that is $t' = f_e(t'_1, \ldots, t'_k)$ and $u' = f_e(u'_1, \ldots, u'_k)$, then, since $t' \ne u'$, there exists an index $1 \le i \le k$ such that $t'_i \ne u'_i$. Moreover, $th(t'_i) = th(u'_i)$, because $th(t') = e[th(t'_1), \ldots, th(t'_k)] = e[th(u'_1), \ldots, th(u'_k)] = th(u')$ and because, by condition (2) in the definition of a generator set, any of the variables $x_1, \ldots, x_k$ appears in e. Thus $height(t'_i) + height(u'_i) < height(t') + height(u')$, $t'_i \ne u'_i$ and $th(t'_i) = th(u'_i)$, which contradicts the minimality of $height(t') + height(u')$. Therefore $root(t') \ne root(u')$, that is $t' = f_{e_1}(t'_1, \ldots, t'_k)$ and $u' = f_{e_2}(u'_1, \ldots, u'_l)$, where $e_1 \in E_k$, $e_2 \in E_l$ and $e_1 \ne e_2$, moreover, $t'_1, \ldots, t'_k, u'_1, \ldots, u'_l \in T_{\Sigma'}$.

We define $t = th(t') = th(u')$. Then, by Statement 3.2.8, $t \in e_1 T_E$ and $t \in e_2 T_E$, proving that E is not a code. □

Finishing proof of Theorem 3.2.7:

In the second step, we finish the proof of our theorem. Let E be an arbitrary generator set. Construct the tree homomorphism h which is injective if and only if E is a code. Then, construct the homomorphism tree transducer $\mathcal{A}$ with the property $th = \tau_{\mathcal{A}}$ (see the note before the present theorem). Then, $\mathcal{A}$ is injective if and only if E is a code which is undecidable in general. This finishes the proof of Theorem 3.2.7. □

In [20] it was shown that, for every deterministic bottom-up tree transducer $\mathcal{B}$, a deterministic top-down tree transducer $\mathcal{A}$ can be constructed effectively such that $\mathcal{B}$ is injective if and only if $\mathcal{A}$ is injective. Hence the result of the previous subsection can be applied to decide if an arbitrarily given deterministic bottom-up tree transducer is injective.

Finally, note that a deterministic homomorphism tree transducer is a special deterministic top-down tree transducer. Hence by Proposition 2.4.23 and Lemma 2.4.26, the following holds.

Theorem 3.2.10 *Let R be a convergent left-linear gsm rewriting system over Σ. Let $L \subseteq T_\Sigma$ be a recognizable tree language. Then it is undecidable if the tree function $\rightarrow_R^* \cap (L \times NF(L, R))$ is injective.*

Chapter 4

Decidability of the inclusions in monoids generated by deterministic tree transformation classes

This chapter is divided into two sections. In Section 4.1, we summarize the results of this chapter. Section 4.2 contains the proof of $DT^R = DT \circ LDB$, $LDT^R \not\subseteq LDT \circ DB$, and $DT \not\subseteq LDTR \circ H$. Using these results and the composition and inclusion results of Engelfriet, Fülöp, and Fülöp and Vágvölgyi, we give a linear time algorithm to determine the correct inclusion relationship between two tree transformation classes which are compositions of some "fundamental" tree transformation classes taken from the set $\{DT^R, LDT^R, DT, LDT, DB, LDB, H, LH\}$.

4.1 Summary of results

In his pioneer papers [14], [15], Engelfriet studied the compositions of deterministic bottom-up tree transducers, deterministic top-down tree transducers, and deterministic top-down tree transducers with regular look-ahead.

Chapter 4

Decidability of the inclusions in monoids generated by deterministic tree transformation classes

This chapter is divided into two sections. In Section 4.1, we summarize the results of this chapter. Section 4.2 contains the proof of $DT^R = DT \circ LDB$, $LDT^R \not\subseteq LDT \circ DB$, and $DT \not\subseteq LDTR \circ H$. Using these results and the composition and inclusion results of Engelfriet, Fülöp, and Fülöp and Vágvölgyi, we give a linear time algorithm to determine the correct inclusion relationship between two tree transformation classes which are compositions of some "fundamental" tree transformation classes taken from the set $\{DT^R, LDT^R, DT, LDT, DB, LDB, H, LH\}$.

4.1 Summary of results

In his pioneer papers [14], [15], Engelfriet studied the compositions of deterministic bottom-up tree transducers, deterministic top-down tree transducers, and deterministic top-down tree transducers with regular look-ahead.

Among several results, he has shown that

$$\begin{array}{llcllrcl}
(a) & DT^R \circ DT^R & = & DT^R & (b) & LDT^R \circ LDT^R & = & LDT^R \\
(c) & DB & \subseteq & DT^R & (d) & LDB \circ DT & = & DT^R \\
(e) & LDT^R & = & LDB \circ LDT & & & & .
\end{array}$$

Moreover, Fülöp and Vágvölgyi [28], [58], [29], [35], [30], [34] systematically studied the compositions of several types of deterministic top-down tree transformation classes and among several inclusion and decomposition results they have shown that

$$\text{(f) } DT \circ LH = DT^2 = DT^3 \qquad\qquad \text{(g) } H \circ H = H\ .$$

Finally, Fülöp [24] studied the compositions of deterministic bottom-up tree transformation classes. He has shown that

$$\text{(h) } DB \circ DB = DB \qquad\qquad \text{(i) } LDB \circ LDB = LDB\ .$$

The set of tree transformation classes appearing in (a) - (i) is

$$M = \{DT^R, LDT^R, DT, LDT, DB, LDB, H, LH\}.$$

We note that there are top-down tree transformation classes, top-down tree transformation classes with regular look-ahead, and also bottom-up tree transformation classes in M.

Our aim is to construct an algorithm which, given arbitrary $Y_1, \ldots, Y_m, Z_1, \ldots, Z_n \in M$, decides which one of the following four conditions hold:

$$\begin{array}{lrclrlcl}
(i) & Y_1 \circ \ldots \circ Y_m & = & Z_1 \circ \ldots \circ Z_n & (ii) & Y_1 \circ \ldots \circ Y_m & \subset & Z_1 \circ \ldots \circ Z_n \\
(iii) & Z_1 \circ \ldots \circ Z_n & \subset & Y_1 \circ \ldots \circ Y_m & (iv) & Y_1 \circ \ldots \circ Y_m & \bowtie & Z_1 \circ \ldots \circ Z_n.
\end{array}$$

In other words, we consider the monoid $[M] = \{Y_1 \circ \ldots \circ Y_m \mid m \geq 0,\ Y_1, \ldots, Y_m \in M\}$ generated by M with composition and give an algorithm which decides, given arbitrary two elements $Y_1 \circ \ldots \circ Y_m$ and $Z_1 \circ \ldots \circ Z_n$ of $[M]$, whether $Y_1 \circ \ldots \circ Y_m \subseteq Z_1 \circ \ldots \circ Z_n$ holds or not.

In their works [34] and [24], Fülöp and Vágvölgyi proposed a method with which such an algorithm can be constructed in certain cases. The method is based on presenting the monoid $[M]$ by defining relations. We note that the presentation of a monoid in this manner is a well-known technics in algebra. In fact, the method of Fülöp and Vágvölgyi is only a general frame and does not give much help in solving the problems that arise in its application actual monoids. Still, they applied the method succesfully for two monoids generated by deterministic top-down and by deterministic bottom-up tree transformation classes, respectively. Moreover, this method was also applied by Slutzki and Vágvölgyi [55] for a monoid generated by deterministic top-down tree transformation classes with regular look-ahead and also by Dányi and Fülöp [9] for a monoid generated by a set containing the class of deterministic superlinear top-down tree transformations. The last application was published in [37] for a monoid generated by total top-down tree transformation classes and for a monoid generated by the class of attributed tree transformations and the class of macro tree transformations.

We also give the algorithm for M by applying this method. Before showing how it is applied to our case, we introduce some further notation. We consider two monoids defined in terms of M: the free monoid M^* (with the operation of concatenation) and $[M]$, defined above, the monoid finitely generated by M (with the operation of composition). Strings over M i.e., elements of M^*, represent tree transformation classes in $[M]$ by means of the homomorphism $|| \; || : M^* \to [M]$ defined by

$$||\lambda|| = I, \text{ and}$$

$$||Y_1 \cdot Y_2 \cdot \ldots \cdot Y_m|| \;=\; Y_1 \circ Y_2 \circ \ldots \circ Y_m \text{ for } m \geq 1 \text{ and } Y_1, \ldots, Y_m \in M.$$

Let ρ be the kernel of $|| \; ||$ i.e., the congruence relation over M^* induced by the homomorphism $|| \; ||$:

$$\rho = ker(|| \; ||) = \{(v, w) \in M^* \times M^* \mid \; ||v|| \;=\; ||w|| \,\}.$$

We construct the algorithm and prove its correctness in the following steps.

- In Subsection 4.3.1, we give a Thue system T over M such that $\leftrightarrow_T^* \subseteq \rho$. (A Thue system is the "two-way" version of a string rewriting system, a detailed explanation follows in Subsection 4.2.1)

The Thue system T consists of 22 formal equations of which the realizations are valid decomposition equations in M. They are, except one, known from the papers [14], [15] of Engelfriet, [28], [58], [35] of Fülöp and Vágvölgyi, and [24] of Fülöp.

- Our contribution to T is $DT^R = DT \circ LDB$, which is proved to be valid in Theorem 4.2.2.

- In Subsection 4.3.2, we give a subset of K of M^* and a string rewriting system S over M such that $\leftrightarrow_S^* = \leftrightarrow_T^*$, Theorem 4.2.6. Moreover, we prove that there is a linear time algorithm which, for every $u \in M^*$, computes a word $v \in K$ in linear time such that $u \rightarrow_S^* v$. (Theorem 4.2.8)

The set K will be proved to be a set of representatives for the equivalence classes of ρ.

- Therefore, in Subsection 4.3.3, we present the inclusion diagram of $||K||$ of the set of tree transformation classes represented by the elements of K. In order to show that the diagram is correct, among others, we prove the non-inclusions $LDT^R \not\subseteq LDT \circ DB$ and $DT \not\subseteq LDT^R \circ H$ in Lemma 4.2.9 and in Lemma 4.2.10.

In Subsection 4.3.4, we complete our results and obtain the desired algorithm.

- We show that $\rho = \leftrightarrow_S^*$ in Theorem 4.2.17.

- We show that $[M] = ||K||$ i.e., that K is a set of representatives of the equivalence classes of ρ. (Theorem 4.2.18)

- Finally, using the algorithm obtained in Theorem 4.2.8, we present the algorithm which decides, for arbitrary elements $Y_1 \circ \ldots \circ Y_m$ and $Z_1 \circ \ldots \circ Z_n$ of $[M]$ whether $Y_1 \circ \ldots \circ Y_m \subseteq Z_1 \circ \ldots \circ Z_n$ holds. (Theorem 4.2.19)

4.2 Decidability of the inclusions in the monoid generated by $\{ DT^R, LDT^R, DT, LDT, DB, LDB, H, LH \}$

4.2.1 The Thue system T

Let Σ be an alphabet. A *Thue system* T over Σ is a finite subset of $\Sigma^* \times \Sigma^*$ and each element (u, v) of T is called a *rewriting rule.* The *Thue congruence generated by* T is the reflexive, transitive closure $\leftrightarrow_T^*$ of the relation $\leftrightarrow_T$ defined as follows: for any $w, z \in \Sigma^*$, $w \leftrightarrow_T z$ if and only if there exist $x, y \in \Sigma^*$ and $(u, v) \in T$ such that either $w = xuy$ and $z = xvy$, or, $w = xvy$ and $z = xuy$. It is well-known that $\leftrightarrow_T^*$ is the least congruence over Σ^* containing T.

It should be clear that a Thue system T over Σ is a "two-way" version of a string rewriting system in the sense that the rewriting rules of T can be used in both directions (cf. Subsection 1.3.3). As mentioned, for a string rewriting system S, the reflexive, symmetric and transitive closure $\leftrightarrow_S^*$ of $\rightarrow_S$ is a congruence over Σ^*. It is called the *Thue congruence generated by* S.

Consider the set of tree transformations

$$M = \{ DT^R, LDT^R, DT, LDT, DB, LDB, H, LH \}$$

and the two monoids defined in terms of M: the monoid M^* (with the operation of concatenation) and $[M] = \{ Y_1 \circ \ldots \circ Y_m \mid m \geq 0, Y_i \in M \text{ for } 1 \leq i \leq m \}$, the monoid finitely generated by M (with the operation of composition). We will freely confuse names (e.g. the symbol DT) with meanings (the class DT). Strings over M represent transformation classes in $[M]$ by means of a homomorphism $|| \; || : M^* \rightarrow [M]$ defined by

$$||Y_1 \cdot Y_2 \cdot \ldots \cdot Y_m|| = Y_1 \circ Y_2 \circ \ldots \circ Y_m \quad .$$

We denote by $I \in [M]$ the tree transformation class consisting of all identity tree transformations i.e., $I = ||\lambda||$. Let ρ be the kernel of $|| \; ||$ i.e., the congruence relation induced by the homomorphism $|| \; ||$:

$$\rho = ker(|| \; ||) = \{(v, w) \in M^* \times M^* \mid ||v|| = ||w||\} \quad .$$

In the rest of this subsection we give a Thue system T over M and show that $\leftrightarrow_T^* \subseteq \rho$.

Let the Thue system $T \subseteq M^* \times M^*$ consist of the following 22 rewriting rules.

(1)	$(LDT^R \cdot LDT, LDT^R)$	(2)	$(LDT^R \cdot LDB, LDT^R)$
(3)	$(DT \cdot LDB, DT^R)$	(4)	$(DT \cdot H, DT^2)$
(5)	$(DT \cdot LH, DT^2)$	(6)	$(LDT \cdot DT^R, DT^R)$
(7)	$(LDT \cdot LDT^R, LDT^R)$	(8)	$(LDT \cdot DT, DT^2)$
(9)	$(LDT \cdot LH, LDT^2)$	(10)	$(LDB \cdot DT, DT^R)$
(11)	$(LDB \cdot LDT, LDT^R)$	(12)	$(LDB \cdot LDB, LDB)$
(13)	$(LDB \cdot H, DB)$	(14)	$(LDB \cdot LH, LDB)$
(15)	$(H \cdot LDT, DT)$	(16)	$(H \cdot LDB, DB)$
(17)	$(H \cdot H, H)$	(18)	$(H \cdot LH, H)$
(19)	$(LH \cdot LDT, LDT)$	(20)	$(LH \cdot LDB, LDB)$
(21)	$(LH \cdot H, H)$	(22)	$(LH \cdot LH, LH)$

Next we will argue that for every $(\alpha, \beta) \in T$, $||\alpha|| = ||\beta||$, or equivalently, $(\alpha, \beta) \in \rho$. For each i $(1 \leq i \leq 22)$, if the i-th rewriting rule of T is (α, β), then the corresponding claim $||\alpha|| = ||\beta||$ will be denoted by (i'). We thus have to prove that (i') holds for $1 \leq i \leq 22$. First we will show that $(3')$ hold. In the proof we will need the following result.

Lemma 4.2.1 *For each* dt^R $\mathcal{A} = (\Sigma, A, \Delta, a_0, R_\mathcal{A})$, *there exists a dbr* $\mathcal{D} = (\Sigma, D, \Sigma, D', R_\mathcal{D})$ *such that for each look-ahead set* L *appearing in a rule of* $\mathcal{A}$, $L = L(\mathcal{D}(C))$ *for some* $C \subseteq D$.

Proof. Let $L_1, \ldots, L_k$ be all the look-ahead sets appearing in the rules of $\mathcal{A}$, where $k \geq 0$. For $1 \leq i \leq k$, let $L_i = L(\mathcal{D}_i)$ for some dbr $\mathcal{D}_i = (\Sigma, D_i, \Sigma, D_i', R_{\mathcal{D}_i})$. Let $D = D_1 \times \ldots \times D_k$, $D' = \emptyset$, and let $\pi_i : D_1 \times \ldots \times D_k \to D_i$ be the ith projection for $1 \leq i \leq k$. The rule $f(d_1(x_1), \ldots, d_m(x_m)) \to d(f(x_1, \ldots, x_m))$ is in $R_\mathcal{D}$ if and only if $f(\pi_i(d_1)(x_1), \ldots, \pi_i(d_m)(x_m)) \to \pi_i(d)(f(x_1, \ldots x_m)) \in R_{\mathcal{D}_i}$ for $1 \leq i \leq k$. It should be clear that for each $1 \leq i \leq k$, $L_i = L(\mathcal{D}(C_i))$ with $C_i = D_1 \times \ldots D_{i-1} \times D_i' \times D_{i+1} \times \ldots \times D_k$. □

Theorem 4.2.2 $DT^R = DT \circ LDB$.

Proof. It should be clear that $DT \subseteq DT^R$ and $LDB \subseteq DB$. By Theorem 3.2 of [15] $DB \subseteq DT^R$. By (i) of Proposition 4.2.3 DT^R is closed under composition. Hence $DT \circ LDB \subseteq DT^R$.

We now show that $DT^R \subseteq DT \circ LDB$. Intuitively, we simulate the computation of a dt^{R} $\mathcal{A}$ by the composition of a dt $\mathcal{B}$ and an ldb $\mathcal{C}$ as follows. All states of $\mathcal{A}$ are in the state set of $\mathcal{B}$. In state a of $\mathcal{A}$, and at a node ν labeled by f, $\mathcal{B}$ applies a rule of which right-hand side contains as subtrees the right-hand sides of all $\mathcal{A}$-rules with left-hand side $a(f(x_1, \ldots, x_m))$. Moreover, the right-hand side of this $\mathcal{B}$-rule also contains the variables $x_1, \ldots, x_m$ corresponding to the subtrees at the sons of ν. Thus, during its computation, $\mathcal{B}$ copies slightly modified versions of the subtrees at the sons of ν. Hence $\mathcal{C}$ is able to compute the look-ahead of $\mathcal{A}$ on these subtrees and is able to choose the right-hand side of the rule applied by $\mathcal{A}$.

More precisely, let $\mathcal{A} = (\Sigma, A, \Delta, a_0, R_{\mathcal{A}})$ be a dt^{R}. We lose no generality by assuming that $\Delta_0 \neq \emptyset$. Let $\mu \in \Delta_0$ be arbitrary. By Lemma 4.2.1, there exists a dbr $\mathcal{D} = (\Sigma, D, \Sigma, D', R_{\mathcal{D}})$ such that for each look-ahead set L appearing in a rule of $\mathcal{A}$, $L = L(\mathcal{D}(E))$ for some $E \subseteq D$. We denote by $L(E)$ the tree language $L(\mathcal{D}(E))$.

Consider the dt $\mathcal{B} = (\Sigma, B, \tilde{\Sigma} \cup \Delta \cup \Gamma, a_0, R_{\mathcal{B}})$, where $B = A \cup \{b\}$, $b \notin A$ is a new state, $\tilde{\Sigma} = \{\tilde{f} \mid f \in \Sigma\}$, and $\tilde{\Sigma} \cap \Delta = \emptyset$. We define the ranked alphabet Γ and $R_{\mathcal{B}}$ in the following way.

(i) For any $a \in A$, $m \geq 0$, and $f \in \Sigma_m$, let

$$\langle\, a(f(x_1, \ldots, x_m)) \to t_1; L(D_{11}), \ldots, L(D_{1m}) \,\rangle$$

$$\vdots$$

$$\langle\, a(f(x_1, \ldots, x_m)) \to t_n; L(D_{n1}), \ldots, L(D_{nm}) \,\rangle$$

be all rules in $R_{\mathcal{A}}$ with left-hand side $a(f(x_1, \ldots, x_m))$ for some $n \geq 0$. Then we put the function symbol $\langle\, a, f \,\rangle$ in Γ_{m+n} and the rule

$$a(f(x_1, \ldots, x_m)) \to \langle\, a, f \,\rangle(b(x_1), \ldots, b(x_m), t_1, \ldots, t_n)$$

in $R_{\mathcal{B}}$.

(Intuitively, the right-hand side of the above rule contains as subtrees the right-hand sides of all $R_{\mathcal{A}}$-rules with left-hand side $a(f(x_1, \ldots, x_m))$. Moreover, it also contains the variables $x_1, \ldots, x_m$. When applying the above rule at a node ν labeled by f, all subtrees at the sons of ν are copied and then $\mathcal{B}$ is able to relabel these subtrees applying rules of (ii).)

(ii) For any $m \geq 0$, and $f \in \Sigma_m$, we put the rule

$$b(f(x_1, \ldots, x_m)) \to \tilde{f}(b(x_1), \ldots, b(x_m))$$

in $R_{\mathcal{B}}$.

(Intuitively, $\mathcal{B}$ in state b rewrites every symbol f into $\tilde{f}$.)

By direct inspection we see that $\mathcal{B}$ is total.

Intuitively, the ldb $\mathcal{C}$ computes the look-ahead of $\mathcal{A}$ on the subtrees containing symbols with "tilde" and hence is able to choose the rule applied by $\mathcal{A}$. Thus $\mathcal{C}$ deletes the other right-hand sides and the subtrees containing symbols with "tilde" and leaves only the chosen right-hand side. To this end the state set of $\mathcal{C}$ contains all states of $\mathcal{D}$. Moreover, $\mathcal{C}$ contains two other states, *yes* and *no* as well. Consider an input subtree with topmost symbol in Γ. If there is a computation of $\mathcal{A}$ encoded in this subtree, then $\mathcal{C}$ arrives above its topmost symbol in state *yes*, otherwise it arrives above its topmost symbol in state *no*.

To be precise, consider the ldb $\mathcal{C} = (\tilde{\Sigma} \cup \Delta \cup \Gamma, C, \Delta, \{\, yes \,\}, R_{\mathcal{C}})$, where $C = \{\, yes, no \,\} \cup D$ and $R_{\mathcal{C}}$ is defined as follows.

(i) For any $m \geq 0$, and $\delta \in \Delta_m$, we put $\delta(yes(x_1), \ldots, yes(x_m)) \to yes(\delta(x_1, \ldots, x_m))$ in $R_{\mathcal{C}}$. Moreover, for any $m \geq 0$, and $\delta \in \Delta_m$, and $c_1, \ldots, c_m \in \{\, yes, no \,\}$, we put $\delta(c_1(x_1), \ldots, c_m(x_m)) \to no(\delta(x_1, \ldots, x_m))$ in $R_{\mathcal{C}}$ if $c_i = no$ for some $1 \leq i \leq m$.

(Intuitively, if $\mathcal{C}$ reaches the topmost symbol of an input subtree in state *yes*, then it contains some computation of $\mathcal{A}$.)

(ii) For any $m \geq 0$, and $f \in \Sigma_m$, if $f(d_1(x_1), \ldots, d_m(x_m)) \to d(f(x_1, \ldots, x_m)) \in R_{\mathcal{D}}$, then we put the rule $\tilde{f}(d_1(x_1), \ldots, d_m(x_m)) \to d(\tilde{f}(x_1, \ldots, x_m))$ in $R_{\mathcal{C}}$.

(Intuitively, $\mathcal{C}$ computes the look-ahead of $\mathcal{A}$ applying the above rules.)

(iii) For any $a \in A$, $m \geq 0$, $f \in \Sigma_m$, with

$$\langle\, a(f(x_1, \ldots, x_m)) \to t_1; L(D_{11}), \ldots, L(D_{1m})\,\rangle$$

$$\vdots$$

$$\langle\, a(f(x_1, \ldots, x_m)) \to t_n; L(D_{n1}), \ldots, L(D_{nm})\,\rangle$$

being all rules in $R_{\mathcal{A}}$ with left-hand side $a(f(x_1, \ldots, x_m))$, for any $c_1, \ldots, c_n \in \{\, yes, no\,\}$, and for any $d_1, \ldots, d_m \in D$, if there exists an integer $1 \leq j \leq n$ such that $d_1 \in D_{j1}, \ldots, d_m \in D_{jm}$, then we put the rule

$$\langle\, a, f\,\rangle(d_1(x_1), \ldots, d_m(x_m), c_1(x_{m+1}), \ldots, c_n(x_{m+n})) \to c_j(x_{m+j})$$

in $R_{\mathcal{C}}$; otherwise, we put the rule

$$\langle\, a, f\,\rangle(d_1(x_1), \ldots, d_m(x_m), c_1(x_{m+1}), \ldots, c_n(x_{m+n})) \to no(\mu)$$

in $R_{\mathcal{C}}$.

(Intuitively, $\mathcal{C}$ chooses the right-hand side of the rule which is applied by $\mathcal{A}$. Then $\mathcal{C}$ deletes the other right-hand sides and the subtrees containing symbols with tilde and leaves only the chosen right-hand side. If no $\mathcal{A}$-rule is applicable by $\mathcal{A}$, then $\mathcal{C}$ enters the state *no*.)

Note that the determinism of $\mathcal{A}$ ensures that in (iii) there exists at most one j such that $d_1 \in D_{j_1}, \ldots, d_m \in D_{j_n}$. Hence there are no two different rules in $R_{\mathcal{C}}$ with the same left-hand side. By direct inspection we see that $\mathcal{C}$ is a total.

We now show that $\tau_{\mathcal{A}} = \tau_{\mathcal{B}} \circ \tau_{\mathcal{C}}$. It is sufficient to show that for any $a \in A$, $p \in T_\Sigma$ and $q \in T_\Delta$,

$$a(p) \overset{*}{\underset{\mathcal{A}}{\Rightarrow}} q \iff \text{there exists } r \in T_{\tilde{\Sigma} \cup \Delta \cup \Gamma} \text{ such that } a(p) \overset{*}{\underset{\mathcal{B}}{\Rightarrow}} r \text{ and } r \overset{*}{\underset{\mathcal{C}}{\Rightarrow}} yes(q)\ .$$

We proceed by induction on the structure of p.

Case 1. $p \in \Sigma_0$.

"$\Longrightarrow$" Let us suppose that $a(p) \Rightarrow^*_{\mathcal{A}} q$. Then the rule $a(p) \to q$ is in $R_{\mathcal{A}}$, the rule $a(p) \to \langle a, p \rangle(q)$ is in $R_{\mathcal{B}}$, and the rule $\langle a, p \rangle(yes(x_1)) \to yes(x_1) \in R_{\mathcal{C}}$. Let $r = \langle a, p \rangle(q)$. Then $a(p) \Rightarrow_{\mathcal{B}} r$. Moreover, $r \Rightarrow^*_{\mathcal{C}} \langle a, p \rangle(yes(q)) \Rightarrow_{\mathcal{C}} yes(q)$.

"$\Longleftarrow$" Let us assume that there exists $r \in T_{\tilde{\Sigma}\cup\Delta\cup\Gamma}$ such that $a(p) \Rightarrow^*_{\mathcal{B}} r$ and $r \Rightarrow^*_{\mathcal{C}} yes(q)$. Since $height(p) = 0$, $a(p) \to r$ is in $R_{\mathcal{B}}$. By the definition of $R_{\mathcal{B}}$, there exists a rule $a(p) \to s$ in $R_{\mathcal{A}}$ such that $r = \langle a, p\rangle(s)$. By the definition of $R_{\mathcal{C}}$, $\langle a, p\rangle(s) \Rightarrow^*_{\mathcal{C}} \langle a, p\rangle(yes(s)) \Rightarrow_{\mathcal{C}} yes(s)$. Hence $yes(s) = yes(q)$. Thus $s = q$ and $a(p) \Rightarrow^*_{\mathcal{A}} q$.

Case 2. $p = f(p_1, \ldots, p_m)$, $f \in \Sigma_m$, $m \geq 1$, $p_1, \ldots, p_m \in T_{\Sigma}$.

"$\Longrightarrow$" Let us suppose that $a(p) \Rightarrow^*_{\mathcal{A}} q$. Let

$$\langle a(f(x_1, \ldots, x_m)) \to t_1; L(D_{11}), \ldots, L(D_{1m}) \rangle$$

$$\vdots$$

$$\langle a(f(x_1, \ldots, x_m)) \to t_n; L(D_{n1}), \ldots, L(D_{nm}) \rangle$$

be all rules in $R_{\mathcal{A}}$ with left-hand side $a(f(x_1, \ldots, x_m))$, where $n \geq 1$. Let us assume that we applied the jth rule in the above derivation. That is,

$$a(f(p_1, \ldots, p_m)) \underset{\mathcal{A}}{\Rightarrow} s_j[a_1(p_{i_1}), \ldots, a_l(p_{i_l})] \overset{*}{\underset{\mathcal{A}}{\Rightarrow}} s_j[q_1, \ldots, q_l] = q\ ,$$

where $s_j \in \bar{T}_{\Delta}(X_l)$, $l \geq 0$, $t_j = s_j[a_1(x_{i_1}), \ldots, a_l(x_{i_l})]$, $p_1 \in L(D_{j1}), \ldots, p_m \in L(D_{jm})$, and $a_1(p_{i_1}) \Rightarrow^*_{\mathcal{A}} q_1$, ..., $a_l(p_{i_l}) \Rightarrow^*_{\mathcal{A}} q_l$. Hence

$$p_1 \overset{*}{\underset{\mathcal{D}}{\Rightarrow}} d_1(p_1), \ldots, p_m \overset{*}{\underset{\mathcal{D}}{\Rightarrow}} d_m(p_m)$$

for some $d_1 \in D_{j1}, \ldots, d_m \in D_{jm}$. By the definition of $R_{\mathcal{B}}$, the rule $a(f(x_1, \ldots, x_m)) \to \langle a, f \rangle(b(x_1), \ldots, b(x_m), t_1, \ldots, t_n) \in R_{\mathcal{B}}$. Hence

$$a(f(p_1, \ldots, p_m)) \underset{\mathcal{B}}{\Rightarrow} \langle a, f \rangle(b(p_1), \ldots, b(p_m), t_1[p_1, \ldots, p_m], \ldots, t_n[p_1, \ldots, p_m]).$$

Since $\mathcal{B}$ is total, for each $1 \leq i \leq n$, there exists exactly one tree $r_i \in T_{\tilde{\Sigma}\cup\Delta\cup\Gamma}$ such that $t_i[p_1, \ldots, p_m] \Rightarrow^*_{\mathcal{B}} r_i$. Moreover, for each $1 \leq i \leq m$, there exists exactly one tree $\tilde{p}_i \in T_{\tilde{\Sigma}}$ such that $b(p_i) \Rightarrow^*_{\mathcal{B}} \tilde{p}_i$.

Thus we get

$$a(p) = a(f(p_1,\dots,p_m)) \underset{\mathcal{B}}{\Rightarrow}$$

$$\langle a, f\rangle(b(p_1),\dots,b(p_m),t_1[p_1,\dots,p_m],\dots,t_j[p_1,\dots,p_m],\dots,t_n[p_1,\dots,p_m]) \overset{*}{\underset{\mathcal{B}}{\Rightarrow}}$$

$$\langle a, f\rangle(\tilde{p}_1,\dots,\tilde{p}_m,r_1,\dots,r_j,\dots,r_n) = r\ ,$$

where $t_j[p_1,\dots,p_m] = s_j[a_1(p_{i_1}),\dots,a_l(p_{i_l})]$, $r_j = s_j[u_1,\dots,u_l]$, $a_1(p_{i_1}) \Rightarrow^*_{\mathcal{B}} u_1, \dots, a_l(p_{i_l}) \Rightarrow^*_{\mathcal{B}} u_l$ for some $u_1,\dots,u_l \in T_{\tilde{\Sigma}\cup\Delta\cup\Gamma}$. By the induction hypothesis

$$u_1 \overset{*}{\underset{\mathcal{C}}{\Rightarrow}} yes(q_1),\dots,u_l \overset{*}{\underset{\mathcal{C}}{\Rightarrow}} yes(q_l)\ .$$

By (i) of the definition of $R_{\mathcal{C}}$

$$s_j[yes(q_{i_1}),\dots,yes(q_{i_l}))] \overset{*}{\underset{\mathcal{C}}{\Rightarrow}} yes(s_j[q_1,\dots,q_l])\ .$$

Hence

$(*)$ $\quad r_j = s_j[u_1,\dots,u_l] \Rightarrow^*_{\mathcal{C}} s_j[yes(q_{i_1}),\dots,yes(q_{i_l}))] \Rightarrow^*_{\mathcal{C}} yes(s_j[q_{i_1},\dots,q_{i_l}])$
$= yes(q)$.

Since $\mathcal{C}$ is total, for each $1 \le i \le n$ there exists exactly one tree $q_i \in T_\Delta$ such that $r_i \Rightarrow^*_{\mathcal{C}} c_i(q_i)$, where $c_i \in \{yes, no\}$. By $(*)$ $c_j = yes$ and $q_j = q$. By (iii) in the definition of $R_{\mathcal{C}}$, the rule

$$\langle a, f\rangle(d_1(x_1),\dots,d_m(x_m),c_1(x_{m+1}),\dots,c_j(x_{m+j}),\dots,c_n(x_{m+n})) \to c_j(x_{m+j})$$

is in $R_{\mathcal{C}}$, where $c_i \in \{yes, no\}$, $i \in \{1,\dots,n\}$. Hence

$$r \overset{*}{\underset{\mathcal{C}}{\Rightarrow}} \langle a, f\rangle(d_1(\tilde{p}_1),\dots,d_m(\tilde{p}_m),c_1(q_1),\dots,c_j(q_j),\dots,c_n(q_n)) \underset{\mathcal{C}}{\Rightarrow} yes(q)\ .$$

" $\Longleftarrow$" Let us assume that there exists $r \in T_{\tilde{\Sigma}\cup\Delta\cup\Gamma}$ such that $a(p) \Rightarrow^*_{\mathcal{B}} r$ and $r \Rightarrow^*_{\mathcal{C}} yes(q)$. Then

$$a(p) = a(f(p_1,\dots,p_m)) \underset{\mathcal{B}}{\Rightarrow}$$

$$\langle a, f\rangle(b(p_1),\dots,b(p_m),t_1[p_1,\dots,p_m],\dots,t_n[p_1,\dots,p_m]) \overset{*}{\underset{\mathcal{B}}{\Rightarrow}}$$

$$\langle a, f\rangle(\tilde{p}_1,\dots,\tilde{p}_m,r_1,\dots,r_n) = r,$$

where $a(f(x_1,\ldots,x_m)) \to \langle\, a, f\,\rangle(b(x_1),\ldots,b(x_m),t_1,\ldots,t_n) \in R_{\mathcal{B}}$, $b(p_i) \Rightarrow^*_{\mathcal{B}} \tilde{p}_i$ for $1 \le i \le m$, and $t_i[p_1,\ldots,p_m] \Rightarrow^*_{\mathcal{B}} r_i$ for $1 \le i \le n$.

For each $1 \le i \le n$, $t_i = s_i[a_{i1}(x_{\phi(i,1)}),\ldots,a_{il_i}(x_{\phi(i,l_i)})]$ for some $l_i \ge 0$, $s_i \in \bar{T}_\Delta(X_{l_i})$, and $a_{i_1},\ldots,a_{il_i} \in A$. Here $\phi : \{\,1,\ldots,n\,\} \times \{\,1,2,\ldots\,\} \to \{\,1,2,\ldots\,\}$ is a mapping. Hence

$$\text{for each } 1 \le i \le n,\ t_i[p_1,\ldots,p_m] = s_i[a_{i1}(p_{\phi(i,1)}),\ldots,a_{il_i}(p_{\phi(i,l_i)})]\ ,$$

and the derivation $t_i[p_1,\ldots,p_m] \Rightarrow^*_{\mathcal{B}} r_i$ has the form

$$t_i[p_1,\ldots,p_m] = s_i[a_{i1}(p_{\phi(i,1)}),\ldots,a_{il_i}(p_{\phi(i,l_i)})] \overset{*}{\underset{\mathcal{B}}{\Rightarrow}} s_i[u_{i1},\ldots,u_{il_i}] = r_i$$

where

$$a_{i1}(p_{\phi(i,1)}) \overset{*}{\underset{\mathcal{B}}{\Rightarrow}} u_{i1},\ldots,a_{il_i}(p_{\phi(i,l_i)}) \overset{*}{\underset{\mathcal{B}}{\Rightarrow}} u_{il_i}\ .$$

We now write the derivation $r \Rightarrow^*_{\mathcal{C}} yes(q)$ in a more detailed form

$$\langle a, f\rangle(\tilde{p}_1,\ldots,\tilde{p}_m,r_1,\ldots,r_n) \overset{*}{\underset{\mathcal{C}}{\Rightarrow}} \langle\, a, f\rangle(d_1(\tilde{p}_1),\ldots,d_m(\tilde{p}_m),c_1(q_1),\ldots,c_n(q_n))$$

$$\underset{\mathcal{C}}{\Rightarrow} yes(q)\ ,$$

where $\tilde{p}_i \Rightarrow^*_{\mathcal{C}} d_i(\tilde{p}_i)$ for $1 \le i \le m$; for each $1 \le i \le n$, $q_i \in T_\Delta$, $c_i \in \{yes, no\}$, and $r_i \Rightarrow^*_{\mathcal{C}} c_i(q_i)$. In the last step we apply the rule

$$\langle\, a, f\,\rangle(d_1(x_1),\ldots,d_m(x_m),c_1(x_{m+1}),\ldots,c_n(x_{m+n})) \to c_j(x_{m+j}) \in R_{\mathcal{C}},$$

where for each $1 \le i \le n$, $c_i \in \{yes, no\}$, $c_j = yes$, and $q_j = q$.

By the definition of $R_{\mathcal{C}}$

$$\langle\, a(f(x_1,\ldots,x_m)) \to t_j; L(D_{j1}),\ldots,L(D_{jm})\,\rangle \in R_{\mathcal{A}}\ ,$$

where $d_1 \in D_{j1},\ldots,d_m \in D_{jm}$, hence $p_1 \in L(D_{j1}),\ldots,p_m \in L(D_{jm})$.

Moreover,

$$r_j = s_j[u_{j1},\ldots,u_{jl_j}] \overset{*}{\underset{\mathcal{C}}{\Rightarrow}} s_j[yes(v_1),\ldots,yes(v_{l_j})] \overset{*}{\underset{\mathcal{C}}{\Rightarrow}} yes(s_j[v_1,\ldots,v_{l_j}]) =$$

$$yes(q_j)\ ,$$

where $v_1,\ldots,v_{l_j} \in T_\Delta$,

$$u_{j1} \overset{*}{\underset{\mathcal{C}}{\Rightarrow}} yes(v_1),\ldots,u_{jl_j} \overset{*}{\underset{\mathcal{C}}{\Rightarrow}} yes(v_{l_j})\ ,$$

and

$$s_j[v_1, \ldots, v_{l_j}] = q_j \ .$$

By the induction hypothesis,

$$a_{j_1}(p_{\phi(j,1)}) \overset{*}{\underset{\mathcal{A}}{\Rightarrow}} v_1, \ldots, a_{jl_j}(p_{\phi(j,l_j)}) \overset{*}{\underset{\mathcal{A}}{\Rightarrow}} v_{l_j} \ .$$

Hence

$$a(p) = a(f(p_1, \ldots, p_m)) \underset{\mathcal{A}}{\Rightarrow} s_j[a_{j_1}(p_{\phi(j,1)}), \ldots, a_{j_l}(p_{\phi(j,l_j)})] \overset{*}{\underset{\mathcal{A}}{\Rightarrow}} s_j[v_1, \ldots, v_{l_j}] =$$

$$q_j = q \ .$$

□

We recall some composition and decomposition results.

Proposition 4.2.3 [15]

$$\begin{array}{llcl} (i) & DT^R \circ DT^R & = & DT^R, \\ (ii) & LDT^R \circ LDT^R & = & LDT^R, \\ (iii) & DT^R & \subseteq & LDB \circ DT, \\ (iv) & LDT^R & \subseteq & LDB \circ LDT. \end{array}$$

Lemma 4.2.4 *For every $(\alpha, \beta) \in T$, $||\alpha|| = ||\beta||$, or equivalently, $(\alpha, \beta) \in \rho$.*

Proof. We obtain the following result by direct inspection.

$$(\dagger) \qquad I \subseteq LDT \subseteq LDT^R \subseteq DT^R \ .$$

In the proof of Theorem 3.2 in [15], Engelfriet showed that $DB \subseteq DT^R$. His proof carries over to the linear case, that is, one can show in the same way that $LDB \subseteq LDT^R$. Thus we get that

$$(\ddagger) \qquad I \subseteq LDB \subseteq LDT^R \subseteq DT^R \ .$$

The following equations hold.

(1′)	$LDT^R \circ LDT$	$=$	LDT^R	(by (†), and (ii) of Proposition 4.2.3)
(2′)	$LDT^R \circ LDB$	$=$	LDT^R	(by (‡), and (ii) of Proposition 4.2.3)
(3′)	$DT \circ LDB$	$=$	DT^R	(by Theorem 4.2.2)
(4′)	$DT \circ H$	$=$	DT^2	(by Table 2 in [36])
(5′)	$DT \circ LH$	$=$	DT^2	(by Table 2 in [36])
(6′)	$LDT \circ DT^R$	$=$	DT^R	(by (†), and (i) of Proposition 4.2.3)
(7′)	$LDT \circ LDT^R$	$=$	LDT^R	(by (†), and (ii) of Proposition 4.2.3)
(8′)	$LDT \circ DT$	$=$	DT^2	(by Table 2 in [36])
(9′)	$LDT \circ LH$	$=$	LDT^2	(by Table 2 in [36])
(10′)	$LDB \circ DT$	$=$	DT^R	(by (‡), (i) and (iii) of Proposition 4.2.3)
(11′)	$LDB \circ LDT$	$=$	LDT^R	(by (‡), (†), (ii) and (iv) of Proposition 4.2.3)
(12′)	$LDB \circ LDB$	$=$	LDB	(by Figure 2 in [24])
(13′)	$LDB \circ H$	$=$	DB	(by Figure 2 in [24])
(14′)	$LDB \circ LH$	$=$	LDB	(by Figure 2 in [24])
(15′)	$H \circ LDT$	$=$	DT	(by Table 2 in [36])
(16′)	$H \circ LDB$	$=$	DB	(by Figure 2 in [24])
(17′)	$H \circ H$	$=$	H	(by Figure 2 in [24])
(18′)	$H \circ LH$	$=$	H	(by Figure 2 in [24])
(19′)	$LH \circ LDT$	$=$	LDT	(by Table 2 in [36])
(20′)	$LH \circ LDB$	$=$	LDB	(by Figure 2 in [24])
(21′)	$LH \circ H$	$=$	H	(by Figure 2 in [24])
(22′)	$LH \circ LH$	$=$	LH	(by Figure 2 in [24])

$\square$

Consequence 4.2.5 $\leftrightarrow_T^* \subseteq \rho$.

The following calculation will be needed in Subsection 4.2.2. However, we present is here so that they are more close to T, which is needed to verify it. We list elements (u_k, v_k) $(23 \leq k \leq 106)$ of $\leftrightarrow_T^*$. For each integer $23 \leq k \leq 106$, we write $u_k \leftrightarrow_T^* v_k$ rather than $(u_k, v_k) \in \leftrightarrow_T^*$. Moreover, the list $(i_1, \ldots, i_{l_k})$ of integers with $1 \leq i_1, \ldots, i_{l_k} \leq k-1$ which follows the pair $u_k \leftrightarrow_T^* v_k$ indicates that u_k can be transformed into v_k by applying the relations $(i_1), \ldots, (i_{l_k})$ in this order. In this way we present the proof that $u_k \leftrightarrow_T^* v_k$.

(23)	$DT^R \cdot DT^R$	$\leftrightarrow_T^*$	DT^R	$(3, 10, 12, 10, 15, 6, 3, 15, 17, 15, 3)$
(24)	$LDT^R \cdot DT^R$	$\leftrightarrow_T^*$	DT^R	$(11, 6, 10, 12, 10)$
(25)	$DT \cdot DT^R$	$\leftrightarrow_T^*$	DT^R	$(15, 6, 3, 15, 17, 15, 3)$
(26)	$DB \cdot DT^R$	$\leftrightarrow_T^*$	DT^R	$(16, 10, 12, 10, 3, 15, 17, 15, 3)$
(27)	$LDB \cdot DT^R$	$\leftrightarrow_T^*$	DT^R	$(10, 12, 10)$
(28)	$H \cdot DT^R$	$\leftrightarrow_T^*$	DT^R	$(3, 15, 17, 15, 3)$
(29)	$LH \cdot DT^R$	$\leftrightarrow_T^*$	DT^R	$(3, 15, 21, 15, 3)$
(30)	$DT^2 \cdot DT^R$	$\leftrightarrow_T^*$	DT^R	$(25, 25)$
(31)	$LDT^2 \cdot DT^R$	$\leftrightarrow_T^*$	DT^R	$(6, 6)$
(32)	$LDT \cdot H \cdot DT^R$	$\leftrightarrow_T^*$	DT^R	$(28, 6)$
(33)	$LDT \cdot LDB \cdot DT^R$	$\leftrightarrow_T^*$	DT^R	$(27, 6)$
(34)	$LDT \cdot DB \cdot DT^R$	$\leftrightarrow_T^*$	DT^R	$(26, 6)$
(35)	$LDT^R \cdot H \cdot DT^R$	$\leftrightarrow_T^*$	DT^R	$(28, 24)$
(36)	$DT^R \cdot LDT^R$	$\leftrightarrow_T^*$	DT^R	$(3, 11, 12, 11, 15, 7, 11, 16, 13, 15, 10)$
(37)	$LDT^R \cdot LDT^R$	$\leftrightarrow_T^*$	LDT^R	$(11, 2, 1)$
(38)	$DT \cdot LDT^R$	$\leftrightarrow_T^*$	DT^R	$(15, 7, 11, 16, 13, 15, 10)$
(39)	$DB \cdot LDT^R$	$\leftrightarrow_T^*$	DT^R	$(16, 11, 12, 16, 13, 15, 10)$
(40)	$LDB \cdot LDT^R$	$\leftrightarrow_T^*$	LDT^R	$(11, 12, 11)$
(41)	$H \cdot LDT^R$	$\leftrightarrow_T^*$	DT^R	$(11, 16, 13, 15, 10)$
(42)	$LH \cdot LDT^R$	$\leftrightarrow_T^*$	LDT^R	$(11, 20, 11)$
(43)	$DT^2 \cdot LDT^R$	$\leftrightarrow_T^*$	DT^R	$(38, 25)$
(44)	$LDT^2 \cdot LDT^R$	$\leftrightarrow_T^*$	LDT^R	$(7, 7)$
(45)	$LDT \cdot H \cdot LDT^R$	$\leftrightarrow_T^*$	DT^R	$(41, 6)$
(46)	$LDT \cdot LDB \cdot LDT^R$	$\leftrightarrow_T^*$	LDT^R	$(40, 7)$
(47)	$LDT \cdot DB \cdot LDT^R$	$\leftrightarrow_T^*$	DT^R	$(39, 6)$
(48)	$LDT^R \cdot H \cdot LDT^R$	$\leftrightarrow_T^*$	DT^R	$(41, 24)$
(49)	$DT^R \cdot DT$	$\leftrightarrow_T^*$	DT^R	$(3, 10, 25)$
(50)	$LDT^R \cdot DT$	$\leftrightarrow_T^*$	DT^R	$(11, 8, 10, 49)$
(51)	$DB \cdot DT$	$\leftrightarrow_T^*$	DT^R	$(16, 10, 28)$
(52)	$H \cdot DT$	$\leftrightarrow_T^*$	DT	$(15, 17, 15)$
(53)	$LH \cdot DT$	$\leftrightarrow_T^*$	DT	$(15, 21, 15)$
(54)	DT^3	$\leftrightarrow_T^*$	DT^2	$(4, 52)$
(55)	$LDT^2 \cdot DT$	$\leftrightarrow_T^*$	DT^2	$(8, 8, 54)$
(56)	$LDT \cdot H \cdot DT$	$\leftrightarrow_T^*$	DT^2	$(52, 8)$
(57)	$LDT \cdot LDB \cdot DT$	$\leftrightarrow_T^*$	DT^R	$(10, 6)$

(58)	$LDT \cdot DB \cdot DT$	$\leftrightarrow_T^*$	DT^R	$(51, 6)$
(59)	$LDT^R \cdot H \cdot DT$	$\leftrightarrow_T^*$	DT^R	$(52, 50)$
(60)	$DT^R \cdot LDT$	$\leftrightarrow_T^*$	DT^R	$(3, 11, 38)$
(61)	$DT \cdot LDT$	$\leftrightarrow_T^*$	DT^2	$(15, 9, 15, 5)$
(62)	$DB \cdot LDT$	$\leftrightarrow_T^*$	DT^R	$(13, 15, 10)$
(63)	$DT^2 \cdot LDT$	$\leftrightarrow_T^*$	DT^2	$(61, 54)$
(64)	LDT^3	$\leftrightarrow_T^*$	LDT^2	$(9, 19)$
(65)	$LDT \cdot H \cdot LDT$	$\leftrightarrow_T^*$	DT^2	$(15, 8)$
(66)	$LDT \cdot LDB \cdot LDT$	$\leftrightarrow_T^*$	LDT^R	$(11, 7)$
(67)	$LDT \cdot DB \cdot LDT$	$\leftrightarrow_T^*$	DT^R	$(62, 6)$
(68)	$LDT^R \cdot H \cdot LDT$	$\leftrightarrow_T^*$	DT^R	$(15, 50)$
(69)	$DT^R \cdot DB$	$\leftrightarrow_T^*$	DT^R	$(3, 13, 12, 13, 16, 4, 3, 25)$
(70)	$LDT^R \cdot DB$	$\leftrightarrow_T^*$	$LDT^R \cdot H$	$(13, 2)$
(71)	$DT \cdot DB$	$\leftrightarrow_T^*$	DT^R	$(16, 4, 3, 25)$
(72)	DB^2	$\leftrightarrow_T^*$	DB	$(16, 13, 12, 16, 13, 17, 13)$
(73)	$LDB \cdot DB$	$\leftrightarrow_T^*$	DB	$(13, 12, 13)$
(74)	$H \cdot DB$	$\leftrightarrow_T^*$	DB	$(16, 17, 16)$
(75)	$LH \cdot DB$	$\leftrightarrow_T^*$	DB	$(16, 21, 16)$
(76)	$DT^2 \cdot DB$	$\leftrightarrow_T^*$	DT^R	$(71, 25)$
(77)	$LDT^2 \cdot DB$	$\leftrightarrow_T^*$	$LDT \cdot DB$	$(9, 75)$
(78)	$LDT \cdot H \cdot DB$	$\leftrightarrow_T^*$	$LDT \cdot DB$	(74)
(79)	$LDT \cdot LDB \cdot DB$	$\leftrightarrow_T^*$	$LDT \cdot DB$	(73)
(80)	$LDT \cdot DB^2$	$\leftrightarrow_T^*$	$LDT \cdot DB$	(72)
(81)	$LDT^R \cdot H \cdot DB$	$\leftrightarrow_T^*$	$LDT^R \cdot H$	$(74, 70)$
(82)	$DT^R \cdot LDB$	$\leftrightarrow_T^*$	DT^R	$(3, 12, 3)$
(83)	$DB \cdot LDB$	$\leftrightarrow_T^*$	DB	$(16, 12, 16)$
(84)	$DT^2 \cdot LDB$	$\leftrightarrow_T^*$	DT^R	$(3, 25)$
(85)	$LDT^2 \cdot LDB$	$\leftrightarrow_T^*$	$LDT \cdot LDB$	$(9, 20)$
(86)	$LDT \cdot H \cdot LDB$	$\leftrightarrow_T^*$	$LDT \cdot DB$	(16)
(87)	$LDT \cdot LDB^2$	$\leftrightarrow_T^*$	$LDT \cdot LDB$	(12)
(88)	$LDT \cdot DB \cdot LDB$	$\leftrightarrow_T^*$	$LDT \cdot DB$	(83)
(89)	$LDT^R \cdot H \cdot LDB$	$\leftrightarrow_T^*$	$LDT^R \cdot H$	$(16, 70)$
(90)	$DT^R \cdot H$	$\leftrightarrow_T^*$	DT^R	$(3, 13, 71)$
(91)	$DB \cdot H$	$\leftrightarrow_T^*$	DB	$(13, 17, 13)$
(92)	$DT^2 \cdot H$	$\leftrightarrow_T^*$	DT^2	$(4, 54)$
(93)	$LDT^2 \cdot H$	$\leftrightarrow_T^*$	$LDT \cdot H$	$(9, 21)$
(94)	$LDT \cdot H^2$	$\leftrightarrow_T^*$	$LDT \cdot H$	(17)

(95)	$LDT \cdot LDB \cdot H$	$\leftrightarrow_T^*$	$LDT \cdot DB$	(13)
(96)	$LDT \cdot DB \cdot H$	$\leftrightarrow_T^*$	$LDT \cdot DB$	(91)
(97)	$LDT^R \cdot H^2$	$\leftrightarrow_T^*$	$LDT^R \cdot H$	(17)
(98)	$DT^R \cdot LH$	$\leftrightarrow_T^*$	DT^R	(3, 14, 3)
(99)	$LDT^R \cdot LH$	$\leftrightarrow_T^*$	LDT^R	(11, 9, 11, 1)
(100)	$DB \cdot LH$	$\leftrightarrow_T^*$	DB	(13, 18, 13)
(101)	$DT^2 \cdot LH$	$\leftrightarrow_T^*$	DT^2	(4, 18, 4)
(102)	$LDT^2 \cdot LH$	$\leftrightarrow_T^*$	LDT^2	(9, 22, 9)
(103)	$LDT \cdot H \cdot LH$	$\leftrightarrow_T^*$	$LDT \cdot H$	(18)
(104)	$LDT \cdot LDB \cdot LH$	$\leftrightarrow_T^*$	$LDT \cdot LDB$	(14)
(105)	$LDT \cdot DB \cdot LH$	$\leftrightarrow_T^*$	$LDT \cdot DB$	(100)
(106)	$LDT^R \cdot H \cdot LH$	$\leftrightarrow_T^*$	$LDT^R \cdot H$	(18)

4.2.2 The string rewriting system S

In this subsection, we give a subset K of M^* and a string rewriting system S over M such that $\leftrightarrow_T^* = \leftrightarrow_S^*$. Moreover, we prove that there is a linear time algorithm which, for every $u \in M^*$, computes a word $v \in K$ in linear time such that $u \rightarrow^* v$.

Let $K \subseteq M^*$ be defined by

$$K = \{\, \lambda, DT^R, LDT^R, DT, LDT, DB, LDB, H, LH \,\} \cup$$

$$\{\, DT^2, LDT^2, LDT \cdot H, LDT \cdot LDB, LDT \cdot DB, LDT^R \cdot H \,\}\,.$$

The string rewriting system S is visualized in two tables. In the Table 4.1 each row corresponds to an element u of $K - \{\lambda\}$ and each column corresponds to an element $Y \in \{\, DT^R, LDT^R, DT, LDT, DB \,\}$. In the Table 4.2 each row corresponds to an element u of $K - \{\lambda\}$ and each column corresponds to an element $Y \in \{\, LDB, H, LH \,\}$. Hence each element u of $K - \{\lambda\}$ and each element $Y \in M$ determine an entry of Table 4.1 or Table 4.2. By direct inspection we now show that the entry determined by $u \in K - \{\lambda\}$ and $Y \in M$ contains a word $v \in K - \{\lambda\}$ such that either $u \cdot Y \leftrightarrow_T^* v$ or $u \cdot Y = v$. First we read the entries in the Table 4.1 and then in the Table 4.2. In both tables we proceed column by column. In each column we proceed from the top to the bottom. Simultaneously, we read the relations (23) – (106) beginning with (23). For each entry determined by $u \in K - \{\lambda\}$ and $Y \in M$ and containing $v \in K$, we carry out the following.

If $u \cdot Y = v$, then we go to the next entry. If $u \cdot Y \neq v$, then we read the current relation (i) of (23) – (106). If (i) is of the form $u \cdot Y \leftrightarrow_T^* v$, then we go to the next entry and to the relation $(i+1)$, otherwise we find the relation $u \cdot Y \leftrightarrow_T^* v$ among (1) – (22).

Let the string rewriting system S be the set of all pairs of the form $(u{\cdot}Y, v)$, where $u \in K - \{\lambda\}$, $Y \in M$, and v is the entry of Table 4.1 or Table 4.2 determined by the row of u and the column of Y, moreover, $u \cdot Y \neq v$. By the above observation, $\leftrightarrow_S^* \subseteq \leftrightarrow_T^*$. Conversely, by direct inspection we obtain that $T \subseteq S$. Hence $\leftrightarrow_T^* \subseteq \leftrightarrow_S^*$. Moreover, by Consequence 4.2.5, $\leftrightarrow_T^* \subseteq \rho$. Thus we get the following result.

Theorem 4.2.6 $\leftrightarrow_T^* = \leftrightarrow_S^* \subseteq \rho$.

In the following discussion, we need the concept of the finite automaton, therefore we introduce it now.

A *finite automaton* is a 4-tuple $\mathcal{A} = (\Sigma, A, \delta, a_0)$, where Σ is an *input (unranked) alphabet*, A is a *finite set of states*, δ is a mapping from $A \times \Sigma$ to A, and $a_0 \in A$ is the *initial state.* A *configuration* of $\mathcal{A}$ is a pair (q, w) in $Q \times \Sigma^*$. The *move relation* $\vdash_{\mathcal{A}}$ is defined as follows. For any configurations (q, w), (q', w') of $\mathcal{A}$, if $w = yw'$ and $\delta(q, y) = q'$, then $(q, w) \vdash_{\mathcal{A}} (q', w')$. The reflexive, transitive closure of $\vdash_{\mathcal{A}}$ is denoted by $\vdash_{\mathcal{A}}^*$.

Consider the automaton $\mathcal{A} = (M, K, \delta, \lambda)$, where for each $Y \in M$, $\delta(\lambda, Y) = Y$, and for any $u \in K - \{\lambda\}$ and $Y \in M$, $\delta(u, Y)$ is equal to the entry of Table 4.1 or Table 4.2 determined by the row of u and the column of Y.

Claim 4.2.7 *For any $w \in M^*$ and $z \in K$, if $(\lambda, w) \vdash_{\mathcal{A}}^* (z, \lambda)$ then $w \to_S^* z$.*

Proof. By the definition of S and $\mathcal{A}$, for any $u \in K$ and $Y \in M$, $\delta(u, Y) = v$ if and only either $u \cdot Y = v$ or $u \cdot Y \to_S v$. Hence the claim follows by an easy induction on the length of w. □

By Claim 4.2.7, for every word $w \in M^*$, the automaton $\mathcal{A}$ computes in $O(|w|)$ time a word $z \in K$ such that $w \to_S^* z$. Hence we have got the following result.

Theorem 4.2.8 *There is a linear time algorithm which, for every word $w \in M^*$, computes a word $z \in K$ such that $w \to_S^* z$.*

	DT^R	LDT^R	DT	LDT	DB
DT^R	DT^R	DT^R	DT^R	DT^R	DT^R
LDT^R	DT^R	LDT^R	DT^R	LDT^R	$LDT^R \cdot H$
DT	DT^R	DT^R	DT^2	DT^2	DT^R
LDT	DT^R	LDT^R	DT^2	LDT^2	$LDT \cdot DB$
DB	DT^R	DT^R	DT^R	DT^R	DB
LDB	DT^R	LDT^R	DT^R	LDT^R	DB
H	DT^R	DT^R	DT	DT	DB
LH	DT^R	LDT^R	DT	LDT	DB
DT^2	DT^R	DT^R	DT^2	DT^2	DT^R
LDT^2	DT^R	LDT^R	DT^2	LDT^2	$LDT \cdot DB$
$LDT \cdot H$	DT^R	DT^R	DT^2	DT^2	$LDT \cdot DB$
$LDT \cdot LDB$	DT^R	LDT^R	DT^R	LDT^R	$LDT \cdot DB$
$LDT \cdot DB$	DT^R	DT^R	DT^R	DT^R	$LDT \cdot DB$
$LDT^R \cdot H$	DT^R	DT^R	DT^R	DT^R	$LDT^R \cdot H$

Table 4.1: The string rewriting system S (part 1).

	LDB	H	LH
DT^R	DT^R	DT^R	DT^R
LDT^R	LDT^R	$LDT^R \cdot H$	LDT^R
DT	DT^R	DT^2	DT^2
LDT	$LDT \cdot LDB$	$LDT \cdot H$	LDT^2
DB	DB	DB	DB
LDB	LDB	DB	LDB
H	DB	H	H
LH	LDB	H	LH
DT^2	DT^R	DT^2	DT^2
LDT^2	$LDT \cdot LDB$	$LDT \cdot H$	LDT^2
$LDT \cdot H$	$LDT \cdot DB$	$LDT \cdot H$	$LDT \cdot H$
$LDT \cdot LDB$	$LDT \cdot LDB$	$LDT \cdot DB$	$LDT \cdot LDB$
$LDT \cdot DB$	$LDT \cdot DB$	$LDT \cdot DB$	$LDT \cdot DB$
$LDT^R \cdot H$	$LDT^R \cdot H$	$LDT^R \cdot H$	$LDT^R \cdot H$

Table 4.2: The string rewriting system S (part 2).

4.2.3 The inclusion diagram of $||K||$

In this subsection we give an inclusion diagram for the set of tree transformation classes $||K|| = \{\, ||w|| \mid w \in K \,\}$. To this end, we first show the following 6 lemmas.

Lemma 4.2.9 *$LDT^R \not\subseteq LDT \circ DB$.*

Proof. Let $\Sigma = \Sigma_0 \cup \Sigma_1 \cup \Sigma_2$, where $\Sigma_0 = \{\sharp, \$\}$, $\Sigma_1 = \{g\}$, and $\Sigma_2 = \{f\}$. Let $\rho \subseteq T_\Sigma \times T_\Sigma$ be defined by

$$\rho = \{\, (f(g^n(\sharp), \$), f(g^n(\$), \$)) \mid n \geq 0 \,\} \cup \{\, (f(g^n(\sharp), \sharp), f(g^n(\sharp), \sharp)) \mid n \geq 0 \,\}.$$

Here and in what follows, $g^0(\sharp) = \sharp$ and $g^{n+1}(\sharp) = g(g^n(\sharp))$ for $n \geq 0$. Consider the ldt^R $\mathcal{A} = (\Sigma, A, \Sigma, a_0, R_\mathcal{A})$, where $A = \{\, a_0, a_1, a_2, a_3, a_4 \,\}$ and $R_\mathcal{A}$ consists of the following 8 rules.

$$\begin{array}{lcl}
\langle\, a_0(f(x_1, x_2)) & \to & f(a_1(x_1), a_2(x_2)); T_\Sigma, \{\, \$ \,\} \,\rangle, \\
\langle\, a_0(f(x_1, x_2)) & \to & f(a_3(x_1), a_4(x_2)); T_\Sigma, \{\, \sharp \,\} \,\rangle, \\
\langle\, a_1(g(x_1)) & \to & g(a_1(x_1)); T_\Sigma \,\rangle, \\
\langle\, a_1(\sharp) & \to & \$; \,\rangle, \\
\langle\, a_2(\$) & \to & \$; \,\rangle, \\
\langle\, a_3(g(x_1)) & \to & g(a_3(x_1)); T_\Sigma \,\rangle, \\
\langle\, a_3(\sharp) & \to & \sharp; \,\rangle, \\
\langle\, a_4(\sharp) & \to & \sharp; \,\rangle.
\end{array}$$

It should be clear that $\tau_\mathcal{A} = \rho$. We now show that $\rho \notin LDT \circ DB$. By way of a contradiction, let us suppose that there is an ldt $\mathcal{B} = (\Sigma, B, \Delta, b_0, R_\mathcal{B})$ and a db $\mathcal{C} = (\Delta, C, \Sigma, C', R_\mathcal{C})$ such that $\rho = \tau_\mathcal{B} \circ \tau_\mathcal{C}$. As $dom(\rho) \subseteq dom(\tau_\mathcal{B})$ and $\mathcal{B}$ is deterministic, there is exactly one rule in $R_\mathcal{B}$ with left-hand side $b_0(f(x_1, x_2))$. Let us suppose that this rule is of the form

$$b_0(f(x_1, x_2)) \to r\ ,$$

where $r \in T_\Delta(B(X_2))$. We now show that both x_1 and x_2 appear in r. Let us suppose that x_1 does not appear in r. Then $\tau_\mathcal{B}(f(\sharp, \sharp)) = \tau_\mathcal{B}(f(g(\sharp), \sharp))$. Hence

$$f(\sharp, \sharp) = \tau_\mathcal{B} \circ \tau_\mathcal{C}(f(\sharp, \sharp)) = \tau_\mathcal{B} \circ \tau_\mathcal{C}(f(g(\sharp), \sharp)) = f(g(\sharp), \sharp)\ .$$

A contradiction. Thus x_1 appears in r. Let us suppose that x_2 does not appear in r. Then $\tau_{\mathcal{B}}(f(\sharp,\sharp)) = \tau_{\mathcal{B}}(f(\sharp,\$))$. Hence

$$f(\sharp,\sharp) = \tau_{\mathcal{B}} \circ \tau_{\mathcal{C}}(f(\sharp,\sharp)) = \tau_{\mathcal{B}} \circ \tau_{\mathcal{C}}(f(\sharp,\$)) = f(\$,\$) .$$

A contradiction. Thus x_2 appears in r. Hence $r = t[b_1(x_1), b_2(x_2)]$, where $t \in T_\Delta(X_2)$, and both x_1 and x_2 appear in t, and $b_1, b_2 \in B$.

Consider the derivations

$$f(g^n(\sharp),\$) \underset{\mathcal{B}}{\Rightarrow} t[b_1(g^n(\sharp)), b_2(\$)] \underset{\mathcal{B}}{\Rightarrow} t[b_1(g^n(\sharp)), p_1] ,$$
$$f(g^n(\sharp),\sharp) \underset{\mathcal{B}}{\Rightarrow} t[b_1(g^n(\sharp)), b_2(\sharp)] \underset{\mathcal{B}}{\Rightarrow} t[b_1(g^n(\sharp)), p_2] ,$$

where $p_1, p_2 \in T_\Delta$, and the rules $b_2(\$) \to p_1$ and $b_2(\sharp) \to p_2$ are in $R_{\mathcal{B}}$. Let

$$U = \{\, s \in \bar{T}_\Sigma(X_1) \mid t[c(x_1), p_1] \overset{*}{\underset{\mathcal{C}}{\Rightarrow}} c'(s) \text{ for some } c \in C, c' \in C' \,\} \cup$$

$$\{\, s \in \bar{T}_\Sigma(X_1) \mid t[c(x_1), p_2] \overset{*}{\underset{\mathcal{C}}{\Rightarrow}} c'(s) \text{ for some } c \in C, c' \in C' \,\} .$$

It should be clear that $|U| \leq 2|C|$. Let

$$J = max\{\, height(s) \mid s \in U \,\} .$$

Let $n > J$. Then

$$b_0(f(g^n(\sharp),\$)) \underset{\mathcal{B}}{\Rightarrow} t[b_1(g^n(\sharp)), b_2(\$)] \underset{\mathcal{B}}{\Rightarrow} t[b_1(g^n(\sharp)), p_1] \overset{*}{\underset{\mathcal{B}}{\Rightarrow}} t[p, p_1] ,$$

$$b_0(f(g^n(\sharp),\sharp)) \underset{\mathcal{B}}{\Rightarrow} t[b_1(g^n(\sharp)), b_2(\sharp)] \underset{\mathcal{B}}{\Rightarrow} t[b_1(g^n(\sharp)), p_2] \overset{*}{\underset{\mathcal{B}}{\Rightarrow}} t[p, p_2]$$

for some $p, p_1, p_2 \in T_\Delta$. Since $f(g^n(\sharp),\sharp) \in dom(\rho)$ and $f(g^n(\sharp),\$) \in dom(\rho)$,

$$t[p, p_1] \overset{*}{\underset{\mathcal{C}}{\Rightarrow}} t[c(q), c_1(q_1)] \overset{*}{\underset{\mathcal{C}}{\Rightarrow}} c'(s_1[q, q_1]) = c'(f(g^n(\$),\$)) ,$$

$$t[p, p_2] \overset{*}{\underset{\mathcal{C}}{\Rightarrow}} t[c(q), c_2(q_2)] \overset{*}{\underset{\mathcal{C}}{\Rightarrow}} c''(s_2[q, q_2]) = c''(f(g^n(\sharp),\sharp))$$

where $c, c_1, c_2 \in C$, $c', c'' \in C'$, $q, q_1, q_2 \in T_\Sigma$, $s_1, s_2 \in T_\Sigma(X_2)$, $p \Rightarrow^*_{\mathcal{C}} c(q)$, $p_1 \Rightarrow^*_{\mathcal{C}} c_1(q_1)$, $p_2 \Rightarrow^*_{\mathcal{C}} c_2(q_2)$,

$$t[c(x_1), c_1(q_1)] \overset{*}{\underset{\mathcal{C}}{\Rightarrow}} c'(s_1[x_1, q_1]) ,$$

$$t[c(x_1), c_2(q_2)] \overset{*}{\underset{C}{\Rightarrow}} c''(s_2[x_1, q_2]) .$$

Hence $s_1[x_1, q_1] \in U$, $s_2[x_1, q_2] \in U$, and

$$s_1[q, q_1] = f(g^n(\$), \$), \quad s_2[q, q_2] = f(g^n(\sharp), \sharp) .$$

By the definition of J and n, $|llp(s_1[x_1, q_1])| \leq J < n$. On the other hand,

$$|llp(f(g^n(\$), \$))| = |llp(s_1[q, q_1])| = n + 1 ,$$

hence $llp(s_1[x_1, q_1])$ is a proper prefix of $llp(s_1[q, q_1])$. Thus x_1 appears in the tree s_1, and $llp(s_1[x_1, q_1])$ leads to an occurrence of x_1. Moreover, as $llp(s_1[q, q_1])$ leads to \$ in the tree $s_1[q, q_1]$, \$ appears in the tree q. By similar arguments, we obtain that x_1 appears in s_2 and $\sharp$ appears in the tree q. Hence q contains both \$ and $\sharp$, and thus $\sharp$ appears in the tree $s_1[q, q_1] = f(g^n(\$), \$)$. A contradiction. □

Lemma 4.2.10 *$DT \not\subseteq LDT^R \circ H$*

Proof. First, we introduce a notation. Let τ be a tree transformation from T_Σ to T_Δ and $L \subseteq T_\Sigma$ be a tree language. Then

$$\tau(L) = \{s \in T_\Delta \mid \text{ there exists } t \in L \text{ such that } (t, s) \in \tau\}.$$

We can generalize this notation. Let C be a class of tree transformations and T be a class of tree languages. Then

$$C(T) = \{\tau(L) \mid \tau \in C \text{ and } L \in T\}.$$

Now, we finish the proof of this lemma by contradiction. Let us suppose that $DT \subseteq LDT^R \circ H$. Then,

$$DT(REC) \subseteq H(LDT^R(REC)).$$

By the Corollary 6.7 in Chapter IV of [40], $LDT^R(REC) = REC$.
Hence,

$$DT(REC) \subseteq H(REC).$$

It is a contradiction because $H(REC) \subset DT(REC)$ by the proof of Theorem 6.12 in Chapter IV of [40]. So, $DT \not\subseteq LDT^R \circ H$. □

Corollary 4.2.11 $LDT^R \circ H \subset DT^R$.

Proof. Since $LDT^R \subseteq DT^R$ and $H \subseteq DT^R$, by (i) of Proposition 4.2.3 we get that $LDT^R \circ H \subseteq DT^R$. The proper inclusion $LDT^R \circ H \subset DT^R$ follows from Lemma 4.2.10 and the inclusion $DT \subseteq DT^R$. □

Lemma 4.2.12 $LDB \not\subseteq DT^2$.

Proof. By the proof of Theorem 4.5 in [33] and by $(*)$ in [32], $dom(DT^2) = DREC$. It is well known that $dom(LDB) = REC$ and $DREC \subset REC$, see [40]. Hence $dom(DT^2) \subset dom(LDB)$ and in this way $LDB \not\subseteq DT^2$. □

Lemma 4.2.13 $LDT \not\subseteq DB$.

Proof. Let $\Sigma = \Sigma_0 \cup \Sigma_1$, where $\Sigma_0 = \{\sharp\}$, $\Sigma_1 = \{f\}$, and let $\Delta = \Delta_0 \cup \Delta_1$, where $\Delta_0 = \{\sharp, \$\}$, and $\Delta_1 = \{f\}$. Let $\rho \subseteq T_\Sigma \times T_\Delta$ be defined by $\rho = \{ (f^n(\sharp), f^n(\sharp)) \mid n \geq 0 \text{ is an even number} \} \cup \{ (f^n(\sharp), f^n(\$)) \mid n \geq 1 \text{ is an odd number} \}$.

Let the ldt $\mathcal{A} = (\Sigma, A, \Delta, a_0, R_\mathcal{A})$ be defined as follows. $A = \{a_0, a_1\}$ and $R_\mathcal{A}$ consists of the following four rules.

$$\begin{array}{lll} a_0(f(x)) & \to & f(a_1(x)), \\ a_1(f(x)) & \to & f(a_0(x)), \\ a_1(\sharp) & \to & \$, \\ a_0(\sharp) & \to & \sharp. \end{array}$$

It should be clear that $\tau_\mathcal{A} = \rho$. Thus $\rho \in LDT$. Let us suppose that $\rho \in DB$. Then there is a db $\mathcal{B} = (\Sigma, B, \Delta, B', R_\mathcal{B})$ such that $\tau_\mathcal{B} = \rho$. Let J bound the heights of the right-hand sides of the rules in $R_\mathcal{B}$. Let $n > J$ be an even number. Then $(f^n(\sharp), f^n(\sharp)) \in \rho$, and hence

$$f^n(\sharp) \Rightarrow^*_\mathcal{B} b_1(f^n(\sharp)) \qquad (*)$$

for some $b_1 \in B'$. As $(f^{n+1}(\sharp), f^{n+1}(\$)) \in \rho$,

$$f^{n+1}(\sharp) \underset{\mathcal{B}}{\overset{*}{\Rightarrow}} b_2(f^{n+1}(\$))$$

for some $b_2 \in B'$. By $(*)$

$$f^{n+1}(\sharp) \underset{\mathcal{B}}{\overset{*}{\Rightarrow}} f(b_1(f^n(\sharp))) \underset{\mathcal{B}}{\Rightarrow} b_2(f^{n+1}(\$)).$$

Hence the rule

$$f(b_1(x_1)) \to b_2(f^{n+1}(\$))$$

is in $R_{\mathcal{B}}$. However, this contradicts the assumption that J bounds the heights of the right-hand sides of the rules in $R_{\mathcal{B}}$. $\square$

Lemma 4.2.14 $H \not\subseteq LDT^R$.

Proof. Let $\Sigma = \Sigma_0 \cup \Sigma_1$, where $\Sigma_0 = \{ \sharp \}$, $\Sigma_1 = \{ f \}$, and let $\Delta = \Delta_0 \cup \Delta_2$, where $\Delta_0 = \{ \sharp \}$, $\Delta_2 = \{ g \}$. The binary balanced tree $p_n \in T_\Delta$ of height n is defined as follows. Let $p_0 = \sharp$, and for any $n > 0$, let $p_n = g(p_{n-1}, p_{n-1})$. Let $\rho \subseteq T_\Sigma \times T_\Delta$ be defined by

$$\rho = \{ (f^n(\sharp), p_n) \mid n \geq 0 \} \, .$$

Consider the th $\mathcal{A} = (\Sigma, A, \Delta, a_0, R)$, where $A = \{ a_0 \}$ and R consists of the following 2 rules.

$a_0(f(x_1)) \to g(a_0(x_1), a_0(x_1))$,
$a_0(\sharp) \to \sharp$.

It should be clear that $\rho = \tau_{\mathcal{A}}$. It is not hard to see that $\rho \notin LDT^R$. $\square$

Theorem 4.2.15 *The diagram in Figure 4.1 is an inclusion diagram for* $||K||$.

Proof. We divide the proof into three steps. In the first step we show that if a class Y is above another class Z and there is an edge between Y and Z, then $Z \subseteq Y$. We go through the edges in the diagram of Figure 4.1 in a top-down and left-to-right order and list all nontrivial inclusions corresponding to the edges. We add the proof of each inclusion in parenthesis.

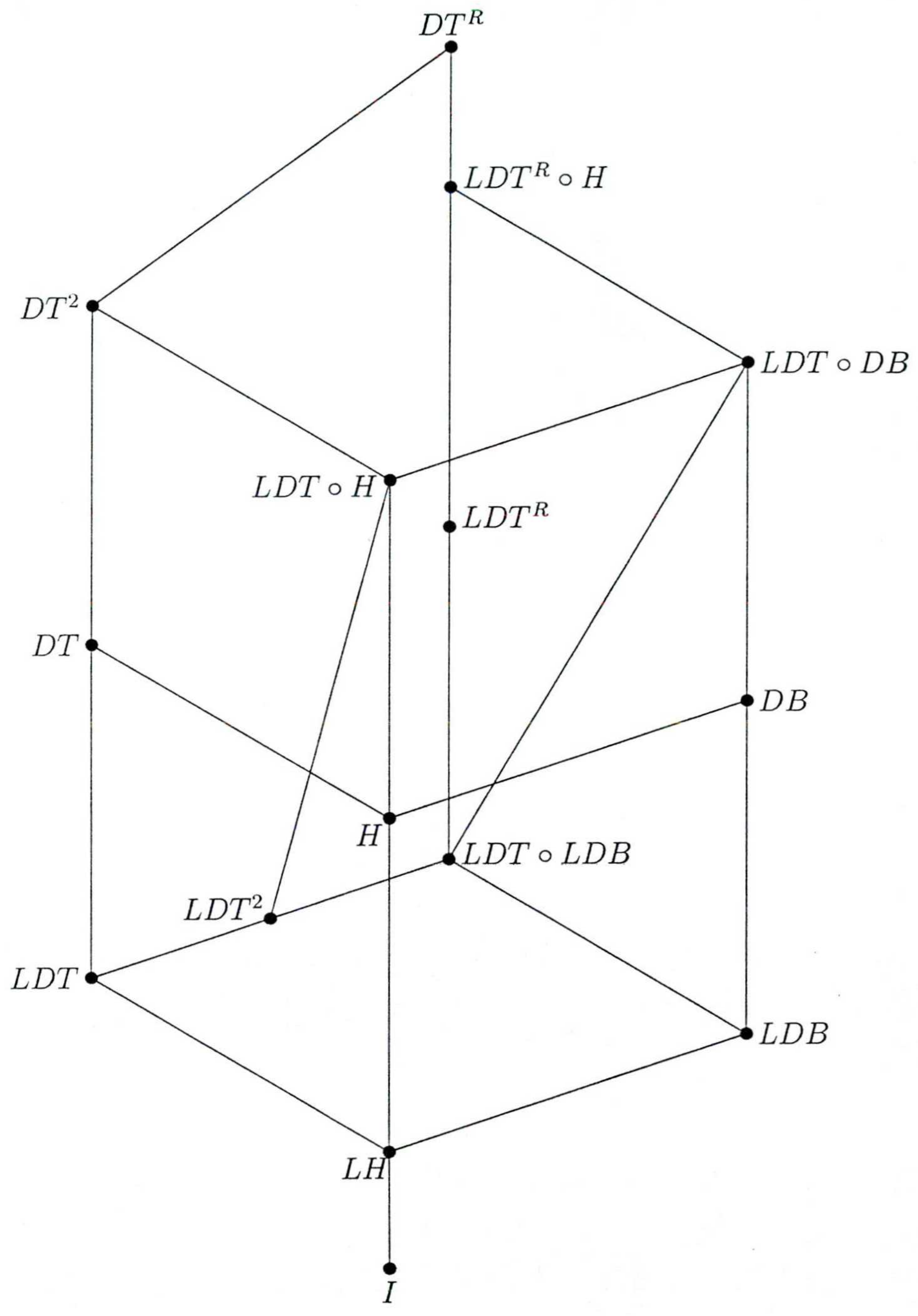

Figure 4.1: The inclusion diagram of the set $||K||$.

$$\begin{array}{rcll}
DT^2 & \subseteq & DT^R & \text{(by the inclusion } DT \subseteq DT^R \text{ and} \\
 & & & \text{by (i) of Proposition 4.2.3)} \\
LDT^R \circ H & \subseteq & DT^R & (LDT^R \subseteq DT^R, H \subseteq DT^R \text{ and} \\
 & & & \text{by (i) of Proposition 4.2.3)} \\
LDT \circ DB & \subseteq & LDT^R \circ H & (LDT \circ DB \overset{(13')}{=} LDT \circ LDB \circ H \subseteq \\
 & & & LDT^R \circ LDT \circ LDB \circ H \overset{(1')}{=} \\
 & & & LDT^R \circ LDB \circ H \overset{(2')}{=} LDT^R \circ H) \\
LDT^2 & \subseteq & LDT \circ H & (LDT^2 \overset{(9')}{=} LDT \circ LH \subseteq LDT \circ H) \\
LDT \circ LDB & \subseteq & LDT^R & \text{(by (}\dagger\text{), (}\ddagger\text{), and (ii) of Proposition 4.2.3)} \\
LDT^2 & \subseteq & LDT \circ LDB & (LDT^2 \overset{(9')}{=} LDT \circ LH \subseteq LDT \circ LDB)
\end{array}$$

In the second step, using the results of the first step, we show that if a class Y is above another class Z and there is an edge between Y and Z, then $Z \subset Y$. We again go through the edges in the diagram of Figure 4.1 in a top-down and left-to-right order, in each horizontal "virtual plane" clockwise. We add the proof of each inclusion in parenthesis. When citing known inclusion results, we ignore the original sources of some easy results and refer instead to the inclusion diagrams appearing in papers [34] and [24] to ensure a convenient progress. Moreover, we shall frequently show an inclusion $X \subset Y$ by referring to a noninclusion $V \not\subseteq Z$, where $V \subseteq Y$ and $X \subseteq Z$.

DT^2	$\subset$	DT^R	(by Theorem 5 in [32] and by Figure 1 in [33])
$LDT^R \circ H$	$\subset$	DT^R	(by Lemma 4.2.11)
$LDT \circ DB$	$\subset$	$LDT^R \circ H$	(by Lemma 4.2.9)
$LDT \circ H$	$\subset$	$LDT \circ DB$	(by Lemma 4.2.12)
$LDT \circ H$	$\subset$	DT^2	(by Figure 1 in [34])
DT	$\subset$	DT^2	(by Figure 1 in [34])
LDT^R	$\subset$	$LDT^R \circ H$	(by Lemma 4.2.14)
DB	$\subset$	$LDT \circ DB$	(by Lemma 4.2.13)
H	$\subset$	$LDT \circ H$	(by Figure 1 in [34])
H	$\subset$	DB	(by Figure 1 in [24])
H	$\subset$	DT	(by Figure 1 in [34])
LDT	$\subset$	DT	(by Figure 1 in [34])
LDT^2	$\subset$	$LDT \circ H$	(by Figure 1 in [34])
$LDT \circ LDB$	$\subset$	LDT^R	(by Lemma 4.2.9)
$LDT \circ LDB$	$\subset$	$LDT \circ DB$	(by Lemma 4.2.14)
LDB	$\subset$	DB	(by Figure 1 in [24])
LH	$\subset$	H	(by Figure 1 in [24])
LDT	$\subset$	LDT^2	(by Figure 1 in [34])
LDT^2	$\subset$	$LDT \circ LDB$	(by Lemma 4.2.12)
LDB	$\subset$	$LDT \circ LDB$	(by Lemma 4.2.13)
LH	$\subset$	LDB	(by Figure 1 in [24])
LH	$\subset$	LDT	(by Figure 1 in [24])
I	$\subset$	LH	(by Figure 1 in [24])

Finally, in the third step we show that if neither of the inclusions $Y \subseteq Z$ and $Z \subseteq Y$ is indicated by the diagram for classes Y and Z in K, then they are incomparable. We traverse the diagram in a top-down and left-to-right order as follows. The diagram is drawn as if its nodes, except the topmost and lowest ones, were fitting to three horizontal planes. We visit first the upper plane, then the middle one and finally the lower one. In each plane we proceed clockwise. For each element Y in K, when visiting Y, we compare Y with all elements Z in K such that there is neither an ascending nor a descending path between the Y and Z, and that Y and Z have not been compared yet. We add the proof of each incomparability in parenthesis.

DT^2	$\bowtie$	$LDT^R \circ H$	(by Lemma 4.2.10 and Lemma 4.2.12)
DT^2	$\bowtie$	$LDT \circ DB$	(by Lemma 4.2.10 and Lemma 4.2.12)
DT^2	$\bowtie$	LDT^R	(by Lemma 4.2.10 and Lemma 4.2.12)
DT^2	$\bowtie$	DB	(by Lemma 4.2.10 and Lemma 4.2.12)
DT^2	$\bowtie$	$LDT \circ LDB$	(by Lemma 4.2.10 and Lemma 4.2.12)
DT^2	$\bowtie$	LDB	(by Lemma 4.2.10 and Lemma 4.2.12)
$LDT^R \circ H$	$\bowtie$	DT	(by Lemma 4.2.12 and Lemma 4.2.10)
$LDT \circ DB$	$\bowtie$	DT	(by Lemma 4.2.12 and Lemma 4.2.10)
$LDT \circ DB$	$\bowtie$	LDT^R	(by Lemma 4.2.14 and Lemma 4.2.9)
$LDT \circ H$	$\bowtie$	DT	(by Figure 1 in [34])
$LDT \circ H$	$\bowtie$	LDT^R	(by Lemma 4.2.14 and Lemma 4.2.12)
$LDT \circ H$	$\bowtie$	DB	(by Lemma 4.2.13 and Lemma 4.2.12)
$LDT \circ H$	$\bowtie$	$LDT \circ LDB$	(by Lemma 4.2.14 and Lemma 4.2.12)
$LDT \circ H$	$\bowtie$	LDB	(by Lemma 4.2.13 and Lemma 4.2.12)
DT	$\bowtie$	LDT^R	(by Lemma 4.2.10 and Lemma 4.2.12)
DT	$\bowtie$	DB	(by Lemma 4.2.13 and Lemma 4.2.12)
DT	$\bowtie$	LDT^2	(by Figure 1 in [34])
DT	$\bowtie$	$LDT \circ LDB$	(by Lemma 4.2.10 and Lemma 4.2.12)
DT	$\bowtie$	LDB	(by Lemma 4.2.13 and Lemma 4.2.12)
LDT^R	$\bowtie$	DB	(by Lemma 4.2.13 and Lemma 4.2.14)
LDT^R	$\bowtie$	H	(by Lemma 4.2.12 and Lemma 4.2.14)
DB	$\bowtie$	LDT	(by Lemma 4.2.14 and Lemma 4.2.13)
DB	$\bowtie$	LDT^2	(by Lemma 4.2.14 and Lemma 4.2.13)
DB	$\bowtie$	$LDT \circ LDB$	(by Lemma 4.2.14 and Lemma 4.2.13)
H	$\bowtie$	LDT	(by Figure 1 in [34])
H	$\bowtie$	LDT^2	(by Figure 1 in [34])
H	$\bowtie$	$LDT \circ LDB$	(by Lemma 4.2.14 and Lemma 4.2.13)
H	$\bowtie$	LDB	(by Figure 1 in [24])
LDT	$\bowtie$	LDB	(by Lemma 4.2.13 and Lemma 4.2.12)
LDT^2	$\bowtie$	LDB	(by Lemma 4.2.13 and Lemma 4.2.12)

□

In the light of Theorem 4.2.15, the following result is obtained by direct inspection of the inclusion diagram of Figure 4.1.

Consequence 4.2.16 *For any $u, v \in K$, $||u|| = ||v||$ if and only if $u = v$.*

4.2.4 The main result

The main results are easy consequences of the work presented in the previous subsections.

Theorem 4.2.17 $\leftrightarrow^*_T = \leftrightarrow^*_S = \rho$.

Proof. By Theorem 4.2.6 it is sufficient to show that $\rho \subseteq \leftrightarrow^*_S$. Let $w, w' \in M^*$ be such that $||w|| = ||w'||$. Then, by Theorem 4.2.8, there exist $z, z' \in K$ such that $w \rightarrow^*_S z$ and $w' \rightarrow^*_S z'$. By Theorem 4.2.6, $||w|| = ||z||$ and $||w'|| = ||z'||$, hence we have $||z|| = ||z'||$. By Consequence 4.2.16, $z = z'$. Therefore we get $w \leftrightarrow^*_S w'$. □

Theorem 4.2.18 $[M] = ||K||$.

Proof. Obviously $||K|| \subseteq ||M^*|| = [M]$. To show that $||M^*|| \subseteq ||K||$, let $w \in M^*$. By Theorem 4.2.8 there exists $z \in K$ such that $w \rightarrow^*_S z$. By Theorem 4.2.6 $||w|| = ||z||$. Hence $||w|| \in ||K||$. □

Finally we turn to the decision problem of the inclusion in $[M]$.

Theorem 4.2.19 *There is a linear time algorithm which for any tree transformation classes $Y_1, \ldots, Y_m, Z_1, \ldots, Z_n \in M$ decides which one of the following four mutually exclusive conditions holds.*

$$\begin{array}{llcl} (i) & Y_1 \circ \ldots \circ Y_m & = & Z_1 \circ \ldots \circ Z_n, \\ (ii) & Y_1 \circ \ldots \circ Y_m & \subset & Z_1 \circ \ldots \circ Z_n, \\ (iii) & Z_1 \circ \ldots \circ Z_n & \subset & Y_1 \circ \ldots \circ Y_m, \\ (iv) & Y_1 \circ \ldots \circ Y_m & \bowtie & Z_1 \circ \ldots \circ Z_n. \end{array}$$

Proof. By Theorem 4.2.8 we find in linear time $y, z \in K$ with $Y_1 \ldots Y_m \rightarrow^*_S y$ and $Z_1 \ldots Z_n \rightarrow^*_S z$. Then by Theorem 4.2.6, we also have $||Y_1 \ldots Y_m|| = ||y||$ and $||Z_1 \ldots Z_n|| = ||z||$. Thus, one of the conditions (i)-(iv) holds for $Y_1 \circ \ldots \circ Y_m$ and $Z_1 \circ \ldots \circ Z_n$ if and only if the corresponding (i')-(iv') holds for $||y||$ and $||z||$, where

$$\begin{array}{llcl} (i') & ||y|| & = & ||z||, \\ (ii') & ||y|| & \subset & ||z||, \\ (iii') & ||z|| & \subset & ||y||, \\ (iv') & ||y|| & \bowtie & ||z||. \end{array}$$

By Figure 4.1 it is decidable in constant time which one of (i')-(iv') holds. □

Conclusions and further research topics

We have introduced the notion of the generalized semi-monadic rewriting system, which is a generalization of well-known rewriting systems: the ground rewriting system, the monadic rewriting system, and the semi-monadic rewriting system. We have shown that lgsm rewriting systems effectively preserve recognizability. We have shown that a tree language L is recognizable if and only if there exists a rewriting system R such that $R \cup R^{-1}$ is an lgsm rewriting system and that L is the union of finitely many $\leftrightarrow_R^*$-classes. We have presented several decidability and undecidability results on gsm rewriting systems.

We have given a simple proof for the decidability of the injectivity problem of linear deterministic top-down tree transducers. Moreover, we have shown that the injectivity problem is undecidable even for homomorphism tree transducers.

Finally, we have shown that $DT^R = DT \circ LDB$, $LDT^R \not\subseteq LDT \circ DB$, and $DT \not\subseteq LDTR \circ H$. Using these results and the composition and inclusion results of Engelfriet, Fülöp, and Fülöp and Vágvölgyi, we have given a linear time algorithm to determine the correct inclusion relationship between two tree transformation classes which are compositions of some "fundamental" tree transformation classes taken from the set $\{DT^R, LDT^R, DT, LDT, DB, LDB, H, LH\}$.

Our results give rise to several open problems.

- Generalize lgsm rewriting systems such that the obtained rewriting systems still effectively preserve recognizability.
- Let R_1 and R_2 be rewriting systems effectively preserving recogniz-

ability (lgsm rewriting systems, respectively) over Σ. Is it decidable if $\rightarrow^*_{R_1} \cap (T_\Sigma \times T_\Sigma) \subseteq \rightarrow^*_{R_2} \cap (T_\Sigma \times T_\Sigma)$? Is it decidable if $\leftrightarrow^*_{R_1} \cap (T_\Sigma \times T_\Sigma) \subseteq \leftrightarrow^*_{R_2} \cap (T_\Sigma \times T_\Sigma)$?

- Let R be a rewriting system effectively preserving recognizability. Is it decidable if R is left-to-right minimal? Is it decidable if R is two-way minimal? Is it decidable if R is left-to-right ground minimal? Is it decidable if R is two-way ground minimal? The last two questions are also open if R is an lgsm rewriting system.

- Dauchet and his colleagues [11], [12] have shown that for a ground rewriting system R, it is decidable if R is confluent and it is decidable if R is noetherian. Give subclasses $\mathbf{C}_1$ and $\mathbf{C}_2$ of lgsm rewriting systems which contain the class of ground rewriting systems such that for any rewriting system $R \in \mathbf{C}_1$ it is decidable if R is noetherian and that for any rewriting system $R \in \mathbf{C}_2$, it is decidable if R is confluent.

- A rewriting system R over Σ is tame if for all critical pairs (u, v) of R

  (i) $R^*(\{u\}) \cup R^*(\{v\})$ is finite,

  (ii) for each $w \in R^*(\{u\}) \cup R^*(\{v\})$, $w \rightarrow^+_R w$ does not hold, and

  (iii) for any $u' \in R^*(\{u\})$ and $v' \in R^*(\{v\})$, there is a $z \in T_\Sigma(X)$ such that $u' \rightarrow^*_R z$ and $v' \rightarrow^*_R z$.

  If R effectively preserves recognizability, then it is decidable if R is tame. If R is convergent, then R is tame as well. It would be worth while studying tame rewriting systems preserving recognizability.

Acknowledgements

I am grateful to Zoltán Fülöp for his expert guidance and valuable suggestions during the course of this work. I am also indebted to Sándor Vágvölgyi for enlightening discussions, counsel and help. I also wish to express my thanks to Gábor Dányi and László Bernátsky for their very useful comments on a draft version of this dissertation.

Összefoglaló (Summary in Hungarian)

A term átíró rendszerek fontos szerepet játszanak az elméleti számítástudományban. Nagyon hasznosnak bizonyultak például a λ-kalkulus, a denotációs szemantika, az automatikus tételbizonyítás és a szimbólikus algebrai számítások kutatásánál.

Ebben a disszertációban term átíró rendszerekre és fatranszformátorokra vonatkozó néhány eldönthetetlenségi kérdést vizsgálunk meg.

A disszertáció első részében regularitást megőrző term átíró rendszereket vizsgálunk. Egy term átíró rendszer megőrzi a regularitást, ha tetszőleges reguláris fanyelv esetén, ezen reguláris fanyelvben lévő fákból képzett leszármazottak halmaza is reguláris. Egy fának egy leszármazottja úgy keletkezik, hogy néhányszor alkalmazzuk a fára a term átíró rendszer szabályait.

Először definiáljuk az általánosított szemi-monadikus term átíró rendszer fogalmát, amely tartalmazza a korábban már ismert regularitást megőrző term átíró rendszereket, nevezetesen a ground, a monadikus és a szemi-monadikus term átíró rendszereket. Ezután, megmutatjuk, hogy a lineáris általánosított szemi-monadikus term átíró rendszerek effektíven megőrzik a regularitást. (Ez azt jelenti, hogy meg tudjuk adni a leszármazottak fanyelvét felismerő faautomatát is, a kiinduló reguláris fanyelvet felismerő faautomatából és a term átíró rendszer szabályaiból.) Az eredmény segítségével adunk egy új jellemzését a felismerhető fanyelveknek.

A továbbiakban bebizonyítunk néhány eldönthetőségi és eldönthetetlenségi eredményt a regularitást effektíven megőrző term átíró rendszerekre és az általánosított szemi-monadikus term átíró rendszerekre vonatkozóan. Többek között megmutatjuk, hogy a regularitást effektíven megőrző tetszőleges term átíró rendszerről eldönthető, hogy lokálisan konfluens-e.

A fenti fogalmakat és eredményeket átvisszük a sztring átíró rendszerekre is, mivel egy sztring átíró rendszer tekinthető úgy, mint egy speciális term átíró rendszer.

A disszertáció további részében fatranszformátorokra vonatkozó eldönthetőségi kérdéseket vizsgálunk meg. A fatranszformátorok számos típusát definiálták és vizsgálták a 70-es évek eleje óta. Mi a determinisztikus top-down, a determinisztikus bottom-up és a determinisztikus regulárís-előrenézésű top-down fatranszformátorokat tekintjük.

Először adunk egy egyszerű bizonyítást arra, hogy a lineáris determinisztikus top-down fatranszformátorok injektivitási problémája eldönthető. Ezután bebizonyítjuk, hogy nem lineáris esetben az injektivitási probléma már a homomorfizmus fatranszformátorok esetében is eldönthetetlen.

A fatranszformátorok által indukált fatranszformációk bináris relációk fák felett, így a fatranszformációk kompozíciója, amelyet $\circ$-rel jelölünk, a szokásos módon definiálható. Megmutatunk néhány kompozíciós és tartalmazási eredményt fatranszformáció osztályokra, amelyek közül a legfontosabbak: $DT^R = DT \circ LDB$, $LDT^R \not\subseteq LDT \circ DB$ és $DT \not\subseteq LDT^R \circ H$. Itt DB jelöli a determinisztikus bottom-up, DT a determinisztikus top-down, H pedig a homomorfizmus fatranszformációk osztályát. Az L prefix jelenti a linearitást és az R felsőindex pedig a reguláris-előrenézést.

Ezen eredmények és a korábban ismert kompozíciós és tartalmazási eredmények (főleg J. Engelfriet, Fülöp Zoltán és Vágvölgyi Sándor eredményei) felhasználásával adunk egy lineáris idejű algoritmust, amelynek segítségével tetszőleges két, a $\{ DT^R, LDT^R, DT, LDT, DB, LDB, H, LH \}$ halmazból vett fatranszformáció osztályokból kompozícióval képzett, osztály közötti tartalmazás eldönthető.

Végül, összegezzük a disszertáció eredményeit és mutatunk néhány megoldatlan problémát a regularitást megőrző term átíró rendszerekre vonatkozóan.

A disszertáció eredményei a következő három cikkben jelentek meg:

- Z. Fülöp, P. Gyenizse, On injectivity of deterministic top-down tree transducers, *Information Processing Letters*, **48** (1993) 183-188.

- P. Gyenizse, S. Vágvölgyi, Composition of Deterministic Bottom-up, Top-down, and Regular Look-ahead Tree Transformations, *Theoretical Computer Science*, **156** (1996) 71-97.

- P. Gyenizse, S. Vágvölgyi, Linear Generalized Semi-monadic Rewrite Systems Effectively Preserve Recognizability, *Theoretical Computer Science*, **188** (1997), megjelenés alatt.

Szeretnék köszönetet mondani témavezetőmnek, Dr. Fülöp Zoltánnak, áldozatos munkájáért, hasznos útmutatásaiért és értékes tanácsaiért. Szeretném megköszönni Dr. Vágvölgyi Sándornak a sok szakmai konzultációt és a felbecsülhetetlen segítséget, valamint Dányi Gábornak és Bernátsky Lászlónak a disszertáció elolvasása közben felmerült észrevételeiket és megjegyzéseiket.

Index

Bibliography

[1] B.S. Baker, Tree transducers and families of tree languages, *Annual ACM STC,* (1973) 200-206.

[2] B.S. Baker, Tree transducers and tree languages, *Information and Control*, **37** (1978) 241-266.

[3] B.S. Baker, Composition of top-down and bottom-up tree transduction, *Information and Control*, **41** (1979) 186-213.

[4] R.V. Book and F.Otto, *String-Rewriting Systems* (Springer Verlag, New-York, 1993).

[5] W.S. Brainerd, Tree generating regular systems, *Information and Control*, **14** (1969) 217-231.

[6] A.C. Caron and J.L. Coquidé, Decidability of reachability for disjoint union of term rewriting systems, *Theoretical Computer Science*, **126** (1994) 31-52.

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

[8] G. Dányi and Z. Fülöp, Superlinear deterministic top-down tree transducers, *Mathematical Systems Theory*, **29** (1996) 507-534.

[9] G. Dányi and Z. Fülöp, Compositions with superlinear deterministic top-down tree transducers, *Theoretical Computer Science*, accepted for publications.

[10] M. Dauchet, It is undecidable whether a finite set of trees is a code, *Technical Report, Laboratorie d'Informatique Fundamentalle De Lille,* Universite de Lille, **109** (1987).

[11] M. Dauchet, P. Heuillard, P. Lescanne, and S. Tison, Decidability of the Confluence of Finite Ground Term Rewrite Systems and of Other Related Term Rewrite Systems, *Information and Control,* **88** (1990) 187-201.

[12] M. Dauchet and S. Tison, The Theory of Ground Rewrite Systems is Decidable, *Proc. 5th IEEE Symposium on Logic in Computer Science,* Philadelphia (1990) 242-248.

[13] N. Dershowitz and J.P. Jouannaud, Rewrite Systems, *Handbook of Theoretical Computer Science,* (J.V. Leeuwen ed.) North-Holland (1990), Vol. B 243-320.

[14] J. Engelfriet, Bottom-up and top-down tree transformations – a comparison, *Mathematical Systems Theory,* **9** (1975) 198-231.

[15] J. Engelfriet, Top-down tree transducers with regular look-ahead, *Mathematical Systems Theory,* **10** (1977) 289-303.

[16] J. Engelfriet, Derivation trees of ground term rewriting systems, *Technical Report,* Leiden University, **96-25** (1996) 1-16.

[17] J. Engelfriet and H. Vogler, Macro tree transducers, *Journal of Computational System Sciences,* **31** (1985) 71-146.

[18] J. Engelfriet and H. Vogler, High level tree transducers and iterated push-down tree transducers, *Acta Informatica,* **26** (1988) 131-192.

[19] J. Engelfriet and H. Vogler, Modular tree transducers, *Theoretical Computer Science,* **78** (1991) 267-303.

[20] Z. Ésik, On decidability of injectivity of tree transformations, *Proc. 3th Coll. Lille, Les arbres en algebre et en programmation,* Lille, (1978) 107-133.

[21] Z. Ésik, Decidability results concerning tree transducers I., *Acta Cybernetica,* **5** (1980) 1-20.

[22] Z. Ésik, Decidability results concerning tree transducers II., *Acta Cybernetica*, **6** (1983) 303-314.

[23] Z. Fülöp, On attributed tree transducers, *Acta Cybernetica*, **5** (1981) 261-279.

[24] Z. Fülöp, A complete description for a monoid of deterministic bottom-up tree transformation classes, *Theoretical Computer Science*, **88** (1991) 253-268.

[25] Z. Fülöp, Undecidable properties of deterministic top-down tree transducers, *Theoretical Computer Science*, **134** (1994) 311-328.

[26] Z. Fülöp and P. Gyenizse, On injectivity of deterministic top-down tree transducers, *Information Processing Letters*, **48** (1993) 183-188.

[27] Z. Fülöp, F. Herrmann, S. Vágvölgyi, and H. Vogler, Tree transducers with external functions, *Theoretical Computer Science*, **108** (1993) 185-236.

[28] Z. Fülöp and S. Vágvölgyi, Results on compositions of deterministic root-to-frontier tree transformations, *Acta Cybernetica*, **8** (1987) 49-61.

[29] Z. Fülöp and S. Vágvölgyi, On ranges of compositions of deterministic root-to-frontier tree transformations, *Acta Cybernetica*, **8** (1988) 259-266.

[30] Z. Fülöp and S. Vágvölgyi, A finite presentation for a monoid of tree transformation classes, in *Proc. 2nd Conference on Automata, Languages and Programming Systems* (Eds. F. Gécseg and I. Peák) ,Dept. of Mathematics, University of Economics (1988), 115-124.

[31] Z. Fülöp and 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.

[32] Z. Fülöp and S. Vágvölgyi, Top-down tree transducers with deterministic top-down look-ahead, *Information Processing Letters*, **33** (1989) 3-5.

[33] Z. Fülöp and S. Vágvölgyi, Variants of top-down tree transducers with look-ahead, *Mathematical Systems Theory*, **21** (1989) 125-145.

[34] Z. Fülöp and S. Vágvölgyi, A complete rewriting system for a monoid of tree transformation classes, *Information and Computation*, **86** (1990) 195-212.

[35] Z. Fülöp and S. Vágvölgyi, A complete classification of deterministic root-to-frontier tree transformation classes, *Theoretical Computer Science*, **81** (1991) 1-15.

[36] Z. Fülöp and S. Vágvölgyi, Decidability of the inclusion in monoids generated by tree transformation classes, *Tree Automata and Languages*, (M. Nivat and A. Podelski eds.) Elsevier Science Publishers B.V. (1992), Amsterdam, pp. 381-408.

[37] Z. Fülöp and H. Vogler, *Formal Models of Syntax-Directed Semantics*, monography, 1997, submitted for publication.

[38] J.H. Gallier and R.V. Book, Reductions in tree replacement systems *Theoretical Computer Science*, **37** (1985) 123-150.

[39] F. Gécseg, Tree transformations preserving recognizability, *Finite Algebra and Multiple-valued Logic,* North-Holland, Amsterdam, (1981) 251-273.

[40] F. Gécseg and M. Steinby, *Tree Automata*, Akadémiai Kiadó, Budapest, 1984.

[41] F. Gécseg and M. Steinby, Tree Languages, *Handbook of Formal Language Theory*, (G. Rozenberg and A. Salomaa eds.) to be issued by Springer Verlag in 1996.

[42] R. Gilleron, Decision problems for term rewriting systems and recognizable languages, Proc. STACS'91, *Lecture Notes in Computer Science*, **480** (1991) 148-159.

[43] R. Gilleron and S. Tison, Regular tree languages and rewrite systems, *Fundamenta Informaticae*, **24** (1995) 157-175.

[44] P. Gyenizse and S. Vágvölgyi, Composition of Deterministic Bottom-up, Top-down, and Regular Look-ahead Tree Transformations, *Theoretical Computer Science*, **156** (1996) 71-97.

[45] P. Gyenizse and S. Vágvölgyi, Linear Generalized Semi-monadic Rewrite Systems Effectively Preserve Recognizability, *Theoretical Computer Science*, **194** (1998) 87-122.

[46] G. Huet, Confluent Reductions: Abstract Properties and Applications to Term Rewriting Systems, *Journal of ACM*, **27** (1980) 797-821.

[47] J.W. Klop, Term Rewriting Systems, *Handbook of Logic in Computer Science*, (S. Abramsky, Dov M. Gabbay, and T.S.E. Maibaum eds.), Clarendon Press, Oxford (1992) Vol. 2. 1-116.

[48] D. Kozen, Complexity of finitely presented algebras, *Proc. 9th Annual ACM Symp. on Theory of Computing*, Boulder Colorado (1977) 164-177.

[49] A. Kühnemann and H. Vogler, Synthesized and inherited functions - a new computational model for syntax-directed semantics, *Acta Informatica*, **31** (1994) 431-477.

[50] J.W. Lloyd, *Foundations of Logic Programming, Second, Extended Edition* (Springer Verlag, Berlin 1993).

[51] F. Otto, On the property of preserving regularity for string-rewriring systems, Proceedings of RTA'97,(H. Common ed.), *Lecture Notes in Computer Science*, **1232** (1997) 83-97.

[52] M. Oyamaguchi, The Church-Rosser property for ground term rewriting systems is decidable, *Theoretical Computer Science*, **49** (1987) 43-79.

[53] W.C. Rounds, Mappings on grammars and trees, *Mathematical Systems Theory*, **4** (1970) 257-287.

[54] K. Salomaa, Deterministic tree pushdown automata and monadic tree rewriting systems, *Journal of Computer and System Science*, **37** (1988) 367-394.

[55] G. Slutzki and S. Vágvölgyi, A Hierarchy of Deterministic Top-down Tree Transformations, in Proc. FCT'93, *Lecture Notes In Computer Science*, Vol. 710, Springer-Verlag, (1993) Berlin pp. 440-451; also Mathematical Systems Theory.

[56] J.W. Thatcher, Generalized sequential machine maps, *Journal of Computational System Science*, **4** (1970) 339-367.

[57] J.W. Thatcher, Tree automata: an informal survey, *Currents in the Theory of Computing*, (A.V. Aho ed.) Prenttice-Hall (1973), 143-172.

[58] S. Vágvölgyi and Z. Fülöp, An infinite hierarchy of tree transformations in the class NDR, *Acta Cybernetica*, **8** (1987) 153-168.

[59] H. Vogler, High level modular tree transducers, *Institutes for Information Processing, Report Nr.* **260**, (H.J. Pongratz and W. Schinnerl eds.), Graz University of Technology (1988).