TREE AUTOMATA

FERENC GÉCSEG

Bolyai Institute
József Attila University
Szeged, Hungary

MAGNUS STEINBY

Department of Mathematics
University of Turku
Turku, Finland

PREFACES

Preface to the Second Edition

When the present book was written in the early 1980s, the theory of tree automata, tree languages and tree transformations was young but already quite extensive. Our aim was to give a systematic and mathematically sound exposition of some central parts of this subject. The presentation uses universal algebra in the spirit of J. R. Büchi and J. B. Wright from whose ideas of automata as algebras tree automata once emerged. That the algebraic formalism encourages and supports precise definitions and rigorous proofs may explain why the book has remained a general reference for many mathematically minded workers in the field ever since its publication in 1984. Unfortunately, it has long been out of print and hard to obtain.

Soon after the regrettable death of Ferenc Gécseg in October 2014, Zoltán Fülöp (Szeged) and Heiko Vogler (Dresden) proposed a reissue of this book. Akadémiai Kiadó, the original publisher, did not find the project feasible but gave us free hands to proceed on our own. Professor Gécseg's family also willingly endorsed the idea. Since the book did not exist in any electronic form, the whole text had to be retyped in Latex. For this exacting task Fülöp and Vogler quickly assembled a highly qualified team that, besides themselves, included Johanna Björklund (Umeå), Frank Drewes (Umeå), Zsolt Gazdag (Budapest), Eija Jurvanen (Turku), Andreas Maletti (Stuttgart), George Rahonis (Thessaloniki), Kai Salomaa (Kingston, Ontario), and Sándor Vágvölgyi (Szeged). Professors Fülöp and Vogler also undertook the overall management of the work. The generous contributions of all these individuals are acknowledged with many thanks.

From the very beginning it was decided that this new edition should be true to the original one. In particular, the terminology was preserved even in cases in which some alternative terms have become prevailing. However, a few mistakes were corrected and a couple of obscure passages were clarified.

Of course, the book was never claimed to offer a complete presentation of its subject matter. In fact, some important topics were totally left out. It was hoped that the extensive bibliography, fairly complete up to around 1982, and the notes and references at the end of each chapter would, at least partly, make up for the shortcomings. Now, over thirty years later, the incompleteness is naturally even more obvious. Much progress has been made in already established areas and many new topics have emerged. Some of the new work is strongly motivated by applications, old or new. No book of this size could do justice to all these developments. Instead, we have to trust that the matters presented here still belong to the core of the theory and are worth studying by anyone who wants to work in this field. Moreover, to account for more recent contributions and lines of research, an appendix has been added to the book. In it several topics are briefly surveyed and some relevant references are given to help an interested reader get started on them. I thank Heiko Vogler and Zoltán Fülöp for some important additions to the bibliography.

Turku
August 2015

Magnus Steinby

Preface to the Original Edition

The purpose of this book is to give a mathematically rigorous presentation of the theory of tree automata, recognizable forests, and tree transformations. Apart from its intrinsic interest this theory offers some new perspectives to various parts of mathematical linguistics. It has also been applied to some decision problems of logic, and it provides tools for syntactic pattern recognition. We have not even tried to discuss all aspects of the subject or any of the applications, but enough central material has been included to give the reader a firm basis for further studies. Being relatively new and very manyfaceted, the field still lacks a uniform widely accepted formalism. We have chosen the language of universal algebra as our vehicle of presentation. However, we have not assumed that the reader is familiar with universal algebra; the preparatory sections in Chapter 1 should make the book self-contained in this respect. On the other hand, it is natural to assume that anyone interested in such a book has some general mathematical training and some knowledge of finite automata and formal languages.

The book consists of four chapters, a bibliography and an index. The first chapter contains an exposition of the necessary universal algebra and lattice theory, as well as a quick review of finite automata and formal languages. We also recommend some books on these subjects. In Chapter 2 trees, forests, tree recognizers, tree grammars, and some operations on forests are introduced. Several characterizations and closure properties of recognizable forests are presented. Chapter 3 is devoted to the connections between recognizable forests and context-free languages. Chapter 4 deals with tree transducers and tree transformations. Chapters 2–4 contain some exercises. Each of these chapters is concluded with some historical and bibliographical comments. We also point out some topics not discussed in the book. We have tried to make the Bibliography as complete as possible. Of course, it has not always been easy to decide whether a given item should be included or not.

We want to thank our colleagues and the staffs at our institutions for the good working atmosphere in which this book was written. Dr. András Ádám and Professor István Peák gave the text a careful scrutiny. We gratefully acknowledge their many remarks. We are also indebted to Dr. Zoltán Ésik for his very helpful comments on Chapter 4. We wish to express our warmest thanks to Mrs. Piroska Folberth for performing very competently the difficult task of typing the manuscript. Also, we want to thank our wives and daughters for their support and for putting so gracefully up with the inconveniences inevitably caused by our undertaking.

The writing of the book has involved several trips between Turku and Szeged. We gratefully acknowledge the financial support provided by the Academy of Finland, the

Hungarian Academy of Sciences, the János Bolyai Mathematical Society, the University of Szeged, and the University of Turku. Our work was also furthered by a possibility for the first-named author to spend a term at the Tampere University of Technology. For this thanks are due Professor Timo Lepistö.

Contents

NOTES TO THE READER

Within each section, there is one counter which is incremented by each of the environments definition, lemma, theorem, corollary, and example. The end of a proof or an example is indicated by the mark $\square$. It appears immediately after a theorem, lemma or corollary if this is not followed by a proof. The references to the literature are by the author(s) and the number with which the publication occurs in the Bibliography. In a few cases we refer to a book mentioned at the end of Chapter 1.

1 PRELIMINARIES

In this chapter we shall review some basic concepts and results from the theories of automata, formal languages, and universal algebras. It is reasonable to assume that a potential reader of this book already knows something about automata and formal languages. On the other hand, we do not presuppose any knowledge of universal algebra. These two assumptions suggested the styles and extents of the following seven sections.

Section 1.1 (Sets, relations and mappings) may be skimmed through for terminology and notation.

Sections 1.2 and 1.3 present the required universal algebraic concepts and results. These are not many, but they should be mastered well as the very basic concepts of the theory of tree automata are defined in terms of universal algebra. We have tried to make the book self-contained in this respect, but a reader who wants to pursue further the algebraic aspects of the theory should certainly consult one of the references on universal algebra.

The lattice theory presented in Section 1.4 is less important here, and the reading of this section may be postponed until needed.

Sections 1.5, 1.6 and 1.7 survey some of the most essential facts about finite recognizers, regular languages, context-free grammars, and (generalized) sequential machines. A reader less familiar with these matters would do wisely to look up these subjects in some of the references given at the end of the chapter.

1.1 SETS, RELATIONS AND MAPPINGS

The set theory needed here is very elementary and most of our set theoretic notation is well-known. However, a few conventions should be pointed out:

(i) $A \subseteq B$ means that the set A is a subset of the set B. Proper inclusion is denoted by $A \subset B$.

(ii) $\emptyset$ denotes the empty set.

(iii) $|A|$ denotes the cardinality of the set A.

(iv) The *power set* of a set A, i.e., the set of all subsets of A, is denoted by $\mathfrak{p}A$.

(v) The union of a family $(A_i \mid i \in I)$ of subsets (indexed by I) of some set is written as $\bigcup(A_i \mid i \in I)$. Similarly, $\bigcap(A_i \mid i \in I)$ is the intersection.

(vi) The set $\{x \in A \mid P_1(x), \ldots, P_k(x)\}$ of all elements x in A with the properties P_1, $\ldots$, P_k may also be written as $\{x \mid P_1(x), \ldots, P_k(x)\}$ when A is understood from the context. We shall use this notation in the following more general form, too. Suppose $f(x_1, \ldots, x_m)$ is an object defined in some way in terms of the objects x_1, $\ldots$, x_m. Then

$$\{f(x_1, \ldots, x_m) \mid P(x_1, \ldots, x_m)\}$$

is the set of all such objects constructed from objects x_1, $\ldots$, x_m satisfying the condition $P(x_1, \ldots, x_m)$. Furthermore, we use

$$\{f_1(x_1, \ldots, x_m), \ldots, f_k(x_1, \ldots, x_m) \mid P(x_1, \ldots, x_m)\}$$

as a short form for the union

$$\{f_1(x_1, \ldots, x_m) \mid P(x_1, \ldots, x_m)\} \cup \cdots \cup \{f_k(x_1, \ldots, x_m) \mid P(x_1, \ldots, x_m)\}.$$

(vii) If there is no danger of confusion, we may write simply a for the one-element set $\{a\}$. Of course, we should not write $\emptyset$ for $\{\emptyset\}$.

Sometimes we employ some notation from logic as abbreviations:

(i) "$(\forall x \in A)\, P(x)$" states that $P(x)$ holds for all $x \in A$.

(ii) "$(\exists x \in A)\, P(x)$" states that there exists an x in A such that $P(x)$ holds.

(iii) "$P \Longrightarrow Q$" means that Q holds if P holds.

(iv) "$P \Longleftrightarrow Q$" states that the conditions P and Q are equivalent, i.e., both of them hold or then neither one holds.

(v) "$P \wedge Q$" is the statement that both P and Q hold. Similarly, "$P \vee Q$" states that at least one of P and Q holds.

The numbers dealt with here are always integers and mostly even non-negative integers. When we write "$\ldots$ for all $n \geq 1$" we mean, in fact, "$\ldots$ for all integers $n \geq 1$". The set of all integers is denoted by $\mathbf{Z}$, the set of the natural numbers 1, 2, $\ldots$ by $\mathbf{N}$, and the set of all non-negative integers by $\mathbf{N_0}$.

Let A and B be sets and $\varrho \subseteq A \times B$ a (binary) *relation* from A to B. The fact that $(a, b) \in \varrho$ ($a \in A$, $b \in B$) is also expressed by writing $a \varrho b$ or $a \equiv b\, (\varrho)$. The opposite case may be expressed by $a \not\varrho\, b$ or by $a \not\equiv b\, (\varrho)$. For any $a \in A$, we put

$$a\varrho = \{b \in B \mid a \varrho b\}.$$

This notation is extended to subsets of A:

$$A_1\varrho = \bigcup (a\varrho \mid a \in A_1) \quad \text{for } A_1 \subseteq A.$$

The *converse* of ϱ is the relation

$$\varrho^{-1} = \{(b, a) \mid (a, b) \in \varrho\} \subseteq B \times A.$$

Obviously,

$$b\varrho^{-1} = \{a \in A \mid a\varrho b\}$$

and

$$B_1\varrho^{-1} = \{a \in A \mid (\exists b \in B_1)a\varrho b\}$$

for all $b \in B$ and $B_1 \subseteq B$. The *domain* of ϱ is the subset $\mathrm{dom}(\varrho) = B\varrho^{-1}$ of A, and its *range* is the subset $\mathrm{range}(\varrho) = A\varrho$ of B.

The *product* or *composition* of two relations $\varrho \subseteq A \times B$ and $\tau \subseteq B \times C$ is the relation

$$\varrho \circ \tau = \{(a, c) \mid (\exists b \in B)a\varrho b\tau c\} \subseteq A \times C.$$

In this definition we used the short form $a\varrho b\tau c$ to express the fact that $a\varrho b$ and $b\tau c$. Often we write $\varrho\tau$ for $\varrho \circ \tau$. The product of relations is associative. We note also the equality $(\varrho \circ \tau)^{-1} = \tau^{-1} \circ \varrho^{-1}$.

Consider now (binary) relations on a set A, i.e. subsets of $A \times A$. These include the *diagonal relation* $\delta_A = \{(a, a) \mid a \in A\}$ and the *total relation* $\iota_A = A \times A$. For any relation ϱ on A we define the *powers* ϱ^n $(n \geq 0)$ with respect to the product of relations:

$$1^\circ \quad \varrho^0 = \delta_A \quad \text{and}$$
$$2^\circ \quad \varrho^{n+1} = \varrho^n \circ \varrho \quad \text{for } n \geq 0.$$

The relation $\varrho \subseteq A \times A$ is called

(a) *reflexive* if $\delta_A \subseteq \varrho$,

(b) *symmetric* if $\varrho^{-1} \subseteq \varrho$,

(c) *antisymmetric* if $\varrho \cap \varrho^{-1} \subseteq \delta_A$ and

(d) *transitive* if $\varrho^2 \subseteq \varrho$.

The intersection of any reflexive relations (on a given A) is reflexive, and the intersection of transitive relations is transitive. Thus there exists for every $\varrho \subseteq A \times A$ a unique minimal reflexive, transitive relation ϱ^* containing ϱ. It is called the *reflexive, transitive closure* of ϱ. One verifies easily that

$$\varrho^* = \delta_A \cup \varrho \cup \varrho^2 \cup \varrho^3 \cup \ldots,$$

i.e., for any a, $b \in A$ we have $a\varrho^* b$ iff

$$a = a_1 \, \varrho \, a_2 \, \varrho \, a_3 \, \ldots \, a_{n-1} \, \varrho \, a_n = b$$

for some $n \geq 1$ and $a_1, \ldots, a_n \in A$.

A relation on A is called an *equivalence relation* on A, if it is reflexive, symmetric and transitive. The set of all equivalence relations on A is denoted by $E(A)$. Clearly, $\delta_A \in E(A)$ and $\iota_A \in E(A)$. Let ϱ be an equivalence relation on A. The *ϱ-class* (or the *equivalence class* modulo ϱ) of an element $a \in A$ is the set $a\varrho$. Obviously, $a\varrho b$ iff

$a\varrho = b\varrho$. We shall also write a/ϱ for $a\varrho$ and extend this notation to subsets $A_1 \subseteq A$ and n-tuples $\mathbf{a} = (a_1, \ldots, a_n)$ of elements of A ($n \geq 1$): $A_1/\varrho = \{a/\varrho \mid a \in A_1\}$ and $\mathbf{a}/\varrho = (a_1/\varrho, \ldots, a_n/\varrho)$. The *quotient set* of A modulo ϱ is A/ϱ. Obviously, A/ϱ is a partition on A, that is, every element of A belongs to exactly one ϱ-class. On the other hand, every partition on A can be obtained this way as the quotient set from a unique equivalence relation and there is a natural one-to-one correspondence between the partitions on A and $E(A)$. The cardinality of A/ϱ is called the *index* of $\varrho \in E(A)$. If $|A/\varrho|$ is finite, we say that ϱ is of *finite index*. We say that $\varrho \in E(A)$ *saturates* the subset $H \subseteq A$ if $H\varrho = H$, i.e., if H is the union of some ϱ-classes.

A *mapping* or a *function* from a set A to a set B is a triple (A, B, φ), where $\varphi \subseteq A \times B$ is a relation such that for every $a \in A$ there exists exactly one $b \in B$ satisfying $a\varphi b$. As usual we write $\varphi\colon A \to B$ and say that φ is a mapping from A to B. If $a\varphi b$ ($a \in A$, $b \in B$), b is called the *image* of a and a an *inverse image* of b. This is expressed by writing $b = a\varphi$, $b = \varphi(a)$ or $\varphi\colon a \mapsto b$. For a subset A_1 of A we also use the two notations $A_1\varphi$ and $\varphi(A_1)$ for the set $\{a\varphi \mid a \in A_1\}$. The converse φ^{-1} of φ is always defined as a relation ($\subseteq B \times A$), but it is usually not a mapping from B to A. Again, $\varphi^{-1}(B_1)$ will sometimes be used instead of $B_1\varphi^{-1}$ when $B_1 \subseteq B$. Note that $\mathrm{dom}(\varphi) = A$ and $\mathrm{range}(\varphi) \subseteq B$. The set of all mappings from A to B is denoted by B^A.

The *composition* or *product* of two mappings $\varphi\colon A \to B$ and $\psi\colon B \to C$ is the mapping

$$\varphi\psi\colon A \to C$$

where $\varphi\psi$ is the product of φ and ψ as relations. Clearly, $a\varphi\psi = (a\varphi)\psi$ for all $a \in A$.

The *restriction* of a mapping $\varphi\colon A \to B$ to a subset C of A is the mapping

$$\varphi|C\colon C \to B$$

where $\varphi|C = \varphi \cap (C \times B)$. If $\psi\colon C \to B$ is obtained from $\varphi\colon A \to B$ as the restriction of φ to C, i.e., $C \subseteq A$ and $\psi = \varphi|C$, then we say also that φ is an *extension* of ψ to A.

The *kernel* $\varphi\varphi^{-1}$ of a mapping $\varphi\colon A \to B$ is an equivalence relation on A and $a_1 \equiv a_2\,(\varphi\varphi^{-1})$ iff $a_1\varphi = a_2\varphi$ (a_1, $a_2 \in A$). On the other hand, one can associate with every $\theta \in E(A)$ a mapping

$$\theta^\natural\colon A \to A/\theta, \quad a \mapsto a\theta, \quad (a \in A)$$

such that the kernel of $\theta^\natural$ is θ. This $\theta^\natural$ is called the *natural mapping* associated with θ.

A mapping $\varphi\colon A \to B$ is called

(i) *injective* (or an *injection*), if $\varphi\varphi^{-1} = \delta_A$,

(ii) *surjective* (or a *surjection*), if $\mathrm{range}(\varphi) = B$, and

(iii) *bijective* (or an *bijection*), if it is injective and surjective.

If $\varphi\colon A \to B$ is surjective, one says also that φ is a mapping of A *onto* B. It is obvious that the natural mapping $\theta^\natural$ is always surjective ($\theta \in E(A)$). The diagonal relation of a set A defines the *identity mapping* $A \to A$, $a \mapsto a$ ($a \in A$). It is denoted by 1_A.

We shall also meet partial mappings, that is, mappings for which the image of some elements may be undefined. A *partial mapping* from A to B is defined by a relation $\varphi \subseteq A \times B$ such that $|a\varphi| \leq 1$ for all $a \in A$. Again, we write $\varphi\colon A \to B$. If $a\varphi = \emptyset$, then we say that φ is *undefined* for a ($a \in A$). The notations and terminology introduced above for mappings apply to partial mappings, too, although $\mathrm{dom}(\varphi)$ may be a proper subset of A when $\varphi\colon A \to B$ is a partial mapping.

It is convenient to think of the elements of a cartesian product $A_1 \times \cdots \times A_n$ as n-tuples $(a_1, \ldots, a_n)$ with $a_1 \in A_1, \ldots, a_n \in A_n$. We adopt the definition of an ordinal number n as the set of all ordinals smaller that n: $0 = \emptyset$, $1 = \{0\}$, $2 = \{0, 1\}$ etc. and, in general, $n = \{0, 1, \ldots, n-1\}$. Then $A_1 \times \cdots \times A_n$ can also be defined as the set of all mappings

$$\varphi\colon n \to A_1 \cup \cdots \cup A_n$$

such that $i\varphi \in A_{i+1}$ for $i = 0, 1, \ldots, n-1$. Of course, we may identify such a φ with the n-tuple $(0\varphi, 1\varphi, \ldots, (n-1)\varphi)$. Now the cartesian power $A^n = A \times \cdots \times A$ (n times) is the set of all mappings $\varphi\colon n \to A$. In particular, $A^0 = \{\emptyset\}$ since $\emptyset$ is the only mapping from $\emptyset$ to A. Note that the notation A^n is consistent with our earlier notation B^A for the set of all mappings from A to B.

We shall also need countably infinite sequences of elements. Let $\omega = \{0, 1, 2, \ldots\}$ be the smallest infinite ordinal and A any set. The elements of A^ω are called *ω-sequences.* Thus an ω-sequence of elements of A is a mapping

$$\varphi\colon \omega \to A$$

which we may also write as

$$(0\varphi, 1\varphi, \ldots, n\varphi, \ldots)_{n<\omega}.$$

We conclude the section by considering operations. These are special mappings and are among the most fundamental concepts of algebra. Let $m \geq 0$. An *m-ary operation* on a set A is a mapping from A^m to A. If $\varphi\colon A^m \to A$ is an m-ary operation on A, then φ assigns to every m-tuple $(a_1, \ldots, a_m)$ of elements of A a unique element of A which we write as $\varphi(a_1, \ldots, a_m)$. The number m is called the *arity* or the *rank* of φ. Most operations encountered in the usual algebraic systems (groups, rings, lattices etc.) have rank 0, 1 or 2. A few comments on these special cases:

(i) A 0-ary operation $\varphi\colon \{\emptyset\} \to A$ is completely determined by its only image $\varphi(\emptyset)$, and often φ is given simply by naming this element. Note that here $\emptyset$ may also be seen as the empty sequence of elements, and often one writes $\varphi(\ )$, or just φ, for $\varphi(\emptyset)$.

(ii) When $m = 1$, we have a mapping from A to itself. Such operations are called *unary.*

(iii) An operation of rank 2 is called a *binary operation.* For example, the addition and the multiplication in a ring are binary operations. In most such concrete examples one uses the *infix notation* for binary operations. Thus it is customary to write the ring operations in the form $a + b$ and $a \cdot b$ instead of $+(a, b)$ and $\cdot(a, b)$, respectively.

A *partial m-ary operation* on a set A is a partial mapping from A^m to A. For any partial m-ary operation $\varphi\colon A^m \to A$ and subset B of A we have a partial mapping

$$\varphi|B\colon B^m \to B,$$

where $\varphi|B = \varphi \cap (B^m \times B)$. If φ is an operation and B is *closed* with respect to φ, i.e., $\varphi(a_1, \ldots, a_m) \in B$ whenever $a_1, \ldots, a_m \in B$, then $\varphi|B$ is an m-ary operation on B called the *restriction* of φ to B. Often the same symbol is used to denote an operation and its restrictions.

Suppose we are given a set A, k m-ary operations $\varphi_1, \ldots, \varphi_k$ on A and a k-ary operation ψ on A (m, $k \geq 0$). The *composition* of $\varphi_1, \ldots, \varphi_k$ with ψ is the m-ary operation $\psi(\varphi_1, \ldots, \varphi_k)$ defined so that

$$\psi(\varphi_1, \ldots, \varphi_k)(a_1, \ldots, a_m) = \psi(\varphi_1(a_1, \ldots, a_m), \ldots, \varphi_k(a_1, \ldots, a_m))$$

for all $a_1, \ldots, a_m \in A$. Note that the possibilities $k = 0$ or $m = 0$ are included. If $k = 0$, then the composition is an m-ary operation with the constant image $\psi(\emptyset)$. If $m = 0$, then the composition is a 0-ary operation with the single value $\psi(\varphi_1(\emptyset), \ldots, \varphi_k(\emptyset))$.

Let φ be an m-ary operation on a set A and $A_1, \ldots, A_m$ any subsets of A. Then we write

$$\varphi(A_1, \ldots, A_m) = \{\varphi(a_1, \ldots, a_m) \mid a_1 \in A_1, \ldots, a_m \in A_m\}.$$

Thus φ is extended to an m-ary operation on the power set $\mathfrak{p}A$. In general, there is no need to introduce a new notation for this extension.

1.2 UNIVERSAL ALGEBRAS

In this and the next section some concepts and results from universal algebra are surveyed. Universal algebra is an extensive field of mathematics, but we need really just certain basic parts of it. On the other hand, a good grasp of the material of these sections is essential to an understanding of the rest of the book.

Generally speaking, an *algebra* (or a *universal algebra*) is a set together with a set of operations on this set. There may be a finite or an infinite number of operations, but we insist that they all are *finitary*, i.e., the ranks are finite as in the definition of operations given in the previous section. As a first example we consider the algebra of subsets of a given set U. In the power set $\mathfrak{p}U$ we have several naturally defined operations. For example, there is a binary operation $\cup$ that forms the union $A \cup B$ of any two A, $B \in \mathfrak{p}U$. Similarly, we have the binary operation $\cap$ that forms the intersection of two subsets of U. A unary operation is obtained if we map every $A \in \mathfrak{p}U$ to its complement $A^c = U - A$. Furthermore, we introduce two 0-ary operations, one that has $\emptyset$ and one that has U as its image. Of course, an infinite number of operations could be defined on $\mathfrak{p}U$, but if we restrict ourselves to those defined above, we get the algebra

$$(\mathfrak{p}U, \cup, \cap, {}^c, \emptyset, U)$$

with two binary, one unary and two 0-ary operations. Note that we get such an algebra for each set U. In fact, all of these algebras can be viewed as special instances of a general class of algebras known as *Boolean algebras.*

The example brings forth an important point. In algebra, and this will be the case here, too, one is generally not interested just in individual algebras, but rather in whole classes of algebras. Algebras in such a class are all "similar" in the sense that there is a natural correspondence between the operations of any two algebras of the class. Such a correspondence of operations is needed when one defines any concept, such as homomorphisms or direct products, involving more than one algebra. For example, the multiplications of any two groups correspond to each other, and a homomorphism of groups should preserve the multiplication. We shall now introduce a convenient vehicle to define such a class of similar algebras.

Definition 1.2.1 An *operator domain* is a set Σ together with a mapping

$$r\colon \Sigma \to \mathbf{N_0}$$

that assigns to every $\sigma \in \Sigma$ an *arity*, or *rank*, $r(\sigma)$. For any $m \geq 0$,

$$\Sigma_m = \{\sigma \in \Sigma \mid r(\sigma) = m\}$$

is the set of the *m-ary operators* (or *operational symbols*).

From now on Σ is an operator domain. The mapping r is usually not mentioned, but we denote by $r(\Sigma)$ the set of all $m \geq 0$ such that $\Sigma_m \neq \emptyset$. One can write Σ as the disjoint union $\Sigma_0 \cup \Sigma_1 \cup \Sigma_2 \cup \ldots$ from which the empty sets will be omitted.

Definition 1.2.2 A *Σ-algebra* $\mathcal{A}$ is a pair consisting of a nonempty set A (of elements of $\mathcal{A}$) and a mapping that assigns to every operator $\sigma \in \Sigma$ an m-ary operation

$$\sigma^{\mathcal{A}}\colon A^m \to A,$$

where m is the arity of σ. The operation $\sigma^{\mathcal{A}}$ is called the *realization* of σ in $\mathcal{A}$. The mapping $\sigma \mapsto \sigma^{\mathcal{A}}$ will not be mentioned explicitly, but we write $\mathcal{A} = (A, \Sigma)$. The Σ-algebra $\mathcal{A}$ is *finite* if A is finite, and it is of *finite type* if Σ is finite. When Σ is not specified, or not emphasized, we speak simply about *"algebras"*. An algebra with just one element is called *trivial.*

In general, $\mathcal{A} = (A, \Sigma)$, $\mathcal{B} = (B, \Sigma)$ and $\mathcal{C} = (C, \Sigma)$, possibly equipped with subscripts, will be Σ-algebras. The realizations of an operator $\sigma \in \Sigma$ in these algebras are denoted by $\sigma^{\mathcal{A}}$, $\sigma^{\mathcal{B}}$ and $\sigma^{\mathcal{C}}$, respectively.

In the previous example of subset algebras we would have $\Sigma = \Sigma_0 \cup \Sigma_1 \cup \Sigma_2$ with (for example) $\Sigma_0 = \{0, 1\}$, $\Sigma_1 = \{\neg\}$ and $\Sigma_2 = \{\wedge, \vee\}$. The algebra of the subsets of a set U is then the Σ-algebra $\mathcal{A}$, where $A = \mathfrak{p}U$ and the operators are realized as follows: $0^{\mathcal{A}} = \emptyset$, $1^{\mathcal{A}} = U$, $\neg^{\mathcal{A}} = {}^c$ (complement in U), $\wedge^{\mathcal{A}} = \cap$ (intersection) and $\vee^{\mathcal{A}} = \cup$ (union).

Note that the possibility $m = 0$ is not excluded when we consider generally an m-ary operation. For $\sigma \in \Sigma_0$ one often writes $\sigma^{\mathcal{A}}$ instead of $\sigma^{\mathcal{A}}(\ )$ or $\sigma^{\mathcal{A}}(\emptyset)$ (this involves the harmless confusion of a 0-ary operation and its value). When $\Sigma = \{\sigma_1, \ldots, \sigma_k\}$ is finite, one usually writes $\mathcal{A} = (A, \sigma_1, \ldots, \sigma_k)$ instead of $\mathcal{A} = (A, \Sigma)$.

We introduce now several concepts related to algebras.

Definition 1.2.3 The Σ-algebra $\mathcal{B}$ is a *subalgebra* of the Σ-algebra $\mathcal{A}$ if $B \subseteq A$ and $\sigma^{\mathcal{B}} = \sigma^{\mathcal{A}}|B$ for all $\sigma \in \Sigma$.

If $\mathcal{B}$ is a subalgebra of $\mathcal{A}$, then B is a *closed subset* of $\mathcal{A}$, i.e., $\sigma^{\mathcal{A}}(b_1, \ldots, b_m) \in B$ for all $\sigma \in \Sigma_m$ ($m \geq 0$) and $b_1, \ldots, b_m \in B$. For every nonempty closed subset B of $\mathcal{A}$, there is exactly one way to realize the operators on B in such a way that we get a subalgebra $\mathcal{B}$ of $\mathcal{A}$: obviously every $\sigma^{\mathcal{B}}$ should be the restriction $\sigma^{\mathcal{A}}|B$ of the corresponding operation of $\mathcal{A}$ to B. Hence, a subalgebra is completely determined by its set of elements and one may call this subset a subalgebra. If σ is a 0-ary operator, then every subalgebra of $\mathcal{A}$ contains the element $\sigma^{\mathcal{A}}$. If Σ_0 is empty, then $\emptyset$ is a closed subset, but we do not count it among the subalgebras.

It is easy to see that the intersection of any family of closed subsets of a given algebra $\mathcal{A}$ is again closed. Thus we have for any $H \subseteq A$ a unique minimal closed subset containing H:

$$[H] = \bigcap (B \mid H \subseteq B \subseteq A, B \text{ closed}).$$

If $H \neq \emptyset$ or $\Sigma_0 \neq \emptyset$, then $[H]$ is also nonempty and thus a subalgebra. It is called the *subalgebra generated* by H. If $\Sigma_0 = \emptyset$, then $[\emptyset] = \emptyset$. A *generating set* of $\mathcal{A}$ is a subset $H \subseteq A$ such that $[H] = A$ and $\mathcal{A}$ is said to be *finitely generated* if it has a finite generating set. It is clear that every finite algebra is finitely generated.

Definition 1.2.4 A *homomorphism* from a Σ-algebra $\mathcal{A}$ to a Σ-algebra $\mathcal{B}$ is a mapping $\varphi \colon A \to B$ such that for all $m \geq 0$, $\sigma \in \Sigma_m$ and $a_1, \ldots, a_m \in A$,

$$\sigma^{\mathcal{A}}(a_1, \ldots, a_m)\varphi = \sigma^{\mathcal{B}}(a_1\varphi, \ldots, a_m\varphi).$$

We write then $\varphi \colon \mathcal{A} \to \mathcal{B}$. This homomorphism is called

(a) an *epimorphism*, if φ is surjective,

(b) a *monomorphism*, if φ is injective, and

(c) an *isomorphism*, if φ is bijective.

If there exists an epimorphism from $\mathcal{A}$ to $\mathcal{B}$, then $\mathcal{B}$ is said to be an *epimorphic image* of $\mathcal{A}$. A monomorphism is also called an *embedding*. If there is an isomorphism from $\mathcal{A}$ to $\mathcal{B}$, then $\mathcal{A}$ and $\mathcal{B}$ are *isomorphic* and we write $\mathcal{A} \cong \mathcal{B}$. Homomorphisms are often also called *morphisms*.

If $\mathcal{A} \cong \mathcal{B}$, then $\mathcal{A}$ and $\mathcal{B}$ are the same algebra from the abstract point of view. An easy computation shows that the composition $\varphi\psi$ of two homomorphisms $\varphi \colon \mathcal{A} \to \mathcal{B}$ and $\psi \colon \mathcal{B} \to \mathcal{C}$ is a homomorphism from $\mathcal{A}$ to $\mathcal{C}$.

A homomorphism is a mapping that is compatible with the operations of the algebras. For example, let $\mathcal{A} = (\mathbf{Z}, +)$ be the algebra of the integers with the usual addition as the only operation, $n \geq 1$ and $\mathcal{B} = (\mathbf{Z}_n, +)$ the algebra where $\mathbf{Z}_n = \{0, 1, \ldots, n-1\}$ and the sum is formed modulo n. Then the mapping $\varphi\colon \mathbf{Z} \to \mathbf{Z}_n$ that maps every $a \in \mathbf{Z}$ to its remainder $r_n(a)$ modulo n ($0 \leq r_n(a) < n$) is an epimorphism from $\mathcal{A}$ to $\mathcal{B}$. Of course, the homomorphisms defined in group theory, lattice theory etc. provide further general examples.

The proof of the following lemma is straightforward and thus it is omitted.

Lemma 1.2.5 *Let $\varphi\colon \mathcal{A} \to \mathcal{B}$ be a homomorphism. If $\mathcal{C}$ is a subalgebra of $\mathcal{A}$, then $C\varphi$ is a subalgebra of $\mathcal{B}$. If $\mathcal{D}$ is a subalgebra of $\mathcal{B}$ and $D\varphi^{-1}$ is nonempty, then $D\varphi^{-1}$ is a subalgebra of $\mathcal{A}$.* □

The following lemma contains an important observation.

Lemma 1.2.6 *Let $\varphi\colon \mathcal{A} \to \mathcal{B}$ and $\psi\colon \mathcal{A} \to \mathcal{B}$ be two homomorphisms and H a generating set of $\mathcal{A}$. If $\varphi|H = \psi|H$, then $\varphi = \psi$. In other words, a homomorphism is completely determined by its restriction to a generating set.*

Proof. Let $C = \{a \in A \mid a\varphi = a\psi\}$. Then $H \subseteq C$ by the assumption. If $m \geq 0$, $\sigma \in \Sigma_m$ and $a_1, \ldots, a_m \in C$, then $\sigma^{\mathcal{A}}(a_1, \ldots, a_m) \in C$:

$$\sigma^{\mathcal{A}}(a_1, \ldots, a_m)\varphi = \sigma^{\mathcal{B}}(a_1\varphi, \ldots, a_m\varphi) = \sigma^{\mathcal{B}}(a_1\psi, \ldots, a_m\psi) = \sigma^{\mathcal{A}}(a_1, \ldots, a_m)\psi.$$

Hence C is closed and we get $C = A$. This implies $\varphi = \psi$. □

We define now two concepts closely related to homomorphisms, namely congruences and quotient algebras.

Definition 1.2.7 A *congruence* (relation) of $\mathcal{A}$ is an equivalence relation on A which is invariant with respect to all operations $\sigma^{\mathcal{A}}$ ($\sigma \in \Sigma$). A relation $\varrho \subseteq A \times A$ is said to be *invariant* with respect to an m-ary operation $f\colon A^m \to A$ if

$$f(a_1, \ldots, a_m) \equiv f(b_1, \ldots, b_m) \ (\varrho)$$

for all elements $a_1, \ldots, a_m, b_1, \ldots, b_m \in A$ such that

$$a_1 \equiv b_1, \ \ldots, \ a_m \equiv b_m \ (\varrho).$$

The set of all congruences of an algebra $\mathcal{A}$ is denoted by $C(\mathcal{A})$.

Every algebra $\mathcal{A}$ has at least the trivial congruences δ_A and ι_A. For $\varrho \in C(\mathcal{A})$, the ϱ-class $a\varrho$ of an element $a \in A$ is also called a *congruence class* (modulo ϱ). The partition A/ϱ of A defined by the congruence classes is *compatible* in the sense that for all $m \geq 0$, $\sigma \in \Sigma_m$ and $a_1\varrho, \ldots, a_m\varrho \in A/\varrho$ there is a class $a\varrho$ such that

$$\sigma^{\mathcal{A}}(a_1\varrho, \ldots, a_m\varrho) \subseteq a\varrho.$$

Obviously, we can choose $a = \sigma^{\mathcal{A}}(a_1, \ldots, a_m)$. It is also easy to see that an equivalence relation $\varrho \in E(A)$ is a congruence of $\mathcal{A}$ only in case A/ϱ is a compatible partition. In fact, in automata theory it is usual to deal with compatible partitions (also called SP partitions) rather than with congruences, but both concepts convey the same idea.

The fact that A/ϱ is a compatible partition for any $\varrho \in C(\mathcal{A})$ also justifies the following definition; the operations are well-defined.

Definition 1.2.8 The *quotient algebra* $\mathcal{A}/\varrho = (A/\varrho, \Sigma)$ of a Σ-algebra $\mathcal{A}$ by a congruence $\varrho \in C(\mathcal{A})$ is defined as follows. For any $m \geq 0$, $\sigma \in \Sigma_m$ and $a_1, \ldots, a_m \in A$ we put

$$\sigma^{\mathcal{A}/\varrho}(a_1\varrho, \ldots, a_m\varrho) = \sigma^{\mathcal{A}}(a_1, \ldots, a_m)\varrho.$$

The definition of $\sigma^{\mathcal{A}/\varrho}$ may be explained as follows. To compute $\sigma^{\mathcal{A}/\varrho}(a_1\varrho, \ldots, a_m\varrho)$ one takes a representative from each of the ϱ-classes, say $a_1, \ldots, a_m$, computes $\sigma^{\mathcal{A}}$ for the representatives and forms then the ϱ-class of the resulting element.

Homomorphisms, congruences and quotient algebras are closely related to each other as the following three theorems show.

Theorem 1.2.9 *For any $\varrho \in C(\mathcal{A})$, the natural mapping $\varrho^\natural \colon a \mapsto a\varrho$ is an epimorphism $\mathcal{A} \to \mathcal{A}/\varrho$ (the* natural homomorphism*).*

Proof. We know that $\varrho^\natural$ is a surjection from A to A/ϱ so it suffices to verify that it is a homomorphism: for all $m \geq 0$, $\sigma \in \Sigma_m$ and $a_1, \ldots, a_m \in A$,

$$\sigma^{\mathcal{A}}(a_1, \ldots, a_m)\varrho^\natural = \sigma^{\mathcal{A}}(a_1, \ldots, a_m)\varrho = \sigma^{\mathcal{A}/\varrho}(a_1\varrho, \ldots, a_m\varrho) = \sigma^{\mathcal{A}/\varrho}(a_1\varrho^\natural, \ldots, a_m\varrho^\natural).$$ □

Theorem 1.2.10 *The kernel $\varphi\varphi^{-1}$ of any homomorphism $\varphi \colon \mathcal{A} \to \mathcal{B}$ is a congruence of $\mathcal{A}$.*

Proof. Consider any $m \geq 0$, $\sigma \in \Sigma_m$ and elements $a_1, \ldots, a_m, a_1', \ldots, a_m' \in A$ such that

$$a_1 \equiv a_1', \ \ldots, \ a_m \equiv a_m' \ (\varphi\varphi^{-1}).$$

Then $a_1\varphi = a_1'\varphi, \ldots, a_m\varphi = a_m'\varphi$, which implies $\sigma^{\mathcal{A}}(a_1, \ldots, a_m)\varphi = \sigma^{\mathcal{B}}(a_1\varphi, \ldots, a_m\varphi) = \sigma^{\mathcal{B}}(a_1'\varphi, \ldots, a_m'\varphi) = \sigma^{\mathcal{A}}(a_1', \ldots, a_m')\varphi$. This means that $\sigma^{\mathcal{A}}(a_1, \ldots, a_m) \equiv \sigma^{\mathcal{A}}(a_1', \ldots, a_m')$ $(\varphi\varphi^{-1})$ as required. □

Theorem 1.2.11 *Every epimorphic image of an algebra $\mathcal{A}$ is isomorphic to some quotient algebra of $\mathcal{A}$.*

Proof. Let $\varphi \colon \mathcal{A} \to \mathcal{B}$ be an epimorphism and $\theta = \varphi\varphi^{-1}$ its kernel. We claim that $\mathcal{B} \cong \mathcal{A}/\theta$. The required isomorphism $\mathcal{A}/\theta \to \mathcal{B}$ is shown to be given by

$$\psi \colon a\theta \mapsto a\varphi \qquad (a \in A).$$

For any a_1, $a_2 \in A$,

$$\begin{aligned} a_1\theta\psi = a_2\theta\psi \quad &\text{iff} \quad a_1\varphi = a_2\varphi \\ &\text{iff} \quad a_1 \equiv a_2 \ (\theta). \end{aligned}$$

This shows that ψ is well-defined (i.e., $a\theta\psi$ is independent of the choice of the representative $a \in A$ of the θ-class $a\theta$) and injective. Since φ is surjective, it is clear that ψ is surjective, too. It remains to be shown that ψ is a homomorphism. Let $m \geq 0$, $\sigma \in \Sigma_m$ and $a_1, \ldots, a_m \in A$. Then

$$\begin{aligned} \sigma^{\mathcal{A}/\theta}(a_1\theta, \ldots, a_m\theta)\psi &= \sigma^{\mathcal{A}}(a_1, \ldots, a_m)\theta\psi \\ &= \sigma^{\mathcal{A}}(a_1, \ldots, a_m)\varphi \\ &= \sigma^{\mathcal{B}}(a_1\varphi, \ldots, a_m\varphi) \\ &= \sigma^{\mathcal{B}}(a_1\theta\psi, \ldots, a_m\theta\psi). \end{aligned}$$ □

Taken together, Theorems 1.2.9 and 1.2.11 say that the epimorphic images of an algebra are exactly its quotient algebras (when one does not distinguish between isomorphic algebras).

Next, direct products of algebras are introduced. We may restrict ourselves to the case of a finite number of factors.

Definition 1.2.12 The *direct product* of two Σ-algebras $\mathcal{A}$ and $\mathcal{B}$ is the Σ-algebra

$$\mathcal{A} \times \mathcal{B} = (A \times B, \Sigma),$$

where the operations are defined so that

$$\sigma^{\mathcal{A}\times\mathcal{B}}((a_1, b_1), \ldots, (a_m, b_m)) = (\sigma^{\mathcal{A}}(a_1, \ldots, a_m), \sigma^{\mathcal{B}}(b_1, \ldots, b_m))$$

for all $m \geq 0$, $\sigma \in \Sigma_m$ and $(a_1, b_1), \ldots, (a_m, b_m) \in A \times B$. The k^{th} $(k \geq 0)$ *direct power* $\mathcal{A}^k$ of the Σ-algebra $\mathcal{A}$ is defined inductively:

(i) $\mathcal{A}^0 = (\{\emptyset\}, \Sigma)$ is the trivial Σ-algebra.

(ii) $\mathcal{A}^{k+1} = \mathcal{A}^k \times \mathcal{A}$ for all $k \geq 0$.

It is easy to see that direct products are associative in the sense that $(\mathcal{A} \times \mathcal{B}) \times \mathcal{C} \cong \mathcal{A} \times (\mathcal{B} \times \mathcal{C})$ for all $\mathcal{A}$, $\mathcal{B}$ and $\mathcal{C}$. Both of these products can be written simply as $\mathcal{A} \times \mathcal{B} \times \mathcal{C}$ and their elements may be identified with the triples (a, b, c) with $a \in A$, $b \in B$ and $c \in C$. More generally, one can define the direct product $\mathcal{A}_1 \times \cdots \times \mathcal{A}_k$ of k $(k \geq 0)$ Σ-algebras as an algebra with $A_1 \times \cdots \times A_k$ as its set of elements and operations performed componentwise. It is easy to see that the projections

$$\pi_i \colon A_1 \times \cdots \times A_k \to A_i, \quad (a_1, \ldots, a_k) \mapsto a_i$$

$(i = 1, \ldots, k)$ are epimorphisms from $\mathcal{A}_1 \times \cdots \times \mathcal{A}_k$ to the respective factor algebras $\mathcal{A}_i$. Hence, every factor in a direct product is an epimorphic image of the direct product.

We shall also need the following, perhaps, less usual, way to construct a new algebra from a given one.

Definition 1.2.13 The *subset algebra* (or *power algebra*) $\mathfrak{p}\mathcal{A} = (\mathfrak{p}A, \Sigma)$ of a Σ-algebra $\mathcal{A}$ is defined as follows. If $m \geq 0$, $\sigma \in \Sigma_m$ and $H_1, \ldots, H_m \in \mathfrak{p}A$, then put

$$\sigma^{\mathfrak{p}\mathcal{A}}(H_1, \ldots, H_m) = \sigma^{\mathcal{A}}(H_1, \ldots, H_m).$$

Note that the singleton sets $\{a\}$ $(a \in A)$ form in $\mathfrak{p}\mathcal{A}$ a subalgebra isomorphic to $\mathcal{A}$. If $\Sigma_0 = \emptyset$, $\mathfrak{p}\mathcal{A}$ has the trivial subalgebra $\{\emptyset\}$.

We conclude this section with a simple example illustrating these constructions.

Example 1.2.14 Suppose Σ consists of one binary operator σ and a nullary operator γ. Let $\mathcal{A} = (\{a, b\}, \Sigma)$ be a Σ-algebra such that $\gamma^{\mathcal{A}} = a$ and $\sigma^{\mathcal{A}}(a, a) = \sigma^{\mathcal{A}}(a, b) = \sigma^{\mathcal{A}}(b, a) = a$, $\sigma^{\mathcal{A}}(b, b) = b$. Consider first the direct power $\mathcal{A}^2 = \mathcal{A} \times \mathcal{A}$. If we write aa for (a, a) etc., then $\gamma^{\mathcal{A}\times\mathcal{A}} = aa$ and $\sigma^{\mathcal{A}\times\mathcal{A}}$ is given by the following multiplication table:

$\sigma^{\mathcal{A}\times\mathcal{A}}$	aa	ab	ba	bb
aa	aa	aa	aa	aa
ab	aa	ab	aa	ab
ba	aa	aa	ba	ba
bb	aa	ab	ba	bb

Let us now construct the subset algebra. The value of the 0-ary operation is $\gamma^{\mathfrak{p}\mathcal{A}} = \{a\}$ and the operation $\sigma^{\mathfrak{p}\mathcal{A}}$ is given by the table below.

$\sigma^{\mathfrak{p}\mathcal{A}}$	$\emptyset$	$\{a\}$	$\{b\}$	$\{a,b\}$
$\emptyset$	$\emptyset$	$\emptyset$	$\emptyset$	$\emptyset$
$\{a\}$	$\emptyset$	$\{a\}$	$\{a\}$	$\{a\}$
$\{b\}$	$\emptyset$	$\{a\}$	$\{b\}$	$\{a,b\}$
$\{a,b\}$	$\emptyset$	$\{a\}$	$\{a,b\}$	$\{a,b\}$

$\square$

1.3 TERMS, POLYNOMIAL FUNCTIONS AND FREE ALGEBRAS

The concepts "term" and "polynomial function" are all-important in our modelling of the theory of tree automata. Let us consider an introductory example. An expression like $(x + y)(y + z)$, such expressions are called terms, represents in a natural manner a function of the three variables x, y, and z. Two things should be pointed out here. First of all, the term defines such a function in any algebra with operations denoted by the operators appearing in the term. In our case it could define, for example, a mapping $\mathbf{Z}^3 \to \mathbf{Z}$ or a mapping $\mathbf{R}^3 \to \mathbf{R}$ depending on whether the addition and multiplication are interpreted as those of integers or those of real numbers. Generally speaking, the terms are determined by the operator domain, but they define operations in all algebras with that operator domain. Secondly, we note that the term not only defines a function, but it also describes a way to compute its values from the values of the variables once the operations of the algebra in question are known. In fact, algebras can be viewed as devices that evaluate terms. When we interpret (in Chapter 2) terms as trees, the step from algebras to tree automata is not long.

From now on, X will be a set disjoint from the operator domain Σ. The elements of X are called *variables*. Other symbols used for sets of variables are Y and Z.

Definition 1.3.1 The set $F_\Sigma(X)$ of Σ-*terms in* X, or ΣX-*terms* for short, is defined as follows:

(i) $X \subseteq F_\Sigma(X)$,

(ii) $\sigma(t_1, \ldots, t_m) \in F_\Sigma(X)$ whenever $m \geq 0$, $\sigma \in \Sigma_m$ and $t_1, \ldots, t_m \in F_\Sigma(X)$, and

(iii) every ΣX-term can be obtained by applying the rules (i) and (ii) a finite number of times.

If σ is a 0-ary operator, then we get by rule (ii) the ΣX-term $\sigma(\ )$. It is convenient to write just σ for such a term. Then the definition of $F_\Sigma(X)$ may be reformulated as follows.

Definition 1.3.1' The set $F_\Sigma(X)$ of ΣX-terms is defined as follows:

(i) $X \cup \Sigma_0 \subseteq F_\Sigma(X)$,

(ii) $\sigma(t_1, \ldots, t_m) \in F_\Sigma(X)$ whenever $m > 0$, $\sigma \in \Sigma_m$ and $t_1, \ldots, t_m \in F_\Sigma(X)$, and

(iii) every ΣX-term can be obtained by applying the rules (i) and (ii) a finite number of times.

When Σ and X are unspecified or unemphasized, we shall speak simply about *terms*. The inductive definition of $F_\Sigma(X)$ suggests a useful method to deal with terms. It could be called *term induction*. If we want to define a property or quantity $c(t)$ for every ΣX-term t, it suffices

(i) to define $c(t)$ for all $t \in X$, and then

(ii) to give a rule how to determine $c(\sigma(t_1, \ldots, t_m))$ in terms of σ ($\in \Sigma_m$) and $c(t_1)$, $\ldots$, $c(t_m)$ ($m \geq 0$).

Sometimes the variation suggested by Definition 1.3.1' is more convenient: in (i) one defines $c(t)$ for $t \in \Sigma_0$, too, but in (ii) one can then restrict oneself to values $m > 0$. *Proofs by term induction* can be modelled according to the same pattern.

Note that $F_\Sigma(X)$ is empty iff $\Sigma_0 = X = \emptyset$. Since we do not want to consider this uninteresting case separately every time, we shall tacitly assume that always $\Sigma_0 \cup X \neq \emptyset$.

Example 1.3.2 Let $\Sigma = \Sigma_0 \cup \Sigma_1 \cup \Sigma_2$, where $\Sigma_0 = \{\mu\}$, $\Sigma_1 = \{\tau\}$ and $\Sigma_2 = \{\sigma\}$. If $X = \{x, y, z\}$, then x, z, μ, $\tau(z)$, $\tau(\mu)$, $\sigma(z, \tau(\mu))$ and $t = \sigma(x, \sigma(z, \tau(\mu)))$ are some examples of ΣX-terms. □

A ΣX-term t is evaluated in a given Σ-algebra as follows. First we assign a value $x\alpha \in A$ to every variable $x \in X$. Then the operations of $\mathcal{A}$ are applied to these elements as indicated by the form of t. For example, given a mapping $\alpha\colon X \to A$, the t of the previous example would yield the element

$$\sigma^{\mathcal{A}}(x\alpha, \sigma^{\mathcal{A}}(z\alpha, \tau^{\mathcal{A}}(\mu^{\mathcal{A}}))).$$

Of course, the result depends on the choice of α, too. This evaluation process can be formalized as follows.

Definition 1.3.3 With every Σ-algebra $\mathcal{A}$ and ΣX-term t we associate a mapping

$$t^{\mathcal{A}}\colon A^X \to A$$

as follows: for any $\alpha\colon X \to A$

(i) $x^{\mathcal{A}}(\alpha) = x\alpha$ $(x \in X)$ and

(ii) $t^{\mathcal{A}}(\alpha) = \sigma^{\mathcal{A}}(t_1^{\mathcal{A}}(\alpha), \ldots, t_m^{\mathcal{A}}(\alpha))$ when $t = \sigma(t_1, \ldots, t_m)$ $(m \geq 0,\ \sigma \in \Sigma_m,\ t_1, \ldots, t_m \in F_\Sigma(X))$. The mappings $t^{\mathcal{A}}$ are called the *polynomial functions* of $\mathcal{A}$ in variables X and their set is denoted by $P_X(\mathcal{A})$.

It may seem strange that the polynomial functions $t^{\mathcal{A}} \in P_X(\mathcal{A})$ are evaluated on mappings from X to A, but this is, in fact, just a modification of the usual way to express polynomial functions. When one writes the value of a polynomial function in the form $p(a_1, \ldots, a_n)$, a given order of the variables is assumed, say $X = \{x_1, \ldots, x_n\}$, and the n-tuple $(a_1, \ldots, a_n)$ is just a convenient way to give the mapping $\alpha\colon X \to A$ such that $x_i\alpha = a_i$ $(i = 1, \ldots, n)$.

In a sense, the polynomial functions of an algebra $\mathcal{A}$ are the operations one can derive by composition from the basic operations $\sigma^{\mathcal{A}}$ $(\sigma \in \Sigma)$ of $\mathcal{A}$, and they share many properties with these. This is exemplified by the following four lemmas.

Lemma 1.3.4 *If $\mathcal{B}$ is a subalgebra of the Σ-algebra $\mathcal{A}$ and $\alpha\colon X \to A$ a mapping such that $X\alpha \subseteq B$, then $t^{\mathcal{A}}(\alpha) \in B$ for all $t \in F_\Sigma(X)$.* $\square$

The lemma states, in other words, that subalgebras are closed with respect to polynomial functions. The proof is a simple exercise in term induction quite similar to that of the next lemma which expresses formally the fact that congruences are invariant with respect to polynomial functions.

Lemma 1.3.5 *Let θ be a congruence of the Σ-algebra $\mathcal{A}$ and $\alpha\colon X \to A$, $\beta\colon X \to A$ two mappings such that*

$$x\alpha \equiv x\beta\ (\theta) \qquad \text{for all } x \in X.$$

Then $t^{\mathcal{A}}(\alpha) \equiv t^{\mathcal{A}}(\beta)\ (\theta)$ for all $t \in F_\Sigma(X)$.

Proof. We proceed by term induction on t. If $t = x \in X$, then

$$t^{\mathcal{A}}(\alpha) = x\alpha \equiv x\beta = t^{\mathcal{A}}(\beta)\ (\theta).$$

Let $t = \sigma(t_1, \ldots, t_m)$ and suppose

$$t_i^{\mathcal{A}}(\alpha) \equiv t_i^{\mathcal{A}}(\beta)\ (\theta) \qquad \text{for all } i = 1, \ldots, m.$$

Then also

$$t^{\mathcal{A}}(\alpha) = \sigma^{\mathcal{A}}(t_1^{\mathcal{A}}(\alpha), \ldots, t_m^{\mathcal{A}}(\alpha)) \equiv \sigma^{\mathcal{A}}(t_1^{\mathcal{A}}(\beta), \ldots, t_m^{\mathcal{A}}(\beta)) = t^{\mathcal{A}}(\beta)\ (\theta)$$

as θ is a congruence. Here the possibility $m = 0$ can be allowed as a trivial special case. □

Lemma 1.3.6 *Let $\varphi\colon \mathcal{A} \to \mathcal{B}$ be a homomorphism of Σ-algebras. Then*

$$t^{\mathcal{A}}(\alpha)\varphi = t^{\mathcal{B}}(\alpha\varphi)$$

for each mapping $\alpha\colon X \to A$ and each ΣX-term t. □

Lemma 1.3.7 *Let $\mathcal{A}$ and $\mathcal{B}$ be Σ-algebras, and $\alpha\colon X \to A$ and $\beta\colon X \to B$ any mappings. If we define a mapping $\gamma\colon X \to A \times B$ by putting*

$$x\gamma = (x\alpha, x\beta) \qquad \textit{for all } x \in X,$$

then

$$t^{\mathcal{A}\times\mathcal{B}}(\gamma) = (t^{\mathcal{A}}(\alpha), t^{\mathcal{B}}(\beta)) \qquad \textit{for all } t \in F_\Sigma(X). \quad □$$

Lemmas 1.3.6 and 1.3.7 can easily be verified by term induction.

The subalgebra generated by a subset can also be described in terms of polynomial functions.

Lemma 1.3.8 *For any subset X of a Σ-algebra $\mathcal{A}$ we have $[X] = \{t^{\mathcal{A}}(\alpha_X) \mid t \in F_\Sigma(X)\}$, where $\alpha_X = 1_A|X$, i.e., α_X is the mapping from X to A such that $x\alpha_X = x$ for all $x \in X$.*

Proof. Denote $\{t^{\mathcal{A}}(\alpha_X) \mid t \in F_\Sigma(X)\}$ by C. For every $x \in X$, $x = x\alpha_X = x^{\mathcal{A}}(\alpha_X) \in C$. Hence $X \subseteq C$. Also, C is closed under the operations of $\mathcal{A}$:

$$\sigma^{\mathcal{A}}(t_1^{\mathcal{A}}(\alpha_X), \ldots, t_m^{\mathcal{A}}(\alpha_X)) = \sigma(t_1, \ldots, t_m)^{\mathcal{A}}(\alpha_X) \in C$$

for all $m \geq 0$, $\sigma \in \Sigma_m$ and $t_1, \ldots, t_m \in F_\Sigma(X)$. Lemma 1.3.4 implies that $C \subseteq B$ for every subalgebra $\mathcal{B}$ which contains X. Hence $C = [X]$. Note that the result is true even if $\Sigma_0 = X = \emptyset$. In this case $[X] = \emptyset$. □

We shall now turn to the Σ-algebra formed by the ΣX-terms.

Definition 1.3.9 The Σ-algebra $\mathcal{F}_\Sigma(X) = (F_\Sigma(X), \Sigma)$ defined so that

$$\sigma^{\mathcal{F}_\Sigma(X)}(t_1, \ldots, t_m) = \sigma(t_1, \ldots, t_m)$$

for all $m \geq 0$, $\sigma \in \Sigma_m$ and $t_1, \ldots, t_m \in F_\Sigma(X)$, is called the *$\Sigma X$-term algebra* or the *free Σ-algebra generated by X*.

We shall first account for the name "free algebra".

Definition 1.3.10 Let K be a class of Σ-algebras. A Σ-algebra $\mathcal{F} = (F, \Sigma)$ is said to be *freely generated* over K by a subset $X \subseteq F$, if the following conditions are satisfied:

(i) $\mathcal{F} \in K$.

(ii) X generates $\mathcal{F}$.

(iii) Every mapping $\alpha\colon X \to A$ of X into any algebra $\mathcal{A}$ in K has an extension to a homomorphism $\widehat{\alpha}\colon \mathcal{F} \to \mathcal{A}$.

If these conditions are satisfied for some subset X of F, then $\mathcal{F}$ is called a *free algebra* over K (with $|X|$ generators), and X is called a *free generating set*.

A well-known example is provided by the free semigroup X^+ generated by a set (alphabet) X. The elements of X^+ are all the finite nonempty strings of elements of X. The product of two such strings u and v is simply their concatenation uv. The associativity of this product is obvious and thus X^+ is a semigroup. As every string $u \in X^+$ is obtained by concatenating individual elements of X, it is clear that X generates X^+. To prove that X^+ is freely generated by X over the class of all semigroups we consider any semigroup S and mapping $\alpha\colon X \to S$. The required (unique) homomorphism

$$\widehat{\alpha}\colon X^+ \to S$$

is obtained by putting

$$(x_1 x_2 \ldots x_k)\widehat{\alpha} = (x_1\alpha) \cdot (x_2\alpha) \cdot \ldots \cdot (x_k\alpha)$$

for all $x_1 x_2 \ldots x_k \in X^+$ (products to the right are formed in $\mathcal{S}$).

Free semigroups are considered later again, but we return now to our term algebras.

Theorem 1.3.11 *The ΣX-term algebra $\mathcal{F}_\Sigma(X)$ is freely generated by X over the class of all Σ-algebras.*

Proof. That X generates $\mathcal{F}_\Sigma(X)$ is quite obvious when we compare the definitions of $F_\Sigma(X)$ and $\mathcal{F}_\Sigma(X)$, but it follows also from the useful observation that

$$t^{\mathcal{F}_\Sigma(X)}(\alpha_X) = t \qquad \text{for all } t \in F_\Sigma(X) \tag{*}$$

(where $\alpha_X = 1_{F_\Sigma(X)}|X$). The proof of (*) goes again by term induction. Let $\mathcal{A}$ be any Σ-algebra and $\alpha\colon X \to A$ any mapping. We claim that the mapping

$$\widehat{\alpha}\colon F_\Sigma(X) \to A, \quad t \mapsto t^{\mathcal{A}}(\alpha) \quad (t \in F_\Sigma(X))$$

is the required homomorphism. For every $x \in X$, $x\widehat{\alpha} = x^{\mathcal{A}}(\alpha) = x\alpha$. Hence, $\widehat{\alpha}|X = \alpha$. It remains to be verified that $\widehat{\alpha}$ is a homomorphism. Indeed,

$$\begin{aligned}\sigma^{\mathcal{F}_\Sigma(X)}(t_1, \dots, t_m)\widehat{\alpha} &= \sigma(t_1, \dots, t_m)^{\mathcal{A}}(\alpha)\\ &= \sigma^{\mathcal{A}}(t_1^{\mathcal{A}}(\alpha), \dots, t_m^{\mathcal{A}}(\alpha))\\ &= \sigma^{\mathcal{A}}(t_1\widehat{\alpha}, \dots, t_m\widehat{\alpha})\end{aligned}$$

for all $m \geq 0$, $\sigma \in \Sigma_m$ and $t_1, \dots, t_m \in F_\Sigma(X)$. $\square$

We add a few general comments on free algebras. First of all, one should note that the homomorphic extension $\widehat{\alpha}\colon \mathcal{F} \to \mathcal{A}$ of a mapping $\alpha\colon X \to A$ ($\mathcal{A} \in K$) is unique. This follows from Lemma 1.2.6. Free algebras over a given class do not always exist, but when they do, they are determined up to isomorphism by the cardinality of the free generating set. This is stated formally in the following lemma.

Lemma 1.3.12 *Any two algebras freely generated over the same class of algebras by sets of the same cardinality are isomorphic.*

Proof. Suppose $\mathcal{A}$ and $\mathcal{B}$ both are free over the same class K and that they have free generating sets X and Y, respectively, such that $|X| = |Y|$. Then there is a bijection $\alpha\colon X \to Y$. The converse of it, $\beta = \alpha^{-1}$, defines a bijection from Y to X. Now there exist morphisms

$$\widehat{\alpha}\colon \mathcal{A} \to \mathcal{B} \quad \text{and} \quad \widehat{\beta}\colon \mathcal{B} \to \mathcal{A}$$

such that $\widehat{\alpha}|X = \alpha$ and $\widehat{\beta}|Y = \beta$. But then

$$\widehat{\alpha}\widehat{\beta}\colon \mathcal{A} \to \mathcal{A} \quad \text{and} \quad \widehat{\beta}\widehat{\alpha}\colon \mathcal{B} \to \mathcal{B}$$

are homomorphisms such that $\widehat{\alpha}\widehat{\beta}|X = 1_X$ and $\widehat{\beta}\widehat{\alpha}|Y = 1_Y$. This means by Lemma 1.2.6 that $\widehat{\alpha}\widehat{\beta} = 1_A$ and $\widehat{\beta}\widehat{\alpha} = 1_B$. Hence, $\widehat{\alpha}$ and $\widehat{\beta}$ are isomorphisms inverse to each other. This implies $\mathcal{A} \cong \mathcal{B}$. $\square$

Lemma 1.3.12 allows us to speak about *the* algebra freely generated over a class K by a set X.

We shall fix the notation $\widehat{\alpha}$ used above for the rest of the book: for any $\mathcal{A}$ and $\alpha\colon X \to A$, $\widehat{\alpha}\colon \mathcal{F}_\Sigma(X) \to \mathcal{A}$ is the homomorphism such that $\widehat{\alpha}|X = \alpha$. To evaluate a ΣX-term t in a Σ-algebra $\mathcal{A}$ for a given assignment $\alpha\colon X \to A$ of values to the variables amounts to the computation of $t\widehat{\alpha}$. Indeed, we showed in the proof of Theorem 1.3.11 that $t^{\mathcal{A}}(\alpha) = t\widehat{\alpha}$ for all $\mathcal{A}$, α and t.

The polynomial functions in variables X of an algebra $\mathcal{A}$ are the mappings one can get from the "projections" $x^{\mathcal{A}}$ ($x \in X$) by iterated compositions with the basic operations

$\sigma^{\mathcal{A}}$ ($\sigma \in \Sigma$). If the generating set of functions is enlarged by the set of all constant mappings ($c \in A$)

$$\gamma_c \colon A^X \to A, \quad \alpha \mapsto c \quad (\alpha \in A^X),$$

then we get, in general, a larger class of functions. These are called algebraic functions. We shall need just the unary (i.e., 1-place) algebraic functions and these only are defined below. In this special case X is a singleton $\{x\}$ and we may identify any mapping $\alpha \colon X \to A$ with the element $x\alpha \in A$. Then the unary algebraic functions can be defined simply as certain mappings from A to A.

Definition 1.3.13 The set of *unary algebraic functions* $\mathrm{Alg}_1(\mathcal{A})$ of a Σ-algebra $\mathcal{A}$ is defined as follows:

(i) $1_A \in \mathrm{Alg}_1(\mathcal{A})$.

(ii) For every $c \in A$, $\mathrm{Alg}_1(\mathcal{A})$ contains the *constant mapping* $\gamma_c \colon A \to A$, $a \mapsto c$ ($a \in A$).

(iii) The composition $\sigma^{\mathcal{A}}(f_1, \ldots, f_m)$ is in $\mathrm{Alg}_1(\mathcal{A})$ whenever $m \geq 0$, $\sigma \in \Sigma_m$ and f_1, $\ldots$, $f_m \in \mathrm{Alg}_1(\mathcal{A})$.

(iv) All members of $\mathrm{Alg}_1(\mathcal{A})$ are obtained by the rules (i)–(iii).

The constant mapping γ_c ($c \in A$) is usually denoted simply by c. It is intuitively clear from Definition 1.3.13 that every $f \in \mathrm{Alg}_1(\mathcal{A})$ can be represented by an expression similar to the terms that gave the polynomial functions. Let $X = A \cup \{x\}$ ($x \notin A$). Following the inductive form of Definition 1.3.13 we associate with every $f \in \mathrm{Alg}_1(\mathcal{A})$ a ΣX-term t_f as follows:

(i) $t_{1_A} = x$.

(ii) $t_c = c$ for all c ($= \gamma_c$) ($c \in A$).

(iii) If $f = \sigma^{\mathcal{A}}(f_1, \ldots, f_m)$, then $t_f = \sigma(t_{f_1}, \ldots, t_{f_m})$.

It is now an easy task to verify that the following lemma holds.

Lemma 1.3.14 *For every $f \in \mathrm{Alg}_1(\mathcal{A})$ there exists a term $t_f \in F_\Sigma(A \cup x)$ such that, for all $a \in A$,*

$$f(a) = t_f^{\mathcal{A}}(\alpha_a)$$

when α_a is the mapping such that $\alpha_a | A = 1_A$ and $x\alpha_a = a$. □

The assignment α_a depends on $a \in A$ only. We may think of t_f as a ΣX-term for a suitable X, in which all variables, save x, have been assigned constant values from A. In other words, the unary algebraic functions are obtained from polynomial functions by fixing the values of some variables. It is now obvious, in view of Lemma 1.3.5, that congruences of $\mathcal{A}$ are invariant with respect to unary algebraic functions. The converse of this observation holds also. In fact, it can be stated in a stronger form in terms of the special unary algebraic functions introduced in the following definition.

Definition 1.3.15 A mapping $f\colon A \to A$ is called an *elementary translation* of the Σ-algebra $\mathcal{A}$, if there exist an $m > 0$, a $\sigma \in \Sigma_m$, a j $(1 \le j \le m)$ and elements $c_1, \ldots, c_{j-1}, c_{j+1}, \ldots, c_m \in A$ such that

$$f(a) = \sigma^{\mathcal{A}}(c_1, \ldots, c_{j-1}, a, c_{j+1}, \ldots, c_m) \quad \text{for all } a \in A.$$

The set of all elementary translations of $\mathcal{A}$ is denoted by $\mathrm{ET}(\mathcal{A})$.

It is obvious that $\mathrm{ET}(\mathcal{A}) \subseteq \mathrm{Alg}_1(\mathcal{A})$.

Lemma 1.3.16 *An equivalence relation $\theta \in E(A)$ is a congruence of $\mathcal{A}$ iff θ is invariant with respect to all elementary translations of $\mathcal{A}$.*

Proof. Suppose $a \equiv b\,(\theta)$ implies $f(a) \equiv f(b)\,(\theta)$ for all $a, b \in A$ and $f \in \mathrm{ET}(\mathcal{A})$. Consider any $m > 0$, $\sigma \in \Sigma_m$ and elements $a_1, \ldots, a_m, b_1, \ldots, b_m \in A$ such that $a_1 \equiv b_1, \ldots, a_m \equiv b_m\,(\theta)$. Define the following m elementary translations:

$$f_j(\xi) = \sigma^{\mathcal{A}}(b_1, \ldots, b_{j-1}, \xi, a_{j+1}, \ldots, a_m) \quad (j = 1, \ldots, m).$$

Then

$$\begin{aligned} \sigma^{\mathcal{A}}(a_1, a_2, \ldots, a_m) &= f_1(a_1) \equiv f_1(b_1)\ (\theta) \\ &= f_2(a_2) \equiv f_2(b_2)\ (\theta) \\ &\ \vdots \\ &= f_m(a_m) \equiv f_m(b_m)\ (\theta) \\ &= \sigma^{\mathcal{A}}(b_1, b_2, \ldots, b_m). \end{aligned}$$

Hence $\sigma^{\mathcal{A}}(a_1, \ldots, a_m)\theta\sigma^{\mathcal{A}}(b_1, \ldots, b_m)$ and we have verified that $\theta \in C(\mathcal{A})$. The converse is obvious. $\square$

1.4 LATTICES

We shall need a few facts from lattice theory, and these are quickly surveyed here.

Definition 1.4.1 Let A be a set. A relation $\varrho \subseteq A \times A$ is called a *partial ordering* of A, if

(1) $\delta_A \subseteq \varrho$ (ϱ is reflexive),

(2) $\varrho \cap \varrho^{-1} \subseteq \delta_A$ (ϱ is antisymmetric), and

(3) $\varrho\varrho \subseteq \varrho$ (ϱ is transitive).

If ϱ is a partial ordering of A, then (A, ϱ) is called a *poset*.

The usual symbol for a partial ordering is $\leq$. Often a set A is called a poset when a certain partial ordering of A is understood.

An example of a poset is $(\mathfrak{p}S, \subseteq)$, where S is a set and $\subseteq$ the usual subset relation in the power set $\mathfrak{p}S$. Another simple example is $(\mathbf{N}, \leq)$ where $\leq$ is the "less than or equal" -relation of natural numbers. This $\leq$ is a *total ordering*, which means that any two elements of the poset are *comparable*, i.e., either $a \leq b$ or $b \leq a$ holds for any two elements a and b. A poset $(A, \leq)$ in which $\leq$ is a total ordering is called a *chain*.

Let $(A, \leq)$ be a poset and a, $b \in A$. We may write $a \geq b$ when $b \leq a$, $a < b$ when $a \leq b$ and $a \neq b$, and $a > b$ when $a \geq b$ and $a \neq b$. Clearly $\geq$ is a partial ordering and the poset $(A, \geq)$ is said to be *dual* to $(A, \leq)$. Each one of the relations $\geq$, $<$ and $>$ determines $\leq$ completely.

An element $a \in A$ is an *upper bound* of a subset $H \subseteq A$ if $b \leq a$ for all $b \in H$. An upper bound a of $H \subseteq A$ is the *least upper bound*, or the *supremum*, of H, if $a \leq c$ for all upper bounds c of H. *Lower bounds* and *greatest lower bounds (infimums)* are defined similarly. The least upper bound and the greatest lower bound of a subset H are denoted, respectively, by $\bigvee H$ and $\bigwedge H$. In case of an indexed family $(a_i \mid i \in I)$ of elements the notations $\bigvee(a_i \mid i \in I)$ and $\bigwedge(a_i \mid i \in I)$ may be used.

An element $c \in A$ is a *zero element* of the poset A if $c \leq a$ for every $a \in A$. If a poset has a zero element, it is unique and usually it is denoted by 0. Similarly, the *unit element* 1, is defined by the condition that $a \leq 1$ for all $a \in A$. Clearly, $\bigwedge A$ exists iff the poset has a zero element 0, and then $\bigwedge A = 0$. Similarly, $\bigvee A$ exists, and then equals 1, iff A has a unit element 1.

Definition 1.4.2 A poset $(A, \leq)$ is a *lattice*, if $\bigvee\{a, b\}$ and $\bigwedge\{a, b\}$ exist for all a, $b \in A$. It is a *complete lattice*, if $\bigvee H$ and $\bigwedge H$ exist for all subsets H of A.

In a lattice one usually writes $a \vee b$ and $a \wedge b$ for $\bigvee\{a, b\}$ and $\bigwedge\{a, b\}$, respectively. The element $a \vee b$ is also called the *join* of a and b, and $a \wedge b$ is the *meet* of a and b. It is easy to see that $\bigvee H$ and $\bigwedge H$ exist for every finite, nonempty subset H of a lattice. However, $\bigvee \emptyset$ exists only in case the lattice has a zero element 0. Then $\bigvee \emptyset = 0$. Similarly, $\bigwedge \emptyset$ exists iff the lattice has a unit element 1; then $\bigwedge \emptyset = 1$.

The following lemma follows directly from the definitions of the join and the meet.

Lemma 1.4.3 *If $(A, \leq)$ is a lattice then $\wedge$ and $\vee$ satisfy the following identities:*

(L1) $x \wedge x = x$, $x \vee x = x$ *(idempotence).*

(L2) $x \wedge y = y \wedge x$, $x \vee y = y \vee x$ *(commutativity).*

(L3) $x \wedge (y \wedge z) = (x \wedge y) \wedge z$, $x \vee (y \vee z) = (x \vee y) \vee z$ *(associativity).*

(L4) $x \wedge (x \vee y) = x$, $x \vee (x \wedge y) = x$ *(absorption).* □

The identities (L1)–(L4) are characteristic of lattices in the following sense. If $(A, \wedge, \vee)$ is an algebra with two binary operations that satisfy these identities, then $(A, \leq)$ is a lattice when $\leq$ is defined so that

$$a \leq b \quad \text{iff} \quad a \wedge b = a \quad (a, b \in A).$$

In this lattice $\bigvee\{a,b\} = a \vee b$ and $\bigwedge\{a,b\} = a \wedge b$ for all a, $b \in A$. In lattice theory lattices are usually defined and considered in parallel both as posets and as algebras. The two aspects of the theory complement each other.

The following lemma is often useful when one wants to show that a certain poset is a complete lattice.

Lemma 1.4.4 *A poset $(A, \leq)$ is a complete lattice, if $\bigwedge H$ exists for each subset $H \subseteq A$.* □

Note that the existence of $\bigwedge \emptyset = 1$ should also be ascertained when Lemma 1.4.4 is used. We shall now apply the lemma to an important example. Let A be a set. It is easy to see that the intersection $\bigcap(\varepsilon_i \mid i \in I)$ of any equivalence relations ε_i $(i \in I)$ of A is again in $E(A)$. This means that

$$\bigwedge(\varepsilon_i \mid i \in I) = \bigcap(\varepsilon_i \mid i \in I)$$

always exists in the poset $(E(A), \subseteq)$. (In particular, $\bigwedge \emptyset = \iota_A$.) Hence, we get

Lemma 1.4.5 *For each set A, $(E(A), \subseteq)$ is a complete lattice.* □

In general, the union of equivalence relations is not an equivalence relation. For any $H \subseteq E(A)$, $\bigvee H$ is the intersection of all equivalence relations which contain the union $\bigcup H$. A more useful description of the supremum is given in the following lemma.

Lemma 1.4.6 *Let $H \subseteq E(A)$ and a, $b \in A$. Then $a \equiv b\,(\bigvee H)$ iff there exist an $n \geq 0$, $\varepsilon_1, \ldots, \varepsilon_n \in H$ and $a_1, \ldots, a_{n-1} \in A$ such that*

$$a\,\varepsilon_1\,a_1\,\varepsilon_2\,a_2\,\ldots\,a_{n-1}\,\varepsilon_n\,b. \qquad \square$$

The lemma may be used to prove the following important fact.

Theorem 1.4.7 *For any algebra $\mathcal{A} = (A, \Sigma)$, $C(\mathcal{A})$ forms a* complete sublattice *of $(E(A), \subseteq)$, that is to say, $\bigvee H \in C(\mathcal{A})$ and $\bigwedge H \in C(\mathcal{A})$ whenever $H \subseteq C(\mathcal{A})$.* □

The *direct product* $(L_1 \times \cdots \times L_n, \leq)$ of posets $(L_1, \leq), \ldots, (L_n, \leq)$ is a poset when we define $\leq$ in $L_1 \times \cdots \times L_n$ so that

$$(a_1, \ldots, a_n) \leq (b_1, \ldots, b_n) \quad \text{iff} \quad a_i \leq b_i \quad \text{for all } i = 1, \ldots, n.$$

If the $(L_i, \leq)$'s are lattices, then the direct product is also a lattice in which

$$(a_1, \ldots, a_n) \vee (b_1, \ldots, b_n) = (a_1 \vee b_1, \ldots, a_n \vee b_n)$$

and

$$(a_1, \ldots, a_n) \wedge (b_1, \ldots, b_n) = (a_1 \wedge b_1, \ldots, a_n \wedge b_n).$$

An *ideal* of a lattice $(A, \leq)$ is a nonempty subset I of A such that, for all a, $b \in A$,

(1) a, $b \in I$ implies $a \vee b \in I$, and

(2) $a \leq b \in I$ implies $a \in I$.

A *dual ideal* of a lattice $(A, \leq)$ is a nonempty subset D of A such that, for all a, $b \in A$,

(1') a, $b \in D$ implies $a \wedge b \in D$, and

(2') $a \geq b \in D$ implies $a \in D$.

General examples are provided by the

(i) *principal ideal* $(a] = \{x \in A \mid x \leq a\}$ generated by an element $a \in A$, and by the

(ii) *principal dual ideal* $[a) = \{x \in A \mid x \geq a\}$ generated by an element $a \in A$.

Let A and B be posets. A mapping $\varphi\colon A \to B$ is said to be *isotone*, if

$$(\forall a_1, a_2 \in A)\, a_1 \leq a_2 \Longrightarrow a_1\varphi \leq a_2\varphi.$$

Suppose now that A and B are complete lattices. The mapping φ is *ω-continuous*, if

$$\bigvee(a_i \mid i \geq 0)\varphi = \bigvee(a_i\varphi \mid i \geq 0)$$

for every ascending ω-sequence

$$a_0 \leq a_1 \leq a_2 \leq \dots$$

of elements $a_i \in A$ $(0 \leq i < \omega)$. An ω-continuous mapping is always isotone, but the converse is false.

Let A be a poset and $\varphi\colon A \to A$ a mapping. An element $a \in A$ is a *fixed-point* of φ, if $a\varphi = a$. It is the *least fixed-point* of φ, if all other fixed-points of φ are above it. Of course, there can be at most one least fixed-point. A well-known theorem by A. Tarski states that every isotone mapping in a complete lattice has a fixed-point. For ω-continuous mappings the following stronger result holds.

Theorem 1.4.8 *Let $(A, \leq)$ be a complete lattice and $\varphi\colon A \to A$ an ω-continuous mapping. Then*

$$[\varphi] = \bigvee(0\varphi^i \mid i \geq 0)$$

is the least fixed-point of φ.

Proof. Since φ is isotone, $0 \leq 0\varphi$ implies

$$0 \leq 0\varphi \leq 0\varphi^2 \leq 0\varphi^3 \leq \dots.$$

By ω-continuity, we get now

$$[\varphi]\varphi = \bigvee(0\varphi^{i+1} \mid i \geq 0) = \bigvee(0\varphi^i \mid i \geq 0) = [\varphi].$$

For any fixed-point a of φ, $0 \leq a$ implies

$$0\varphi \leq a\varphi = a,$$

and in general by induction on $i \geq 0$, $0\varphi^i \leq a$. Hence $[\varphi] \leq a$, and $[\varphi]$ is the least fixed-point of φ. □

1.5 FINITE RECOGNIZERS AND REGULAR LANGUAGES

In this section several basic concepts and facts from the theory of finite automata are reviewed. For many readers there is probably nothing really new. The presentation is quite telegraphic and proofs are sketched at most. Much of the material will be generalized to tree automata in Chapter 2, and the present section is intended mainly as an outline of the proper background scenery.

An *alphabet* is a finite nonempty set of symbols which are called *letters*. We shall usually use the letters X, Y and Z to indicate alphabets. A finite string of letters from an alphabet X is called an *X-word* or a *word* over X. Consider an arbitrary X-word

$$w = x_1 x_2 \dots x_n \quad (n \geq 0,\ x_1, \dots, x_n \in X).$$

Here $x_i = x_j$ is possible even for $i \neq j$. If $n = 0$, then w is the *empty word* which is denoted by e. The *length* of w is n and we write it $|w|$. Obviously, $|w| = 0$ iff $w = e$. The set of all X-words is denoted by X^*, and the set of all nonempty X-words is denoted by X^+. The letters of an alphabet are viewed as indivisible symbols. This means, in particular, that for any $m \geq 0$, $n \geq 0$ and $x_1, \dots, x_m,\ y_1, \dots, y_n \in X$,

$$x_1 x_2 \dots x_m = y_1 y_2 \dots y_n$$

holds just in case $m = n$ and $x_i = y_i$ for all $i = 1, ..., m$. Letters are considered words of length 1. Hence, we may write $X \subset X^+ \subset X^*$ and $X^* = X^+ \cup e$.

In Section 3 we noted that X^+ is the free semigroup generated by X, when the product of two words is defined to be their catenation. Similarly, X^* is the *free monoid* generated by X. The identity element is the empty word: $ew = we = w$ for each $w \in X^*$.

A *language* over X, or an *X-language*, is simply a subset of X^*. An X-language is *e-free* if it does not include the empty word. Of course, formal language theory concerns itself with such languages only that can be specified in some effective manner.

A *family of languages* $\mathcal{L}$ is defined by indicating for each alphabet the set $\mathcal{L}(X)$ of X-languages belonging to the family. For example, $\mathcal{L}(X)$ could consist of all languages recognized by automata of a given type with input alphabet X. If $L \in \mathcal{L}(X)$, one may write just $L \in \mathcal{L}$. Two families of languages $\mathcal{K}$ and $\mathcal{L}$ are equal, which we write $\mathcal{K} = \mathcal{L}$, if $\mathcal{K}(X) = \mathcal{L}(X)$ for every alphabet X. Similarly, the inclusion $\mathcal{K} \subseteq \mathcal{L}$ means that $\mathcal{K}(X) \subseteq \mathcal{L}(X)$ for every X.

One way to specify a language $L \subseteq X^*$ is to give an automaton that can examine any given X-word and then tell whether the word is in L or not. Such automata are called *recognizers*. The most basic type of recognizers is the following:

Definition 1.5.1 An *X-recognizer* (also called a *Rabin-Scott recognizer*) $\mathbf{A}$ consists of

(1) a finite (nonvoid) set A of *states*,

(2) the *input alphabet* X,

(3) a *next-state function* $\delta\colon A \times X \to A$,

(4) an *initial state* $a_0 \in A$, and

(5) a set $A' \subseteq A$ of *final states*.

We write $\mathbf{A} = (A, X, \delta, a_0, A')$.

If the X-recognizer $\mathbf{A}$ of Definition 1.5.1 is in state a $(\in A)$ and receives the input x $(\in X)$, it enters state $\delta(a, x)$ and remains in this state until it reads the next input letter. The next-state function is extended to a function

$$\hat{\delta} \colon A \times X^* \to A$$

as follows:

1° $\hat{\delta}(a, e) = a$ for each $a \in A$, and

2° $\hat{\delta}(a, wx) = \delta(\hat{\delta}(a, w), x)$ for all $a \in A$, $w \in X^*$ and $x \in X$.

We will omit the cap from $\hat{\delta}$. For any $a \in A$ and $w \in X^*$, $\delta(a, w)$ is the state of $\mathbf{A}$ when it has read the whole input word w, from left to right, and the state in the beginning was a. As a language recognizer $\mathbf{A}$ operates as follows. The word w to be tested for membership is entered to $\mathbf{A}$ so that the state of $\mathbf{A}$ initially is a_0. Now w is *accepted* by $\mathbf{A}$ if $\delta(a_0, w)$ is a final state. Otherwise w is said to be *rejected* by $\mathbf{A}$. The *language recognized* by $\mathbf{A}$ consists of all X-words accepted by $\mathbf{A}$, i.e., it is the X-language

$$L(\mathbf{A}) = \{w \in X^* \mid \delta(a_0, w) \in A'\}.$$

An X-language L is called *recognizable*, if there exists an X-recognizer $\mathbf{A}$ such that $L = L(\mathbf{A})$. The family of recognizable languages is denoted by Rec, and Rec X denotes the set of all recognizable X-languages.

In the definition of X-recognizers the finiteness of the state set is essential. Otherwise, every X-language would be recognizable.

We shall now prepare for the first of the many characterizations of recognizable languages.

The *product* of two X-languages U and V is the X-language

$$UV = \{uv \mid u \in U, v \in V\}.$$

The product is associative:

$$U(VW) = (UV)W \quad \text{for all } U, V, W \subseteq X^*.$$

Furthermore,

$$U\emptyset = \emptyset U = \emptyset \quad \text{and} \quad U\{e\} = \{e\}U = U$$

for every X-language U.

The *powers* U^n $(n \geq 0)$ of an X-language U are defined inductively:

1° $U^0 = \{e\}$ and
2° $U^n = U^{n-1}U$ for $n > 0$.

By means of the powers we may define the *iteration* of U

$$U^* = \bigcup(U^n \mid n \geq 0).$$

Excluding U^0, we get the language

$$U^+ = \bigcup(U^n \mid n \geq 1).$$

Clearly, $U^* = U^+ \cup \{e\}$, and $U^+ = U^*$ iff $e \in U$. A word $w \in X^*$ belongs to U^* iff it can be expressed in the form $w = u_1 u_2 \ldots u_n$, where $n \geq 0$ and $u_1, \ldots u_n \in U$.

Note that X^n is the set of all X-words of length n ($n \geq 0$) and the set X^* of all X-words really is the iteration of X (when X is viewed as the set of X-words of length 1).

Union, product and iteration are called the *regular language operations*.

Definition 1.5.2 The set Reg X of *regular* X-languages is the smallest set R such that

1° $\emptyset \in R$ and $\{x\} \in R$ for each $x \in X$, and
2° $U, V \in R$ implies $U \cup V$, UV, $U^* \in R$.

Regular languages are also called rational languages. All finite languages are regular. Hence Reg X is the smallest set of X-languages containing the finite X-languages which is closed under the three regular operations.

The form of Definition 1.5.2 implies that every regular X-language can be represented by a *regular expression* which shows how the language is obtained from $\emptyset$ and the languages $\{x\}$ by forming unions, products and iterations.

Example 1.5.3 Let $X = \{x, y\}$. Some members of Reg X are $\emptyset$, $\{x\}$, $\{y\}$, $\{xy\} = \{x\}\{y\}$, $\{xy, yy\} = \{x\}\{y\} \cup \{y\}\{y\} = (\{x\} \cup \{y\})\{y\}$ and

$$U = \{x^i y^j \mid i \geq 1, j \geq 0\} \cup \{yx^{2k} \mid k \geq 0\}.$$

A possible regular expression for the language U would be $\eta = (x(x)^*(y)^*) + (y(xx)^*)$ (usually '+' is used for union). If we agree on the usual hierarchy of regular operations (first iterations, then products, and unions last), then some parentheses can be omitted and η becomes $xx^*y^* + y(xx)^*$. The language U is recognized by the X-recognizer defined by the state graph of Fig 1.1 (the initial state is a_0 and the final states are a, b and c).

□

The following theorem is one of the cornerstones of finite automaton theory.

Theorem 1.5.4 *(S. C. Kleene 1956)* Rec = Reg. □

The theorem is effective in the following sense. There are algorithms to construct a recognizer for any regular language given by a regular expression. Conversely, a regular expression representing $L(\mathbf{A})$ can be found for any given recognizer $\mathbf{A}$.

Kleene's theorem implies also that the family Rec is closed under the regular operations. We shall present some more closure properties of the family Rec.

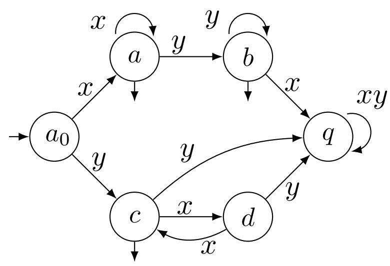

Figure 1.1

Theorem 1.5.5 *Let X and Y be arbitrary alphabets.*

(a) *If U, $V \in \mathrm{Rec}\ X$, then $U \cap V$, $U - V \in \mathrm{Rec}\ X$.*

(b) *If U is a recognizable X-language, then so is its* mirror image *(or* reversal*)*

$$mi(U) = \{x_n \dots x_2 x_1 \mid n \geq 0,\ x_1 x_2 \dots x_n \in U\ (x_i \in X)\}.$$

(c) *If U and V are recognizable X-languages, then so are the* quotient languages

$$U^{-1}V = \{w \in X^* \mid uw = v \text{ for some } u \in U, v \in V\}$$

and

$$UV^{-1} = \{w \in X^* \mid wv = u \text{ for some } u \in U, v \in V\}.$$

(d) *Let $\varphi \colon X^* \to Y^*$ be a homomorphism (of monoids). If $U \in \mathrm{Rec}\ X$, then $U\varphi \in \mathrm{Rec}\ Y$. If $V \in \mathrm{Rec}\ Y$, then $V\varphi^{-1} \in \mathrm{Rec}\ X$.*

(e) *If $U \in \mathrm{Rec}\ X$ and $\varphi \colon \mathfrak{p}X^* \to \mathfrak{p}Y^*$ is such a* substitution mapping *that $x\varphi \in \mathrm{Rec}\ Y$ for all $x \in X$, then $U\varphi \in \mathrm{Rec}\ Y$.* □

Recall that a mapping $\varphi \colon \mathfrak{p}X^* \to \mathfrak{p}Y^*$ is a substitution, if

1° $\{e\}\varphi = \{e\}$,
2° $\{wx\}\varphi = (w\varphi)(x\varphi)$ for all $w \in X^*$, $x \in X$, and
3° $U\varphi = \bigcup(u\varphi \mid u \in U)$ for all $U \subseteq X^*$.

Obviously, the substitution is completely defined when the languages $x\varphi$ ($x \in X$) are given. Extended to mappings of languages, homomorphisms $\varphi \colon X^* \to Y^*$ are special substitutions for which every $x\varphi$ ($x \in X$) consists of exactly one word.

Often it is convenient to allow a recognizer to be nondeterministic. In a *nondeterministic X-recognizer* $\mathbf{A} = (A, X, \delta, A_0, A')$ the next-state function is a mapping

$$\delta \colon A \times X \to \mathfrak{p}A.$$

Also, the recognizer has a set $A_0 \subseteq A$ of initial states. If $\mathbf{A}$ receives in state a the input letter x, then it may enter any one of the states in $\delta(a, x)$. The operation of $\mathbf{A}$ may be started in any initial state $a_0 \in A_0$. A word $w = x_1 x_2 \dots x_n$ ($n \geq 0, x_1, \dots, x_n \in X$) is accepted by $\mathbf{A}$ if there is such a choice of states $a_0, a_1, \dots, a_n$ that

(i) $a_0 \in A_0$,

(ii) $a_i \in \delta(a_{i-1}, x_i)$ for all $i = 1, \ldots, n$, and

(iii) $a_n \in A'$.

The mapping δ extends to a mapping

$$\hat{\delta} \colon \mathfrak{p}A \times X^* \to \mathfrak{p}A$$

as follows:

1° $\hat{\delta}(H, e) = H$ for all $H \subseteq A$, and

2° $\hat{\delta}(H, wx) = \bigcup(\delta(a, x) \mid a \in \hat{\delta}(H, w))$ for all $H \subseteq A, w \in X^*$ and $x \in X$.

Obviously, $\hat{\delta}(H, w)$ is the set of states $\mathbf{A}$ may reach under the input word w from at least one state in H. The *language recognized* by A can now be defined formally as

$$L(\mathbf{A}) = \{w \in X^* \mid \hat{\delta}(A_0, w) \cap A' \neq \emptyset\}.$$

Every X-recognizer may be interpreted as a nondeterministic X-recognizer $\mathbf{A}$, where A_0 and the sets $\delta(a, x)$ all are singletons. On the other hand, every nondeterministic X-recognizer $\mathbf{A}$ may be turned into the equivalent X-recognizer

$$\mathbf{B} = (\mathfrak{p}A, X, \hat{\delta}, A_0, A''),$$

where $A'' = \{H \in \mathfrak{p}A \mid H \cap A' \neq \emptyset\}$; this is the well-known "subset construction". Hence, a language can be recognized by a nondeterministic recognizer iff it is recognizable in our original sense of the word.

Now we recall some algebraic characterizations of Rec.

An equivalence relation ϱ on a semigroup $\mathcal{S}$ is a *right congruence*, if $a\varrho b$ implies $ac\varrho bc$ for all $a, b, c \in S$. Every X-recognizer $\mathbf{A} = (A, X, \delta, a_0, A')$ defines a right congruence $\varrho_{\mathbf{A}}$ of the free monoid X^* as follows:

$$u \equiv v \ (\varrho_{\mathbf{A}}) \quad \text{iff} \quad \delta(a_0, u) = \delta(a_0, v) \quad (u, v \in X^*).$$

The index of $\varrho_{\mathbf{A}}$ is at most $|A|$ and

$$L(\mathbf{A}) = \bigcup(u\varrho_{\mathbf{A}} \mid u \in X^*, \delta(a_0, u) \in A').$$

This shows that every recognizable X-language is saturated by a right congruence of X^* of finite index.

Suppose now that the X-language L is saturated by a right congruence ϱ of X^* of finite index. The X-recognizer

$$\mathbf{A} = (X^*/\varrho, X, \delta, e\varrho, L/\varrho),$$

where δ is defined by the condition

$$\delta(u\varrho, x) = (ux)\varrho \quad (u \in X^*, x \in X),$$

is then well-defined and

$$\delta(e\varrho, u) = u\varrho$$

for each $u \in X^*$. This implies $L(\mathbf{A}) = L \in \mathrm{Rec}\ X$. Among all right congruences of X^* saturating a given X-language there is a greatest one which is called the *Nerode congruence* of L. We denote it by ϱ_L and it can be defined by the condition that

$$u \equiv v\ (\varrho_L) \quad \text{iff} \quad (\forall w \in X^*)\ (uw \in L \Leftrightarrow vw \in L)$$

for all $u, v \in X^*$. From these observations it is easy to construct a proof for the following theorem.

Theorem 1.5.6 *(A. Nerode 1957). For any X-language L the following three conditions are equivalent:*

(1) $L \in \mathrm{Rec}\ X$.

(2) *L is saturated by a right congruence of X^* of finite index.*

(3) *The Nerode congruence ϱ_L is of finite index.* □

There is a similar characterization which uses congruences of X^*. Every X-recognizer $\mathbf{A}$ defines a congruence $\theta_{\mathbf{A}}$ of X^* of finite index which saturates $L(\mathbf{A})$:

$$u \equiv v\ (\theta_{\mathbf{A}}) \quad \text{iff} \quad (\forall a \in A)\ \delta(a, u) = \delta(a, v).$$

If $L \subseteq X^*$ is saturated by a congruence, then a recognizer for L can be constructed as above in the case of right congruences. The greatest congruence θ_L saturating L is called the *syntactic congruence* of L. It may be defined by the condition that

$$u \equiv v\ (\theta_L) \quad \text{iff} \quad (\forall w, w' \in X^*)\ (wuw' \in L \Leftrightarrow wvw' \in L)$$

for all $u, v \in X^*$.

Theorem 1.5.7 *(J. R. Myhill 1957). For every X-language L the following three conditions are equivalent:*

(1) $L \in \mathrm{Rec}\ X$.

(2) *L is saturated by a congruence of X^* of finite index.*

(3) *The syntactic congruence θ_L is of finite index.* □

Let θ be a congruence of X^* saturating an X-language L. Then $L = (L\theta^\natural)\theta^{\natural^{-1}}$, where

$$\theta^\natural \colon X^* \to X^*/\theta$$

is the canonical homomorphism, and X^*/θ is finite iff θ is of finite index. This applies, in particular, to the syntactic congruence θ_L. The monoid X^*/θ_L is called the *syntactic monoid* of L. On the other hand, if we have a finite monoid M, a homomorphism

$$\varphi \colon X^* \to M$$

and a subset $H \subseteq M$ for which $L = H\varphi^{-1}$, then $\varphi\varphi^{-1}$ is a congruence of X^* of finite index saturating L. It is now clear that Myhill's theorem can be reformulated as follows.

Theorem 1.5.8 *For any X-language L the following three conditions are equivalent:*

(1) $L \in \mathrm{Rec}\ X$.

(2) *There exist a finite monoid M, a homomorphism $\varphi \colon X^* \to M$ and a subset $H \subseteq M$ such that $L = H\varphi^{-1}$.*

(3) *The syntactic monoid of L is finite.* $\square$

An X-language L is called *local*, if there exist sets $H, K \subseteq X$ and $I \subseteq X^2$ such that

$$L - \{e\} = (HX^* \cap X^*K) - X^*IX^*.$$

The membership of a nonempty word w in such an L can be tested by checking that the first letter of w is in H, the last letter of w is in K, and that no two consecutive letters of w form a pair belonging to I. Note that a local language may, according to our definition, contain the empty word.

A homomorphism $\varphi \colon X^* \to Y^*$ is called *length-preserving* if $|w\varphi| = |w|$ for all $w \in X^*$. Obviously φ is length-preserving iff $X\varphi \subseteq Y$.

In terms of these concepts one more characterization of Rec can be given.

Theorem 1.5.9 *An X-language L is recognizable iff $L = U\varphi$ for some alphabet Y, local Y-language U and length-preserving morphism $\varphi \colon Y^* \to X^*$.* $\square$

An X-recognizer $\mathbf{A}$ is said to be *minimal*, if no X-recognizer with fewer states recognizes $L(\mathbf{A})$. It is obvious that every regular language has a minimal recognizer. To say more than that, we need a few concepts.

Let $\mathbf{A} = (A, X, \delta, a_0, A')$ be an X-recognizer. It is said to be *connected*, if there exists for every $a \in A$ a word $w \in X^*$ such that $a = \delta(a_0, w)$. Two states a and b of $\mathbf{A}$ are said to be *equivalent*, and we write $a \sim b$, if

$$(\forall w \in X^*)\,(\delta(a, w) \in A' \Longleftrightarrow \delta(b, w) \in A').$$

The recognizer $\mathbf{A}$ is *reduced*, if $a \sim b$ implies $a = b$.

A relation $\theta \in E(A)$ is a *congruence* of $\mathbf{A}$, if

(1) $a\theta b$ implies $\delta(a,x)\theta\delta(b,x)$ for all $a,b \in A$ and $x \in X$, and

(2) θ saturates A'.

Let $C(\mathbf{A})$ be the set of all congruences of $\mathbf{A}$. It is not hard to prove that $\sim$ is a congruence of $\mathbf{A}$. In fact, it is the greatest congruence of $\mathbf{A}$.

If $\theta \in C(\mathbf{A})$, then one can define a *quotient recognizer*

$$\mathbf{A}/\theta = (A/\theta, X, \delta', a_0\theta, A'/\theta)$$

by putting

$$\delta'(a\theta, x) = \delta(a,x)\theta \quad \text{for all } a \in A \text{ and } x \in X.$$

The congruence property (1) guarantees that δ' is well-defined. An easy induction on $|w|$ shows that

$$\delta'(a\theta, w) = \delta(a,w)\theta \quad \text{for all } a \in A \text{ and } w \in X^*.$$

This implies $L(\mathbf{A}/\theta) = L(\mathbf{A})$. In particular, $L(\mathbf{A}/{\sim}) = L(\mathbf{A})$. It is now obvious that a minimal recognizer should be reduced and, of course, connected.

Let $\mathbf{A} = (A, X, \delta, a_0, A')$ and $\mathbf{B} = (B, X, \eta, b_0, B')$ be two X-recognizers. A *homomorphism* $\varphi\colon \mathbf{A} \to \mathbf{B}$ is a mapping $\varphi\colon A \to B$ such that

(1) $\delta(a,x)\varphi = \eta(a\varphi, x)$ for all $a \in A$ and $x \in X$,

(2) $a_0\varphi = b_0$, and

(3) $B'\varphi^{-1} = A'$.

Epimorphisms and *isomorphisms* of X-recognizers are, respectively, surjective and bijective homomorphisms.

Homomorphisms, congruences and quotients of X-recognizers are related to each other the same way as the corresponding concepts in algebra. Hence, for any $\theta \in C(\mathbf{A})$, the natural mapping $\theta^\natural$ is an epimorphism $\mathbf{A} \to \mathbf{A}/\theta$. If $\varphi\colon \mathbf{A} \to \mathbf{B}$ is an epimorphism, then $\varphi\varphi^{-1}$ is a congruence of $\mathbf{A}$ and $\mathbf{A}/\varphi\varphi^{-1}$ is isomorphic to $\mathbf{B}$. Moreover,

$$\delta(a,w)\varphi = \eta(a\varphi, w) \quad \text{for all } a \in A, w \in X^*.$$

This implies $L(\mathbf{A}) = L(\mathbf{B})$.

The X-recognizer $\mathbf{B}$ is a *subrecognizer* of $\mathbf{A}$ if $B \subseteq A$, $b_0 = a_0$, $B' = A' \cap B$ and $\eta = \delta|B \times X$. The subset B determines such a subrecognizer completely. The *connected part*

$$A_c = \{\delta(a_0, w) \mid w \in X^*\}$$

of an X-recognizer is the state set of a subrecognizer

$$\mathbf{A}_c = (A_c, X, \delta_c, a_0, A' \cap A_c)$$

where $\delta_c = \delta|A_c \times X$.

The following theorem summarizes the main facts concerning minimal and reduced recognizers.

Theorem 1.5.10 (a) *The minimal recognizer of a regular language is unique up to isomorphism, i.e., if two recognizers are minimal and equivalent to each other, then they are isomorphic.*

(b) *A recognizer is minimal iff it is connected and reduced.*

(c) *For any recognizer* $\mathbf{A}$, *the quotient* $\mathbf{A}/\sim$ *is reduced and its connected part* $(\mathbf{A}/\sim)_c$ *is minimal. The recognizer* $\mathbf{A}_c/\sim$ *is isomorphic to* $(\mathbf{A}/\sim)_c$.

(d) *If* $\mathbf{A}$ *is minimal,* $\mathbf{B}$ *is connected and* $L(\mathbf{A}) = L(\mathbf{B})$, *then there exists a unique epimorphism* $\varphi\colon \mathbf{B} \to \mathbf{A}$. □

Theorem 1.5.10 implies that one can find a minimal recognizer for a regular language L by starting with any recognizer $\mathbf{A}$ of L; first one finds the connected part A_c and then one has to determine the equivalent pairs of states in $\mathbf{A}_c$. For both tasks there are simple algorithms. The order may also be reversed; first form $\mathbf{A}/\sim$ and then find the connected part of this reduced recognizer.

The decidability of the emptiness, finiteness and equality questions for regular languages follows from the following simple observation.

Lemma 1.5.11 *Let* $\mathbf{A}$ *be an* X*-recognizer with* n *states.*

(a) *If* $L(\mathbf{A})$ *contains a word* w *of length* $\geq n$, *then one may write* $w = uvz$ *so that* $0 < |v| \leq n$ *and* $uv^kz \in L(\mathbf{A})$ *for all* $k \geq 0$.

(b) $L(\mathbf{A})$ *is nonempty iff it contains a word of length* $< n$.

(c) $L(\mathbf{A})$ *is infinite iff it contains a word* w *such that* $n \leq |w| < 2n$. □

Statement (a) is often referred to as the "pumping lemma" for finite recognizers.

To test whether $L(\mathbf{A})$ is nonempty it suffices to try all input words of length $< |A|$. Similarly, the finiteness of $L(\mathbf{A})$ can be checked by applying all input words w such that $|A| \leq |w| < 2|A|$. From any two X-recognizers $\mathbf{A}$ and $\mathbf{B}$ one can construct a recognizer for $(L(\mathbf{A}) - L(\mathbf{B})) \cup (L(\mathbf{B}) - L(\mathbf{A}))$. But this language is empty exactly in case $L(\mathbf{A}) = L(\mathbf{B})$. Hence, the equivalence of $\mathbf{A}$ and $\mathbf{B}$ can also be decided.

1.6 GRAMMARS AND CONTEXT-FREE LANGUAGES

We shall now consider the most important tools of formal language theory, Chomsky's grammars. A grammar is a device to define a language by showing how to generate the strings of the language. The concept is very flexible, and by imposing various restrictions on grammars several interesting families of languages can be obtained. A good example is provided by the celebrated *Chomsky hierarchy* consisting of four families of languages. At the bottom of the hierarchy we find, once more, the recognizable languages. However, most of this section will be devoted to context-free languages. These form the second step in the hierarchy.

Definition 1.6.1 A *grammar* is a 4-tuple (N, X, P, a_0), where

(1) N is a finite nonempty set of *nonterminal symbols*,

(2) X is the *terminal alphabet*,

(3) P is the finite set of *productions*, and

(4) $a_0 \in N$ is the *initial symbol*.

It is required that $N \cap X = \emptyset$. Every production is of the form $\beta \to \gamma$, where $\beta, \gamma \in (N \cup X)^*$ and β contains at least one nonterminal symbol.

Let $G = (N, X, P, a_0)$ be a grammar. For $u, v \in (N \cup X)^*$ we write $u \Rightarrow_G v$ (or just $u \Rightarrow v$, when G is understood) if there exist $u', u'' \in (N \cup X)^*$ and a production $\beta \to \gamma \in P$ so that $u = u'\beta u''$ and $v = u'\gamma u''$. If $u \Rightarrow_G v$, then u is said to *generate* v *directly* in G. If there exists a *derivation*

$$u_0 \Rightarrow_G u_1 \Rightarrow_G u_2 \Rightarrow_G \ldots \Rightarrow_G u_n \quad (n \geq 0)$$

such that $u_0 = u$ and $u_n = v$, then we write $u \Rightarrow_G^* v$ (or just $u \Rightarrow^* v$). The *language generated* by G is the X-language

$$L(G) = \{w \in X^* \mid a_0 \Rightarrow_G^* w\}.$$

Two grammars are *equivalent*, if they generate the same language.

The grammars of Definition 1.6.1 are very general and every recursively enumerable language can be generated by such a grammar.

Definition 1.6.2 A grammar (N, X, P, a_0) is called *right linear*, if each production is of the form

$$a \to xb, \quad a \to x \quad \text{or} \quad a \to e,$$

where $a, b \in N$ and $x \in X$. A language is *right linear*, or of *type 3* (in the Chomsky hierarchy), if it can be generated by a right linear grammar.

A right linear grammar $G = (N, X, P, a_0)$ can be converted into a nondeterministic X-recognizer

$$\mathbf{A} = (N \cup \{c\}, X, \delta, \{a_0\}, A') \quad (c \notin N)$$

which recognizes $L(G)$ as follows. For any $a, b \in N$ and $x \in X$, put

(i) $b \in \delta(a, x)$ iff $a \to xb \in P$,

(ii) $c \in \delta(a, x)$ iff $a \to x \in P$, and

(iii) $\delta(c, x) = \emptyset$.

Finally, let $A' = \{c\} \cup \{a \in N \mid a \to e \in P\}$. Conversely, every X-recognizer $\mathbf{A} = (A, X, \delta, a_0, A')$ can be replaced by the right linear grammar $G = (A, X, P, a_0)$, where

$$P = \{a \to xb \mid \delta(a, x) = b\} \cup \{a \to e \mid a \in A'\}.$$

These observations lead to one more characterization of Rec:

Theorem 1.6.3 *The type 3 languages are exactly the regular languages.* □

Now we proceed to the main topic of this section.

Definition 1.6.4 A grammar (N, X, P, a_0) is *context-free* (CF, for short) if each production is of the form

$$a \to \gamma$$

where $a \in N$ and $\gamma \in (N \cup X)^*$. A language is *context-free* (CF) if it is generated by a CF grammar. The family of all CF languages is denoted by CF and the set of CF X-languages by $\mathrm{CF}(X)$.

The CF languages are the type 2 languages in Chomsky's hierarchy. Every right linear grammar is CF. Hence Rec $\subseteq$ CF. If $|X| = 1$, then $\mathrm{Rec}\, X = \mathrm{CF}(X)$, but in all other cases the inclusion is proper.

Example 1.6.5 Suppose X contains two distinct letters x and y. Every derivation in the CF grammar

$$G = (\{a\}, X, \{a \to xay, a \to xy\}, a)$$

is of the form

$$a \Rightarrow xay \Rightarrow xxayy \Rightarrow \ldots \Rightarrow x^{n-1}ay^{n-1} \Rightarrow x^n y^n \quad (n \geq 1).$$

Hence, $L(G)$ is the nonregular language $\{x^n y^n \mid n \geq 1\}$. □

The main fact to connect CF languages with tree automata is that context-free derivations can be represented by *derivation trees*. A derivation tree is a description of the syntax of a word of the CF language. (Here it would be more natural to speak about "sentences" of a language.) Derivation trees have proved very useful tools in the theory of CF languages. Later we shall define "trees" in a way suitable for our purposes, but here there is no need to define the concept too formally.

Let $G = (N, X, P, a_0)$ be a CF grammar. The derivation tree representing a derivation of a word $u \in (X \cup N)^*$ from a symbol $a \in (X \cup N)$ in G is defined by induction on the number k of steps in the derivation:

1° If $k = 0$, then $u = a$ and the derivation tree consists of a single node labelled by a.

2° Consider a derivation

$$a \Rightarrow u_1 \Rightarrow u_2 \Rightarrow \ldots \Rightarrow u_{k-1} \Rightarrow u \qquad (*)$$

where $k \geq 1$. Suppose $u_1 = d_1 \ldots d_m$, where $m \geq 0$ and $d_1, \ldots, d_m \in N \cup X$. At this point the context-freeness of G becomes essential. Every application of a production in (*) rewrites exactly one d_i or a nonterminal derived from exactly one d_i. This means that (*) may be decomposed into a number of "subderivations"

$$d_i \Rightarrow \ldots \Rightarrow v_i \quad (i = 1, \ldots, m)$$

each of which yields a segment v_i of u and $u = v_1 v_2 \ldots v_m$. If the derivation trees of the subderivations are $t_1, \ldots, t_m$, respectively, then the derivation tree of (*) is that shown in Fig. 1.2.

The possibility $m = 0$ was not excluded. Then $k = 1$, $u = e$ and the derivation tree reduces to a single node labelled by a.

The word $xxxyyy$ has the derivation

$$a \Rightarrow xay \Rightarrow xxayy \Rightarrow xxxyyy$$

in the grammar of Example 1.6.5. The corresponding derivation tree is shown in Fig. 1.3.

Consider any derivation

$$a_0 \Rightarrow \ldots \Rightarrow w$$

of a terminal word $w \in L(G)$ from the initial symbol. The corresponding derivation

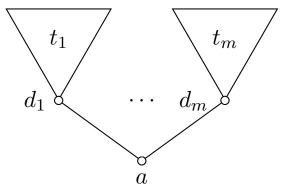

Figure 1.2

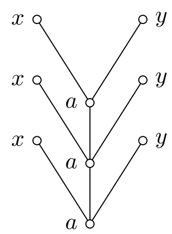

Figure 1.3

tree is also called a derivation tree of w, and w can be read from the "leaves" of the tree.

The grammar G of Example 1.6.5 has the rather special property that every word in $L(G)$ has just one derivation in G.

Example 1.6.6 Consider the CF grammar

$$G = (\{a_0, a, b\}, \{x, y\}, P, a_0)$$

where P consists of the productions

$$a_0 \to ab, \quad a \to xay, \quad a \to xy, \quad b \to ybx \text{ and } b \to yx.$$

Obviously, $L(G) = \{x^m y^{m+n} x^n \mid m, n \geq 1\}$. The word $xyyx \in L(G)$ has the two derivations

$$a_0 \Rightarrow ab \Rightarrow xyb \Rightarrow xyyx$$

and

$$a_0 \Rightarrow ab \Rightarrow ayx \Rightarrow xyyx$$

both of which are represented by the derivation tree shown in Fig. 1.4. In general, the word $x^m y^{m+n} x^n$ has $\binom{m+n}{n}$ different derivations all of which are represented by the same derivation tree. $\square$

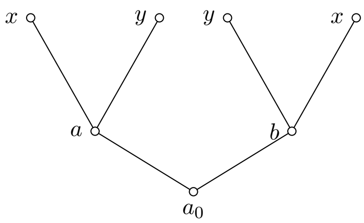

Figure 1.4

In Example 1.6.6 the different derivations of the same word do not represent different syntactic descriptions of the word. In fact, they can all be obtained from each other by changing the order in which the individual steps are carried out. If we agree on some fixed order in which the subderivations are to be carried out, then there would be just one derivation for each derivation tree of a word in the language.

Definition 1.6.7 A derivation

$$u_0 \Rightarrow u_1 \Rightarrow u_2 \Rightarrow \ldots \Rightarrow u_k$$

in a CF grammar $G = (N, X, P, a_0)$ is called a *leftmost derivation*, if we can write, for every $i = 0, \ldots, k-1$,

$$u_i = w_i a u_i' \quad \text{and} \quad u_{i+1} = w_i \gamma u_i'$$

so that $w_i \in X^*$, $a \in N$ and $a \to \gamma \in P$. The grammar G is *ambiguous* if some word w in $L(G)$ has two different leftmost derivations from a_0. Otherwise G is *unambiguous*. A CF language generated by at least one unambiguous CF grammar is said to be *unambiguous*. If all CF grammars generating a given CF language are ambiguous, then the language is said to be *inherently ambiguous*.

A CF grammar G is unambiguous if every word $w \in L(G)$ has exactly one derivation tree. It is ambiguous, if at least one word $w \in L(G)$ has more than one derivation tree. The grammars of Examples 1.6.5 and 1.6.6 are unambiguous. Every regular language is unambiguous. Of course, a language generated by an ambiguous CF grammar may be unambiguous. The language

$$\{x^i y^j z^k \mid i = j \text{ or } j = k \quad (i, j, k \geq 1)\}$$

is a well-known example of an inherently ambiguous language.

There are many simplifying additional conditions that a CF grammar may always be assumed to satisfy. Some of these are listed below.

Definition 1.6.8 Let $G = (N, X, P, a_0)$ be a CF grammar.

(a) G is *reduced* if either $P = \emptyset$ and $N = \{a_0\}$, or then for every $a \in N$,

$$a_0 \Rightarrow^* uav \Rightarrow^* w$$

for some $u, v \in (N \cup X)^*$ and $w \in X^*$.

(b) G is in *Chomsky normal form* if each production is of the form

(i) $a \to bc \quad (a \in N,\, b, c \in N - a_0)$,

(ii) $a \to x \quad (a \in N,\, x \in X)$, or

(iii) $a_0 \to e$.

(c) G is in *Greibach normal form* if each production is of the form

(i) $a \to xa_1 \dots a_m \quad (m \geq 0,\, a \in N,\, a_1, \dots, a_m \in N - a_0,\, x \in X)$, or

(ii) $a_0 \to e$.

If $m \leq k$ for all productions of type (i), then G is said to be in *Greibach k-form* $(k \geq 0)$.

Proofs for the following facts can be found in the references given at the end of the section.

Theorem 1.6.9 (a) *Every CF grammar (N, X, P, a_0) can be converted into an equivalent reduced CF grammar (N', X, P', a_0), where $N' \subseteq N$ and $P' \subseteq P$.*

(b) *Every CF grammar can be converted into an equivalent CF grammar in any one of the following normal forms: Chomsky normal form, Greibach normal form, and Greibach 2-form. In all cases the grammar can be made reduced.* □

We recall now some of the closure properties of the family CF.

Theorem 1.6.10 *If the languages U and V are CF, then so are $U \cup V$, UV and U^*.* □

The languages $U = \{x^m y^n z^n \mid m, n \geq 1\}$ and $V = \{x^n y^n z^m \mid m, n \geq 1\}$ are CF, but $U \cap V = \{x^n y^n z^n \mid n \geq 1\}$ is not. This observation implies also that the difference $U - V$ of two CF languages U and V may be noncontext-free. However, the following theorem holds.

Theorem 1.6.11 *If U is a CF language and V is a regular language, then $U \cap V$ and $U - V$ are CF languages.* □

The following theorem implies, as a special case, that CF is closed under morphisms.

Lemma 1.6.12 *Let $\varphi\colon \mathfrak{p}X^* \to \mathfrak{p}Y^*$ be a substitution mapping such that $x\varphi \in \mathrm{CF}(Y)$ for all $x \in X$. If $U \in \mathrm{CF}(X)$, then $U\varphi \in \mathrm{CF}(Y)$.* □

The following useful lemma is obtained most naturally by considering derivation trees.

Lemma 1.6.13 *(Bar-Hillel's pumping lemma). For each CF grammar G one can find two natural numbers p and q such that the following holds for every word $w \in L(G)$: if $|w| > p$, then we may write $w = u_1 v_1 w' v_2 u_2$ so that*

(i) $|v_1 w' v_2| \leq q$,

(ii) $v_1 v_2 \neq e$, *and*

(iii) $u_1 v_1^i w' v_2^i u_2 \in L(G)$ *for every* $i \geq 0$. □

Next we recall some decidability properties of CF languages. A CF language is always assumed to be given by a CF grammar generating it.

Theorem 1.6.14 *There are algorithms for deciding the following questions:*

(1) *Is a given word in a given* CF *language?*

(2) *Is a given* CF *language empty?*

(3) *Is a given* CF *language finite?* □

The decidability of the finiteness problem follows from Bar-Hillel's lemma. The other two statements can be justified quite directly.

Theorem 1.6.15 *The following questions are undecidable:*

(a) *Are two given* CF *languages equal?*

(b) *Is the intersection of two given* CF *languages empty? | finite? | regular? | context-free?*

(c) *Is the complement $X^* - U$ of a* CF *X-language U empty? | finite? | regular? | context-free?*

(d) *Is a given* CF *grammar ambiguous?*

(e) *Is a given* CF *language inherently ambiguous?* □

In the previous section we noted that every regular language has a minimal recognizer. One might want to find a CF grammar equivalent to a given one with the smallest possible number of nonterminals (*nonterminal minimization problem*) or with a minimum number of productions (*production minimization problem*). However, the following theorem holds.

Theorem 1.6.16 *Both the nonterminal minimization problem and the production minimization problem are unsolvable.* □

Let n be a fixed natural number. The sum of two n-tuples of nonnegative integers

$$\mathbf{a} = (a_1, \ldots, a_n) \text{ and } \mathbf{b} = (b_1, \ldots, b_n)$$

is formed componentwise:

$$\mathbf{a} + \mathbf{b} = (a_1 + b_1, \ldots, a_n + b_n).$$

Similarly, we put

$$k\mathbf{a} = (ka_1, \ldots, ka_n)$$

for all $k \in \mathbf{N_0}$ and $\mathbf{a} \in \mathbf{N_0^n}$.

A subset K of $\mathbf{N_0^n}$ is called *linear*, if there exist an $m \geq 0$ and n-tuples $\mathbf{a}_1, \ldots, \mathbf{a}_m$, $\mathbf{b} \in \mathbf{N_0^n}$ such that

$$K = \{k_1\mathbf{a}_1 + \ldots + k_m\mathbf{a}_m + \mathbf{b} \mid k_1, \ldots, k_m \in \mathbf{N_0}\}.$$

A subset of $\mathbf{N_0^n}$ is *semilinear* if it is the union of finitely many linear sets.

Let X be an alphabet with n letters ($n \geq 1$). It is convenient to think that the letters of X are listed in some fixed order, $x_1, \ldots, x_n$. The *Parikh vector* of a word $w \in X^*$ is the n-tuple

$$\mathrm{Par}(w) = (a_1, \ldots, a_n)$$

where a_i is the number of occurrences of x_i in w ($i = 1, \ldots, n$). The resulting *Parikh mapping*

$$\mathrm{Par}\colon X^* \to \mathbf{N_0^n}$$

satisfies the conditions

$$\text{(i) } \mathrm{Par}(e) = (0, \ldots, 0)$$

and

$$\text{(ii) } \mathrm{Par}(uv) = \mathrm{Par}(u) + \mathrm{Par}(v) \quad (u, v \in X^*).$$

The mapping Par is extended to X-languages in the natural way:

$$\mathrm{Par}(L) = \{\mathrm{Par}(w) \mid w \in L\}$$

for all $L \subseteq X^*$.

Theorem 1.6.17 *For every* CF *language* L*, the Parikh set* $\mathrm{Par}(L)$ *is semilinear.* □

1.7 SEQUENTIAL MACHINES

Automata that produce outputs in response to inputs are generally called sequential machines. The basic example of these is provided by the Mealy-machine which arose as an abstract model of digital circuits with memory. A *Mealy-machine* is a system $\mathbf{A} = (X, A, Y, a_0, \delta, \lambda)$, where

(1) X is the *input alphabet*,

(2) A is a finite, nonempty set of *states*,

(3) Y is the *output alphabet*,

(4) $a_0 \in A$ is the *initial state*,

(5) $\delta\colon A \times X \to A$ is the *next-state function*, and

(6) $\lambda\colon A \times X \to Y$ is the *output function*.

In many applications there is no fixed initial state, and a_0 is then omitted from the definition. The operation of $\mathbf{A}$ can be described as follows. If $\mathbf{A}$ is in state a $(\in A)$ and receives an input x $(\in X)$, then it enters state $\delta(a, x)$ and emits the letter $\lambda(a, x)$. In order to describe the behaviour of $\mathbf{A}$ under an arbitrary input word $w \in X^*$ we extend δ and λ to mappings

$$\hat{\delta}\colon A \times X^* \to A, \quad \hat{\lambda}\colon A \times X^* \to Y^*$$

as follows:

1° $\hat{\delta}(a, e) = a$ and $\hat{\lambda}(a, e) = e$ for every $a \in A$.

2° $\hat{\delta}(a, wx) = \delta(\hat{\delta}(a, w), x)$ and $\hat{\lambda}(a, wx) = \hat{\lambda}(a, w)\lambda(\hat{\delta}(a, w), x)$ for all $a \in A$, $w \in X^*$, $x \in X$.

If $\mathbf{A}$ receives in state a the input word w, it emits the word $\hat{\lambda}(a, w)$ $(\in Y^*)$ and ends up in state $\hat{\delta}(a, w)$. The *translation* induced by $\mathbf{A}$ is defined as the relation

$$\tau_{\mathbf{A}} = \{(w, \hat{\lambda}(a_0, w)) \mid w \in X^*\} \quad (\subseteq X^* \times Y^*).$$

Two Mealy-machines are said to be *equivalent* if they define the same translation.

In the case of a Mealy-machine $\mathbf{A}$ every input word w has exactly one translation $\hat{\lambda}(a_0, w)$ and this has the same length as w. Mealy-machines enjoy a number of desirable properties and they have a well-developed theory. For example, the following facts are known:

(a) The translations induced by Mealy-machines have a very simple characterization.

(b) The equivalence problem of Mealy-machines is decidable.

(c) For any Mealy-machine one can find an equivalent minimal Mealy-machine and this is unique up to isomorphism.

(d) Let $\mathbf{A}$ be the Mealy-machine defined above. If $L \in \text{Rec } X$, then $L\tau_{\mathbf{A}} \in \text{Rec } Y$. If $L \in \text{Rec } Y$, then $L\tau_{\mathbf{A}}^{-1} \in \text{Rec } X$.

There are several ways to generalize Mealy-machines. First of all, both the next-state and the output behaviour may be nondeterministic. Another generalization allows the sequential machine to emit a word in response to each input letter. Moreover, one may add a set of final states. Then a translation of a word is accepted just in case it leaves the machine in a final state. We shall now define a generalized sequential machine which includes all these features. It is now convenient to use a set of productions which will account both for the next-state behaviour and for the outputs. We arrive at the following concept.

Definition 1.7.1 A *(nondeterministic) generalized sequential machine* (gsm) is a system $\mathbf{A} = (X, A, Y, a_0, P, A')$ where

(1) X is the *input alphabet*,

(2) A is a finite, nonempty set of *states*,

(3) Y is the *output alphabet*,

(4) a_0 $(\in A)$ is the *initial state*,

(5) P is a set of *productions* of the form $ax \to wb$ with $a, b \in A$, $x \in X$ and $w \in Y^*$, and

(6) $A' \subseteq A$ is the set of *final states*.

It is assumed that $A \cap (X \cup Y) = \emptyset$. The gsm $\mathbf{A}$ is said to be *deterministic* if there exists for each pair $(a, x) \in A \times X$ exactly one production of the form $ax \to wb$.

Let $\mathbf{A}$ be the above gsm. A production $ax \to wb$ is interpreted as follows. If $\mathbf{A}$ is in state a and receives the input x, $\mathbf{A}$ may enter state b and simultaneously emit the word w. We shall now define the translation performed by $\mathbf{A}$. For any two words $p, q \in (A \cup X \cup Y)^*$, we write $p \Rightarrow_{\mathbf{A}} q$ if there exist a production $ax \to wb$ in P and words p' and p'' such that $p = p'axp''$ and $q = p'wbp''$. The reflexive, transitive closure of $\Rightarrow_{\mathbf{A}}$ is denoted by $\Rightarrow_{\mathbf{A}}^*$. Thus $p \Rightarrow_{\mathbf{A}}^* q$ $(p, q \in (A \cup X \cup Y)^*)$ holds iff there exists a *derivation* of the form

$$p = p_0 \Rightarrow_{\mathbf{A}} p_1 \Rightarrow_{\mathbf{A}} \ldots \Rightarrow_{\mathbf{A}} p_k = q \quad (k \geq 0).$$

Now, the *translation* induced by $\mathbf{A}$ is defined as the relation

$$\tau_A = \{(u, v) \mid u \in X^*, v \in Y^*, a_0 u \Rightarrow_{\mathbf{A}}^* vb \text{ for some } b \in A'\}.$$

If $(u, v) \in \tau_{\mathbf{A}}$, then v is a *translation* of u. If $\mathbf{A}$ is deterministic, then each X-word w has at most one translation. Two gsm's are *equivalent* if they induce the same translation.

The tree transducers, which form the subject matter of Chapter 4, may be viewed as further generalizations of gsm's in which trees replace words as inputs and as outputs. The following two theorems may be compared with some of the results to be presented in Chapter 4.

Theorem 1.7.2 *Let* $\mathbf{A} = (X, A, Y, a_0, P, A')$ *be a* gsm. *If* $L \in \text{Rec } X$, *then* $L\tau_{\mathbf{A}} \in \text{Rec } Y$. *If* $L \in \text{Rec } Y$, *then* $L\tau_{\mathbf{A}}^{-1} \in \text{Rec } X$. □

Theorem 1.7.3 *The equivalence problem of deterministic* gsm*'s is decidable, but the equivalence problem of nondeterministic* gsm*'s is undecidable.* □

The next-state behaviour of a gsm is identical to that of a nondeterministic Rabin-Scott recognizer. Thus the following fact, which will be needed in Chapter 4, is obvious.

Lemma 1.7.4 *Let* $\mathbf{A}$ *be a* gsm *as defined above. For any two states* $a, b \in A$, *the language*

$$L(a, b) = \{u \in X^* \mid au \Rightarrow_{\mathbf{A}}^* bv \text{ for some } v \in Y^*\}$$

is regular. □

1.8 REFERENCES

Extensive treatments of universal algebra can be found in the following two standard references:

- P. M. Cohn, Universal algebra, D. Reidel, Dordrecht (2. ed. 1981).
- G. Grätzer, Universal algebra, Springer-Verlag, New York (2. ed. 1979).

The following more concise texts may also be recommended:

- H. Lugowski, Grundzüge der universellen Algebra, Teubner, Leipzig (1976).
- H. Werner, Einführung in die allgemeine Algebra, Bibliographisches Institut, Mannheim (1978).

A good introduction to lattice theory (available in German and in French, too):

- G. Szász, Introduction to lattice theory, Academic Press, New York (1963).

Two general texts on finite automata and regular expressions:

- F. Gécseg and I. Peák, Algebraic theory of automata, Akadémiai Kiadó, Budapest (1972).
- A. Salomaa, Theory of automata, Pergamon Press, Oxford (1969).

An extensive algebraic treatment of the theory of finite automata can be found in the following two volumes:

- S. Eilenberg, Automata, languages, and machines, Academic Press, New York (Vol. A 1974, Vol. B 1976).

The general area of formal language theory is covered, for example, by the following books:

- A. V. Aho and J. D. Ullman, The theory of parsing, translation, and compiling, Prentice-Hall, Englewood Cliffs, N. J. (1972).

- M. A. Harrison, Introduction to formal language theory, Addison-Wesley, Reading, Mass. (1978).
- J. E. Hopcroft and J. D. Ullmann, Formal languages and their relation to automata, Addison-Wesley, Reading. Mass. (1969).
- A. Salomaa, Formal languages, Academic Press, New York (1973).

A highly recommendable classic on context-free languages is:

- S. Ginsburg, The mathematical theory of context-free languages, McGraw-Hill, New York (1966).

2 TREE RECOGNIZERS AND RECOGNIZABLE FORESTS

This chapter is devoted to finite-state tree recognizers and the family of forests recognizable by them. Here trees are defined as terms over a finite operator domain, and a forest (or tree language) is just a set of trees. As in the case of formal languages, there are two particularly natural ways to effectively define a forest; a forest can be recognized by an automaton, or it can be generated by a grammar. In Section 2.2 we introduce the tree recognizers which correspond to Rabin–Scott recognizers. It does not make any difference whether Rabin–Scott recognizers are defined to read words from left to right or from right to left, but here we should consider both recognizers that read trees from the leaves down towards the root (frontier-to-root recognizers) and recognizers which work in the opposite direction (root-to-frontier tree recognizers). In both cases the recognizer may be either deterministic or nondeterministic. This gives us four types of finite-state tree recognizers. Three of these define the same family of forests, the family Rec of recognizable forests. Deterministic root-to-frontier recognizers are essentially weaker and they define a proper subfamily of Rec. In Section 2.3 we define regular tree grammars. After having shown that these can be reduced to a very simple normal form, we prove that regular tree grammars generate exactly the recognizable forests. Often it will be convenient to use regular tree grammars in the study of recognizable forests. In Section 2.4 several operations on forests are considered. Many of these arise as a generalization of some basic language operation. Usually Rec can be shown to be closed under such operations. However, one should note that there are often many ways to generalize from languages to forests, and a right choice among the alternatives is essential if one wants to generalize the corresponding results, too. For example, there is a natural generalization of the product of languages with respect to which Rec is not even closed. A related point is demonstrated by the case of tree homomorphisms. Here the greater generality of trees compared with words admits of some entirely new phenomena, such as the copying of subtrees.

In Section 2.5 regular expressions to denote forests are defined, and the appropriate generalized Kleene theorem can then be proved. Section 2.6 contains the minimization theory of deterministic frontier-to-root tree recognizers. In Sections 2.7 to 2.9 the family Rec is characterized in some further ways. Recognizable forests are described by means of congruences of the term algebra, as solutions of fixed-point equations, and in terms of local forests. Moreover, a Medvedev-type characterization in terms of certain elementary forests and elementary operations is given. In Section 2.10 we show that the emptiness, the finiteness, and the equivalence problems of recognizable forests are decidable. Section 2.11 is devoted to deterministic root-to-frontier recognizers. The forests recognizable by them are characterized by means of a certain closure property. Furthermore, we show that these recognizers have canonical minimal forms.

In this chapter we try to cover the central parts of what could be called "the generalized theory of finite automata", but many topics had to be excluded. Some of these are mentioned in the Notes and references. There we shall also indicate a few other developments not directly related to this chapter as well as some applications of the theory of tree automata.

2.1 TREES AND FORESTS

The "trees" which appear in tree automata theory may be visualized as tree-like directed labelled graphs. Such a tree has exactly one node, the root, to which no edge enters. From the root there is exactly one path to every node. Moreover, it is essential that the edges leaving a given node have a specified left-to-right order. This concept has been formalized in several ways, but the variations in the definition are of little or no consequence. We shall choose a definition that suits well an algebraic treatment of the theory.

For the labelling of the nodes of a tree we need two alphabets of different kind, a ranked alphabet and a frontier alphabet. As a rule, these two are assumed to be disjoint. A *ranked alphabet* is a finite nonempty operator domain (cf. Sect. 1.2). From now on Σ always represents a ranked alphabet. Other symbols to be used for ranked alphabets include Ω and Γ. The inclusion $\Sigma \subseteq \Omega$ means that $\Sigma_m \subseteq \Omega_m$ for all $m \geq 0$. If $\Sigma_m \cap \Omega_n = \emptyset$ whenever $m \neq n$, then $\Sigma \cup \Omega$ may be defined:

$$(\Sigma \cup \Omega)_m = \Sigma_m \cup \Omega_m \text{ for all } m \geq 0.$$

A *frontier alphabet* is simply an alphabet in the usual sense, but sometimes we should let it be empty. In fact, in most cases there is no need to exclude this possibility. Our usual symbols for frontier alphabets are X, Y and Z.

For any Σ and X, a ΣX*-tree* is simply a ΣX-term. Thus the set of ΣX-trees is $F_\Sigma(X)$. In many cases Σ or X, or both, are either understood or unspecified. In such cases we often speak about Σ*-trees*, X*-trees* or just *trees*. A similar situation will arise whenever a concept involves a ranked alphabet and a frontier alphabet. We shall not lengthen such definitions by listing the modified names, but they will be used without explanation whenever convenient.

The letters p, q, r, s and t are reserved for trees.

Although trees are defined as strings, they can be visualized as, and are in fact intended as representations of, such tree structures as described above.

Example 2.1.1 Let $\Sigma = \Sigma_0 \cup \Sigma_1 \cup \Sigma_2$ be a ranked alphabet, where $\Sigma_0 = \{\gamma\}$, $\Sigma_1 = \{\omega\}$ and $\Sigma_2 = \{\sigma\}$. As the frontier alphabet we take $X = \{x, y\}$. Then $t = \omega\,(\sigma(y, \sigma(\gamma, x)))$ is the ΣX-tree shown in Fig. 2.1. □

Any other way of writing ΣX-terms would suit our purpose equally well. For example, in Polish notation the tree t of Example 2.1.1 would be written as $\omega\sigma y\sigma\gamma x$, but it would still be treated in tree automaton theory as the "tree" shown in Fig. 2.1.

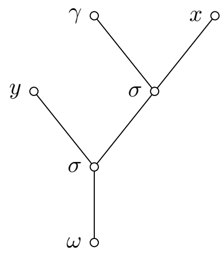

Figure 2.1.

Term induction will now be called *tree induction.* Below some important concepts are defined by tree induction.

Definition 2.1.2 The *height* $\mathrm{hg}(t)$, the *root* $\mathrm{root}(t)$ and the set of *subtrees* $\mathrm{sub}(t)$ of a ΣX-tree t are defined as follows:

1° If $t \in X \cup \Sigma_0$, then $\mathrm{hg}(t) = 0$, $\mathrm{root}(t) = t$ and $\mathrm{sub}(t) = \{t\}$.

2° If $t = \sigma(t_1, \ldots, t_m)$ $(m > 0)$, then

$$\begin{aligned} \mathrm{hg}(t) &= \max(\mathrm{hg}(t_i) \mid i = 1, \ldots, m) + 1, \\ \mathrm{root}(t) &= \sigma, \text{ and} \\ \mathrm{sub}(t) &= \bigcup(\mathrm{sub}(t_i) \mid 1 \le i \le m) \cup t. \end{aligned}$$

For the tree of Example 2.1.1 we get $\mathrm{hg}(t) = 3$, $\mathrm{root}(t) = \omega$ and $\mathrm{sub}(t) = \{t, \sigma\,(y, \sigma(\gamma, x))\,,$ $y, \sigma(\gamma, x), \gamma, x\}$.

Subtrees of height 0 are referred to as the *leaves* of the tree. A leaf is labelled by a letter from the frontier alphabet or by a nullary operator. The *length* $|t|$ of a tree t is simply its length as a word. The leaves of tree t of our example are y, γ and x. Its length is 15 (when parentheses and commas are counted, too). Of course, one can define and prove things about trees by induction on the length; but in practice this mostly reduces to tree induction. Induction on the height $\mathrm{hg}(t)$ is equivalent to tree induction.

We shall use the term *frontier* in a rather informal way to designate the part of a tree consisting of the leaves. The frontier of the tree of Example 2.1.1 consists of the nodes labelled by y, γ and x. The same letter or nullary operator could appear several times as a leaf in the frontier. The visual picture of a tree also suggests the notions of a *branch* and that of a *path.* In our t there are two main branches leaving the lower σ. They correspond to the subtrees y and $\sigma(\gamma, x)$. There are three paths from the root to the frontier. They spell out the words $\omega\sigma y$, $\omega\sigma\sigma\gamma$ and $\omega\sigma\sigma x$, respectively. These terms are used in a descriptive manner to aid the intuition and no precise definitions are needed.

Note. In the literature the root is often called the “top” of the tree, while its frontier is referred to as the “bottom”. Then “top-down” indicates the direction from the root towards the frontier, and “bottom-up” means the opposite direction. This terminology is connected with the common practice of drawing trees upside-down.

The same tree may occur several times as a subtree of a given tree and one should distinguish between a subtree and an *occurrence of a subtree*. It is possible to assign coordinates to the nodes of a tree and then indicate a certain occurrence of a subtree by the coordinates of its root. However, the following simple device to specify an occurrence of a subtree will suffice. For any occurrence of a subtree s of a tree t, there is a unique way to write $t = usv$. Here u and v are just words and the occurrence of s is uniquely determined by u.

We shall now consider some ways to construct new trees from given ones. The very definition of $F_\Sigma(X)$ suggests such a construction. If $m \geq 0$, $\sigma \in \Sigma_m$ and $t_1, \ldots, t_m \in F_\Sigma(X)$, then $\sigma(t_1, \ldots, t_m)$ is a new ΣX-tree which could be called the *σ-catenation* of $t_1, \ldots, t_m$. It is obtained by connecting the roots of the trees $t_1, \ldots, t_m$ to a new root labelled by σ. The construction is illustrated by Fig. 2.2.

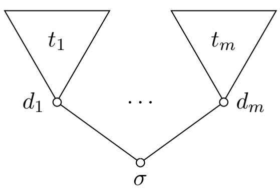

Figure 2.2.

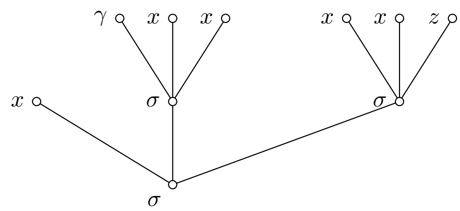

Figure 2.3.

Note that the σ-catenation is the σ-operation of the ΣX-term algebra $\mathcal{F}_\Sigma(X)$:

$$\sigma(t_1, \ldots, t_m) = \sigma^{\mathcal{F}_\Sigma(X)}(t_1, \ldots, t_m).$$

Let t be a ΣX-tree and suppose we are given a tree s_x for every $x \in X$. The tree denoted by

$$t(x \leftarrow s_x \mid x \in X), \quad \text{or just} \quad t(x \leftarrow s_x),$$

is obtained by substituting in t, simultaneously for every $x \in X$, s_x for each occurrence of x. The formal definition by tree induction reads as follows:

1° If $t = z \in X$, then $t(x \leftarrow s_x) = s_z$.

2° If $t = \sigma \in \Sigma_0$, then $t(x \leftarrow s_x) = \sigma$.

3° If $t = \sigma(t_1, \ldots, t_m)$, then

$$t(x \leftarrow s_x) = \sigma\left(t_1(x \leftarrow s_x), \ldots, t_m(x \leftarrow s_x)\right).$$

If the trees s_x are ΣX-trees, then $t(x \leftarrow s_x)$ is also a ΣX-tree. However, the construction works also in the more general case where the trees s_x are ΩY-trees for some Ω and Y such that $\Sigma_m \cap \Omega_n = \emptyset$ whenever $m \neq n$. Then $t(x \leftarrow s_x) \in F_{\Sigma\cup\Omega}(Y)$.

Suppose $X = \{x_1, \ldots, x_n\}$. One may then write $t(x \leftarrow s_x)$ in the more explicit form

$$t\left(x_1 \leftarrow s_{x_1}, \ldots, x_n \leftarrow s_{x_n}\right).$$

If the order $x_1, \dots, x_n$ is understood, we may write simply $t\,(s_{x_1}, \dots, s_{x_n})$.

A letter x may be left unrewritten by choosing $s_x = x$. The notation $t(x_1 \leftarrow s_1, \dots, x_n \leftarrow s_n)$ is used more generally to indicate a substitution where the letters x_i are rewritten as the corresponding s_i $(i = 1, \dots, n)$, but all the other letters of X are left unchanged in the tree t.

Example 2.1.3 Suppose $\gamma \in \Sigma_0$, $\sigma \in \Sigma_3$ and $x, y, z \in X$. If $t = \sigma\,(y, \sigma(\gamma, x, y), z)$, then

$$t\,(y \leftarrow x, z \leftarrow \sigma(x, x, z)) = \sigma\,(x, \sigma(\gamma, x, x), \sigma(x, x, z))\,.$$

The tree is shown in Fig. 2.3. □

Often a certain occurrence of a subtree s of a tree t should be replaced by a tree r. If the presentation $t = usv$ indicates the particular occurrence of s, then the result is urv. It is easy to show that urv is also a ΣX-tree whenever $t, r \in F_\Sigma(X)$. The operation may also be described as follows. Let ξ be a new letter. There is a unique tree $t' \in F_\Sigma(X \cup \xi)$ with exactly one occurrence of ξ such that $t = t'(\xi \leftarrow s)$. Then $urv = t'(\xi \leftarrow r)$. Other ways to operate on trees will be encountered later on.

Trees define polynomial functions in algebras. These will be very important, and we shall now see how the basic tree operations are reflected in them. Let $\mathcal{A} = (A, \Sigma)$ be a Σ-algebra. If $t \in F_\Sigma(X)$ is obtained by σ-catenation from the trees $t_1, \dots, t_m$ $(m \geq 0, \sigma \in \Sigma_m)$, then

$$t^{\mathcal{A}} = \sigma^{\mathcal{A}}(t_1^{\mathcal{A}}, \dots, t_m^{\mathcal{A}})$$

is simply the composition of $t_1^{\mathcal{A}}, \dots, t_m^{\mathcal{A}}$ with $\sigma^{\mathcal{A}}$. Now we consider the substitution operation. Let $X = \{x_1, \dots, x_n\}$ and $t, s_1, \dots, s_n \in F_\Sigma(X)$. The polynomial function

$$t(s_1, \dots, s_n)^{\mathcal{A}} : A^X \to A$$

is computed as follows. For any $\alpha : X \to A$,

$$t(s_1, \dots, s_n)^{\mathcal{A}}(\alpha) = t^{\mathcal{A}}(\beta),$$

where $\beta : X \to A$ is defined so that $x_i\beta = s_i^{\mathcal{A}}(\alpha)$ for all $i = 1, \dots, n$.

Finally, consider the replacing of an occurrence of a subtree s of a ΣX-tree t by a ΣX-tree r. Write $t = t'(\xi \leftarrow s)$ as explained above. For any $\alpha : X \to A$, we get then

$$t'(\xi \leftarrow r)^{\mathcal{A}}(\alpha) = t'^{\mathcal{A}}(\alpha')$$

where $\alpha' : X \cup \xi \to A$ is defined so that $\alpha'|X = \alpha$ and $\xi\alpha = r^{\mathcal{A}}(\alpha)$.

A ΣX-forest is simply a subset of $F_\Sigma(X)$. Many authors call forests *tree languages*. In general, we use the letters R, S and T for forests.

If $\Sigma \subseteq \Omega$ and $X \subseteq Y$, then all ΣX-trees are ΩY-trees, too. Thus every ΣX-forest may be viewed as an ΩY-forest. In most cases this can safely be done. For example, a ΣX-forest is recognizable (in the sense defined in the next section) as a ΣX-forest iff it is recognizable as an ΩY-forest.

Of course, those forests only are of interest that can be defined in some natural way. This chapter is devoted to a family of such forests, the forests recognizable by finite tree automata. In the theory of these forests many concepts and results familiar from the theory of recognizable languages can be perceived. The generalization from words and languages to trees and forests will be considered in the next section.

2.2 TREE RECOGNIZERS

In this section we introduce tree recognizers, that is, tree automata which define forests. There are four basic types of these recognizers. A tree recognizer may be defined in such a way that it reads its input trees from the frontier towards the root. Then it is called a *frontier-to-root recognizer*, or an *F-recognizer* for short. A tree recognizer which reads the trees starting at the root proceeding then towards the frontier is called a *root-to-frontier recognizer*, or simply an *R-recognizer*. In both cases the recognizer may be either *deterministic* or *nondeterministic*. As a rule, all tree recognizers considered here are *finite*, i.e., they have a finite number of states.

Our first task will be to compare the families of forests recognizable by these four types of tree recognizers. It turns out that we get just two families. Deterministic F-recognizers, nondeterministic F-recognizers and nondeterministic R-recognizers all have the same recognition power. The forests recognized by them are termed *recognizable*. Deterministic R-recognizers are considerably weaker and they yield a rather special subfamily of the recognizable forests.

As stated in the previous section, Σ is always a ranked alphabet and X is a frontier alphabet.

Definition 2.2.1 A *frontier-to-root ΣX-recognizer* or an *$(F)\Sigma X$-recognizer*, for short, $\mathbf{A}$ consists of

(1) a finite Σ-algebra $\mathcal{A} = (A, \Sigma)$,

(2) an *initial assignment* $\alpha : X \to A$ and

(3) a set $A' \subseteq A$ of *final states*.

We write $\mathbf{A} = (\mathcal{A}, \alpha, A')$ or $\mathbf{A} = (A, \Sigma, X, \alpha, A')$. The *forest recognized* by $\mathbf{A}$ is the ΣX-forest

$$T(\mathbf{A}) = \{t \in F_\Sigma(X) \mid t^{\mathcal{A}}(\alpha) \in A'\}.$$

A ΣX-forest T is said to be *recognizable*, if there exists a ΣX-recognizer $\mathbf{A}$ such that $T = T(\mathbf{A})$. The family of recognizable forests is denoted by Rec, and $\mathrm{Rec}(\Sigma, X)$ denotes the set of all recognizable ΣX-forests.

The recognizers defined above are finite and deterministic although this has not been emphasized in the name. They are our "basic" type of tree recognizer and we shall usually omit the label "F" which distinguishes them from root-to-frontier tree recognizers. The elements of the underlying algebra $\mathcal{A}$ are called the *states* of $\mathbf{A}$ and A is its *state set*.

If not otherwise specified, **A** will be the ΣX-recognizer $(\mathcal{A}, \alpha, A')$. Also **B** and **C** will usually be the ΣX-recognizers $(\mathcal{B}, \beta, B')$ and $(\mathcal{C}, \gamma, C')$, respectively. Here $\mathcal{B} = (B, \Sigma)$ and $\mathcal{C} = (C, \Sigma)$ are Σ-algebras, $\beta : X \to B$ and $\gamma : X \to C$ are the initial assignments, and $B' \subseteq B$ and $C' \subseteq C$.

In algebraic terms the operation of the ΣX-recognizer **A** can be explained as follows. Given an input tree $t \in F_\Sigma(X)$ the polynomial function $t^{\mathcal{A}}$ is evaluated on the initial assignment α. The tree is accepted exactly in case the result $t^{\mathcal{A}}(\alpha)$ is a final state. If

$$\hat{\alpha} : \mathcal{F}_\Sigma(X) \to \mathcal{A}$$

is the extension of α to a homomorphism, then

$$t^{\mathcal{A}}(\alpha) = t\hat{\alpha} \quad \text{for every} \quad t \in F_\Sigma(X),$$

and we may write

$$T(\mathbf{A}) = \{t \in F_\Sigma(X) \mid t\hat{\alpha} \in A'\} = A'\hat{\alpha}^{-1}.$$

A more pictorial description of the operation of **A** in automata theoretic terms is also possible. Given an input tree t, **A** starts reading it from the leaves in states that depend on the labels of the leaves. If a certain leaf is labelled by a frontier letter x, then **A** is in state $x\alpha$ at that leaf. If the label is a nullary operator σ, then **A** starts from that leaf in state $\sigma^{\mathcal{A}}$. Now **A** moves down all the branches towards the root step by step as follows. If a given node v is labelled by the m-ary operator σ $(m > 0)$, then **A** enters v in state $\sigma^{\mathcal{A}}(a_1, \ldots, a_m)$, where $a_1, \ldots, a_m$ are the states of **A** at the nodes immediately above v, listed in order from left to right. The tree is accepted if **A** enters the root in a final state.

Example 2.2.2 Let $\Sigma = \Sigma_1 \cup \Sigma_2$, $\Sigma_1 = \{\sim\}$, $\Sigma_2 = \{\wedge, \vee\}$ and $X = \{x, y\}$. Define the operations of the Σ-algebra $\mathcal{A} = (\{0,1\}, \Sigma)$ by the tables below:

a	$\sim^{\mathcal{A}}(a)$
0	1
1	0

a	b	$\wedge^{\mathcal{A}}(a,b)$	$\vee^{\mathcal{A}}(a,b)$
0	0	0	0
0	1	0	1
1	0	0	1
1	1	1	1

Define an initial assignment so that $x\alpha = 1$ and $y\alpha = 0$. To complete the definition of our ΣX-recognizer **A** we choose $\{1\}$ as the set of final states. The computation of **A** on the tree

$$t = \wedge(\sim(\wedge(y,x)), \vee(\sim(y), x))$$

is shown in Fig. 2.4. The states of **A** at the nodes are shown in parentheses. The tree is accepted since the state at the root is 1. Let $\sim, \wedge$ and $\vee$ have their usual meanings as symbols for the logical connectives "not", "and" and "or". Then ΣX-trees are expressions of propositional logic in the two propositional variables x and y. If 0 and 1 are interpreted as the truth values "false" and "true", respectively, then **A** computes the truth values of propositions, when the truth values of the variables are given. The forest recognized by **A** consists of the propositions (in variables x and y) that are true when x is true and y is false. □

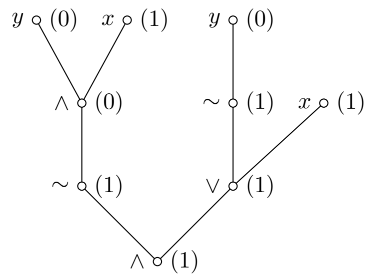

Figure 2.4.

Example 2.2.3 Let $\Sigma = \Sigma_2 = \{+, \cdot\}$ and $X = \{x_1, \ldots, x_n\}$ for some $n \geq 1$. The ΣX-trees may now be interpreted as arithmetic expressions in variables $x_1, \ldots, x_n$. Using the customary infix notation one could write, for example $x_1 + x_1 \cdot x_2$ rather than $+(x_1, \cdot(x_1, x_2))$. Let $m > 0$ and define the Σ-algebra $\mathcal{A} = (\{0, 1, \ldots, m-1\}, \Sigma)$ so that

$$a +^{\mathcal{A}} b = a + b \pmod{m}$$

and

$$a \cdot^{\mathcal{A}} b = a \cdot b \pmod{m}$$

for all $a, b = 0, 1, \ldots, m-1$. If t is a ΣX-tree and $\alpha : X \to A$ is any mapping, then $t^{\mathcal{A}}(\alpha)$ is the value of the expression $t \pmod{m}$ when the variables are assigned values according to α. Thus any ΣX-recognizer $\mathbf{A} = (\mathcal{A}, \alpha, A')$ based on the algebra $\mathcal{A}$ recognizes a set of arithmetic expressions which get a value (mod m) in A' when each variable x_i is given a certain value $x_i\alpha$ $(i = 1, \ldots, n)$. □

The examples suggest some useful general observations on tree recognizers. A tree recognizer is a device that evaluates an expression (a tree) for given values of the variables (given by the initial assignment) and decides then on the basis of this value whether the expression belongs to given set or not. Since the state set is finite such an evaluation is always "modulo something". For example, we could not construct a tree recognizer which would find out whether the value of an arithmetic expression is a prime or not. Similarly, there is no tree recognizer that recognizes the set of all trees in which two given operators appear the same number of times. The following example discusses another manifestation of the same phenomenon.

Example 2.2.4 Let $\Sigma = \Sigma_2 = \{\sigma\}$ and let X be an arbitrary nonempty frontier alphabet. Then the forest

$$T = \{\sigma(t, t) \mid t \in F_\Sigma(X)\}$$

is not recognizable. For suppose $T = T(\mathbf{A})$ for some ΣX-recognizer $\mathbf{A}$. Since A is finite, there must exist two different ΣX-trees s and t such that $s\hat{\alpha} = t\hat{\alpha}$. But then we would have that

$$\sigma(s, t)\hat{\alpha} = \sigma^{\mathcal{A}}(s\hat{\alpha}, t\hat{\alpha}) = \sigma^{\mathcal{A}}(s\hat{\alpha}, s\hat{\alpha}) = \sigma(s, s)\hat{\alpha} \in A',$$

which implies the contradiction $\sigma(s, t) \in T$. □

Let us now look how tree recognizers arise as generalizations of the Rabin–Scott recognizers through a universal algebraic interpretation. First, let $\mathbf{A} = (A, I, \delta, a_0, A')$ be an I-recognizer as defined in Sect. 1.5 (to avoid confusion we use I as the input alphabet). Define a ranked alphabet Σ such that $\Sigma_1 = I$ and $\Sigma_m = \emptyset$ for all $m \neq 1$. The next-state mapping of $\mathbf{A}$ is completely determined by the Σ-algebra $\mathcal{A} = (A, \Sigma)$ which is defined so that

$$\sigma^{\mathcal{A}}(a) = \delta(a, \sigma) \quad \text{for all} \quad a \in A \quad \text{and} \quad \sigma \in I.$$

If we put $X = \{x\}$, then I-words and ΣX-trees can be identified as follows. The empty word e corresponds to the tree x, and a nonempty word $\sigma_1 \ldots \sigma_k$ ($k \geq 1, \sigma_i \in I$) may be interpreted as the tree $\sigma_k(\ldots \sigma_1(x) \ldots)$ (the reverse Polish notation for trees would make the identification even more natural). Define $\alpha : X \to A$ so that $x\alpha = a_0$. Then

$$\delta(a_0, t) = t^{\mathcal{A}}(\alpha) \quad \text{for all} \quad t \in I^*(= F_\Sigma(X)!).$$

This implies that the forest recognized by the ΣX-recognizer $(\mathcal{A}, \alpha, A')$ is, interpreted as an I-language, the language recognized by $\mathbf{A}$. Hence a Rabin–Scott recognizer may be viewed as a tree recognizer over a unary ranked alphabet and a one-element frontier alphabet. The general ΣX-recognizers result when one does not require Σ to be unary and allows also an arbitrary frontier alphabet X.

The nondeterministic frontier-to-root tree recognizers that we soon shall define may be viewed as generalized F-tree recognizers in which nondeterminism is allowed both in the assignment of states to the leaves and in the next-state behaviour. First we have to introduce nondeterministic operations and nondeterministic algebras.

An *m-ary nondeterministic* (ND) *operation* on a set A is a mapping from A^m to $\mathfrak{p}A$ ($m \geq 0$). Thus an m-ary ND operation

$$f : A^m \to \mathfrak{p}A$$

assigns to every m-tuple of elements from A a subset of A. A nullary ND operation

$$f : \{\emptyset\} \to \mathfrak{p}A$$

fixes a subset of A, and f may be identified with this subset $f(\emptyset)$. A *nondeterministic* (ND) Σ-*algebra* $\mathcal{A} = (A, \Sigma)$ consists of a nonempty set A and a family $\{\sigma^{\mathcal{A}} \mid \sigma \in \Sigma\}$ of ND operations on A such that for each $\sigma \in \Sigma$, $\sigma^{\mathcal{A}}$ is m-ary if $\sigma \in \Sigma_m$. The ND Σ-algebra is *finite* if A is finite. A Σ-algebra may be viewed as an ND Σ-algebra when elements $a \in A$ are identified with the corresponding singletons $\{a\}$.

On the other hand, we associate with every ND Σ-algebra $\mathcal{A} = (A, \Sigma)$ an ordinary Σ-algebra, namely the subset algebra

$$\mathfrak{p}\mathcal{A} = (\mathfrak{p}A, \Sigma)$$

where

$$\sigma^{\mathfrak{p}\mathcal{A}}(A_1, \ldots, A_m) = \bigcup \left(\sigma^{\mathcal{A}}(a_1, \ldots, a_m) \mid a_1 \in A_1, \ldots, a_m \in A_m\right)$$

for all $m \geq 0$, $\sigma \in \Sigma_m$ and $A_1, \ldots, A_m \subseteq A$. Now any mapping

$$\alpha : X \to \mathfrak{p}A$$

may be extended to a homomorphism

$$\hat{\alpha} : \mathcal{F}_\Sigma(X) \to \mathfrak{p}\mathcal{A}.$$

Consider a ΣX-tree t. The computation of the set $t\hat{\alpha}$ may be described in automata theoretic terms as follows. If a leaf is labelled by a letter x, then the "automaton" $\mathcal{A}$ may start at that leaf in any one of the states in $x\alpha$. If a leaf is labelled by a nullary operator, then $\sigma^{\mathcal{A}}$ is the set of the possible starting states. Let v be any node in the tree labelled by an m-ary symbol σ ($m > 0$). Let $\sigma(t_1, \ldots, t_m)$ be the subtree of t which has v as its root. Then $t_1\hat{\alpha}, \ldots, t_m\hat{\alpha}$ are the respective sets of possible states of $\mathcal{A}$ at the nodes immediately above v. Now $\mathcal{A}$ may enter v in any one of the states from $\sigma^{\mathfrak{p}\mathcal{A}}(t_1\hat{\alpha}, \ldots, t_m\hat{\alpha})$. Clearly, $t\hat{\alpha}$ is the set of all states in which $\mathcal{A}$ may be at the root of t.

Definition 2.2.5 A *nondeterministic frontier-to-root ΣX-recognizer*, or an NDF ΣX-recognizer for short, $\mathbf{A}$ consists of

(1) a finite ND Σ-algebra $\mathcal{A} = (A, \Sigma)$,

(2) an *initial assignment* $\alpha : X \to \mathfrak{p}A$ and

(3) a set $A' \subseteq A$ of *final states.*

We write $\mathbf{A} = (\mathcal{A}, \alpha, A')$ or $\mathbf{A} = (A, \Sigma, X, \alpha, A')$. The *forest recognized* by $\mathbf{A}$ is the ΣX-forest

$$T(\mathbf{A}) = \{t \in F_\Sigma(X) \mid t\hat{\alpha} \cap A' \neq \emptyset\}.$$

The definition of $T(\mathbf{A})$ means that a tree t is accepted by $\mathbf{A}$ iff there is a set of choices of initial states for the leaves and next-states for the other nodes such that $\mathbf{A}$ enters the root of t in a final state. It is rather obvious that the ΣX-recognizer

$$\mathfrak{p}\mathbf{A} = (\mathfrak{p}A, \alpha, A''),$$

where

$$A'' = \{A_1 \in \mathfrak{p}A \mid A_1 \cap A' \neq \emptyset\},$$

recognizes the same forest as $\mathbf{A}$. Indeed, for any $t \in F_\Sigma(X)$,

$$\begin{aligned} t \in T(\mathfrak{p}\mathbf{A}) \quad &\text{iff} \quad t^{\mathfrak{p}\mathcal{A}}(\alpha) \in A'' \quad \text{iff} \quad t\hat{\alpha} \in A'' \\ &\text{iff} \quad t\hat{\alpha} \cap A' \neq \emptyset \quad \text{iff} \quad t \in T(\mathbf{A}). \end{aligned}$$

This is the natural generalization of the usual subset construction as applied to ND Rabin–Scott recognizers, and $\mathfrak{p}\mathbf{A}$ is the "subset recognizer" corresponding to $\mathbf{A}$. Since every ΣX-recognizer may be viewed as an equivalent NDF ΣX-recognizer we have verified the following theorem.

Theorem 2.2.6 *The forests recognized by nondeterministic frontier-to-root recognizers are exactly the recognizable forests.* □

We begin the discussion of root-to-frontier tree recognizers with the nondeterministic version. In a *nondeterministic root-to-frontier* Σ*-algebra* (NDR Σ*-algebra*, for short) $\mathcal{A} = (A, \Sigma)$, A is a nonempty set and every $\sigma \in \Sigma_m$ with $m \geq 1$ is realized as a mapping

$$\sigma^{\mathcal{A}} : A \to \mathfrak{p}(A^m).$$

For $\sigma \in \Sigma_0$, $\sigma^{\mathcal{A}}$ is a subset of A. We call $\mathcal{A}$ *finite*, if A is finite.

Definition 2.2.7 A *nondeterministic root-to-frontier* ΣX*-recognizer* $\mathbf{A}$, or an NDR ΣX*-recognizer*, consists of

(1) a finite NDR Σ-algebra $\mathcal{A} = (A, \Sigma)$,

(2) a set $A' \subseteq A$ of *initial states*, and

(3) a *final assignment* $\alpha : X \to \mathfrak{p}A$.

We write $\mathbf{A} = (\mathcal{A}, A', \alpha)$ or $\mathbf{A} = (A, \Sigma, X, A', \alpha)$. The elements of A are called *states*.

In order to make the formal definition of the forest recognized by such an $\mathbf{A}$ easier to understand, we shall first describe its intended operation. At the root of a given ΣX-tree t, $\mathbf{A}$ may be in any initial state $a \in A'$. Consider now any node v of t labelled by some $\sigma \in \Sigma_m$ with $m \geq 1$. If a is a possible state of $\mathbf{A}$ at v and $(a_1, \ldots, a_m) \in \sigma^{\mathcal{A}}(a)$, then $\mathbf{A}$ may assume state a_1 at the leftmost node immediately above v, state a_2 at the node immediately to the right of this node etc. For every m-tuple in $\sigma^{\mathcal{A}}(a)$, $\mathbf{A}$ has such a sequence of possible next-states for the nodes directly above v. Note that the possible states at these nodes are connected with each other: $(a_1, \ldots, a_m), (a_1', \ldots, a_m') \in \sigma^{\mathcal{A}}(a)$ does not imply, for example, $(a_1', a_2, \ldots, a_m) \in \sigma^{\mathcal{A}}(a)$. The tree t is accepted by $\mathbf{A}$ if it is possible to choose the initial state for the root and then make the consecutive choices of next-state vectors in such a way that $\mathbf{A}$ arrives at each leaf labelled by a frontier letter x in a state belonging to $x\alpha$, and at each leaf labelled by a 0-ary symbol σ in a state belonging to $\sigma^{\mathcal{A}}$. It is easier to formalize this recognition process by tracing it from the leaves back to the root. The idea is to see which states at each node can lead to acceptance. For the leaves this is clear. If a leaf is labelled by $x \in X$, then the accepting states for that leaf form the set $x\alpha$. If a leaf is labelled by $\sigma \in \Sigma_0$, then the accepting states are those belonging to $\sigma^{\mathcal{A}}$. Now one can infer the states that are accepting at the nodes immediately below the leaves. When these have been found, we may determine the states in which $\mathbf{A}$ should be at nodes one level deeper in a tree. Finally one finds out the accepting states for the root. The tree is accepted iff at least one of these is an initial state.

Definition 2.2.8 Let $\mathbf{A} = (\mathcal{A}, A', \alpha)$ be an NDR ΣX-recognizer. A mapping

$$\tilde{\alpha} : F_\Sigma(X) \to \mathfrak{p}A$$

is defined as follows:

1° If $x \in X$, then $x\tilde{\alpha} = x\alpha$.

2° If $\sigma \in \Sigma_0$, then $\sigma\tilde{\alpha} = \sigma^{\mathcal{A}}$.

3° If $t = \sigma(t_1, \ldots, t_m)$ $(m \geq 1)$, then
$t\tilde{\alpha} = \{a \in A \mid \sigma^{\mathcal{A}}(a) \cap (t_1\tilde{\alpha} \times \ldots \times t_m\tilde{\alpha}) \neq \emptyset\}$.

The *forest recognized* by **A** is the ΣX-forest

$$T(\mathbf{A}) = \{t \in F_\Sigma(X) \mid t\tilde{\alpha} \cap A' \neq \emptyset\}.$$

Example 2.2.9 Let us consider again the arithmetic expressions, defined in Example 2.2.3. We shall construct an NDR $\Sigma\{x_1, x_2\}$-recognizer which accepts an expression in variables x_1 and x_2 iff the value of the expression is divisible by 4 when $x_1 = 0$ or 2 (mod 4) and $x_2 = 3$ (mod 4). An obvious choice for a state set is $A = \{0, 1, 2, 3\}$. The set of initial states is $\{0\}$, and the final assignment is defined by $x_1\alpha = \{0, 2\}$ and $x_2\alpha = \{3\}$. The next-state behaviour is determined by inferring the possible summands or factors from the sum or product, respectively. We get

$$+^{\mathcal{A}}(0) = \{(0,0), (1,3), (2,2), (3,1)\}$$

$$+^{\mathcal{A}}(1) = \{(0,1), (1,0), (2,3), (3,2)\}$$

etc., and

$$\cdot^{\mathcal{A}}\,(0) = \{0\} \times A \cup A \times \{0\} \cup \{(2,2)\}$$

$$\cdot^{\mathcal{A}}(1) = \{(1,1), (3,3)\}$$

etc..

Note that we would get an equivalent NDF-recognizer by "inverting" these operations $(0 +^{\mathcal{A}} 0 = 0$ etc.), and making $\{0\}$ the set of final states and α the initial assignment. □

The concluding observation of Example 2.2.9 can be generalized as follows. We say that the NDF ΣX-recognizer $\mathbf{A} = (A, \Sigma, X, \alpha, A')$ and the NDR ΣX-recognizer $\mathbf{B} = (B, \Sigma, X, B', \beta)$ are *associated* if

(1) $A = B, A' = B'$ and $\alpha = \beta$,

(2) $(a_1, \ldots, a_m) \in \sigma^{\mathcal{B}}(a)$ iff $a \in \sigma^{\mathcal{A}}(a_1, \ldots, a_m)$, for all $m \geq 1$, $\sigma \in \Sigma_m$ and $a_1, \ldots, a_m, a \in A$, and

(3) $\sigma^{\mathcal{A}} = \sigma^{\mathcal{B}}$ for every $\sigma \in \Sigma_0$.

It easy to see that $\hat{\alpha} = \tilde{\beta}$ if **A** and **B** are associated. Since every NDF tree recognizer has an associated NDR tree recognizer, and conversely, we get

Theorem 2.2.10 *The forests recognizable by NDR tree recognizers are exactly the recognizable forests.* □

A *deterministic root-to-frontier* ΣX*-recognizer*, or a DR ΣX*-recognizer*, is a NDR ΣX-recognizer $\mathbf{A} = (\mathcal{A}, A', \alpha)$ such that A' and all of the sets $\sigma^{\mathcal{A}}(a)$ $(\sigma \in \Sigma_m, m \geq 1, a \in A)$ and $\sigma^{\mathcal{A}}$ with $\sigma \in \Sigma_0$ contain exactly one element. Thus a DR ΣX-recognizer $\mathbf{A}$ has exactly one initial state and in every situation there is exactly one choice of next-state vector. Moreover, there is exactly one final state for each leaf labelled by a nullary symbol. The forest recognized by $\mathbf{A}$ is defined the same way as in the general case.

That determinism is a real limitation in the case of root-to-frontier recognizers is shown by the following example.

Example 2.2.11 Suppose $\sigma \in \Sigma_2$ and $x, y \in X$. If a DR ΣX-recognizer accepts the trees $\sigma(x, y)$ and $\sigma(y, x)$, then it must accept $\sigma(x, x)$, too. Hence, the forest $T = \{\sigma(x, y), \sigma(y, x)\}$ cannot be recognized by any DR ΣX-recognizer. On the other hand, it is obvious that $T \in \mathrm{Rec}(\Sigma, X)$. □

The inability of these recognizers to cope with situations such as that in Example 2.2.11 is due to the fact that they have to read disjoint subtrees separately without any possibility to combine the information gathered from the individual subtrees. In an NDR tree recognizer this handicap is compensated for by their ability to make several guesses about the subtrees jointly before reading them separately.

2.3 REGULAR TREE GRAMMARS

So far, the recognizable forests have been characterized by means of three types of tree recognizers. Now we shall introduce a class of tree grammars that also defines the family of recognizable forests. These grammars are the natural counterparts to type 3 grammars.

Definition 2.3.1 A *regular* ΣX*-grammar* G consists of

(1) a finite nonempty set N of *nonterminal symbols*,

(2) a finite set P of *productions* of the form $a \to r$, where $a \in N$ and $r \in F_\Sigma(N \cup X)$, and

(3) an *initial symbol* $a_0 \in N$.

It is assumed that $N \cap (\Sigma \cup X) = \emptyset$. We write $G = (N, \Sigma, X, P, a_0)$.

When Σ and X are not specified, we speak about *regular tree grammars* or just *grammars*, if there is no danger of confusion.

Let G be a regular tree grammar as in the definition above. The right-hand side of a production is a tree in which nonterminal symbols may appear at the leaves only. For $p, q \in F_\Sigma(X \cup N)$, we write

$$p \Rightarrow_G q \quad \text{(or just } p \Rightarrow q)$$

if there exist $a \in N, r \in F_\Sigma(X \cup N)$ and words u, v such that $p = uav, q = urv$ and $a \to r \in P$, i.e., $p \Rightarrow_G q$ means that q is obtained by replacing an occurrence of a nonterminal symbol a by a tree r, where $a \to r$ is a production of the grammar. More generally, we write

$$p \Rightarrow_G^* q \quad \text{(or just } p \Rightarrow^* q\text{)}$$

if $p = q$ or there exists a (nontrivial) derivation

$$p \Rightarrow_G p_1 \Rightarrow_G \ldots \Rightarrow_G p_{n-1} \Rightarrow_G q \quad (n \geq 1)$$

of q from p. Hence, $\Rightarrow^*$ is the reflexive, transitive closure of $\Rightarrow$, when we view it as a relation in $F_\Sigma(X \cup N)$.

Definition 2.3.2 The *forest generated* by a regular ΣX-grammar $G = (N, \Sigma, X, P, a_0)$ is the ΣX-forest

$$T(G) = \{t \in F_\Sigma(X) \mid a_0 \Rightarrow_G^* t\}.$$

Two regular ΣX-grammars G_1 and G_2 are said to be *equivalent* if $T(G_1) = T(G_2)$.

Example 2.3.3 Let $\Sigma = \Sigma_0 \cup \Sigma_2$, $\Sigma_0 = \{\omega\}$, $\Sigma_2 = \{\sigma\}$ and $X = \{x\}$. Define the regular ΣX-grammar

$$G = (\{a, b\}, \Sigma, X, P, a),$$

where

$$P = \{a \to \sigma(x, \sigma(x, b)), a \to \sigma(\omega, a), b \to \sigma(x, x)\}.$$

The tree

$$t = \sigma\,(\omega, \sigma(x, \sigma(x, \sigma(x, x))))$$

is in $T(G)$ and it has the derivation

$$a \Rightarrow \sigma(\omega, a) \Rightarrow \sigma(\omega, \sigma(x, \sigma(x, b))) \Rightarrow t.$$

If the graphical representation of trees is used, this derivation can be written as in Fig. 2.5. □

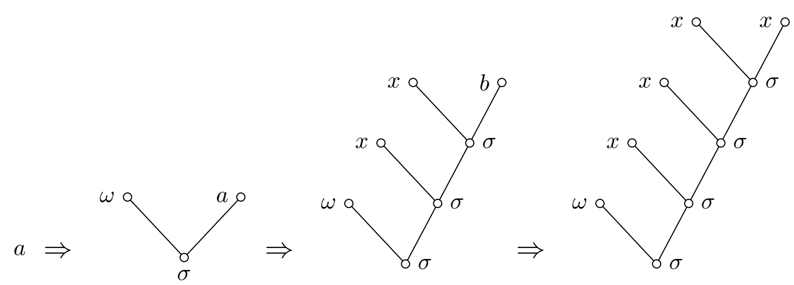

Figure 2.5.

A regular ΣX-grammar may be viewed as a context-free grammar with a terminal alphabet consisting of Σ, X, the parentheses and the comma. Thus, if we treat trees as words, then the forests generated by regular tree grammars are special CF languages. However, we are mainly interested in them as forests, and we shall prove that exactly the recognizable forests can be generated by these grammars. To facilitate the proof first we show that the form of the productions may be restricted considerably without limiting the generative power of regular tree grammars.

To begin with, we note that productions of the form

$$a \to b \quad (a, b \in N)$$

are not needed. All such productions can be deleted if we add to P all productions $a \to r$ $(a \in N, r \in F_\Sigma(X \cup N) - N)$ such that $a \Rightarrow^* b$ and $b \to r \in P$ for some $b \in N$. (It is easy to see that $a \Rightarrow^* b$ is decidable for $a, b \in N$.)

Call $\mathrm{hg}(r)$ the *height* of the production $a \to r$. If the height of a production $a \to r$ is > 1, then r is of the form $\sigma(r_1, \ldots, r_m)$, where $m \geq 1, \sigma \in \Sigma_m$ and $\mathrm{hg}(r_i) < \mathrm{hg}(r)$ for each $i = 1, \ldots, m$. If we introduce new nonterminal symbols $a_1, \ldots, a_m$ and the productions

$$a \to \sigma(a_1, \ldots, a_m) \tag{*}$$

and

$$a_i \to r_i \quad (i = 1, \ldots, m), \tag{**}$$

then the production $a \to r$ may be deleted without changing the forest generated. Indeed, any application of $a \to r$ can be replaced by an application of (*) followed by applications of the productions (**). On the other hand, none of the productions (**) can be used unless (*) has first been used, and when (*) has been applied it must be followed by applications of all productions (**) as there is no other way to rewrite the new nonterminals a_i. The total effect of these steps is the same as that of a single application of $a \to r$. Thus every production of height > 1 can be replaced by productions of lesser height. The process can be repeated until there are no productions of height > 1. In (**) there may be productions of the type $a \to b$, but they can be eliminated. Hence each production of height 0 may be assumed to be of the type

$$a \to x \quad (a \in N, x \in X) \tag{i}$$

or of the form

$$a \to \sigma \quad (a \in N, \sigma \in \Sigma_0). \tag{ii}$$

A production of height 1 is of the form

$$a \to \sigma(r_1, \ldots, r_m) \quad (m \geq 1, \sigma \in \Sigma_m, a \in N),$$

where each r_i is a frontier letter, a 0-ary operator or a nonterminal symbol. If r_i is a letter from X or a 0-ary operator, then we may substitute a new nonterminal symbol d for it and introduce the production $d \to r_i$ of height 0 without changing the forest generated. Thus we may assume that all productions of height 1 are of the form

$$a \to \sigma(a_1, \ldots, a_m) \quad (m \geq 1, \sigma \in \Sigma_m, a, a_1, \ldots, a_m \in N). \tag{iii}$$

We say that a regular tree grammar is in *normal form* if each of its productions is of type (i), (ii) or (iii). The previous discussion amounts to the following lemma.

Lemma 2.3.4 *Every regular tree grammar can be transformed into an equivalent regular tree grammar in normal form.* □

Example 2.3.5 None of the productions of the grammar considered in Example 2.3.3 is in normal form. The production $a \to \sigma(x, \sigma(x, b))$ can be replaced by the following set:

$$a \to \sigma(a_1, a_2), \quad a_1 \to x, \quad a_2 \to \sigma(a_1, b).$$

Notice that we could use the new nonterminal symbol a_1 twice since in both functions it should be rewritten as x. Similarly, the production $a \to \sigma(\omega, a)$ is replaced by the two productions

$$a \to \sigma(a_3, a) \quad \text{and} \quad a_3 \to \omega,$$

and the production $b \to \sigma(x, x)$ is replaced by $b \to \sigma(a_1, a_1)$ (we already have $a_1 \to x$). We have got a grammar in normal form with five nonterminal symbols a, b, a_1, a_2 and a_3, and the productions

$$\begin{array}{rrr} a \to \sigma(a_1, a_2), & a \to \sigma(a_3, a), & b \to \sigma(a_1, a_1), \\ a_1 \to x, & a_2 \to \sigma(a_1, b) & \text{and} \quad a_3 \to \omega. \end{array}$$

□

The following minor generalization of regular tree grammars is introduced as a technical aid. An *extended regular ΣX-grammar*

$$G = (N, \Sigma, X, P, A')$$

is defined otherwise exactly as a regular ΣX-grammar, but it has a set $A' \subseteq N$ of initial symbols. Also $\Rightarrow_G^*$ is defined the same way as for regular tree grammars. The forest generated by such a G is

$$T(G) = \{t \in F_\Sigma(X) \mid a_0 \Rightarrow_G^* t \text{ for some } a_0 \in A'\}.$$

It is immediately clear that every language generated by an extended regular tree grammar can be generated by an ordinary regular tree grammar, too.

Theorem 2.3.6 *The forests generated by regular tree grammars are exactly the recognizable forests.*

Proof. We associate with every NDF ΣX-recognizer $\mathbf{A} = (A, \Sigma, X, \alpha, A')$ an extended regular ΣX-grammar

$$G = (A, \Sigma, X, P, A'),$$

where

$$P = \{a \to x \mid x \in X, a \in x\alpha\} \cup \{a \to \sigma \mid \sigma \in \Sigma_0, a \in \sigma^{\mathcal{A}}\} \cup$$
$$\{a \to \sigma(a_1, \ldots, a_m) \mid m \geq 1, \sigma \in \Sigma_m, a, a_1, \ldots, a_m \in A, a \in \sigma^{\mathcal{A}}(a_1, \ldots, a_m)\}.$$

The grammar G is in normal form (i.e., the productions are of type (i)–(iii)). It is clear that every extended regular ΣX-grammar in normal form arises this way from a NDF ΣX-recognizer. To prove the theorem it suffices now to show that $T(\mathbf{A}) = T(G)$ for such an associated pair $\mathbf{A}$ and G. To do this we show by tree induction that

$$a \in t\hat{\alpha} \quad \text{iff} \quad a \Rightarrow_G^* t \tag{*}$$

holds for all $a \in A$ and $t \in F_\Sigma(X)$.

1° For $t = x \in X$, $a \in x\hat{a}$ iff $a \to x \in P$ iff $a \Rightarrow^* x$ (here we needed the fact that G has no productions of the form $a \to b$).

2° The case $t = \sigma \in \Sigma_0$ is similar: $a \in \sigma\hat{\alpha}$ iff $a \in \sigma^{\mathcal{A}}$ iff $a \to \sigma \in P$ iff $a \Rightarrow^* \sigma$.

3° Let $t = \sigma(t_1, \ldots, t_m)$ $\quad (m \geq 1)$ and suppose that (*) holds for $t_1, \ldots, t_m$ and all states. If $a \Rightarrow^* t$, then there is a derivation of the form

$$a \Rightarrow \sigma(a_1, \ldots, a_m) \Rightarrow^* \sigma(t_1, \ldots, t_m),$$

where $a_1, \ldots, a_m \in N$ and

$$a_i \Rightarrow^* t_i \quad \text{for} \quad i = 1, \ldots, m.$$

Then $a \in \sigma^{\mathcal{A}}(a_1, \ldots, a_m)$ by the definition of P, and (*) implies that $a_1 \in t_1\hat{\alpha}, \ldots, a_m \in t_m\hat{\alpha}$. Hence,

$$a \in \sigma^{\mathfrak{p}\mathcal{A}}(t_1\hat{\alpha}, \ldots, t_m\hat{\alpha}) = t\hat{\alpha}.$$

Conversely, $a \in t\hat{\alpha}$ means that

$$a \in \sigma^{\mathcal{A}}(a_1, \ldots, a_m)$$

for some $a_1 \in t_1\hat{\alpha}, \ldots, a_m \in t_m\hat{\alpha}$. But then (*) implies $a_1 \Rightarrow^* t_1, \ldots, a_m \Rightarrow^* t_m$. Also, P contains the production $a \to \sigma(a_1, \ldots, a_m)$ and we get the required derivation

$$a \Rightarrow \sigma(a_1, \ldots, a_m) \Rightarrow^* \sigma(t_1, \ldots, t_m) = t.$$

This completes the proof of (*), and we have for every ΣX-tree t,

$$\begin{aligned} t \in T(\mathbf{A}) \quad &\text{iff} \quad t\hat{\alpha} \cap A' \neq \emptyset \\ &\text{iff} \quad a \in t\hat{\alpha} \ \text{ for some } \ a \in A' \\ &\text{iff} \quad a \Rightarrow_G^* t \ \text{ for some } \ a \in A' \\ &\text{iff} \quad t \in T(G). \end{aligned}$$

Hence $T(\mathbf{A}) = T(G)$ as required. $\square$

2.4 OPERATIONS ON FORESTS

In this section some more insight into the family of recognizable forests is gained by studying its closure properties with respect to various forest operations. In the following definitions and theorems all forests usually have the same ranked alphabet and the same frontier alphabet. To show that this is no serious limitation, we note the following simple fact.

Lemma 2.4.1 *Let Σ and Ω be ranked alphabets such that $\Sigma \subseteq \Omega$, and let X and Y be frontier alphabets such that $X \subseteq Y$. Then*

$$\mathrm{Rec}(\Sigma, X) = \mathrm{Rec}(\Omega, Y) \cap \mathfrak{p}F_\Sigma(X). \qquad \square$$

Of course, the lemma presupposes the point of view that every ΣX-forest is also an ΩY-forest. Now let Σ and Ω be any ranked alphabets such that $\Sigma_m \cap \Omega_n = \emptyset$ whenever $m \neq n$. Also, let X and Y be arbitrary frontier alphabets. The lemma implies that if $S \in \mathrm{Rec}(\Sigma, X)$ and $T \in \mathrm{Rec}(\Omega, Y)$, then S and T can be regarded as recognizable forests over a common ranked alphabet $\Sigma \cup \Omega$ and a common frontier alphabet $X \cup Y$.

Theorem 2.4.2 *If $S, T \in \mathrm{Rec}(\Sigma, X)$, then $S \cap T$, $S \cup T$ and $S - T$ are also recognizable ΣX-forests.*

Proof. Suppose S and T are recognized by the ΣX-recognizers $\mathbf{A}$ and $\mathbf{B}$, respectively. Let $\mathcal{C} = \mathcal{A} \times \mathcal{B}$ and define

$$\gamma : X \to C \quad \text{by} \quad x \mapsto (x\alpha, x\beta).$$

Then

$$t\hat{\gamma} = (t\hat{\alpha}, t\hat{\beta}) \quad \text{for all } t \in F_\Sigma(X).$$

This implies that we get from $\mathcal{C}$ and γ ΣX-recognizers for $S \cap T$, $S \cup T$ and $S - T$ by choosing, respectively, as the set of final states $A' \times B'$, $A' \times B \cup A \times B'$, and $A' \times (B - B')$. For example, let

$$\mathbf{C} = (\mathcal{C}, \gamma, A' \times B').$$

For any $t \in F_\Sigma(X)$,

$$\begin{aligned} t \in T(\mathbf{C}) \quad &\text{iff} \quad t\hat{\gamma} = (t\hat{\alpha}, t\hat{\beta}) \in A' \times B' \\ &\text{iff} \quad t \in T(\mathbf{A}) \cap T(\mathbf{B}). \end{aligned}$$

That is, $T(\mathbf{C}) = S \cap T$. $\square$

Note that the complement $F_\Sigma(X) - T$ of a recognizable ΣX-forest T is recognizable. If T is recognized by a ΣX-recognizer $\mathbf{A}$, then the complement is recognized by $(\mathcal{A}, \alpha, A - A')$.

Definition 2.4.3 Let $(T_x \mid x \in X)$ be an X-indexed family of ΣX-forests. For each ΣX-tree t we define a forest $t(x \leftarrow T_x \mid x \in X)$, mostly written simply $t(x \leftarrow T_x)$, as follows:

1° If $t = z \in X$, then $t(x \leftarrow T_x) = T_z$.

2° If $t = \sigma \in \Sigma_0$, then $t(x \leftarrow T_x) = \sigma$.

3° If $t = \sigma(t_1, \ldots, t_m)(m \geq 1)$, then

$$t(x \leftarrow T_x) = \{\sigma(s_1, \ldots, s_m) \mid s_i \in t_i(x \leftarrow T_x) \text{ for } i = 1, \ldots, m\}.$$

The *forest product* of the family $(T_x \mid x \in X)$ with the ΣX-forest T is defined as the ΣX-forest

$$T(x \leftarrow T_x \mid x \in X) = \bigcup(t(x \leftarrow T_x \mid x \in X) \mid t \in T).$$

We shall usually write just $T(x \leftarrow T_x)$. If T consists of a single ΣX tree t, then

$$T(x \leftarrow T_x) = t(x \leftarrow T_x).$$

The trees $t(x \leftarrow T_x)$ are obtained from t by replacing every occurrence of each letter x by a tree from the corresponding forest T_x. Different occurrences of the same letter x may be rewritten as different trees from T_x.

If $x_1, \ldots, x_n \in X$, then we use the notation

$$T(x_1 \leftarrow T_1, \ldots, x_n \leftarrow T_n)$$

for the forest product $T(x \leftarrow T_x)$, where

$$T_x = \begin{cases} T_i & \text{for} \quad x = x_i \quad (i = 1, \ldots, n), \\ x & \text{for} \quad x \notin \{x_1, \ldots, x_n\}. \end{cases}$$

If the letters $x_1, \ldots, x_n$ and their order are understood, then this notation may be further simplified to $T(T_1, \ldots, T_n)$.

The comments presented at the beginning of the section show that the definition of forest products also includes the cases, where $T \subseteq F_\Sigma(X)$ and $T_x \subseteq F_\Omega(Y)$ $(x \in X)$ for any such alphabets that $\Sigma_m \cap \Omega_n = \emptyset$ whenever $m \neq n$. If T is a ΣX-forest and the forests T_x are ΩY-forests, then $T(x \leftarrow T_x)$ is a $(\Sigma \cup \Omega)Y$-forest.

Example 2.4.4 Let $\Sigma = \Sigma_0 \cup \Sigma_2$, $\Sigma_0 = \{\omega\}$, $\Sigma_2 = \{\sigma\}$, $X = \{x, y\}$ and $Y = \{y, z\}$. If $t = \sigma(x, \sigma(y, x))$, $T_x = \{\sigma(y, z), z\}$ and $T_y = \{\sigma(\omega, y), \sigma(z, z)\}$, then $t(x \leftarrow T_x, y \leftarrow T_y)$ contains eight trees, among them the tree $\sigma(\sigma(y, z), \sigma(\sigma(\omega, y), z))$. □

The following special type of forest products is important.

Definition 2.4.5 Let S and T be ΣX-forests and $z \in X$. The *z-product* of S and T is the forest product

$$S \cdot_z T = T(x \leftarrow T_x \mid x \in X)$$

where $T_z = S$ and $T_x = x$ for all $x \in X$, $x \neq z$.

The trees in $S \cdot_z T$ are obtained by taking a tree t from T and substituting a tree from S for every occurrence of z in t. Different occurrences of z may be replaced by different trees from S.

Theorem 2.4.6 *If $T \in \mathrm{Rec}(\Sigma, X)$ and $T_x \in \mathrm{Rec}(\Sigma, X)$ for all $x \in X$, then $T(x \leftarrow T_x) \in \mathrm{Rec}(\Sigma, X)$. In particular, $\mathrm{Rec}(\Sigma, X)$ is closed under all x-products $(x \in X)$.*

Proof. Here it is convenient to use regular tree grammars. Suppose T and the forests T_x $(x \in X)$ are generated by the regular ΣX-grammars $G = (N, \Sigma, X, P, a_0)$ and $G_x = (N_x, \Sigma, X, P_x, a_x)$ $(x \in X)$, respectively. We may assume that the grammars are in normal form and that their sets of nonterminal symbols are pairwise disjoint. Construct a regular ΣX-grammar

$$G' = (N', \Sigma, X, P', a_0)$$

with $N' = N \cup \bigcup(N_x \mid x \in X)$ and

$$P' = P'' \cup \{a \to a_x \mid x \in X, \quad a \to x \in P\} \cup \bigcup(P_x \mid x \in X),$$

where P'' is P with all productions of the form $a \to x$ $(a \in N, x \in X)$ deleted.

We claim that $T(G') = T(x \leftarrow T_x)$. The idea is that every derivation $a_0 \Rightarrow_G \ldots \Rightarrow_G t$ of a tree $t \in T$ can be imitated by the productions in P'' up to the point where frontier letters $x \in X$ are to be generated. Instead of generating a leaf x one transfers then by a production $a \to a_x$ to the beginning of a derivation which generates any tree $t_x \in T_x$ in place of the leaf. This means that G' can generate all of $T(x \leftarrow T_x)$. On the other hand, every derivation in G' can be brought into this form by rearranging the applications of the productions suitably. Hence, $T(G') \subseteq T(x \leftarrow T_x)$. For a formal proof it suffices to show that

$$a \Rightarrow^*_{G'} p \quad \text{iff} \quad (\exists q \in F_\Sigma(X))\ a \Rightarrow^*_G q, \quad p \in q(x \leftarrow T_x) \qquad (*)$$

holds for all $a \in N$ and $p \in F_\Sigma(X)$. We proceed by tree induction on p. The fact that the grammars G and G_x are in normal form is used without comment.

1° Let $p = y \in X$. Suppose there is a $q \in F_\Sigma(X)$ such that $a \Rightarrow^*_G q$ and $y \in q(x \leftarrow T_x)$. This is possible only in case $q = z$ and $y \in T_z$ for some $z \in X$. Then $a \to z \in P$ and hence $a \to a_z$, $a_z \to y \in P'$. We get the derivation

$$a \Rightarrow_{G'} a_z \Rightarrow_{G'} y.$$

On the other hand, all derivations of y from a in G' are of this form. Hence, if $a \Rightarrow^*_{G'} y$, then $a \to a_z$, $a_z \to y \in P'$ for some $z \in X$. This means that $a \to z \in P$ and $a_z \to y \in P_z$, and thus z is the required tree q.

2° Let $p = \sigma \in \Sigma_0$.

(2a) If there is a q such that $a \Rightarrow^*_G q$ and $\sigma \in q(x \leftarrow T_x)$, then there are two possibilities. The first one is that $q = \sigma$. Then P and P' both contain $a \to \sigma$ and we get the required derivation $a \Rightarrow^*_{G'} \sigma$ in one step. The other possibility is that $q = x \in X$ and P_x contains $a_x \to \sigma$. Then $a \to a_x$ and $a_x \to \sigma$ are in P' and we get the derivation

$$a \Rightarrow_{G'} a_x \Rightarrow_{G'} \sigma.$$

(2b) Suppose $a \Rightarrow^*_{G'} \sigma$. One possibility is that $a \to \sigma \in P'$. Then $a \to \sigma$ is in P, too, and we may choose $q = \sigma$. The only alternative is that the derivation is of the form $a \Rightarrow_{G'} a_x \Rightarrow_{G'} \sigma$ for some $x \in X$. Then $a \to x \in P$ and $\sigma \in T_x$, and we may put $q = x$.

3° Let $p = \sigma(p_1, \ldots, p_m)$ $(m > 0)$.

(3a) Suppose we have a tree q such that $a \Rightarrow^*_G q$ and $p \in q(x \leftarrow T_x)$. Again there are two cases to consider. If $q = z \in X$, then $p \in T_z$, $a \to z \in P$ and $a_z \Rightarrow^*_{G_z} p$. Now $a \to a_z \in P'$ and, since $P_z \subseteq P'$, we get

$$a \Rightarrow_{G'} a_z \Rightarrow^*_{G'} p.$$

The other possibility is that

$$q = \sigma(q_1, \ldots, q_m)$$

for some $q_1, \ldots, q_m \in F_\Sigma(X)$. Then

$$p_i \in q_i(x \leftarrow T_x) \quad (i = 1, \ldots, m)$$

and the derivation $a \Rightarrow^*_G q$ must begin with a step

$$a \Rightarrow_G \sigma(a_1, \ldots, a_m)$$

such that

$$a_i \Rightarrow^*_G q_i \quad \text{for} \quad i = 1, \ldots, m.$$

Our silent inductive assumption yields

$$a_i \Rightarrow^*_{G'} p_i \quad \text{for} \quad i = 1, \ldots, m.$$

Combining these derivations with $a \to \sigma(a_1, \ldots, a_m) \in P'$ we get $a \Rightarrow^*_{G'} p$.

(3b) Suppose $a \Rightarrow^*_{G'} p$. This could mean that $a \to z \in P$ and $a_z \Rightarrow^*_{G_z} p$ for some $z \in X$. Then we may choose $q = z$. The other possibility is that the derivation takes the form

$$a \Rightarrow_{G'} \sigma(a_1, \ldots, a_m) \Rightarrow^*_{G'} \sigma(p_1, \ldots, p_m).$$

Then there exist ΣX-trees q_i such that

$$a_i \Rightarrow^*_G q_i, \quad p_i \in q_i(x \leftarrow T_x) \quad (i = 1, \ldots, m).$$

Now we may put $q = \sigma(q_1, \ldots, q_m)$. □

Next we generalize the iteration operation taking the x-products as the starting point.

Definition 2.4.7 Let T be any ΣX-forest and let $x \in X$. Put $T^{0,x} = \{x\}$ and

$$T^{j+1,x} = T^{j,x} \cdot_x T \cup T^{j,x}$$

for all $j \geq 0$. Then the *x-iteration* of T is the ΣX-forest

$$T^{*x} = \bigcup(T^{j,x} \mid j \geq 0).$$

The forest T^{*x} is obtained as follows. First include x. New members of T^{*x} are obtained by substituting in some $t \in T$ for every occurrence of x some tree already known to be in T^{*x}. Note that $T^{1,x} = T \cup x$ and $T^{j,x} \subseteq T^{j+1,x}$ for every $j \geq 0$.

Theorem 2.4.8 *If $T \in \mathrm{Rec}(\Sigma, X)$, then $T^{*x} \in \mathrm{Rec}(\Sigma, X)$ for each $x \in X$.*

Proof. Let $G = (N, \Sigma, X, P, a_0)$ be a regular tree grammar in normal form generating the forest T. Construct an extended regular ΣX-grammar $G' = (N', \Sigma, X, P', A')$, where

(1) $N' = N \cup \{d\}$ ($d \notin N$),

(2) $P' = P \cup \{d \to x\} \cup \{a \to r \mid a \to x \in P,\ a_0 \to r \in P\}$, and

(3) $A' = \{a_0, d\}$.

It is not hard to see that $T(G') = T^{*x}$. □

The following operation may be seen as a converse to the x-product.

Definition 2.4.9 Let S and T be ΣX-forests and let $x \in X$. The *x-quotient* of T by S is the forest

$$S^{-x}T = \{p \in F_\Sigma(X) \mid S \cdot_x \{p\} \cap T \neq \emptyset\}.$$

If $S = \{s\}$ is a singleton, then we write $S^{-x}T = s^{-x}T$.

A tree p is in $S^{-x}T$ iff one can convert it into a tree in T by substituting for every occurrence of x a tree from S. If Σ is unary and $X = \{x\}$, and if we identify the tree $\sigma_k(\ldots\sigma_1(x)\ldots)$ with the word $\sigma_1 \ldots \sigma_k$, then

$$S^{-x}T = S^{-1}T = \{u \in \Sigma^* \mid Su \cap T \neq \emptyset\}$$

is the usual (left) quotient language.

Theorem 2.4.10 *If $T \in \mathrm{Rec}(\Sigma, X)$ and S is any ΣX-forest, then $S^{-x}T$ is recognizable for every $x \in X$. Moreover, the number of different x-quotients $S^{-x}T$ for any fixed $T \in \mathrm{Rec}(\Sigma, X)$ is finite.*

Proof. Let $\mathbf{A}$ be a ΣX-recognizer for T. We define an NDF ΣX-recognizer

$$\mathbf{B} = (\mathcal{A}, \beta, A')$$

which is identical to $\mathbf{A}$ (when states $a \in A$ and singleton sets $\{a\}$ are identified) except for the initial assignment which is defined so that

$$x\beta = S\hat{\alpha}$$

and

$$z\beta = \{z\alpha\} \quad \text{for all} \quad z \in X, z \neq x.$$

Here $S\hat{\alpha}$ is the set of all states $s\hat{\alpha}$ in which $\mathbf{A}$ may be after reading a tree s from S. By tree induction one verifies that

$$t\hat{\beta} = (S \cdot_x t)\hat{\alpha}$$

for all $t \in F_\Sigma(X)$. Hence

$$\begin{aligned} t \in T(\mathbf{B}) \quad &\text{iff} \quad t\hat{\beta} \cap A' \neq \emptyset \\ &\text{iff} \quad (S \cdot_x t)\hat{\alpha} \cap A' \neq \emptyset \\ &\text{iff} \quad S \cdot_x t \cap T \neq \emptyset \\ &\text{iff} \quad t \in S^{-x}T \end{aligned}$$

for all $t \in F_\Sigma(X)$. This implies $S^{-x}T = T(\mathbf{B})$. The second statement follows from this construction as the number of possible β's is finite. □

Next we introduce the forest operation corresponding to the σ-catenation of trees which was defined in Section 2.1.

Definition 2.4.11 Let $\sigma \in \Sigma$ be an m-ary operator and let $T_1, \dots, T_m$ be m ΣX-forests for some $m \geq 0$. The *σ-product* of the forests $T_1, \dots, T_m$ is the forest

$$\sigma(T_1, \dots, T_m) = \{\sigma(t_1, \dots, t_m) \mid t_1 \in T_1, \dots, t_m \in T_m\}.$$

If $m = 0$, then the σ-product is always $\{\sigma\}$. In general,

$$\sigma(T_1, \dots, T_m) = \{\sigma(x_1, \dots, x_m)\}(x_1 \leftarrow T_1, \dots, x_m \leftarrow T_m).$$

From Theorem 2.4.6 we get the following result which could easily be proved directly, too.

Corollary 2.4.12 *If $\sigma \in \Sigma_m$ and $T_1, \dots, T_m \in \mathrm{Rec}(\Sigma, X)$ ($m \geq 0$), then $\sigma(T_1, \dots, T_m) \in \mathrm{Rec}(\Sigma, X)$.* □

We shall now consider some operations in which forests are generally transformed into forests over another ranked alphabet. The ranked alphabets will be Σ and Ω. Moreover, we introduce for every $m \geq 0$, a new alphabet

$$\Xi_m = \{\xi_1, \dots, \xi_m\}$$

which is assumed to be disjoint from all other alphabets.

Definition 2.4.13 Suppose we are given a mapping

$$h_X : X \to F_\Omega(Y)$$

and for each $m \geq 0$ a mapping

$$h_m : \Sigma_m \to F_\Omega(Y \cup \Xi_m).$$

The *tree homomorphism* determined by these mappings is the mapping

$$h : F_\Sigma(X) \to F_\Omega(Y)$$

defined as follows:

1° $h(x) = h_X(x)$ for each $x \in X$.

2° $h(\sigma(t_1, \ldots, t_m)) = h_m(\sigma)(\xi_1 \leftarrow h(t_1), \ldots, \xi_m \leftarrow h(t_m))$ for all $m \geq 0$, $\sigma \in \Sigma_m$ and $t_1, \ldots, t_m \in F_\Sigma(X)$.

The tree homomorphism h is said to be *linear* if no letter ξ_i appears more than once in $h_m(\sigma)$ for any $m \geq 0$ and $\sigma \in \Sigma_m$.

To define such an h it obviously suffices to give h_X and the mappings h_m for which $\Sigma_m \neq \emptyset$.

Example 2.4.14 Let $\Sigma = \Sigma_2 = \{\,|\,\}$, $\Omega = \Omega_1 \cup \Omega_2$, $\Omega_1 = \{'\}$, $\Omega_2 = \{\vee\}$ and $X = Y = \{x, y\}$. Define h_X and h_2 by the conditions

$$h_X(x) = x, \quad h_X(y) = y \text{ and } h_2(|) = \vee('(\xi_1),'(\xi_2)).$$

If we interpret $|$ as the Sheffer stroke (i.e., the 2-place NAND), $\vee$ as the symbol of disjunction and $'$ as the symbol of negation, then the tree homomorphism h defined by h_X and h_2 transforms $|$-expressions in variables x and y into equivalent expressions which use $\vee$ and $'$ only. If the more customary way to write Boolean expressions is used, we get, for example,

$$\begin{aligned} h((x|y)|(x|x)) &= h(x|y)' \vee h(x|x)' \\ &= (x' \vee y')' \vee (x' \vee x')'. \end{aligned}$$

This tree homomorphism is linear. □

Tree homomorphisms are not really homomorphisms in the sense of algebra. The concept is the result of the dual nature of words. When one generalizes from languages to forests, words are usually treated as unary terms. On the other hand, many concepts in language theory arise from the interpretation of words as elements of a free monoid. Here the initial concept was that of a homomorphism from the free monoid generated by an alphabet Σ to the free monoid generated by another alphabet Ω. Such a homomorphism rewrites every letter in a word over Σ as a word over Ω. When Σ and Ω are now viewed as unary ranked alphabets, this means that every operator from Σ is rewritten as a piece of Ω-tree to be combined with other such pieces to form the image of a given Σ-word. The generalization of such mappings to the case of arbitrary ranked alphabets gives tree homomorphisms.

The following example shows that tree homomorphisms do not always preserve recognizability.

Example 2.4.15 Put $\Sigma = \Sigma_1 = \{\sigma\}$, $X = Y = \{x\}$ and $\Omega = \Omega_2 = \{\omega\}$. Define h_X and h_1 so that

$$h_X(x) = x \text{ and } h_1(\sigma) = \omega(\xi_1, \xi_1).$$

All ΣX-trees are of the type

$$t_k = \sigma(\sigma(\ldots\sigma(x)\ldots)) = \sigma^k(x) \quad (k \geq 0).$$

Obviously, $h(t_0) = h_X(x) = x$ and, for all $k \geq 0$,

$$h(t_{k+1}) = \omega(h(t_k), h(t_k)).$$

Thus $h(F_\Sigma(X))$ consists of the trees

$$s_0 = x, \quad s_1 = \omega(x, x), \ldots, s_{k+1} = \omega(s_k, s_k), \ldots .$$

Suppose $\mathbf{A} = (A, \Omega, Y, \alpha, A')$ is an ΩY-recognizer such that $T(\mathbf{A}) = h(F_\Sigma(X))$. There must exist two integers $i, j \geq 0$, $i \neq j$, such that $s_i\hat{\alpha} = s_j\hat{\alpha}$. But then

$$\omega(s_i, s_j)\hat{\alpha} = \omega^{\mathcal{A}}(s_i\hat{\alpha}, s_j\hat{\alpha}) = \omega^{\mathcal{A}}(s_i\hat{\alpha}, s_i\hat{\alpha}) = s_{i+1}\hat{\alpha} \in A'$$

would imply $\omega(s_i, s_j) \in h(F_\Sigma(X))$. Thus $h(F_\Sigma(X))$ cannot be recognizable. □

The nonpreservation of recognizability in Example 2.4.15 is due to the ability of the tree homomorphism to create arbitrarily large identical subtrees by copying. No tree recognizer can check whether trees of unbounded height are identical or not. Such copying is precluded by linearity, and the following closure theorem holds.

Theorem 2.4.16 *If $h : F_\Sigma(X) \to F_\Omega(Y)$ is a linear tree homomorphism and $T \in \mathrm{Rec}(\Sigma, X)$, then $h(T) \in \mathrm{Rec}(\Omega, Y)$.*

Proof. Let $G = (N, \Sigma, X, P, a_0)$ be a regular tree grammar in normal form generating T. We may assume that G has no superfluous nonterminal symbols from which no ΣX-tree can be generated. Let Σ' and Ω' be the ranked alphabets which are obtained by adding all nonterminal symbols $a \in N$ to Σ and Ω, respectively, as nullary operators. We extend h to a tree homomorphism

$$h' : F_{\Sigma'}(X) \to F_{\Omega'}(Y)$$

by continuing h_0 to a mapping

$$h'_0 : \Sigma_0 \cup N \to F_{\Omega'}(Y)$$

so that $h_0'(a) = a$ for all $a \in N$. Now let

$$G' = (N, \Omega, Y, P', a_0)$$

be the regular ΩY-grammar, where

$$P' = \{a \to h'(p) \mid a \to p \in P\},$$

i.e., G' is obtained simply by replacing in every production $a \to p \in P$ the right-hand side by the tree $h'(p)$. The theorem follows when we show that $T(G') = h(T)$. This again is obvious once we have shown that

$$a \Rightarrow_{G'}^* t \quad \text{iff} \quad (\exists s \in F_\Sigma(X))\, h(s) = t, \quad a \Rightarrow_G^* s \tag{*}$$

holds for all $a \in N$ and $t \in F_\Omega(Y)$. We prove the two directions of (*) separately.

Suppose $a \Rightarrow_{G'}^* t$ for some $a \in N$ and ΩY-tree t. We prove the existence of the required s by induction on the length of the shortest derivation of t from a.

1° If t is obtained by a one-step derivation, then P' contains the production $a \to t$. Then P contains a production $a \to r$ such that $h'(r) = t$. If r does not contain any nonterminal symbols, we may put $s = r$. Otherwise we choose for every $b \in N$ appearing in r a tree $r_b \in F_\Sigma(X)$ such that $b \Rightarrow_G^* r_b$. Let s be the tree obtained by substituting in r these trees for the corresponding nonterminal symbols. Then $h(s) = h'(r) = t$ since h' deletes all nonterminal symbols from r. Moreover,

$$a \Rightarrow_G r \Rightarrow_G^* s,$$

and s is the required tree.

2° Suppose now that the derivation consists of k steps ($k > 1$) and that (*) holds whenever a shorter derivation exists. The first step must be the application of a production $a \to h'(p)$, where $a \to p \in P$. Since G is in normal form,

$$p = \sigma(a_1, \ldots, a_m)$$

for some $m > 0$, $\sigma \in \Sigma_m$ and $a_1, \ldots, a_m \in N$. The derivation of t can now be written in the form

$$a \Rightarrow_{G'} h_m(\sigma)(\xi_1 \leftarrow a_1, \ldots, \xi_m \leftarrow a_m) \Rightarrow_{G'} \ldots \Rightarrow_{G'} t.$$

For each ξ_i ($i = 1, \ldots, m$) which is present in $h_m(\sigma)$ we have a subderivation

$$a_i \Rightarrow_{G'} \ldots \Rightarrow_{G'} t_i (\in F_\Omega(Y))$$

of length less than k. The linearity of h implies that such a ξ_i appears in $h_m(\sigma)$ exactly once, and hence t_i is unique. For every t_i there is an $s_i \in F_\Sigma(X)$ such that $h(s_i) = t_i$ and $a_i \Rightarrow_G^* s_i$. If a certain ξ_i does not appear in $h_m(\sigma)$, then we choose any $s_i \in F_\Sigma(X)$ such that $a_i \Rightarrow_G^* s_i$ and put $t_i = h(s_i)$. With these choices we get a tree

$$s = \sigma(s_1, \ldots, s_m) \in F_\Sigma(X)$$

such that

$$a \Rightarrow_G \sigma(a_1, \ldots, a_m) \Rightarrow_G^* \sigma(s_1, \ldots, s_m) = s$$

and

$$h(s) = h_m(\sigma)(\xi_1 \leftarrow h(s_1), \ldots, \xi_m \leftarrow h(s_m)) = t.$$

Now we shall prove the converse part of (*). Suppose $a \Rightarrow_G^* s$ and $h(s) = t$ for some $a \in N$, $s \in F_\Sigma(X)$ and $t \in F_\Omega(Y)$. To show that this implies $a \Rightarrow_{G'}^* t$ we proceed by induction on the length of the shortest derivation $a \Rightarrow_G \ldots \Rightarrow_G s$.

1° If there is a derivation of length one, then it consists simply of the application of the production $a \to s$. But then $a \to t$ is a production of G' and $a \Rightarrow_{G'} t$ is the required derivation.

2° Suppose now that the derivation is of the form

$$a \Rightarrow_G \sigma(a_1, \ldots, a_m) \Rightarrow_G \ldots \Rightarrow_G \sigma(s_1, \ldots, s_m) = s,$$

where $m > 0$, $\sigma \in \Sigma_m$ and $a_1, \ldots, a_m \in N$. For every $i = 1, \ldots, m$ there is a shorter derivation

$$a_i \Rightarrow_G \ldots \Rightarrow_G s_i.$$

Hence, $a_i \Rightarrow_{G'}^* h(s_i)$ for each $i = 1, \ldots, m$. Moreover, P' contains the production

$$a \to h_m(\sigma)(\xi_1 \leftarrow a_1, \ldots, \xi_m \leftarrow a_m)$$

corresponding to the production $a \to \sigma(a_1, \ldots, a_m)$ of G. Now the required derivation is

$$\begin{aligned} a \Rightarrow_{G'} h_m(\sigma)(\xi_1 \leftarrow a_1, \ldots, \xi_m \leftarrow a_m) &\Rightarrow_{G'} \ldots \\ \Rightarrow_{G'} h_m(\sigma)(\xi_1 \leftarrow h(s_1), \ldots, \xi_m \leftarrow h(s_m)) \\ = h(s) = t. \end{aligned}$$

This concludes the proof. □

Next we show that arbitrary inverse tree homomorphisms preserve recognizability. We need the following technical lemma. Its proof is left as an exercise.

Lemma 2.4.17 *Consider a Σ-algebra $\mathcal{A}$ and a mapping $\alpha : X \to A$, where $X \cap A = \emptyset$. Let*

$$\overline{\alpha} : \mathcal{F}_\Sigma(X \cup A) \to \mathcal{A}$$

be the unique homomorphism such that $\overline{\alpha}|X = \alpha$ and $\overline{\alpha}|A = 1_A$. Then $\overline{\alpha}|\mathcal{F}_\Sigma(X) = \hat{\alpha}$ and

$$p(\xi_1 \leftarrow p_1, \ldots, \xi_k \leftarrow p_k)\hat{\alpha} = p(\xi_1 \leftarrow p_1\hat{\alpha}, \ldots, \xi_k \leftarrow p_k\hat{\alpha})\overline{\alpha}$$

for all $k \geq 0$, $p \in F_\Sigma(X \cup \Xi_k)$ and $p_1, \ldots, p_k \in F_\Sigma(X)$. □

Theorem 2.4.18 *Let $h : F_\Sigma(X) \to F_\Omega(Y)$ be a tree homomorphism. If $T \in \mathrm{Rec}(\Omega, Y)$, then $h^{-1}(T) \in \mathrm{Rec}(\Sigma, X)$.*

Proof. Let $\mathbf{A} = (A, \Omega, Y, \alpha, A')$ be an ΩY-recognizer for T. We construct a ΣX-recognizer $\mathbf{B} = (A, \Sigma, X, \beta, A')$ as follows. For any $m \geq 0$, $\sigma \in \Sigma_m$ and $a_1, \ldots, a_m \in A$, we put

$$\sigma^{\mathcal{B}}(a_1, \ldots, a_m) = h_m(\sigma)(\xi_1 \leftarrow a_1, \ldots, \xi_m \leftarrow a_m)\overline{\alpha},$$

where $\overline{\alpha} : F_\Omega(Y \cup A) \to A$ is the homomorphism for which $\overline{\alpha}|X = \alpha$ and $\overline{\alpha}|A = 1_A$. In the special case $m = 0$, we get $\sigma^{\mathcal{B}} = h_0(\sigma)\overline{\alpha} = h_0(\sigma)\hat{\alpha}$. The initial assignment is defined by putting

$$x\beta = h(x)\hat{\alpha} \quad \text{for all} \quad x \in X.$$

Now a proof by tree induction shows that

$$s\hat{\beta} = h(s)\hat{\alpha}$$

for all $s \in F_\Sigma(X)$. Hence, $s \in T(\mathbf{B})$ iff $h(s) \in T(\mathbf{A})$. This means that $h^{-1}(T) = T(\mathbf{B})$ is recognizable. □

As a conclusion we consider a simple, but very important special type of tree homomorphisms.

Definition 2.4.19 A tree homomorphism $h : F_\Sigma(X) \to F_\Omega(Y)$ is called *alphabetic* if the defining mappings h_X and h_m ($m \geq 0$) satisfy the following conditions:

(1) $h_X(x) \in Y$ for all $x \in X$.

(2) $h_m(\sigma) = \omega(\xi_1, \ldots, \xi_m)$, where $\omega \in \Omega_m$, for all $m \geq 0$, $\sigma \in \Sigma_m$.

An alphabetic tree homomorphism $F_\Sigma(X) \to F_\Omega(Y)$ can be defined only in case $\Omega_m \neq \emptyset$ for all such $m \geq 0$ that $\Sigma_m \neq \emptyset$. Alphabetic tree homomorphisms are often called *projections*.

Consider the general alphabetic tree homomorphism of the definition. For any $t \in F_\Sigma(X)$, the image $h(t)$ is obtained simply by rewriting every x in t as the letter $h_X(x)$ and every $\sigma \in \Sigma_m$ as the operator ω, where $h_m(\sigma) = \omega(\xi_1, \ldots, \xi_m)$. Hence h preserves completely the "shape" of the tree t. Obviously, h is linear. From Theorems 2.4.16 and 2.4.18 we get

Corollary 2.4.20 *Let $h : F_\Sigma(X) \to F_\Omega(Y)$ be an alphabetic tree homomorphism.*

(i) If $T \in \mathrm{Rec}(\Sigma, X)$, then $h(T) \in \mathrm{Rec}(\Omega, Y)$.

(ii) If $T \in \mathrm{Rec}(\Omega, Y)$, then $h^{-1}(T) \in \mathrm{Rec}(\Sigma, X)$. □

2.5 REGULAR EXPRESSIONS. KLEENE'S THEOREM

Kleene's theorem is of central importance in the theory of finite automata and it is quite natural that it was among the first results to be generalized to the theory of tree automata. Although the greater generality adds some technical complications, the standard development of the theory can be followed quite close here, too, once the right generalizations of the basic concepts have been found.

We fix again an arbitrary ranked alphabet Σ and an arbitrary frontier alphabet X. It turns out that some additional frontier symbols are needed in the construction of regular forests. Therefore we will operate with an extended alphabet Z which contains X as a subset.

Definition 2.5.1 The set of *regular* ΣZ*-expressions* $\mathrm{RE}(\Sigma, Z)$ is defined as the smallest set RE such that the following conditions are satisfied:

1° $\emptyset \in \mathrm{RE}$.

2° $\Sigma_0 \cup Z \subseteq \mathrm{RE}$.

3° If $\zeta, \eta \in \mathrm{RE}$, then $(\zeta + \eta) \in \mathrm{RE}$.

4° If $\zeta, \eta \in \mathrm{RE}$ and $z \in Z$, then $(\zeta \cdot_z \eta) \in \mathrm{RE}$.

5° If $\zeta \in \mathrm{RE}$ and $z \in Z$, then $(\zeta^{*z}) \in \mathrm{RE}$.

6° If $m > 0$, $\sigma \in \Sigma_m$, $\eta_1, \ldots, \eta_m \in \mathrm{RE}$, then $\sigma(\eta_1, \ldots, \eta_m) \in \mathrm{RE}$.

Thus regular ΣZ-expressions are strings of symbols from $\Sigma \cup Z$, of commas etc. Parts 2° and 6° of the definition imply that every ΣZ-tree is a regular ΣZ-expression. Regular expressions are intended as representations of forests.

Definition 2.5.2 The forest $|\eta|$ *represented* by a regular expression $\eta \in \mathrm{RE}(\Sigma, Z)$ is defined following the inductive form of Definition 2.5.1:

1° $|\emptyset| = \emptyset$ (the empty forest).

2° If $\eta \in \Sigma_0 \cup Z$, then $|\eta| = \{\eta\}$.

3° $|(\zeta + \eta)| = |\zeta| \cup |\eta|$.

4° $|(\zeta \cdot_z \eta)| = |\zeta| \cdot_z |\eta|$.

5° $|(\zeta^{*z})| = |\zeta|^{*z}$.

6° $|\sigma(\eta_1, \ldots, \eta_m)| = \sigma(|\eta_1|, \ldots, |\eta_m|)$.

Note that the operations in the right-hand sides of $3° - 6°$ are forest operations which have been defined in Section 2.4. It is easy to see that every tree $t \in F_\Sigma(Z)$ represents, as a regular expression, the one-element forest $\{t\}$.

With this interpretation in mind we may simplify regular expressions by omitting parentheses that are not needed in order to specify the intended order of the operations. First of all, the outermost parentheses in $(\zeta + \eta)$, $(\zeta \cdot_z \eta)$ and (ζ^{*z}) are obviously superfluous if the expressions do not appear as parts of other expressions. We may also agree that iterations precede products and that products precede unions. Then the parentheses around ζ^{*z} can always be omitted and, for example,

$$\zeta + \eta \cdot_x \theta^{*y}$$

is interpreted as a short form for

$$(\zeta + (\eta \cdot_x (\theta^{*y}))).$$

Example 2.5.3 Let $\Sigma = \Sigma_0 \cup \Sigma_2$, $\Sigma_0 = \{\omega\}$ and $\Sigma_2 = \{\sigma\}$ and $Z = \{x, y\}$. The forest represented by

$$\eta = \omega \cdot_y \sigma(x, y)^{*y}$$

contains the trees ω, $\sigma(x, \omega)$, $\sigma(x, \sigma(x, \omega))$ etc. Note that y has a purely auxiliary function; it does not appear in any tree of the forest $|\eta|$. □

In the following definition we make the formal distinction between letters that may appear in trees of the forest represented by a regular expression and those letters that are used just to mark leaves to be rewritten when products of forests are formed.

Definition 2.5.4 Suppose a regular ΣZ-expression ζ can be written in the form

$$\zeta = u(\eta \cdot_z \theta)v$$

where $\eta, \theta \in \mathrm{RE}(\Sigma, Z)$ and $z \in Z$. Then every occurrence of z within the string $\cdot_z\theta$ is said to be *bound*. An occurrence of a letter $z \in Z$ which is not bound is *free*. A letter $z \in Z$ is *bound* in $\zeta \in \mathrm{RE}(\Sigma, Z)$, if all occurrences of z in ζ are bound, and it is *free* in ζ if it has at least one free occurrence in ζ. We denote by Z_ζ the set of letters $z \in Z$ free in ζ.

In Example 2.5.3 $Z_\eta = \{x\}$ and y is bound by the y-product.

Lemma 2.5.5 *For any* $\eta \in \mathrm{RE}(\Sigma, Z)$, $|\eta| \in \mathrm{Rec}(\Sigma, Z_\eta)$.

Proof. We proceed by induction following the six parts in Definitions 2.5.1 and 2.5.2.

1° $Z_\emptyset = \emptyset$ and $|\emptyset| = \emptyset \in \mathrm{Rec}(\Sigma, \emptyset)$.

2° For each $z \in Z$, $Z_z = \{z\}$ and $|z| = \{z\} \in \mathrm{Rec}(\Sigma, \{z\})$. For $\sigma \in \Sigma_0$, $Z_\sigma = \emptyset$, but still $|\sigma| = \{\sigma\} \in \mathrm{Rec}(\Sigma, \emptyset)$.

3° If $\eta = \zeta + \theta$, then $Z_\eta = Z_\zeta \cup Z_\theta$ and $|\eta| = |\zeta| \cup |\theta| \in \mathrm{Rec}(\Sigma, Z_\eta)$ by Lemma 2.4.1 and Theorem 2.4.2.

4° If $\eta = \zeta \cdot_z \theta$, then (if we omit the trivial case $z \notin Z_\theta$, $|\eta| = |\theta|$) $Z_\eta = Z_\zeta \cup (Z_\theta - z)$. There are two cases to consider. If $z \notin Z_\zeta$, then $Z_\eta = (Z_\zeta \cup Z_\theta) - z$. From Theorem 2.4.6 we know that $|\eta| \in \mathrm{Rec}(\Sigma, Z_\zeta \cup Z_\theta)$. Thus it suffices to show that no tree $t \in |\eta|$ contains any occurrence of z. But this is obvious since every such t is obtained from some $s \in |\theta|$ by replacing every occurrence of z by a tree from $|\zeta|$, and no tree in $|\zeta|$ contains z. If $z \in Z_\zeta$, then $Z_\eta = Z_\zeta \cup Z_\theta$ and $|\eta| \in \mathrm{Rec}(\Sigma, Z_\eta)$ follows directly from Theorem 2.4.6.

5° If $\eta = \zeta^{*z}$ $(z \in Z)$, then $Z_\eta = Z_\zeta \cup z$. Thus $|\zeta| \in Rec(\Sigma, Z_\eta)$ by Lemma 2.4.1. This implies $|\zeta^{*z}| \in Rec(\Sigma, Z_\eta)$ by Theorem 2.4.8.

6° If $\eta = \sigma(\eta_1, \ldots, \eta_m)$, where $m > 0$, $\sigma \in \Sigma_m$ and $|\eta_i| \in Rec(\Sigma, Z_{\eta_i})$ $(i = 1, \ldots, m)$, then $Z_\eta = Z_{\eta_1} \cup \ldots \cup Z_{\eta_m}$ and every $|\eta_i|$ is also a recognizable ΣZ_η-forest. Corollary 2.4.12 yields now $|\eta| \in \mathrm{Rec}(\Sigma, Z_\eta)$. □

The operations (finite) union, z-product and z-iteration are called the *regular operations.* A forest is *regular* if it can be constructed from finite forests by applying a finite number of regular operations. In view of the preceding discussion regularity can also be defined as follows:

Definition 2.5.6 A ΣX-forest T is *regular* if there exist an alphabet Z $(X \subseteq Z)$ and a regular ΣZ-expression η such that $|\eta| = T$.

Note that an unlimited number of auxiliary letters $(z \in Z - X)$ is allowed in a regular expression representing a regular forest, but that in any particular case just a finite number of them are needed. Lemma 2.5.5 implies now that all regular forests are recognizable. The next lemma contains the converse statement.

Lemma 2.5.7 *For any ΣX-recognizer $\mathbf{A}$ one can construct a regular expression $\eta \in \mathrm{RE}(\Sigma, X \cup A)$ (we assume $X \cap A = \emptyset$) such that $|\eta| = T(\mathbf{A})$.*

Proof. The proof is modelled after the almost standard proof for the corresponding fact in the language case (due to R. McNaughton and H. Yamada (1960)). The notation can be simplified by assuming that

$$A = \{1, 2, \ldots, k\} \quad \text{for some} \quad k \geq 1.$$

As in Lemma 2.4.17 let

$$\overline{\alpha} : \mathcal{F}_\Sigma(X \cup A) \to \mathcal{A}$$

be the homomorphism such that $\overline{\alpha}|X = \alpha$ and $\overline{\alpha}|A = 1_A$. For any $i \in A$, $K \subseteq A$ and h, $0 \leq h \leq k$, we denote by $T(K, h, i)$ the set of all $t \in F_\Sigma(X \cup K)$ such that

(1) $t\overline{\alpha} = i$ and

(2) $s\overline{\alpha} \in \{1, \ldots, h\}$ for all $s \in \mathrm{sub}(t) - (X \cup \Sigma_0 \cup t)$.

Thus $t \in T(K, h, i)$ means that the leaves of t may be labelled, besides frontier letters and nullary symbols, by states from K. Moreover, the computation of $\mathbf{A}$ on t results in state i and the state of $\mathbf{A}$ at any node between the frontier and the root is in the set $\{1, \ldots, h\}$. Obviously,

$$T(\mathbf{A}) = \bigcup (T(\emptyset, k, i) \mid i \in A').$$

It suffices therefore to show that all sets $T(K, h, i)$ are regular. To do this we proceed by induction on the number h.

1° When $h = 0$, no intermediate states between the frontier and the root are allowed. Every tree t in $T(K, 0, i)$ must hence be of one of the following types:

(i) $t = x \in X$ and $x\alpha = i$.

(ii) $t = i \in K$.

(iii) $t = \sigma \in \Sigma_0$ with $\sigma^{\mathcal{A}} = i$.

(iv) $t = \sigma(d_1, \ldots, d_m)$ with $m > 0$, $d_j \in X \cup \Sigma_0 \cup K$ $(j = 1, \ldots, m)$ and $t\overline{\alpha} = i$.

In each case a regular expression for $\{t\}$ can be written. The number of such trees t is finite and we get a regular expression for $T(K, 0, i)$.

2° Suppose we already have a regular expression for each $T(K, j, i)$ such that $j \leq h$ for some $h < k$. We show that

$$T(K, h+1, i) = \tag{*}$$

$$T(K, h, i) \cup T(K, h, h+1) \cdot_{h+1} T(K \cup h+1, h, h+1)^{*h+1} \cdot_{h+1} T(K \cup h+1, h, i)$$

holds for all $K \subseteq A$ and $i \in A$. This will complete the induction because the right-hand side of (*) is obtained by regular operations from forests for which we already have regular expressions.

Let T be the right-hand side of (*). From the construction of T it is obvious that $T \subseteq T(K, h+1, i)$. If $t \in T(K, h+1, i)$, then either $t \in T(K, h, i)$ or t has a proper subtree $s \notin X \cup \Sigma_0$ such that $s\overline{\alpha} = h+1$. In the former case we get $t \in T$ directly. In the second case we have

$$t \in \{p_1, \ldots, p_d\} \cdot_{h+1} \{q_{11}, \ldots, q_{1e_1}\} \cdot_{h+1} \cdots \cdot_{h+1} \{q_{j1}, \ldots, q_{je_j}\} \cdot_{h+1} \{r\},$$

for some

$$p_1, \ldots, p_d \in T(K, h, h+1), q_{11}, \ldots, q_{1e_1}, \ldots, q_{j1}, \ldots, q_{je_j} \in T(K \cup h+1, h, h+1)$$

and $r \in T(K \cup h+1, h, i)$. This means that t belongs to the second part of T. □

Combining Lemma 2.5.5 and Lemma 2.5.7 we get the following generalized form of Kleene's theorem.

Theorem 2.5.8 *A forest is recognizable iff it is regular.* □

2.6 MINIMAL TREE RECOGNIZERS

The number of states is a simple and natural measure of the complexity of a finite automaton. In this section we consider minimal recognizers of forests. In the case of a recognizable forest minimality means simply a minimal number of states, and there is always a minimal recognizer which is unique up to isomorphism. All tree recognizers recognizing a nonregular forest must be infinite and counting the number of states does not make any sense. Nevertheless, the general definition of minimality is such that the minimal recognizer of a forest remains unique even in such a case. The minimal recognizer of a forest can be derived from any recognizer of this forest. If the forest is recognizable, then the minimalization procedure is effective. Otherwise, the finiteness of the recognizers is not needed in this section. Also, some of the concepts and results presented here will be applied to infinite tree recognizers in the next section. Thus we will temporarily drop our general assumption that all tree recognizers dealt with are finite. In all other respects the earlier definitions and conventions remain valid.

We shall now define homomorphisms, congruences and quotients of tree recognizers. The reader may find it helpful to review the corresponding material from Section 1.2 before going on. Algebraic functions and elementary translations (cf. Sect. 1.3) will also be needed.

Definition 2.6.1 A *homomorphism* from a ΣX-recognizer $\mathbf{A}$ to a ΣX-recognizer $\mathbf{B}$ is a mapping $\varphi : A \to B$ such that

(1) φ is a homomorphism from the Σ-algebra $\mathcal{A}$ to the Σ-algebra $\mathcal{B}$,

(2) $\alpha\varphi = \beta$, and

(3) $B'\varphi^{-1} = A'$.

If φ is a homomorphism from $\mathbf{A}$ to $\mathbf{B}$, we write $\varphi : \mathbf{A} \to \mathbf{B}$. A homomorphism of tree recognizers is an *epimorphism* if it is surjective, a *monomorphism* if it is injective, and it is called an *isomorphism* if it is bijective. If there exists an isomorphism $\varphi : \mathbf{A} \to \mathbf{B}$, then we write $\mathbf{A} \cong \mathbf{B}$ and say that $\mathbf{A}$ and $\mathbf{B}$ are *isomorphic*. If there exists an epimorphism $\varphi : \mathbf{A} \to \mathbf{B}$, then $\mathbf{B}$ is said to be an *epimorphic image* of $\mathbf{A}$. A monomorphism is also called an *embedding*.

Part (3) of Definition 2.6.1 means that the final states, and these only, map to final states in a homomorphism. If φ is an epimorphism, then (3) implies $A'\varphi = B'$.

Lemma 2.6.2 *Let* $\mathbf{A}$ *and* $\mathbf{B}$ *be two* ΣX*-recognizers. If there exists a homomorphism* $\varphi : A \to B$*, then* $T(\mathbf{A}) = T(\mathbf{B})$*.*

Proof. The clauses (1) and (2) of Definition 2.6.1 imply together with Lemma 1.3.6 that

$$t^{\mathcal{A}}(\alpha)\varphi = t^{\mathcal{B}}(\alpha\varphi) = t^{\mathcal{B}}(\beta)$$

for every $t \in F_\Sigma(X)$. Now clause (3) shows that

$$\begin{aligned} t \in T(\mathbf{B}) \quad & \text{iff} \quad t^{\mathcal{B}}(\beta) = t^{\mathcal{A}}(\alpha)\varphi \in B' \\ & \text{iff} \quad t^{\mathcal{A}}(\alpha) \in A' \\ & \text{iff} \quad t \in T(\mathbf{A}) \end{aligned}$$

for every $t \in F_\Sigma(X)$, and the lemma follows. □

Definition 2.6.3 A *congruence* of a ΣX-recognizer $\mathbf{A}$ is a congruence ϱ of the algebra $\mathcal{A}$ saturating A', that is, such that $A'\varrho = A'$. The set of all congruence relations of $\mathbf{A}$ is denoted by $C(\mathbf{A})$.

Lemma 2.6.4 $C(\mathbf{A})$ *is a principal ideal of the complete lattice* $C(\mathcal{A})$*, and thus* $(C(\mathbf{A}), \subseteq)$ *is a complete lattice itself, too.*

Proof. It suffices to verify the following simple facts:

(i) $\delta_{\mathbf{A}} \in C(\mathbf{A})$ (which implies $C(\mathbf{A}) \neq \emptyset$).

(ii) $\theta \subseteq \varrho \in C(\mathbf{A})$ and $\theta \in C(\mathcal{A})$ imply $\theta \in C(\mathbf{A})$.

(iii) $\vee(\varrho \mid \varrho \in C(\mathbf{A})) \in C(\mathbf{A})$.

In (iii) the supremum is to be formed in $C(\mathcal{A})$. It is the generating element of the principal ideal. □

In Theorem 2.6.10 we shall get a more useful description of the greatest element of $C(\mathbf{A})$.

Definition 2.6.5 The *quotient* ΣX*-recognizer* of a ΣX-recognizer $\mathbf{A}$ with respect to a congruence ϱ is the ΣX-recognizer

$$\mathbf{A}/\varrho = (\mathcal{A}/\varrho, \alpha_\varrho, A'/\varrho),$$

where α_ϱ is defined so that $x\alpha_\varrho = (x\alpha)\varrho$ for each $x \in X$.

The usual relations between homomorphisms, congruences and quotients hold for tree recognizers, too. Some of them are listed in the following theorem. We omit the proofs since they can be constructed exactly as the corresponding proofs in algebra.

Theorem 2.6.6 (a) *If* $\varrho \in C(\mathbf{A})$*, the natural mapping*

$$\varrho^\natural : A \to A/\varrho,\ a \mapsto a\varrho \quad (a \in A),$$

is an epimorphism $\mathbf{A} \to \mathbf{A}/\varrho$ *(called the* natural epimorphism*).*

(b) *If* $\varphi : \mathbf{A} \to \mathbf{B}$ *is a homomorphism, then the kernel* $\varphi\varphi^{-1}$ *is a congruence of* $\mathbf{A}$ *and the image*

$$\mathbf{A}\varphi = (\mathcal{A}\varphi, \beta, A'\varphi)$$

of $\mathbf{A}$ *is isomorphic to* $\mathbf{A}\varphi\varphi^{-1}$. *(In* $\mathbf{A}\varphi$ $\mathcal{A}\varphi$ *is the* Σ*-algebra* $(A\varphi, \Sigma)$ *such that* $\sigma^{\mathcal{A}\varphi} = \sigma^{\mathcal{B}}|A\varphi$ *and* β *is to be interpreted as a mapping from* X *to* $A\varphi$.*)*

(c) *If* $\pi \subseteq \varrho$ *for some* $\pi, \varrho \in C(\mathbf{A})$, *then* $\mathbf{A}/\varrho$ *is an epimorphic image of* $\mathbf{A}/\pi$. □

From Theorem 2.6.6 and Lemma 2.6.2 we get

Corollary 2.6.7 *If* $\varrho \in C(\mathbf{A})$, *then* $T(\mathbf{A}/\varrho) = T(\mathbf{A})$. □

Thus any congruence of a tree recognizer yields an equivalent recognizer which is an epimorphic image of the original one. If the recognizer is finite and the congruence is nontrivial, then a real reduction in the number of states is achieved. Obviously, the greatest congruence gives the smallest quotient recognizer. The construction of the quotient recognizer involves a merging of states which are equivalent in the sense that one can be substituted for another in any computation without affecting the end result. We shall now give a precise meaning to this equivalence of states and show then that the greatest congruence consists exactly of the pairs of equivalent states.

Definition 2.6.8 Two states a and b of a ΣX-recognizer $\mathbf{A}$ are said to be *equivalent* and we write $a \sim_{\mathbf{A}} b$ or just $a \sim b$, iff

$$(\forall f \in \mathrm{Alg}_1(\mathcal{A})) \quad f(a) \in A' \Longleftrightarrow f(b) \in A'.$$

To get a better intuitive grasp of this definition we recall the fact that for each algebraic function $f \in \mathrm{Alg}_1(\mathcal{A})$ there exists a tree $t \in F_\Sigma(A \cup \xi)$ such that for all $a \in A$,

$$f(a) = t\hat{\alpha}_a,$$

where $\alpha_a : A \cup \xi \to A$ is defined by $\alpha_a|A = 1_A$ and $\xi\alpha_a = a$ (Lemma 1.3.14). This means that $\mathcal{A}$ computes $f(a)$ from the tree t when one assigns state a to all leaves labelled by ξ. On the other hand, every tree $t \in F_\Sigma(A \cup \xi)$ defines this way a unary algebraic function. Such a tree may be thought of as the unprocessed part of a ΣX-tree where a leaf labelled by a state $c \in A$ corresponds to a subtree s such that $s\hat{\alpha} = c$. Once a value $a \in A$ has been assigned to the leaves labelled by ξ the computation may be completed. The equivalence of two states a and b means that the assignments $\xi = a$ and $\xi = b$ give always the same result (mod A') when such a computation is completed.

Definition 2.6.9 The ΣX-recognizer $\mathbf{A}$ is

(a) *reduced* if $\sim_{\mathbf{A}} = \delta_{\mathbf{A}}$,

(b) *connected* if every state of $\mathbf{A}$ is *reachable*, i.e., there exists for every $a \in A$ a tree $t \in F_\Sigma(X)$ such that $t\hat{\alpha} = a$, and $\mathbf{A}$ is

(c) *minimal* if it is connected and reduced.

That a recognizer is reduced means that no two distinct states are equivalent. To be connected means that every state is possible in some computation performed by the recognizer on some tree. By Lemma 1.3.8, a tree recognizer $\mathbf{A}$ is connected iff $X\alpha$ generates $\mathcal{A}$. In the case of a finite recognizer minimality really means a minimal number of states among equivalent recognizers. If a recognizer is not connected, then the nonreachable states can be discarded without changing the forest recognized. If $\mathbf{A}$ is finite and $\sim_{\mathbf{A}} > \delta_{\mathbf{A}}$, then $\mathbf{A}/\sim_{\mathbf{A}}$ is a properly smaller recognizer equivalent to $\mathbf{A}$. Hence, a finite tree recognizer can be minimal with respect to the number of states only if it is minimal in the sense of Definition 2.6.9. The converse will be established later.

Theorem 2.6.10 *For any ΣX-recognizer $\mathbf{A}$, $\sim$ is the greatest congruence of $\mathbf{A}$ and $\mathbf{A}/\sim$ is a reduced ΣX-recognizer equivalent to $\mathbf{A}$.*

Proof. It is obvious that $\sim$ is an equivalence relation on A. Let $a \sim b$ $(a, b \in A)$. For any two unary algebraic functions $f, g \in \mathrm{Alg}_1(\mathcal{A})$, the composition

$$f \circ g : \xi \mapsto g(f(\xi)) \quad (\xi \in A)$$

is a unary algebraic function. Hence

$$g(f(a)) \in A' \quad \text{iff} \quad g(f(b)) \in A',$$

and this implies $f(a) \sim f(b)$. By Lemma 1.3.16, $\sim$ is a congruence of $\mathcal{A}$. If $a \sim b$ and $a \in A'$, then $b = 1_{\mathbf{A}}(b) \in A'$. Thus $A' \sim = A'$ and $\sim$ is a congruence of $\mathbf{A}$. Let ϱ be any congruence of $\mathbf{A}$. If $a \varrho b$ and $f \in \mathrm{Alg}_1(\mathcal{A})$, then $\varrho \in C(\mathcal{A})$ implies $f(a) \varrho f(b)$. Now $A' \varrho = A'$ implies

$$f(a) \in A' \quad \text{iff} \quad f(b) \in A'.$$

Hence $a \sim b$ and we have shown that $\sim$ is the greatest among the congruences of $\mathbf{A}$. Corollary 2.6.7 tells us that $T(\mathbf{A}) = T(\mathbf{A}/\sim)$. That $\mathbf{A}/\sim$ is reduced follows directly from the fact, well-known in universal algebra, that the lattice $C(\mathbf{A}/\sim)$ is isomorphic to the principal dual ideal $[\sim)$ generated by $\sim$ in $C(\mathbf{A})$. Since $\sim$ is the greatest element of $C(\mathbf{A})$, $[\sim)$ is trivial and thus $\sim_{\mathbf{A}/\sim}$ must be the diagonal relation of $A/\sim$. A more direct proof is possible, too. It is not hard to show that $(a\sim) \sim_{\mathbf{A}/\sim} (b\sim)$ implies $a \sim b$, and hence $a\sim = b\sim$. □

The quotient recognizer $\mathbf{A}/\sim_{\mathbf{A}}$ is often called the *reduced form* of $\mathbf{A}$. It is clear from Theorem 2.6.10 that two tree recognizers having isomorphic reduced forms are equivalent. We show that the converse holds for connected recognizers. In other words, equivalent minimal recognizers are shown to be isomorphic.

Theorem 2.6.11 *Let $\mathbf{A}$ and $\mathbf{B}$ be two minimal tree recognizers. If $\mathbf{A}$ and $\mathbf{B}$ are equivalent, then they are also isomorphic.*

Proof. Define $\varphi : A \to B$ so that

$$(t\hat{\alpha})\varphi = t\hat{\beta} \quad \text{for all} \quad t \in F_\Sigma(X).$$

We show that φ gives the required isomorphism from $\mathbf{A}$ to $\mathbf{B}$. This involves the following seven points:

(i) φ associates with every $a \in A$ a state of $\mathbf{B}$ since $\mathbf{A}$ is connected.

(ii) To show that φ is well-defined we consider the possibility that $s\hat{\alpha} = t\hat{\alpha}$ for two ΣX-trees s and t. If $s\hat{\beta} \neq t\hat{\beta}$, then $s\hat{\beta}$ and $t\hat{\beta}$ are nonequivalent and there exists an algebraic function $f \in \mathrm{Alg}_1(\mathcal{B})$ such that $f(s\hat{\beta}) \in B'$ and $f(t\hat{\beta}) \notin B'$ (or conversely). By Lemma 1.3.14 there exists a tree $p \in F_\Sigma(B \cup \xi)$ $(\xi \notin B \cup X)$ such that for all $b \in B$,

$$f(b) = p^{\mathcal{B}}(\beta_b),$$

where $\beta_b : B \cup \xi \to B$ is defined so that $\beta_b|B = 1_B$ and $\xi\beta_b = b$. Since $\mathbf{B}$ is connected there exists for each $b \in B$ a ΣX-tree p_b such that $p_b\hat{\beta} = b$. Let

$$q = p(b \leftarrow p_b \mid b \in B)(\in F_\Sigma(X \cup \xi)).$$

Consider the ΣX-trees $q_s = q(\xi \leftarrow s)$ and $q_t = q(\xi \leftarrow t)$. Now

$$q_s\hat{\beta} = p^{\mathcal{B}}(\beta_{s\hat{\beta}}) = f(s\hat{\beta}) \in B'$$

and

$$q_t\hat{\beta} = p^{\mathcal{B}}(\beta_{t\hat{\beta}}) = f(t\hat{\beta}) \notin B'.$$

If we assign in q to every letter $x \in X$ the value $x\alpha$, we get a function $g \in \mathrm{Alg}_1(\mathcal{A})$ such that for each $a \in A$,

$$g(a) = q^{\mathcal{A}}(\alpha_a)$$

where $\alpha_a : X \cup \xi \to A$ is defined so that $\alpha_a|X = \alpha$ and $\xi\alpha_a = a$. Applying Lemma 1.3.6 we get now

$$g(s\hat{\alpha})\varphi = q^{\mathcal{A}}(\alpha_{s\hat{\alpha}})\varphi = q_s\hat{\alpha}\varphi = q_s\hat{\beta} \in B'$$

and

$$g(t\hat{\alpha})\varphi = q^{\mathcal{A}}(\alpha_{t\hat{\alpha}})\varphi = q_t\hat{\alpha}\varphi = q_t\hat{\beta} \notin B'.$$

This is in contradiction with our original assumption that $s\hat{\alpha} = t\hat{\alpha}$. Hence $q_s \in T(\mathbf{B})$, but $q_t \notin T(\mathbf{B})$. On the other hand, $s\hat{\alpha} = t\hat{\alpha}$ implies $q_s\hat{\alpha} = q_t\hat{\alpha}$, and a contradiction with our assumption that $T(\mathbf{A}) = T(\mathbf{B})$ results.

(iii) Reversing the roles of $\mathbf{A}$ and $\mathbf{B}$ in Part (ii) one sees that $s\hat{\beta} = t\hat{\beta}$ implies $s\hat{\alpha} = t\hat{\alpha}$ for all ΣX-trees s and t. This means that φ is injective.

(iv) φ is surjective since $\mathbf{B}$ is connected.

(v) Let $m \geq 0$, $\sigma \in \Sigma_m$ and $a_1, \ldots, a_m \in A$. There are trees $t_1, \ldots, t_m \in F_\Sigma(X)$ such that $a_1 = t_1\hat{\alpha}, \ldots, a_m = t_m\hat{\alpha}$. Then

$$\begin{aligned}
\sigma^{\mathcal{A}}(a_1, \ldots, a_m)\varphi &= \sigma^{\mathcal{A}}(t_1\hat{\alpha}, \ldots, t_m\hat{\alpha})\varphi \\
&= \sigma(t_1, \ldots, t_m)\hat{\alpha}\varphi \\
&= \sigma(t_1, \ldots, t_m)\hat{\beta} \\
&= \sigma^{\mathcal{B}}(t_1\hat{\beta}, \ldots, t_m\hat{\beta}) \\
&= \sigma^{\mathcal{B}}(a_1\varphi, \ldots, a_m\varphi).
\end{aligned}$$

Hence φ is a homomorphism from $\mathcal{A}$ to $\mathcal{B}$.

(vi) For each $x \in X$, $x\alpha\varphi = x\hat{\alpha}\varphi = x\hat{\beta} = x\beta$. Thus $\alpha\varphi = \beta$.

(vii) If $t\hat{\alpha} \in A'$ ($t \in F_\Sigma(X)$), then $t\hat{\alpha}\varphi = t\hat{\beta} \in B'$ since $t \in T(\mathbf{A}) = T(\mathbf{B})$. Similarly, $t\hat{\alpha}\varphi \in B'$ implies $t\hat{\alpha} \in A'$. Hence, $B'\varphi^{-1} = A'$. □

Corollary 2.6.12 *If* $\mathbf{A}$ *and* $\mathbf{B}$ *are connected* ΣX*-recognizers such that* $T(\mathbf{A}) = T(\mathbf{B})$*, then* $\mathbf{A}/{\sim_\mathbf{A}} \cong \mathbf{B}/{\sim_\mathbf{B}}$. □

For every ΣX-forest T there is at least the infinite ΣX-recognizer

$$\mathbf{F}_T = (\mathcal{F}_\Sigma(X), 1_X, T)$$

where $\mathcal{F}_\Sigma(X) = (F_\Sigma(X), \Sigma)$ is the ΣX-term algebra. Indeed, for each $t \in F_\Sigma(X)$ we have

$$t^{\mathcal{F}_\Sigma(X)}(1_X) = t \in T(\mathbf{F}_T) \quad \text{iff} \quad t \in T.$$

Obviously $\mathbf{F}_T$ is connected. Hence, $\mathbf{F}_T/{\sim}$ is a minimal recognizer for T (the relation $\sim$ will be examined more closely in the next section). To show it we shall verify that every quotient recognizer of a connected tree recognizer is connected.

Let $\varphi : \mathbf{A} \to \mathbf{B}$ be an epimorphism of ΣX-recognizers. If $\mathbf{A}$ is connected, then so is $\mathbf{B}$. Indeed, let b be any state of $\mathbf{B}$. There exists an $a \in A$ such that $a\varphi = b$. Since $\mathbf{A}$ is connected there is a tree $t \in F_\Sigma(X)$ so that $a = t^{\mathcal{A}}(\alpha)$. Using Lemma 1.3.6 we get

$$t^{\mathcal{B}}(\beta) = t^{\mathcal{B}}(\alpha\varphi) = t^{\mathcal{A}}(\alpha)\varphi = a\varphi = b.$$

In particular, $\mathbf{A}/{\sim_\mathbf{A}}$ is connected for every tree recognizer $\mathbf{A}$.

We now have everything needed for the main theorem of the section.

Theorem 2.6.13 *For every forest T there exists a minimal tree recognizer, and it is unique up to isomorphism. If* $\mathbf{A}$ *is any connected recognizer of T, then the minimal recognizer is an epimorphic image of* $\mathbf{A}$*. In fact,* $\mathbf{A}/{\sim_\mathbf{A}}$ *is minimal.* □

The theorem is valid for every forest. It suggests the following two-step procedure for finding the minimal recognizer for T once any recognizer $\mathbf{A}$ of T is given:

1° Discard all nonreachable states from **A**. We get a connected recognizer **B** such that $T(\mathbf{B}) = T$.

2° Reduce **B** by finding $\sim_{\mathbf{B}}$ and then constructing $\mathbf{B}/{\sim_{\mathbf{B}}}$ which is the required minimal recognizer.

Both of these steps become effective when T is a recognizable forest and the given recognizer **A** is finite.

The reachable states of **A** form the subalgebra of $\mathcal{A}$ generated by the subset $X\alpha$. This can be found as follows. Let $H_0 = X\alpha \cup \{\sigma^{\mathcal{A}} \mid \sigma \in \Sigma_0\}$ and put

$$H_{i+1} = H_i \cup \{\sigma^{\mathcal{A}}(a_1, \ldots, a_m) \mid m > 0, \sigma \in \Sigma_m, a_1, \ldots, a_m \in H_i\}.$$

Then

$$H_0 \subseteq H_1 \subseteq \ldots \subseteq A$$

and $H_i = [X\alpha]$ $(i \geq 0)$ if $H_{i+1} = H_i$. Such an i must exist since A is finite.

Suppose now that we have a finite connected ΣX-recognizer **B** and consider step 2°. First one should find $\mathrm{Alg}_1(\mathcal{B})$. It is finite and can be formed repeating the inductive step of Definition 1.3.13 a finite number of times. Then $\sim_{\mathbf{B}}$ can be determined directly, using the definition. Although the minimal recognizer $\mathbf{B}/{\sim_{\mathbf{B}}}$ certainly can be found this way, the procedure would be quite tedious in most cases. A computationally simpler method can be derived from the following lemma. The proof is left as an exercise. The crucial aid is Lemma 1.3.16: an equivalence is a congruence iff it is invariant with respect to all elementary translations.

Lemma 2.6.14 *Define a descending sequence* $\sim_0 \supseteq \sim_1 \supseteq \ldots$ *of equivalences on* **B** *as follows: (i)* $B/{\sim_0} = \{B', B - B'\}$ *and (ii) for all* $i \geq 0$ *and* $a, b \in B$, $a \sim_{i+1} b$ *iff* $a \sim_i b$ *and* $f(a) \sim_i f(b)$ *for all* $f \in \mathrm{ET}(\mathcal{B})$*. Then* $\sim_i = \sim_{\mathbf{B}}$ *if* $\sim_{i+1} = \sim_i$*, and this holds for some* $i < |B|$. □

2.7 ALGEBRAIC CHARACTERIZATIONS OF RECOGNIZABILITY

In this section two strictly algebraic characterizations of the recognizable forests are presented. First some ideas from the previous section are applied to derive a generalization of Nerode's theorem on regular languages and right congruences of the free monoid (cf. Theorem 1.5.6). Then we show that the recognizable forests can be obtained by solving fixed-point equations of a certain kind. Again, there is a well-known precursor in the theory of finite automata. In fact, in the unary case the equations considered here reduce to Arden's equations which give the regular languages as their solutions.

Let Σ and X be fixed and denote the ΣX-term algebra $\mathcal{F}_\Sigma(X)$ by $\mathcal{F}$, for short. In the previous section we noted that each ΣX-forest T has the (infinite) ΣX-recognizer $\mathbf{F}_T = (\mathcal{F}, 1_X, T)$. Consider any ΣX-recognizer **A** such that $T(\mathbf{A}) = T$. It is easy to verify that the extension of the initial assignment $\alpha\colon X \to A$ to a homomorphism

$$\hat{\alpha}\colon \mathcal{F} \to \mathcal{A}$$

is also a homomorphism of ΣX-recognizers from $\mathbf{F}_T$ to $\mathbf{A}$. Indeed, $1_X\hat{\alpha} = \hat{\alpha}$ and $A'\hat{\alpha}^{-1} = T(\mathbf{A}) = T$. The kernel $\hat{\alpha}\hat{\alpha}^{-1}$ is a congruence of $\mathbf{F}_T$ with a congruence class for each reachable state of $\mathbf{A}$. If T is recognizable, $\mathbf{A}$ may be chosen as finite, and then $\hat{\alpha}\hat{\alpha}^{-1}$ is of finite index. Now, suppose $\mathbf{F}_T$ has a congruence ϱ of finite index. Then $\mathbf{F}_T/\varrho$ is a finite ΣX-recognizer such that $T(\mathbf{F}_T/\varrho) = T(\mathbf{F}_T) = T$ (by Corollary 2.6.7). Hence T is recognizable. The congruences of $\mathbf{F}_T$ are simply the congruences of $\mathcal{F}$ which saturate T. Among these there is one of finite index iff the greatest congruence $\sim_{\mathbf{F}_T}$ of $\mathbf{F}_T$ is of finite index. The congruence $\sim_{\mathbf{F}_T}$ ($\sim_T$ for short) is the *Nerode congruence of* T. These observations may be summed up as

Theorem 2.7.1 *For every ΣX-forest T the following three conditions are equivalent:*

(i) $T \in \mathrm{Rec}(\Sigma, X)$.

(ii) The term algebra $\mathcal{F}_\Sigma(X)$ has a congruence of finite index which saturates T.

(iii) The index of the Nerode congruence $\sim_T$ is finite. □

The recognizer $\mathbf{F}_T$ is connected and Theorem 2.6.10 implies therefore that $\mathbf{F}_T/\sim_T$ is the minimal recognizer of the forest T. To find $\sim_T$ for a given ΣX-forest T one could try to apply Definition 2.6.8 to $\mathbf{F}_T$: for any $s, t \in F_\Sigma(X)$,

$$s \sim_T t \quad \text{iff} \quad \big(\forall p \in F_\Sigma(X \cup \xi)\big)\big(p(\xi \leftarrow s) \in T \Longleftrightarrow p(\xi \leftarrow t) \in T\big) \ .$$

A part of Theorem 2.7.1 can be restated as follows.

Corollary 2.7.2 *A ΣX-forest T is recognizable iff there exist a finite Σ-algebra $\mathcal{A}$, a homomorphism $\varphi\colon \mathcal{F}_\Sigma(X) \to \mathcal{A}$ and a subset $A' \subseteq A$ such that $T = A'\varphi^{-1}$.* □

The corollary gives, in fact, just an obvious reformulation of the definition of recognizability. Without going into the subject any further here, we note that in this form recognizability may be defined for subsets of arbitrary algebras (and not just term algebras): a subset T of a Σ-algebra $\mathcal{A}$ is said to be recognizable, if there exist a finite Σ-algebra $\mathcal{B}$, a homomorphism $\varphi\colon \mathcal{A} \to \mathcal{B}$ and a subset $H \subseteq B$ such that $H\varphi^{-1} = T$. If here $\mathcal{A} = \mathcal{F}_\Sigma(X)$, then we get the recognizable ΣX-forests, and if $\mathcal{A}$ is the free monoid X^*, then we get the recognizable X-languages.

As an introduction to the theory of fixed-point equations we first look at an example of Arden equations.

Example 2.7.3 Consider the two-state Rabin-Scott recognizer $\mathbf{A}$ defined by the state graph shown in Fig. 2.6. The input alphabet is $\Sigma = \{\sigma, \tau\}$.
Let L_1 and L_2 be the languages of all words taking $\mathbf{A}$ from the initial state 1 to state 1 and 2, respectively. Then the following equations hold:

$$\begin{aligned} L_1 &= L_1\sigma \cup L_2\sigma \cup e \\ L_2 &= L_1\tau \cup L_2\tau \ . \end{aligned} \tag{1}$$

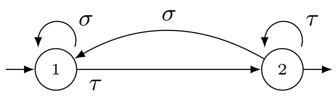

Figure 2.6.

If we define a mapping

$$\hat{\Pi}\colon (\mathfrak{p}\Sigma^*)^2 \to (\mathfrak{p}\Sigma^*)^2$$

so that for all $U, V \subseteq \Sigma^*$,

$$\hat{\Pi}(U,V) = (U\sigma \cup V\sigma \cup e,\ U\tau \cup V\tau) \ ,$$

then (1) means that (L_1, L_2) is a solution of the fixed-point equation

$$(v_1, v_2) = \hat{\Pi}(v_1, v_2) \ . \tag{2}$$

Moreover, (L_1, L_2) is the least solution of (2) when $(\mathfrak{p}\Sigma^*)^2$ is partially ordered in the natural way:

$$(U_1, V_1) \le (U_2, V_2) \quad \text{iff} \quad U_1 \subseteq U_2 \quad \text{and} \quad V_1 \subseteq V_2 \ .$$

If we view Σ as a unary ranked alphabet and identify $\Sigma\{x\}$-trees and Σ-words as shown in Section 2.2 ($x = e$, $\sigma_k(\cdots\sigma_1(x)\cdots) = \sigma_1\cdots\sigma_k$), then the term algebra $\mathcal{F}_\Sigma(\{x\})$ may be taken to be

$$\mathcal{F} = (\Sigma^*, \Sigma) \ ,$$

where

$$\sigma^{\mathcal{F}}(u) = u\sigma \quad (\sigma \in \Sigma,\ u \in \Sigma^*) \ .$$

In the corresponding subset algebra

$$\mathfrak{p}\mathcal{F} = (\mathfrak{p}\Sigma^*, \Sigma)$$

we have the operations

$$\sigma^{\mathfrak{p}\mathcal{F}}(L) = L\sigma \quad (\sigma \in \Sigma,\ L \subseteq \Sigma^*) \ .$$

The mapping $\hat{\Pi}$ can be defined in terms of these operations, the empty word and unions:

$$\hat{\Pi}(U,V) = \big(\sigma^{\mathfrak{p}\mathcal{F}}(U) \cup \sigma^{\mathfrak{p}\mathcal{F}}(V) \cup x,\ \tau^{\mathfrak{p}\mathcal{F}}(U) \cup \tau^{\mathfrak{p}\mathcal{F}}(V)\big) \ .$$

Using forest products we may write this as follows:

$$\begin{aligned} \hat{\Pi}(U,V) = \big(&\{\sigma(v_1),\ \sigma(v_2),\ x\}(v_1 \leftarrow U,\ v_2 \leftarrow V) \ , \\ &\{\tau(v_1),\ \tau(v_2)\}(v_1 \leftarrow U,\ v_2 \leftarrow V)\big) \ . \end{aligned} \tag{3}$$

Finally, we write (2) in the more readable form

$$\begin{aligned} v_1 &= \sigma(v_1) + \sigma(v_2) + x \\ v_2 &= \tau(v_1) + \tau(v_2) \end{aligned} \tag{4}$$

as a system of equations to be solved in the forest algebra $\mathfrak{p}\mathcal{F}$ which is augmented by union as an operation. Union is denoted here by $+$. □

It is obvious that Example 2.7.3 could be repeated for any regular language and that the language itself is always the union of those components of the minimal fixed-point which correspond to final states. The interpretation of the equations in terms of forest operations serves as the starting point for a generalization to equations for regular forests.

Fix again a ranked alphabet Σ and a frontier alphabet X. For any $k \geq 1$, let

$$F_k = (\mathfrak{p}F_\Sigma(X))^k$$

be the set of k-tuples of ΣX-forests. We order F_k partially by componentwise inclusion:

$$(S_1, \ldots, S_k) \leq (T_1, \ldots, T_k) \quad \text{iff} \quad S_1 \subseteq T_1, \ldots, S_k \subseteq T_k \ .$$

Then F_k becomes a complete lattice in which least upper bounds and greatest lower bounds are obtained, respectively, by forming componentwise unions and intersections, thus

$$\bigvee\big((S_{i1}, \ldots, S_{ik}) \mid i \in I\big) = \Big(\bigcup(S_{i1} \mid i \in I), \ldots, \bigcup(S_{ik} \mid i \in I)\Big)$$

and

$$\bigwedge\big((S_{i1}, \ldots, S_{ik}) \mid i \in I\big) = \Big(\bigcap(S_{i1} \mid i \in I), \ldots, \bigcap(S_{ik} \mid i \in I)\Big) \ .$$

The least element is $\mathbf{0} = (\emptyset, \ldots, \emptyset)$. (We refer the reader to Section 1.4 for the lattice theory needed here.)

Let $V_k = \{v_1, \ldots, v_k\}$ be a set of variables disjoint from Σ and X. With every $\Sigma(X \cup V_k)$-forest P we associate the mapping

$$\hat{P} \colon F_k \to \mathfrak{p}F_\Sigma(X)$$

defined so that

$$\hat{P}(T_1, \ldots, T_k) = P(v_1 \leftarrow T_1, \ldots, v_k \leftarrow T_k)$$

for all $(T_1, \ldots, T_k) \in F_k$. A k-tuple $\Pi = (P_1, \ldots, P_k)$ of finite $\Sigma(X \cup V_k)$-forests is called a (Σ, X, k)-*polynomial* and we associate with it the mapping

$$\hat{\Pi} \colon F_k \to F_k$$

defined so that

$$\hat{\Pi}(\mathbf{T}) = \big(\hat{P}_1(\mathbf{T}), \ldots, \hat{P}_k(\mathbf{T})\big) \quad (\mathbf{T} \in F_k) \ .$$

Lemma 2.7.4 *For any* (Σ, X, k)*-polynomial* Π*, the mapping* $\hat{\Pi} \colon F_k \to F_k$ *is* ω*-continuous.*

Proof. Let $\Pi = (P_1, \ldots, P_k)$. The mapping $\hat{\Pi}$ is isotone as

$$P(v_1 \leftarrow S_1, \ldots, v_k \leftarrow S_k) \subseteq P(v_1 \leftarrow T_1, \ldots, v_k \leftarrow T_k)$$

obviously holds for all $P \subseteq F_\Sigma(X \cup V_k)$ and ΣX-forests $S_1, \ldots, S_k, T_1, \ldots, T_k$ such that $S_1 \subseteq T_1, \ldots, S_k \subseteq T_k$. Let

$$\mathbf{T}_0 \subseteq \mathbf{T}_1 \subseteq \mathbf{T}_2 \subseteq \ldots$$

be any ascending ω-sequence of vectors

$$\mathbf{T}_i = (T_{i1}, \ldots, T_{ik}) \in F_k \quad (i \geq 0)$$

of ΣX-forests. Now write

$$\mathbf{T} = \Big(\bigcup (T_{i1} \mid i \geq 0), \ldots, \bigcup (T_{ik} \mid i \geq 0) \Big) \ .$$

In order to prove ω-continuity we should show that

$$\hat{\Pi}(\mathbf{T}) = \Big(\bigcup \big(\hat{P}_1(\mathbf{T}_i) \mid i \geq 0\big), \ldots, \bigcup \big(\hat{P}_k(\mathbf{T}_i) \mid i \geq 0\big) \Big) \ ,$$

or equivalently, that

$$\hat{P}_j(\mathbf{T}) = \bigcup \big(\hat{P}_j(\mathbf{T}_i) \mid i \geq 0\big) \quad (j = 1, \ldots, k) \ . \tag{5}$$

Every tree $t \in \hat{P}_j(\mathbf{T})$ is obtained from some $p \in P_j$ by substituting a tree from $\bigcup (T_{im} \mid i \geq 0)$ for every occurrence of each variable v_m and each $m = 1, \ldots, k$. The number of occurrences of variables in p is finite. Hence there exists an $i \geq 0$ such that all trees used in this substitution appear in a component of $\mathbf{T}_i$. Then $t \in \hat{P}_j(\mathbf{T}_i)$. This shows that the left side of (5) is included in the right side of (5) for each $j = 1, \ldots, k$. The converse inclusions are obvious since $\hat{\Pi}$ is isotone and $\mathbf{T}_i \leq \mathbf{T}$ for all $i \geq 0$. □

Now, using Theorem 1.4.8 we get

Corollary 2.7.5 *For any* (Σ, X, k)*-polynomial* Π*, the mapping* $\hat{\Pi}\colon F_k \to F_k$ *has the least fixed-point*

$$[\hat{\Pi}] = \bigvee (\mathbf{0}\hat{\Pi}^i \mid i \geq 0) \ .$$ □

The corollary means that $[\hat{\Pi}]$ is the least solution of the fixed-point equation

$$(v_1, \ldots, v_k) = \hat{\Pi}(v_1, \ldots, v_k) \ , \tag{6}$$

where the v_i's are "unknowns" that assume ΣX-forests as their values. The equation (6) can also be written as a system of equations

$$\begin{cases} v_1 = P_1 \\ \quad\vdots \\ v_k = P_k \ , \end{cases} \tag{7}$$

where the P's are usually expressed as formal sums of their elements (as we did in Example 2.7.3).

The finiteness of the components P_i was not used in the proof of Lemma 2.7.4. However, it will be essential for obtaining the main result of this section. In fact, it will be convenient, although not necessary, to work with an even more restricted class of fixed-point equations, which we shall soon introduce. Example 2.7.3 provides us with a guideline here, too.

Let us extend the height function of $F_\Sigma(X)$ to $F_\Sigma(X \cup V_k)$ so that

$$\mathrm{hg}(v_i) = -1 \quad (i = 1, \dots, k) \ .$$

Then the $\Sigma(X \cup V_k)$-trees of height 0 are

(i) the frontier letters $x \in X$,

(ii) the 0-ary operators $\sigma \in \Sigma_0$, and

(iii) the trees of the form $\sigma(v_{i_1}, \dots, v_{i_m})$, where $m > 0$, $\sigma \in \Sigma_m$ and $v_{i_1}, \dots, v_{i_m} \in V_k$.

Definition 2.7.6 A (Σ, X, k)-polynomial $\Pi = (P_1, \dots, P_k)$ is *regular*, if every $\Sigma(X \cup V_k)$-tree of height 0 belongs to exactly one P_j, and the P_j's do not contain any other trees. If Π is regular, then $\hat{\Pi}$ and the corresponding fixed-point equation (6) are also said to be *regular*. A ΣX-forest T is called *equational* if it can be expressed as the union of some components of the least solution of a regular fixed-point equation.

The fixed-point equation in Example 2.7.3 is regular. It is easy to see that the same procedure applied to any Rabin-Scott recognizer will yield a regular fixed-point equation. Hence, every regular language is equational when viewed as a unary forest. It is also well-known, and easy to prove, that the components of the least solution of a system of Arden equations are regular.

Example 2.7.7 Let $\Sigma = \Sigma_0 \cup \Sigma_2$, $\Sigma_0 = \{\gamma\}$, $\Sigma_2 = \{\sigma\}$ and $X = \{x, y\}$. Then

$$\Pi = \big(\{x,\ \gamma,\ \sigma(v_1, v_2),\ \sigma(v_2, v_1)\},\ \{y,\ \sigma(v_1, v_1),\ \sigma(v_2, v_2)\}\big)$$

is a regular $(\Sigma, X, 2)$-polynomial. The corresponding regular fixed-point equation can be written as the system

$$\begin{cases} v_1 = x + \gamma + \sigma(v_1, v_2) + \sigma(v_2, v_1) \\ v_2 = y + \sigma(v_1, v_1) + \sigma(v_2, v_2) \ . \end{cases}$$

The least solution is the pair (T_1, T_2), where

$$T_1 = \big\{x,\ \gamma,\ \sigma(x, y),\ \sigma(\gamma, y),\ \sigma(y, x),\ \sigma(y, \gamma),\ \sigma\big(x, \sigma(x, x)\big), \dots\big\}$$

and

$$T_2 = \big\{y,\ \sigma(x, x),\ \sigma(\gamma, \gamma),\ \sigma(y, y), \dots\big\} \ . \qquad \square$$

Let $[\hat{\Pi}] = (T_1, \ldots, T_k)$ be the least fixed-point for a given (Σ, X, k)-polynomial Π. We define a binary relation $\varrho(\Pi)$ in $F_\Sigma(X)$:

$$\varrho(\Pi) = \{(s,t) \mid s, t \in T_i \text{ for some } i = 1, \ldots, k\} \ .$$

Lemma 2.7.8 *If Π is a regular (Σ, X, k)-polynomial, then $\varrho(\Pi)$ is a congruence of $\mathcal{F}_\Sigma(X)$ with at most k equivalence classes. For each congruence ϱ of $\mathcal{F}_\Sigma(X)$ of index k $(k \geq 1)$ there exists a regular (Σ, X, k)-polynomial Π such that $\varrho(\Pi) = \varrho$.*

Proof. Let $\Pi = (P_1, \ldots, P_k)$ be a regular (Σ, X, k)-polynomial and $[\hat{\Pi}] = (T_1, \ldots, T_k)$ the corresponding least fixed-point. From the definition of $\varrho(\Pi)$ it is clear that the relation is symmetric. To prove that it is reflexive and transitive, too, we show that every ΣX-tree t belongs to exactly one T_i. First we note that

$$T_i = P_i(v_1 \leftarrow T_1, \ldots, v_k \leftarrow T_k) \quad (i = 1, \ldots, k) \tag{8}$$

as $[\hat{\Pi}]$ is a fixed-point of $\hat{\Pi}$. We proceed now by induction on $\mathrm{hg}(t)$.

1° If $\mathrm{hg}(t) = 0$, then t is in exactly one of the sets P_i $(i = 1, \ldots, k)$ because Π is regular. From (8) we see that t is in the corresponding T_i and that it could belong to some other T_j $(j \neq i)$ only in case $v_i \in P_j$. But $\mathrm{hg}(v_i) = -1$ and v_i does not appear in Π.

2° Consider a tree $t = \sigma(t_1, \ldots, t_m)$ $(m > 0)$ and assume that all trees of lesser height belong to exactly one T_i. Then there exists for each $j = 1, \ldots, m$ exactly one i_j $(1 \leq i_j \leq k)$ such that $t_j \in T_{i_j}$. Also, there is exactly one i $(1 \leq i \leq k)$ such that $p = \sigma(v_{i_1}, \ldots, v_{i_m}) \in P_i$. Clearly,

$$t \in p(v_1 \leftarrow T_1, \ldots, v_k \leftarrow T_k) \subseteq T_i \ .$$

The uniqueness of the indices i_j implies that p is the only tree of height 0 in $F_\Sigma(X \cup V_k)$ from which t can be obtained by the substitutions $v_1 \leftarrow T_1, \ldots, v_k \leftarrow T_k$. Hence t belongs to T_i only.

Now we know that $\varrho(\Pi) \in E\big(F_\Sigma(X)\big)$. It is obvious that it has at most k equivalence classes. (There may be less than k classes as some T's could be empty.) To prove that it is a congruence relation we consider any $m \geq 1$, $\sigma \in \Sigma_m$ and $s_1, \ldots, s_m, t_1, \ldots, t_m \in F_\Sigma(X)$ such that

$$s_1 \equiv t_1, \ldots, s_m \equiv t_m \ \big(\varrho(\Pi)\big) \ .$$

There are indices $i_1, \ldots, i_m$ such that

$$s_j, t_j \in T_{i_j} \text{ for } j = 1, \ldots, m \ .$$

Let $\sigma(v_{i_1}, \ldots, v_{i_m})$ be in P_i. Then

$$\sigma(s_1, \ldots, s_m),\ \sigma(t_1, \ldots, t_m) \in T_i$$

by (8). Hence
$$\sigma^{\mathcal{F}_\Sigma(X)}(s_1,\ldots,s_m) \equiv \sigma^{\mathcal{F}_\Sigma(X)}(t_1,\ldots,t_m)\ \big(\varrho(\Pi)\big)$$
as required.

Now, suppose $\varrho \in C\big(\mathcal{F}_\Sigma(X)\big)$ and let $S_1,\ldots,S_k$ be the equivalence classes of ϱ. We define a (Σ, X, k)-polynomial $\Pi = (P_1,\ldots,P_k)$ so that

$$P_i = \{p \in F_\Sigma(X \cup V_k) \mid \mathrm{hg}(p) = 0,\ p(v_1 \leftarrow S_1,\ldots,v_k \leftarrow S_k) \subseteq S_i\}$$

for all $i = 1,\ldots,k$. The fact that ϱ is a congruence means that for each p of height 0 there is exactly one i $(1 \le i \le k)$ such that

$$p(v_1 \leftarrow S_1,\ldots,v_k \leftarrow S_k) \subseteq S_i\ .$$

Hence Π is regular. We claim that $\varrho(\Pi) = \varrho$. Let $[\hat{\Pi}] = (T_1,\ldots,T_k)$. In order to prove the second statement of the lemma we show by induction on $\mathrm{hg}(t)$ that for all $i = 1,\ldots,k$,
$$\big(\forall t \in F_\Sigma(X)\big)\big(t \in S_i \Longleftrightarrow t \in T_i\big)\ .$$

1° If $\mathrm{hg}(t) = 0$, then there is exactly one i such that $t \in P_i$. This means $t \in S_i$. From (8) it follows that $t \in T_i$ for the same i.

2° Let $t = \sigma(t_1,\ldots,t_m)$ $(m > 0)$ and suppose the claim holds for all trees of height $<$ $\mathrm{hg}(t)$. Then there are unique indices $i_1,\ldots,i_m$ such that

$$t_j \in S_{i_j} \cap T_{i_j} \quad (j = 1,\ldots,m)\ .$$

Also, there is a unique i such that

$$p = \sigma(v_{i_1},\ldots,v_{i_m}) \in P_i\ .$$

Then
$$t \in \sigma(S_{i_1},\ldots,S_{i_m}) = p(v_1 \leftarrow S_1,\ldots,v_k \leftarrow S_k) \subseteq S_i$$
by the definition of P_i. On the other hand, (8) implies $t \in T_i$. □

If we combine Lemma 2.7.8 and Theorem 2.7.1, we get

Theorem 2.7.9 *A forest is equational iff it is recognizable.* □

From the first part of this section it is clear that a ΣX-forest T can be recognized by a k-state tree recognizer iff T is saturated by a congruence of $\mathcal{F}_\Sigma(X)$ of index $\le k$. From Lemma 2.7.8 we get a similar connection between the number of states and the number of variables in a regular fixed-point equation which defines the forest.

There is also a very close connection between regular tree grammars and the fixed-point equations considered here. For example, the equations of Example 2.7.7 can be converted into the following set of productions in which v_1 and v_2 are nonterminal symbols:

$$\begin{array}{llll} v_1 \to x, & v_1 \to \gamma, & v_1 \to \sigma(v_1,v_2), & v_1 \to \sigma(v_2,v_1), \\ v_2 \to y, & v_2 \to \sigma(v_1,v_1), & v_2 \to \sigma(v_2,v_2)\ . & \end{array}$$

The resulting regular tree grammar generates T_1 if v_1 is the initial symbol, and it generates T_2 if v_2 is the initial symbol.

On the other hand, every regular ΣX-grammar with k nonterminal symbols can be converted into a fixed-point system with k equations. This system is not necessarily regular, but the components of the least solution are nevertheless the regular forests generated by the grammar from the different nonterminal symbols. For example, if Σ and X are as in Example 2.7.7 and the productions are

$$a \to x, \quad a \to \gamma, \quad a \to \sigma(a,b), \quad b \to \sigma(b,b), \quad b \to y,$$

then the corresponding equations would be

$$\begin{aligned} a &= x + \gamma + \sigma(a,b) \quad \text{and} \\ b &= y + \sigma(b,b) \ , \end{aligned}$$

where a and b now are the unknowns. The least solution is $\big(T(G_a), T(G_b)\big)$, where G_a and G_b are the grammars which we obtain by choosing a and b, respectively, as the initial symbol.

2.8 A MEDVEDEV-TYPE CHARACTERIZATION

Our next description of the recognizable forests is a streamlined generalization of a well-known characterization of the regular languages given by J. Medvedev in 1956. First we define the family of representable forests. The theorem states then that the representable forests are exactly the recognizable forests. The representable forests are defined collectively for all ranked alphabets as the definition involves tree homomorphisms and these may take us from one alphabet to another. Recall that $r(\Sigma)$ is the finite set of nonnegative integers m for which $\Sigma_m \neq \emptyset$.

Definition 2.8.1 For every pair (Σ, X) we define the "next-to-root function"

$$\text{nroot}\colon F_\Sigma(X) - (\Sigma_0 \cup X) \to \bigcup\big((\Sigma \cup X)^m \mid m \in r(\Sigma)\big)$$

so that

$$\text{nroot}\big(\sigma(t_1, \ldots, t_m)\big) = \big(\text{root}(t_1), \ldots, \text{root}(t_m)\big)$$

for all $m > 0$, $\sigma \in \Sigma_m$ and $t_1, \ldots, t_m \in F_\Sigma(X)$.

Definition 2.8.2 The *elementary ΣX-forests* are the forests

$$U(d) = \text{root}^{-1}(d) \quad (d \in \Sigma \cup X) \ , \text{ and} \tag{i}$$

$$V(d_1, \ldots, d_m) = \text{nroot}^{-1}(d_1, \ldots, d_m) \ , \tag{ii}$$

where $m > 0$, $m \in r(\Sigma)$, and $d_1, \ldots, d_m \in \Sigma \cup X$.

U, V are inverse homomorphisms

Note that the definitions of the $U(d)$- and $V(d_1,\dots,d_m)$-forests presume a Σ and an X although the notations do not show this. Clearly, $U(d)$ is the set of all ΣX-trees with the root labelled by d, and $V(d_1,\dots,d_m)$ consists of all ΣX-trees of height ≥ 1 in which the nodes immediately above the root are labelled, from left to right, by $d_1,\dots,d_m$, respectively. Note also that $U(d) = \{d\}$ when $d \in \Sigma_0 \cup X$. We need three more definitions.

Definition 2.8.3 The *restriction* of a forest T is the forest

$$\mathrm{rest}(T) = \{t \in T \mid \mathrm{sub}(t) \subseteq T\} \ .$$

Definition 2.8.4 The *elementary operations* on forests are the formation of

(i) the union of two forests,

(ii) the intersection of two forests,

(iii) an alphabetic tree homomorphic image of a forest, and

(iv) the restriction of a forest.

Definition 2.8.5 A forest is *representable* if it can be constructed from elementary forests by a finite number of applications of elementary operations.

Now the theorem can be stated.

Theorem 2.8.6 *A forest is representable iff it is recognizable.*

Proof. To prove that the representable forests are recognizable it suffices to note that the elementary forests are recognizable and that the elementary operations preserve recognizability. Consider any Σ and X. If $d \in \Sigma_0 \cup X$, then $U(d) = \{d\} \in \mathrm{Rec}(\Sigma, X)$. If $d \in \Sigma_m$ $(m > 0)$, then

$$U(d) = d(y_1,\dots,y_m)\big(y_1 \leftarrow F_\Sigma(X),\dots,y_m \leftarrow F_\Sigma(X)\big)$$

is again recognizable. Similarly,

$$V(d_1,\dots,d_m) = \bigcup\big(\sigma(y_1,\dots,y_m)\big(y_1 \leftarrow U(d_1),\dots,y_m \leftarrow U(d_m)\big) \mid \sigma \in \Sigma_m\big)$$

is recognizable for all $m \in r(\Sigma)$ and $d_1,\dots,d_m \in \Sigma \cup X$. We have already seen in Section 2.4 that unions, intersections and alphabetic tree homomorphisms preserve recognizability. Let T be the forest recognized by a ΣX-recognizer $\mathbf{A}$. We construct a recognizer for $\mathrm{rest}(T)$. First define a Σ-algebra $\mathcal{B} = (A \cup b, \Sigma)$ $(b \notin A)$ so that

$$\sigma^{\mathcal{B}}(b_1,\dots,b_m) = \begin{cases} \sigma^{\mathcal{A}}(b_1,\dots,b_m) & \text{if } b_1,\dots,b_m \in A \text{ and } \sigma^{\mathcal{A}}(b_1,\dots,b_m) \in A' \ , \\ b & \text{in all other cases,} \end{cases}$$

for all $m \geq 0$, $\sigma \in \Sigma_m$ and $b_1, \ldots, b_m \in A \cup b$. The initial assignment $\beta\colon X \to A \cup b$ is defined so that for each $x \in X$,

$$x\beta = \begin{cases} x\alpha & \text{if } x\alpha \in A' \ , \\ b & \text{if } x\alpha \notin A' \ . \end{cases}$$

Consider any ΣX-tree t. It is easy to show that

$$t\hat{\beta} = \begin{cases} t\hat{\alpha} & \text{if } \mathrm{sub}(t) \subseteq T \ , \\ b & \text{otherwise.} \end{cases}$$

Hence, $\mathbf{B} = (\mathcal{B}, \beta, A')$ recognizes $\mathrm{rest}(T)$.

We shall now show that every recognizable forest is representable. Let $T = T(\mathbf{A})$ for some ΣX-recognizer $\mathbf{A}$. First define a new ranked alphabet Ω such that

$$\Omega_m = \Sigma_m \times (A \cup X)^m \quad \text{for all} \quad m \geq 0 \ .$$

We construct two representable ΩX-forests R and S as follows. For $c \in A \cup X$ we introduce the notation

$$\overline{c} = \begin{cases} c & \text{if } c \in A \ , \\ c\alpha & \text{if } c \in X \ . \end{cases}$$

Then

$$\begin{aligned} R = \{x \in X \mid x\alpha \in A'\} \cup \\ \bigcup \big(U((\sigma, c_1, \ldots, c_m)) \mid (\sigma, c_1, \ldots, c_m) \in \Omega,\ \sigma^{\mathcal{A}}(\overline{c}_1, \ldots, \overline{c}_m) \in A'\big) \ . \end{aligned}$$

The forest S is the union of all intersections

$$V(u_1, \ldots, u_m) \cap U\big((\sigma, b_1, \ldots, b_m)\big) \ ,$$

where for each $i = 1, \ldots, m$, either

(i) $u_i \in X$ and $\overline{b}_i = u_i\alpha$, or

(ii) $u_i = (\tau, c_1, \ldots, c_k) \in \Omega_k$ $(k \geq 0)$ and $b_i = \tau^{\mathcal{A}}(\overline{c}_1, \ldots, \overline{c}_k)$.

Note that the possibility $m = 0$ is included at appropriate places in the definitions of R and S.

Define the tree homomorphism

$$h\colon F_\Omega(X) \to F_\Sigma(X)$$

so that

$$h_m\big((\sigma, b_1, \ldots, b_m)\big) = \sigma \ , \quad \big(m \geq 0,\ (\sigma, b_1, \ldots, b_m) \in \Omega_m\big)$$

and $h_X = 1_X$. Clearly, h is alphabetic. We claim that

$$T = h(P)$$

for the representable forest

$$P = R \cap \operatorname{rest}(S \cup \Omega_0 \cup X) \ .$$

Let $p \in P$. If $p = (\sigma, e) \in \Omega_0$, then $p \in R$ implies $\sigma^{\mathcal{A}} \in A'$. Hence $h(p) = \sigma \in T$. If $p = x \in X$, then $p \in R$ implies $h(x)\hat{\alpha} = x\alpha \in A'$. Again $h(p) = x \in T$. Next we show that for every $p \in \operatorname{rest}(S \cup \Omega_0 \cup X)$ of height ≥ 1

$$h(p)\hat{\alpha} = \sigma^{\mathcal{A}}(\overline{b}_1, \dots, \overline{b}_m) \ , \qquad \text{where} \qquad (\sigma, b_1, \dots, b_m) = \operatorname{root}(p) \ . \tag{1}$$

We proceed by induction on $\operatorname{hg}(p)$.

1° If $\operatorname{hg}(p) = 1$, then $m \geq 1$ and

$$p = (\sigma, b_1, \dots, b_m)(u_1, \dots, u_m)$$

for some $u_1, \dots, u_m \in \Omega_0 \cup X$. Since $p \in S$ we have $h(u_i)\hat{\alpha} = \overline{b}_i$ for all $i = 1, \dots, m$. But this implies that (1) holds for p.

2° Now let

$$p = (\sigma, b_1, \dots, b_m)(p_1, \dots, p_m)$$

and assume that (1) holds for the trees $p_1, \dots, p_m$. As p is in S and

$$h(p)\hat{\alpha} = \sigma^{\mathcal{A}}\big(h(p_1)\hat{\alpha}, \dots, h(p_m)\hat{\alpha}\big) \ ,$$

it suffices to show that $h(p_i)\hat{\alpha} = \overline{b}_i$ for every $i = 1, \dots, m$. We should consider three cases.

(a) If p_i is of the form $(\tau, c_1, \dots, c_k)(r_1, \dots, r_k)$ $(k > 0)$, then the induction hypothesis yields

$$h(p_i)\hat{\alpha} = \tau^{\mathcal{A}}(\overline{c}_1, \dots, \overline{c}_k) \ .$$

Moreover, $\tau^{\mathcal{A}}(\overline{c}_1, \dots, \overline{c}_k) = b_i = \overline{b}_i$ since $p \in S$.

(b) If $p_i = (\sigma, e) \in \Omega_0$, then $h(p_i)\hat{\alpha} = \sigma^{\mathcal{A}} = b_i = \overline{b}_i$.

(c) If $p_i = x \in X$, then $h(p_i)\hat{\alpha} = x\alpha = \overline{b}_i$.

Now we have completed the proof of (1). Consider any tree

$$p = (\sigma, b_1, \dots, b_m)(p_1, \dots, p_m) \in P \ .$$

By using (1) and the fact that $p \in R$ we get

$$h(p)\hat{\alpha} = \sigma^{\mathcal{A}}(\overline{b}_1, \dots, \overline{b}_m) \in A' \ .$$

This implies $h(p) \in T$ and we have shown that $h(P) \subseteq T$.

In order to prove the converse inclusion we show first by tree induction how to construct for each $t \in F_\Sigma(X)$ a tree $p \in \operatorname{rest}(S \cup \Omega_0 \cup X)$ such that $h(p) = t$:

1° If $t = x \in X$, then we may choose $p = x$.

2° If $t = \sigma \in \Sigma_0$, put $p = (\sigma, e)$.

3° Let $t = \sigma(t_1, \ldots, t_m)$ $(m > 0)$ and suppose we have trees $p_1, \ldots, p_m \in \mathrm{rest}(S \cup \Omega_0 \cup X)$ such that $h(p_i) = t_i$ $(i = 1, \ldots, m)$. If we put

$$p = (\sigma, b_1, \ldots, b_m)(p_1, \ldots, p_m) \ ,$$

where $b_i = t_i\hat{\alpha}$ for $i = 1, \ldots, m$, then $h(p) = t$ and $p \in \mathrm{rest}(S \cup \Omega_0 \cup X)$ as required.

Let $t \in T$ and construct a p for t as above. To prove $t \in h(P)$ it suffices to show that $p \in R$. This can again be done by tree induction:

1° If $t = x \in X$, then $x\alpha \in A'$ and hence $p = x \in R$.

2° If $t = \sigma \in \Sigma_0$, then $\sigma^{\mathcal{A}} \in A'$ and $p = (\sigma, e) \in U(\sigma, e) \subseteq R$.

3° Let $t = \sigma(t_1, \ldots, t_m)$ $(m > 0)$. If we use (1) and its notation, we get

$$\sigma^{\mathcal{A}}(\overline{b}_1, \ldots, \overline{b}_m) = h(p)\hat{\alpha} = t\hat{\alpha} \in A' \ .$$

This shows that $p \in R$. □

2.9 LOCAL FORESTS

In this section a proper subfamily of the recognizable forests is introduced. We will then also get one more characterization of the recognizable forests, not quite unrelated to that given in the preceding section.

We need the following auxiliary concept

Definition 2.9.1 The set of *forks* $\mathrm{fork}(t)$ of a ΣX-tree t is defined as follows:

1° If $t \in \Sigma_0 \cup X$, then $\mathrm{fork}(t) = \emptyset$.

2° If $t = \sigma(t_1, \ldots, t_m)$ $(m > 0)$, then

$$\mathrm{fork}(t) = \mathrm{fork}(t_1) \cup \cdots \cup \mathrm{fork}(t_m) \cup \big\{\sigma\big(\mathrm{root}(t_1), \ldots, \mathrm{root}(t_m)\big)\big\} \ .$$

The set of all forks of ΣX-trees $\bigcup\big(\mathrm{fork}(t) \mid t \in F_\Sigma(X)\big)$ will be denoted by $\mathrm{fork}(\Sigma, X)$.

Example 2.9.2 Let $\Sigma = \Sigma_0 \cup \Sigma_1 \cup \Sigma_2$, $\Sigma_0 = \{\gamma\}$, $\Sigma_1 = \{\tau\}$, $\Sigma_2 = \{\sigma\}$ and $X = \{x, y\}$. For the ΣX-tree

$$t = \sigma\big(\tau(\gamma),\, \sigma(x,\, \tau(y))\big) \ ,$$

we have

$$\mathrm{fork}(t) = \big\{\sigma(\tau, \sigma),\, \tau(\gamma),\, \sigma(x, \tau),\, \tau(y)\big\} \ .$$

Graphically these forks are represented as in Fig. 2.7 respectively. Obviously, $\mathrm{fork}(\Sigma, X)$ is always finite and here it consists of 30 forks. □

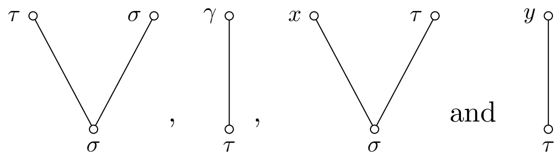

Figure 2.7.

Local forests may now be defined.

Definition 2.9.3 A ΣX-forest T is *local* if there are sets $R(\subseteq \Sigma \cup X)$ and $F\big(\subseteq \mathrm{fork}(\Sigma, X)\big)$ such that, for each $t \in F_\Sigma(X)$,

$$t \in T \quad \text{iff} \quad \mathrm{root}(t) \in R \quad \text{and} \quad \mathrm{fork}(t) \subseteq F \ .$$

Then we write $T = \mathrm{Loc}(R, F)$.

Hence the membership of a ΣX-tree t in the local forest $\mathrm{Loc}(R, F)$ can be decided by testing for the local properties $\mathrm{root}(t) \in R$ and $\mathrm{fork}(t) \subseteq F$.

A ΣX-recognizer for $\mathrm{Loc}(R, F)$ can be constructed as follows. First we define a Σ-algebra $\mathcal{A} = (A, \Sigma)$. Let $A = \Sigma \cup X \cup 0$ $(0 \notin \Sigma \cup X)$. For every $\sigma \in \Sigma_0$, put $\sigma^{\mathcal{A}} = \sigma$. For $m > 0$, $\sigma \in \Sigma_m$ and $a_1, \ldots, a_m \in A$ let

$$\sigma^{\mathcal{A}}(a_1, \ldots, a_m) = \begin{cases} \sigma & \text{if } \sigma(a_1, \ldots, a_m) \in F \ , \\ 0 & \text{otherwise.} \end{cases}$$

Let $\alpha\colon X \to A$ be the embedding $x \mapsto x$ $(x \in X)$. It is easy to show that for all $t \in F_\Sigma(X)$,

$$t\hat{\alpha} = \begin{cases} \mathrm{root}(t) & \text{if } \mathrm{fork}(t) \subseteq F \ , \\ 0 & \text{otherwise.} \end{cases}$$

This readily implies $T(\mathbf{A}) = \mathrm{Loc}(R, F)$ for $\mathbf{A} = (\mathcal{A}, \alpha, R)$. Hence we have

Theorem 2.9.4 *Every local forest is recognizable.* □

The converse of Theorem 2.9.4 does not hold. For example, the forest consisting of the single tree of Example 2.9.2 is not local as there are many other trees with the same root and the same forks. However, the following fact can be proved.

Theorem 2.9.5 *For every recognizable ΣX-forest T there exist a ranked alphabet Ω, a frontier alphabet Y, a local ΩY-forest S and an alphabetic tree homomorphism*

$$h\colon F_\Omega(Y) \to F_\Sigma(X)$$

such that $T = h(S)$.

Proof. Let $G = (N, \Sigma, X, P, a_0)$ be a regular ΣX-grammar generating T. We assume that G is in normal form. A new ranked alphabet Ω is defined so that

$$\Omega_m = \{[a \to \sigma(a_1, \dots, a_m)] \mid a \to \sigma(a_1, \dots, a_m) \in P,\ \sigma \in \Sigma_m\}$$

for all $m \geq 0$. Also, let

$$Y = \{[a \to x] \mid a \to x \in P,\ x \in X\} \ .$$

The local ΩY-forest $S = \mathrm{Loc}(R, F)$ is defined by the sets

$$R = \{[a_0 \to p] \mid a_0 \to p \in P\}$$

and

$$F = \big\{[a \to \sigma(a_1, \dots, a_m)]([a_1 \to p_1], \dots, [a_m \to p_m]) \mid m > 0,\\ a \to \sigma(a_1, \dots, a_m),\ a_1 \to p_1, \dots, a_m \to p_m \in P\big\} \ .$$

Finally, define an alphabetic tree homomorphism

$$h \colon F_\Omega(Y) \to F_\Sigma(X)$$

by the mappings

$$h_Y \colon Y \to F_\Sigma(X), \quad [a \to x] \mapsto x,$$

and

$$h_m \colon \Omega_m \to F_\Sigma(X \cup \Xi_m), \quad [a \to \sigma(a_1, \dots, a_m)] \mapsto \sigma(\xi_1, \dots, \xi_m) \ .$$

Now $h(S) = T$, and thereby the theorem, follows from (1) and (2):

(1) If $a \Rightarrow_G^* t$, for some $a \in N$ and $t \in F_\Sigma(X)$, then there is a tree $s \in F_\Omega(Y)$ such that $h(s) = t$, $\mathrm{fork}(s) \subseteq F$ and $\mathrm{root}(s)$ is of the form $[a \to p]$.

(2) If $s \in F_\Omega(Y)$ is such that $\mathrm{fork}(s) \subseteq F$ and $\mathrm{root}(s) = [a \to p]$ for some $p \in F_\Sigma(N \cup X)$, then $a \Rightarrow_G^* h(s)$.

Part (1) can be proved by induction on the length of the derivation of t and (2) by tree induction on s. □

Note that $h(S)$ is always recognizable when S is a local forest and h an alphabetic tree homomorphism (Theorem 2.9.4 and Corollary 2.4.20).

2.10 SOME BASIC DECISION PROBLEMS

In this section we shall show that some of the first questions one might ask about given tree recognizers are algorithmically decidable. To begin with, we have the *emptiness problem:* Is the forest recognized by a given tree recognizer empty? Or one may ask whether this forest is finite or infinite. This is the *finiteness problem.* Finally, we have the important *equivalence problem:* Do two given tree recognizers recognize the same forest? In fact, the more general *inclusion problem:* "$T(\mathbf{A}) \subseteq T(\mathbf{B})$?" is shown to be decidable. The problems are quite easy and the proofs follow the strategy familiar from finite automata theory with a "pumping lemma" as the key result. We have seen in Section 2.2 that any nondeterministic frontier-to-root, or root-to-frontier, tree recognizer can be converted into an equivalent deterministic F-recognizer. Hence we may again restrict ourselves to our basic type of tree recognizers.

We need the following special notation. Let Σ and X be given. Introduce a new letter ξ and let T_ξ be the set of all $\Sigma(X \cup \xi)$-trees in which ξ appears exactly once. For any $q \in T_\xi$ and $p \in F_\Sigma(X) \cup T_\xi$ we denote $q(\xi \leftarrow p)$ by $p \cdot q$. Also, we define the powers q^k as follows:

1° $q^0 = \xi$,

2° $q^{n+1} = q \cdot q^n \quad (n \geq 0)$.

Using these notations we may formulate the pumping lemma of tree recognizers as follows.

Lemma 2.10.1 *Let* $\mathbf{A}$ *be a* k*-state* ΣX*-recognizer. If* $t \in T(\mathbf{A})$ *and* $\mathrm{hg}(t) \geq k$*, then there are trees* $p \in F_\Sigma(X)$ *and* $q, r \in T_\xi$ *such that*

(a) $t = p \cdot q \cdot r$,

(b) $\mathrm{hg}(q) \geq 1$ *and*

(c) $p \cdot q^i \cdot r \in T(\mathbf{A})$ *for all* $i = 0, 1, 2, \ldots$.

Proof. Suppose $t \in T(\mathbf{A})$ and $\mathrm{hg}(t) \geq k (= |A|)$. Then we can write $t = \sigma(t_1, \ldots, t_m)$ ($m > 0$, $\sigma \in \Sigma_m$). Choose some j ($1 \leq j \leq m$) such that $\mathrm{hg}(t_j) = \mathrm{hg}(t) - 1$. Then

$$t = t_j \cdot s_1 \ ,$$

where

$$s_1 = \sigma(t_1, \ldots, t_{j-1}, \xi, t_{j+1}, \ldots, t_m) \in T_\xi \ .$$

If $\mathrm{hg}(t_j) > 0$, we may decompose t_j the same way. Since $\mathrm{hg}(t) \geq k$ the process can be repeated k times and finally we obtain a representation

$$t = t' \cdot s_k \cdot \ldots \cdot s_2 \cdot s_1 \ ,$$

where $t' \in F_\Sigma(X)$ and $s_1, \ldots, s_k \in T_\xi$. Moreover, $\mathrm{hg}(s_i) \geq 1$ for every $i = 1, \ldots, k$. Let

$$u_{k+1} = t' \ , \quad u_k = t' \cdot s_k \ , \ldots , \quad u_1 = t' \cdot s_k \cdot \ldots \cdot s_1 = t \ .$$

There must be indices h and j, $k+1 \geq h > j \geq 1$, such that

$$u_h \hat{\alpha} = u_j \hat{\alpha} \ .$$

Now let $p = u_h$, $q = s_{h-1} \cdot \ldots \cdot s_j$ and $r = s_{j-1} \cdot \ldots \cdot s_1$ (if $j = 1$, then $r = \xi$). Then $t = p \cdot q \cdot r$ and $\mathrm{hg}(q) \geq 1$. Also, our choice of p and q implies

$$p\hat{\alpha} = (p \cdot q)\hat{\alpha} \ . \tag{1}$$

We assume that $A \cap X = \emptyset$, and extend $\hat{\alpha}$ to a homomorphism

$$\overline{\alpha}\colon \mathcal{F}_\Sigma(X \cup A) \to \mathcal{A}$$

so that $\hat{\alpha}|A = 1_A$. By Lemma 2.4.17 $s\overline{\alpha} = s\hat{\alpha}$ whenever $s \in F_\Sigma(X)$. We verify now by induction on i that

$$(p \cdot q^i)\hat{\alpha} = (p \cdot q)\hat{\alpha} \tag{2}$$

for every $i \geq 0$. From (1) we know that this is true for $i = 0$. Suppose (2) holds for a given i. This assumption and (1) imply

$$\begin{aligned}(p \cdot q^{i+1})\hat{\alpha} &= q\big(\xi \leftarrow (p \cdot q^i)\hat{\alpha}\big)\overline{\alpha} = q\big(\xi \leftarrow (p \cdot q)\hat{\alpha}\big)\overline{\alpha} \\ &= q\big(\xi \leftarrow p\hat{\alpha}\big)\overline{\alpha} = (p \cdot q)\hat{\alpha} \ .\end{aligned}$$

Using (2) we get for each $i \geq 0$,

$$\begin{aligned}(p \cdot q^i \cdot r)\hat{\alpha} &= r\big(\xi \leftarrow (p \cdot q^i)\hat{\alpha}\big)\overline{\alpha} \\ &= r\big(\xi \leftarrow (p \cdot q)\hat{\alpha}\big)\overline{\alpha} \\ &= (p \cdot q \cdot r)\hat{\alpha} \ .\end{aligned}$$

Hence, $p \cdot q^i \cdot r \in T(\mathbf{A})$ for all $i \geq 0$. □

Theorem 2.10.2 *Let* $\mathbf{A}$ *be a* k*-state* ΣX*-recognizer. Then* $T(\mathbf{A})$ *is nonempty iff it contains a tree of height less than* k*. Hence the emptiness problem of recognizable forests is decidable.*

Proof. Suppose $T(\mathbf{A})$ is nonempty. Let t be a tree in $T(\mathbf{A})$ of minimal length. If $\mathrm{hg}(t) \geq k$, we apply the pumping lemma and write $t = p \cdot q \cdot r$. But then $T(\mathbf{A})$ would contain the tree $p \cdot r$ which is properly shorter than t as $\mathrm{hg}(q) \geq 1$. Hence $\mathrm{hg}(t) < k$ must hold. The converse part is trivial. The emptiness of $T(\mathbf{A})$ can always be decided by going through the finite set of trees of height $< k$. □

Suppose two ΣX-recognizers $\mathbf{A}$ and $\mathbf{B}$ are given. Clearly, $T(\mathbf{A}) \subseteq T(\mathbf{B})$ iff $T(\mathbf{A}) - T(\mathbf{B}) = \emptyset$. But $T(\mathbf{A}) - T(\mathbf{B})$ is recognized by

$$\mathbf{C} = \big(\mathcal{A} \times \mathcal{B}, \gamma, A' \times (B - B')\big) \ ,$$

where $x\gamma = (x\alpha, x\beta)$ for $x \in X$. Thus the question "$T(\mathbf{A}) \subseteq T(\mathbf{B})$?" can be answered by deciding whether $T(\mathbf{C})$ is empty or not. The equivalence problem can similarly be reduced to the emptiness problem. Of course, its decidability follows also from the decidability of the inclusion problem. We have justified

Theorem 2.10.3 *The inclusion problem and the equivalence problem of tree recognizers are decidable.* □

Finally we consider the finiteness problem.

Theorem 2.10.4 *It is decidable whether the forest recognized by a given tree recognizer is finite or infinite.*

Proof. Let $\mathbf{A}$ be a k-state ΣX-recognizer and write

$$T = T(\mathbf{A}) - \{t \in F_\Sigma(X) \mid \mathrm{hg}(t) < k\} \ .$$

We claim that $T(\mathbf{A})$ is finite iff $T = \emptyset$. Obviously the condition is sufficient since the set of ΣX-trees of height $< k$ is finite. If $T \neq \emptyset$ and $t \in T$, then $\mathrm{hg}(t) \geq k$ and we may apply the pumping lemma and write $t = p \cdot q \cdot r$ so that

$$p \cdot q^i \cdot r \in T(\mathbf{A}) \quad \text{for all } i \geq 0 \ .$$

These trees are pairwise distinct since $\mathrm{hg}(q) \geq 1$. Hence $T(\mathbf{A})$ is infinite. The forest T is recognizable and one can easily construct a recognizer for it. This means that the condition $T = \emptyset$ is effectively testable. □

The decidability of the finiteness problem may also be deduced from the following corollary of the pumping lemma. The proof is an exercise.

Lemma 2.10.5 *Let $\mathbf{A}$ be a k-state tree recognizer. Then $T(\mathbf{A})$ is infinite iff it contains a tree t such that*

$$k \leq \mathrm{hg}(t) < 2k \ .$$ □

2.11 DETERMINISTIC R-RECOGNIZERS

In Section 2.2 it was shown that NDR-recognizers recognize exactly the family Rec, but that there are recognizable forests that cannot be recognized by any deterministic R-recognizer. The limited recognition power of DR-recognizers is due to the fact that they have no way of combining the information gathered from disjoint subtrees. This implies that a DR-recognizer will accept any tree in which every path from the root to the frontier appears in some tree accepted by the recognizer. It will turn out that this closure property characterizes the forests recognizable by DR-recognizers. Here a "path" contains, not only a list of the labels of the nodes traversed, but also the information about the directions taken at the nodes. In the later part of this section we shall consider the minimization of DR-recognizers. It will be shown that every DR-recognizer can be reduced to a canonical minimal form which is unique up to isomorphism.

Let Σ be a fixed ranked alphabet. In order to avoid some troublesome technicalities, we shall assume that $\Sigma_0 = \emptyset$. We associate with Σ a unary ranked alphabet

$$\Gamma = \Gamma_1 = \bigcup(\Gamma(\sigma) \mid \sigma \in \Sigma),$$

where for all $\sigma, \tau \in \Sigma$,

(i) $\Gamma(\sigma) = \{\sigma_1, \ldots, \sigma_m\}$ if $\sigma \in \Sigma_m$ $(m \geq 1)$, and

(ii) $\Gamma(\sigma) \cap \Gamma(\tau) = \emptyset$ if $\sigma \neq \tau$.

The paths in Σ-trees can now be defined as Γ-trees.

Definition 2.11.1 Let X be any frontier alphabet. For each $x \in X$ the set $g_x(t)$ of *x-paths* of a ΣX-tree t is defined as follows:

1° $g_x(x) = \{x\}$, and $g_x(y) = \emptyset$ for all $y \neq x$, $y \in X$.

2° If $t = \sigma(t_1, \ldots, t_m)$ $(\sigma \in \Sigma_m,\ m > 0)$, then $g_x(t) = \sigma_1(g_x(t_1)) \cup \cdots \cup \sigma_m(g_x(t_m))$.

We extend g_x to a mapping from $\mathfrak{p}F_\Sigma(X)$ to $\mathfrak{p}F_\Gamma(X)$ in the natural way. Moreover, we put

$$g(T) = \bigcup(g_x(T) \mid x \in X)$$

for each $T \subseteq F_\Sigma(X)$.

Label the edges of the graph representing a tree $t \in F_\Sigma(X)$ so that the i^{th} edge (counted from the left) leaving a node labelled by a symbol σ always gets the label σ_i. Then the elements of $g_x(t)$ $(x \in X)$ are spelled out by the paths leading from the root to a leaf labelled by x when we interpret a word $\sigma_{1i_1} \ldots \sigma_{ki_k} x$ $(k \geq 0, \sigma_{1i_1}, \ldots, \sigma_{ki_k} \in \Gamma)$ as the ΓX-tree $\sigma_{1i_1}(\ldots \sigma_{ki_k}(x) \ldots)$. Moreover, every such path gives an element of $g_x(t)$.

Lemma 2.11.2 *If* $T \in \mathrm{Rec}(\Sigma, X)$, *then* $g(T) \in \mathrm{Rec}(\Gamma, X)$.

Proof. Let $G = (N, \Sigma, X, P, a_0)$ be a regular ΣX-grammar in normal form generating T. The case $T = \emptyset$ being trivial, we may assume that every $G_a = (N, \Sigma, X, P, a)$ $(a \in N)$ generates a nonempty forest. Let $G' = (N, \Gamma, X, P', a_0)$ be the regular ΓX-grammar, where

$$P' = \{a \to \sigma_i(a_i) \mid a \to \sigma(a_1, \ldots, a_m) \in P,\ m > 0,\ 1 \leq i \leq m\} \cup \\ \{a \to x \mid a \to x \in P, x \in X\}.$$

We claim that $T(G') = g(T)$. This follows when we show that, for every tree

$$p = \sigma_{1i_1}(\ldots \sigma_{ki_k}(x) \ldots) \in F_\Gamma(X)$$

and every $a \in N$,

$$p \in T(G'_a) \quad \text{iff} \quad p \in g(T(G_a)), \tag{*}$$

where $G'_a = (N, \Gamma, X, P', a)$.

We proceed by induction on $\mathrm{hg}(p)$.

1° If $\mathrm{hg}(p) = 0$, then $p = x$. In this case (*) obviously holds as $a \to x$ is in P' iff it is in P.

2° Suppose $\mathrm{hg}(p) > 0$ and that (*) holds for all trees of lesser height.

If $p \in T(G'_a)$, then $a \Rightarrow^*_{G'} \sigma_{1i_1}(a_{i_1})$ and $a_{i_1} \Rightarrow^*_{G'} \sigma_{2i_2}(\ldots \sigma_{ki_k}(x) \ldots)$ for some $a_{i_1} \in N$, and P contains a production $a \to \sigma_1(a_1, \ldots, a_m)$ such that $i_1 \leq m$. By the inductive assumption there exists a tree $t_{i_1} \in T(G_{a_{i_1}})$ such that $\sigma_{2i_2}(\ldots \sigma_{ki_k}(x) \ldots) \in g_x(t_{i_1})$. Moreover, we may choose for every $i \neq i_1$, $1 \leq i \leq m$, a tree $t_i \in T(G_{a_i})$. Then $t = \sigma_1(t_1, \ldots, t_m) \in T(G_a)$ and $p \in g_x(t) \subseteq g(T(G_a))$.

Conversely, let $p \in g(T(G_a))$. Then $p \in g_x(t)$ for some $t \in T(G_a)$. Obviously, t is of the form $\sigma_1(t_1, \ldots, t_m)$, where $i_1 \leq m$, and it has a derivation

$$a \Rightarrow_G \sigma_1(a_1, \ldots, a_m) \Rightarrow^*_G t.$$

This means that P' contains the production $a \to \sigma_{1i_1}(a_{i_1})$. Moreover, $t_{i_1} \in T(G_{a_{i_1}})$ and $\sigma_{2i_2}(\ldots \sigma_{ki_k}(x) \ldots) \in g_x(t_{i_1})$. Hence, we get a derivation

$$a \Rightarrow^*_G \sigma_{1i_1}(a_{i_1}) \Rightarrow^*_{G'} p,$$

which shows that $p \in T(G'_a)$. □

Let g be the mapping of Definition 2.11.1 associated with a given frontier alphabet X. Then we write $\tau_X = gg^{-1}$. It is clear that τ_X is a closure operation in $F_\Sigma(X)$, i.e., for all $S, T \subseteq F_\Sigma(X)$,

(i) $S \subseteq S\tau_X$,

(ii) $S \subseteq T$ implies $S\tau_X \subseteq T\tau_X$, and

(iii) $S\tau_X\tau_X = S\tau_X$.

For any $T \subseteq F_\Sigma(X)$, $T\tau_X$ is the *closure* of T, and T is said to be *closed* if $T\tau_X = T$.

Now, consider an arbitrary NDR ΣX-recognizer $\mathbf{A} = (\mathcal{A}, A', \alpha)$. For each $a \in A$, let

$$T(\mathbf{A}, a) = \{t \in F_\Sigma(X) \mid a \in t\tilde{\alpha}\}.$$

A state $a \in A$ is a 0-*state*, if $T(\mathbf{A}, a) = \emptyset$. We say that $\mathbf{A}$ is *normalized* if for all $m > 0$, $\sigma \in \Sigma_m$ and $a \in A$ one of the following two alternatives holds:

(1) Each component of every vector in $\sigma^{\mathcal{A}}(a)$ is a 0-state.

(2) No component of any vector of $\sigma^{\mathcal{A}}(a)$ is a 0-state.

A normalized NDR ΣX-recognizer $\mathbf{A}$ has the following important property. Let $p \in g_x(s)$ $(x \in X)$ for some ΣX-tree s such that $\mathbf{A}$ has a computation on s which begins at the root in an initial state and ends at the leaf corresponding to p in a state which belongs to $x\alpha$. Then there exists a tree t in $T(\mathbf{A})$ such that $p \in g_x(t)$. Such a t can be built around the x-path p by completing it with trees from appropriate $T(\mathbf{A}, a)$-forests.

An NDR ΣX-recognizer $\mathbf{A}$ becomes normalized if we omit from each set $\sigma^{\mathcal{A}}(a)$ every vector which contains a 0-state. This does not change $T(\mathbf{A})$ because the use of a vector containing a 0-state cannot lead to an accepting computation. Hence, we have

Lemma 2.11.3 *For every NDR-recognizer there is an equivalent normalized NDR-recognizer.* □

We associate with each NDR ΣX-recognizer $\mathbf{A}$ a DR ΣX-recognizer $\mathfrak{p}\mathbf{A} = (\mathfrak{p}\mathcal{A}, A', \beta)$ defined as follows:

(i) $\mathfrak{p}\mathcal{A} = (\mathfrak{p}A, \Sigma)$ is the deterministic root-to-frontier algebra such that

$$\sigma^{\mathfrak{p}\mathcal{A}}(H) = \Big(\bigcup(\pi_1(\sigma^{\mathcal{A}}(a)) \mid a \in H), \ldots, \bigcup(\pi_m(\sigma^{\mathcal{A}}(a)) \mid a \in H)\Big)$$

for all $H \in \mathfrak{p}A$, $m > 0$ and $\sigma \in \Sigma_m$. Here π_i $(1 \leq i \leq m)$ is the i^{th} projection.

(ii) For each $x \in X$, $x\beta = \{H \in \mathfrak{p}A \mid H \cap x\alpha \neq \emptyset\}$.

Lemma 2.11.4 *For every normalized NDR ΣX-recognizer $\mathbf{A}$, $T(\mathfrak{p}\mathbf{A}) = T(\mathbf{A})\tau_X$.*

Proof. In order to prove the inclusion $T(\mathfrak{p}\mathbf{A}) \subseteq T(\mathbf{A})\tau_X$, we consider an arbitrary tree $s \in T(\mathfrak{p}\mathbf{A})$ and an x-path $p \in g_x(s)$ $(x \in X)$. We should show that $p \in g(T(\mathbf{A}))$. Let $p = \sigma_{1i_1}(\ldots(\sigma_{ki_k}(x))\ldots)$. By the definition of $\mathfrak{p}\mathbf{A}$ there are states $a_0, a_1, \ldots, a_k \in A$ such that

(i) $a_0 \in A'$ and $a_k \in x\alpha$, and

(ii) $a_j \in \pi_{i_j}(\sigma_j^{\mathcal{A}}(a_{j-1}))$ for $j = 1, \ldots, k$.

Since $\mathbf{A}$ is normalized, this implies that there is a tree $t \in T(\mathbf{A})$ such that $p \in g_x(t)$. Hence $p \in g(T(\mathbf{A}))$. Now, let $s \in T(\mathbf{A})\tau_X$ and consider any x-path

$$p = \sigma_{1i_1}(\ldots\sigma_{ki_k}(x)\ldots) \in g_x(s) \quad (x \in X).$$

Then $p \in g_x(t)$ for some $t \in T(\mathbf{A})$ and there are states $a_0, a_1, \ldots, a_k \in A$ such that the above conditions (i) and (ii) hold. But the definition of $\mathfrak{p}\mathbf{A}$ implies that the state of $\mathfrak{p}\mathbf{A}$ at the leaf corresponding to p includes a_k for any tree in which p is an x-path. Hence $\mathfrak{p}\mathbf{A}$ arrives at the leaf of s corresponding to p in a state belonging to $x\alpha$. This holds for every leaf of s and therefore $s \in T(\mathfrak{p}\mathbf{A})$. □

Corollary 2.11.5 *If $T \in \mathrm{Rec}(\Sigma, X)$, then $T\tau_X \in \mathrm{Rec}(\Sigma, X)$.* □

Lemmas 2.11.3 and 2.11.4 also imply that every closed recognizable forest is recognized by a DR recognizer. But it is easy to see that $T(\mathfrak{p}\mathbf{A}) = T(\mathbf{A})$ if $\mathbf{A}$ is deterministic. Hence we may state the following result.

Theorem 2.11.6 *A recognizable forest can be recognized by a DR recognizer iff it is closed.* □

The rest of this section deals with the minimization of DR-recognizers. First two general remarks. When $\mathbf{A} = (\mathcal{A}, a_0, \alpha)$ is a DR ΣX-recognizer, then the NDR algebra $\mathcal{A} = (A, \Sigma)$ is deterministic and we may view each $\sigma^{\mathcal{A}}$ ($\sigma \in \Sigma_m$, $m > 0$) as a mapping

$$\sigma^{\mathcal{A}}\colon A \to A^m.$$

Hence we write $\sigma^{\mathcal{A}}(a) = (a_1, \ldots, a_m)$ rather than $\sigma^{\mathcal{A}}(a) = \{(a_1, \ldots, a_m)\}$. The second remark concerns normalized DR recognizers. If the DR ΣX-recognizer $\mathbf{A}$ is normalized, one of the following conditions holds for each pair $(a, \sigma) \in A \times \Sigma$:

(1) Every component of $\sigma^{\mathcal{A}}(a)$ is a 0-state.

(2) No component of $\sigma^{\mathcal{A}}(a)$ is a 0-state.

Of course, Lemma 2.11.3 and the construction which led to it are valid here, too, but we define a "standard" normalized form $\mathbf{A}^* = (\mathcal{A}^*, a_0, \alpha)$ of $\mathbf{A}$ as follows:

(i) If $\mathbf{A}$ has no 0-state, then put $\mathbf{A}^* = \mathbf{A}$.

(ii) If $\mathbf{A}$ has a 0-state, choose one of them, say d, and define then for all $m > 0$, $\sigma \in \Sigma_m$, and $a \in A$,

$$\sigma^{\mathcal{A}^*}(a) = \begin{cases} (d, \ldots, d)\ (\in A^m) & \text{if } \sigma^{\mathcal{A}}(a) \text{ contains a 0-state,} \\ \sigma^{\mathcal{A}}(a) & \text{otherwise.} \end{cases}$$

It is easy to prove that $\mathbf{A}^*$ is normalized and deterministic, and that $T(\mathbf{A}^*) = T(\mathbf{A})$.

Normalized DR recognizers have also the following useful property.

Lemma 2.11.7 *Let $\mathbf{A}$ and $\mathbf{B}$ be normalized DR ΣX-recognizers, and let $a \in A$, $b \in B$, $m > 0$, $\sigma \in \Sigma_m$, $\sigma^{\mathcal{A}}(a) = (a_1, \ldots, a_m)$ and $\sigma^{\mathcal{B}}(b) = (b_1, \ldots, b_m)$. If $T(\mathbf{A}, a) = T(\mathbf{B}, b)$, then $T(\mathbf{A}, a_i) = T(\mathbf{B}, b_i)$ for all $i = 1, \ldots, m$.*

Proof. If one of the states a_i $(1 \leq i \leq m)$ is a 0-state, then all of them are. Moreover, $T(\mathbf{A}, a) = T(\mathbf{B}, b)$ does not contain any tree of the form $\sigma(t_1, \ldots, t_m)$. Hence, one of the forests $T(\mathbf{B}, b_i)$ $(1 \leq i \leq m)$, and therefore every one of them, is empty. Thus $T(\mathbf{A}, a_i) = T(\mathbf{B}, b_i) = \emptyset$ for all $i = 1, \ldots, m$.

Suppose now that $T(\mathbf{A}, a_i) \neq \emptyset$ and $T(\mathbf{B}, b_i) \neq \emptyset$ for all $i = 1, \ldots, m$. Consider any i $(1 \leq i \leq m)$ and $t_i \in T(\mathbf{A}, a_i)$. Choose any $t_1 \in T(\mathbf{A}, a_1), \ldots, t_{i-1} \in T(\mathbf{A}, a_{i-1}), t_{i+1} \in T(\mathbf{A}, a_{i+1}), \ldots, t_m \in T(\mathbf{A}, a_m)$. Then $\sigma(t_1, \ldots, t_m) \in T(\mathbf{A}, a) = T(\mathbf{B}, b)$ implies $t_i \in T(\mathbf{B}, b_i)$. By a symmetrical argument, $T(\mathbf{B}, b_i) \subseteq T(\mathbf{A}, a_i)$ holds for every $i = 1, \ldots, m$. Hence, $T(\mathbf{A}, a_i) = T(\mathbf{B}, b_i)$ for every $i = 1, \ldots, m$, as required. $\square$

We shall now define a few algebraic concepts for DR recognizers. Let $\mathbf{A} = (\mathcal{A}, a_0, \alpha)$ and $\mathbf{B} = (\mathcal{B}, b_0, \beta)$ be DR ΣX-recognizers.

A *homomorphism* from $\mathbf{A}$ to $\mathbf{B}$ is a mapping $\varphi\colon A \to B$ such that

(i) for all $m > 0$, $\sigma \in \Sigma_m$ and $a \in A$, $\sigma^{\mathcal{B}}(a\varphi) = (a_1\varphi, \ldots, a_m\varphi)$, where $(a_1, \ldots, a_m) = \sigma^{\mathcal{A}}(a)$,

(ii) $a_0\varphi = b_0$, and

(iii) for every $x \in X$, $x\beta\varphi^{-1} = x\alpha$.

If φ is a homomorphism from $\mathbf{A}$ to $\mathbf{B}$, we write $\varphi\colon \mathbf{A} \to \mathbf{B}$. If such a φ is surjective, it is called an *epimorphism.* For an epimorphism condition (iii) implies $x\alpha\varphi = x\beta$, too. If there exists an epimorphism φ from $\mathbf{A}$ onto $\mathbf{B}$, then $\mathbf{B}$ is an *epimorphic* image of $\mathbf{A}$. If $\varphi\colon \mathbf{A} \to \mathbf{B}$ is bijective, then $\mathbf{A}$ and $\mathbf{B}$ are *isomorphic*, and we write $\mathbf{A} \cong \mathbf{B}$.

A *congruence* on $\mathbf{A}$ is an equivalence relation ϱ on A such that

(i) for all $m > 0$, $\sigma \in \Sigma_m$ and $a, a' \in A$, $a\varrho = a'\varrho$ implies $\sigma^{\mathcal{A}}(a)/\varrho = \sigma^{\mathcal{A}}(a')/\varrho$ (recall the notation from Section 1.1), and

(ii) ϱ saturates every set $x\alpha$ $(x \in X)$.

If ϱ is a congruence on $\mathbf{A}$, then the *quotient recognizer* determined by ϱ is the DR ΣX-recognizer

$$\mathbf{A}/\varrho = (\mathcal{A}/\varrho, a_0\varrho, \alpha_\varrho),$$

where $\mathcal{A}/\varrho = (A/\varrho, \Sigma)$ is defined by

$$\sigma^{\mathcal{A}/\varrho}(a\varrho) = \sigma^{\mathcal{A}}(a)/\varrho \quad (\sigma \in \Sigma_m, m > 0, a \in A),$$

and $\alpha_\varrho\colon X \to A/\varrho$ is defined by $x\alpha_\varrho = x\alpha/\varrho$ $(x \in X)$. It is easy to see that $\mathbf{A}/\varrho$ is well-defined.

The following theorem is easily obtained by modifying the proofs of the corresponding facts from algebra.

Theorem 2.11.8 *Let* $\mathbf{A}$ *and* $\mathbf{B}$ *be DR* ΣX*-recognizers.*

(a) If ϱ *is a congruence of* $\mathbf{A}$*, then the natural mapping* $\varrho^\sharp\colon A \to A/\varrho$ *defines an epimorphism of* $\mathbf{A}$ *onto* $\mathbf{A}/\varrho$*.*

(b) If $\varphi\colon \mathbf{A} \to \mathbf{B}$ *is an epimorphism, then* $\varrho = \varphi\varphi^{-1}$ *is a congruence on* $\mathbf{A}$*, and* $\mathbf{A}/\varrho \cong \mathbf{B}$*.* □

The following fact will be needed later.

Theorem 2.11.9 *If* $\mathbf{B}$ *is an epimorphic image of* $\mathbf{A}$*, then* $T(\mathbf{A}) = T(\mathbf{B})$*.*

Proof. Let $\varphi\colon \mathbf{A} \to \mathbf{B}$ be an epimorphism. We verify by tree induction that

$$t\tilde{\alpha} = t\tilde{\beta}\varphi^{-1} \text{ and } t\tilde{\alpha}\varphi = t\tilde{\beta}, \tag{*}$$

for every $t \in F_\Sigma(X)$.

1° For $t = x \in X$, (*) follows directly from the fact that φ is an epimorphism.

2° Let $t = \sigma(t_1, \ldots, t_m)$ and assume that (*) holds for $t_1, \ldots, t_m$. Suppose $a \in t\tilde{\alpha}$. If $\sigma^{\mathcal{A}}(a) = (a_1, \ldots, a_m)$, this means that $a_1 \in t_1\tilde{\alpha}, \ldots, a_m \in t_m\tilde{\alpha}$. Hence, $a_1\varphi \in t_1\tilde{\beta}, \ldots, a_m\varphi \in t_m\tilde{\beta}$. This implies

$$\sigma^{\mathcal{B}}(a\varphi) = (a_1\varphi, \ldots, a_m\varphi) \in t_1\tilde{\beta} \times \cdots \times t_m\tilde{\beta}.$$

Hence, $a\varphi \in t\tilde{\beta}$. Suppose now that $a\varphi \in t\tilde{\beta}$, and let $\sigma^{\mathcal{A}}(a) = (a_1, \ldots, a_m)$. Then $a_1\varphi \in t_1\tilde{\beta}, \ldots, a_m\varphi \in t_m\tilde{\beta}$, which implies $a_1 \in t_1\tilde{\alpha}, \ldots, a_m \in t_m\tilde{\alpha}$. Hence, $a \in t\tilde{\alpha}$. The equality $t\tilde{\alpha} = t\tilde{\beta}\varphi^{-1}$ implies $t\tilde{\alpha}\varphi = t\tilde{\beta}$ as φ is surjective.

Now, (*) implies that for every $t \in F_\Sigma(X)$,

$$\begin{aligned} t \in T(\mathbf{A}) \quad &\text{iff} \quad a_0 \in t\tilde{\alpha} \\ &\text{iff} \quad a_0\varphi(= b_0) \in t\tilde{\alpha}\varphi(= t\tilde{\beta}) \\ &\text{iff} \quad t \in T(\mathbf{B}). \end{aligned}$$ □

We call two states a and a' of a DR ΣX-recognizer $\mathbf{A}$ *equivalent*, and we write $a \sim_{\mathbf{A}} a'$ (or just $a \sim a'$), if $T(\mathbf{A}, a) = T(\mathbf{A}, a')$. Obviously, $\sim_{\mathbf{A}}$ is an equivalence relation on A. We say that $\mathbf{A}$ is *reduced*, if $\sim_{\mathbf{A}} = \delta_A$.

Lemma 2.11.10 *If* $\mathbf{A}$ *is a normalized DR* ΣX*-recognizer, then* $\sim$ *is a congruence on* $\mathbf{A}$ *and* $\mathbf{A}/\sim$ *is reduced.*

Proof. First we show that $\sim$ is a congruence relation.

(i) Consider any $m > 0$, $\sigma \in \Sigma_m$ and $a, a' \in A$ such that $a \sim a'$. Let

$$\sigma^{\mathcal{A}}(a) = (a_1, \ldots, a_m) \quad \text{and} \quad \sigma^{\mathcal{A}}(a') = (a'_1, \ldots, a'_m).$$

But $a \sim a'$ means that $T(\mathbf{A}, a) = T(\mathbf{A}, a')$, and Lemma 2.11.7 implies that

$$T(\mathbf{A}, a_i) = T(\mathbf{A}, a'_i) \quad \text{for all} \quad i = 1, \ldots, m.$$

Hence, $a_i \sim a'_i$ for all $i = 1, \ldots, m$.

(ii) If $a \in x\alpha$ and $a \sim a'$, for some $x \in X$ and $a, a' \in A$, then $x \in T(\mathbf{A}, a) = T(\mathbf{A}, a')$ implies $a' \in x\alpha$. Hence, $\sim$ saturates $x\alpha$.

Now we know that the quotient recognizer $\mathbf{A}/\sim$ can be defined. It is reduced as $(a\sim) \sim_{\mathbf{A}/\sim} (a'\sim)$ implies $a\sim = a'\sim$ $(a, a' \in A)$ because, by Theorem 2.11.9,

$$T(\mathbf{A}, a) = T(\mathbf{A}/\sim, a\sim) = T(\mathbf{A}/\sim, a'\sim) = T(\mathbf{A}, a').$$ □

Let $a, a' \in A$. We write $a \vdash a'$ if there exist an $m > 0$ and a $\sigma \in \Sigma_m$ such that a' appears in $\sigma^{\mathcal{A}}(a)$. The reflexive, transitive closure of $\vdash$ is denoted by $\vdash^*$. If $a \vdash^* a'$, we say that a' is *reachable* from a. The DR recognizer $\mathbf{A}$ is said to be *connected* if every state is reachable from the initial state.

The *connected component*

$$\mathbf{A}^c = (\mathcal{A}^c, a_0, \alpha^c)$$

of a DR ΣX-recognizer $\mathbf{A}$ is defined as follows:

(i) $\mathcal{A}^c = (A^c, \Sigma)$, where $A^c = \{a \in A \mid a_0 \vdash^* a\}$ and $\sigma^{\mathcal{A}^c}(a) = \sigma^{\mathcal{A}}(a)$ for all $\sigma \in \Sigma$ and $a \in A^c$.

(ii) $x\alpha^c = x\alpha \cap A^c$ for each $x \in X$.

Clearly, the operations $\sigma^{\mathcal{A}^c} \colon A^c \to (A^c)^m$ are completely defined ($m > 0, \sigma \in \Sigma_m$).

The proof of Lemma 2.11.11 is quite straightforward and we shall omit it.

Lemma 2.11.11 *Let* $\mathbf{A}$ *be any DR* ΣX*-recognizer. Then*

(a) $\mathbf{A}^c$ *is connected and deterministic,*

(b) $\mathbf{A}^c = \mathbf{A}$ *iff* $\mathbf{A}$ *is connected,*

(c) $T(\mathbf{A}^c) = T(\mathbf{A})$*, and*

(d) if $\mathbf{A}$ *is normalized, then so is* $\mathbf{A}^c$*.* □

We are now ready to present the main theorem of the minimization theory of DR recognizers.

Theorem 2.11.12 *Let* $\mathbf{A}$ *and* $\mathbf{B}$ *be connected, normalized DR* ΣX*-recognizers. Then* $T(\mathbf{A}) = T(\mathbf{B})$ *iff* $\mathbf{A}/{\sim_\mathbf{A}} \cong \mathbf{B}/{\sim_\mathbf{B}}$*.*

Proof. If $\mathbf{A}/{\sim_\mathbf{A}}$ and $\mathbf{B}/{\sim_\mathbf{B}}$ are isomorphic, then

$$T(\mathbf{A}) = T(\mathbf{A}/{\sim_\mathbf{A}}) = T(\mathbf{B}/{\sim_\mathbf{B}}) = T(\mathbf{B})$$

by Theorems 2.11.8 and 2.11.9.

Assume now that $T(\mathbf{A}) = T(\mathbf{B})$. We define a mapping

$$\varphi \colon A/{\sim_\mathbf{A}} \to B/{\sim_\mathbf{B}}$$

by the condition that

$$(a{\sim_\mathbf{A}})\varphi = b{\sim_\mathbf{B}} \quad \text{if} \quad T(\mathbf{A}, a) = T(\mathbf{B}, b) \quad (a \in A, b \in B).$$

The following steps (i)–(v) show that φ is the required isomorphism.

(i) $(a{\sim_\mathbf{A}})\varphi$ is defined for all $a{\sim_\mathbf{A}} \in A/{\sim_\mathbf{A}}$. Since $\mathbf{A}$ is connected, there exist for every $a \in A$ a $k \geq 0$ and states $a_1, \ldots, a_k \in A$ such that

$$a_0 \vdash a_1 \vdash a_2 \vdash \cdots \vdash a_k = a.$$

Using Lemma 2.11.7 one shows by induction on the smallest k (corresponding to the given a) that there is a b such that $T(\mathbf{A}, a) = T(\mathbf{B}, b)$.

(ii) φ is well-defined. If $a \sim_\mathbf{A} a'$, $T(\mathbf{A}, a) = T(\mathbf{B}, b)$ and $T(\mathbf{A}, a') = T(\mathbf{B}, b')$ for some $a, a' \in A$ and $b, b' \in B$, then $b{\sim_\mathbf{B}} = b'{\sim_\mathbf{B}}$.

(iii) φ is injective. Similarly as (ii).

(iv) φ is surjective. If we exchange the roles of $\mathbf{A}$ and $\mathbf{B}$ in (i), we see that there exists for every $b \in B$ an $a \in A$ such that $T(\mathbf{A}, a) = T(\mathbf{B}, b)$.

(v) φ is a homomorphism. That φ preserves the operations follows from Lemma 2.11.7. If $a{\sim_{\mathbf{A}}} \in x\alpha/{\sim_{\mathbf{A}}}$ $(x \in X)$ and $(a{\sim_{\mathbf{A}}})\varphi = b{\sim_{\mathbf{B}}}$, then $x \in T(\mathbf{A}, a) = T(\mathbf{B}, b)$ implies $b{\sim_{\mathbf{B}}} \in x\beta/{\sim_{\mathbf{B}}}$. Likewise, $(a{\sim_{\mathbf{A}}})\varphi = b{\sim_{\mathbf{B}}} \in x\beta/{\sim_{\mathbf{B}}}$ implies $a{\sim_{\mathbf{A}}} \in x\alpha/{\sim_{\mathbf{A}}}$. Thus $x\beta{\sim_{\mathbf{B}}}\varphi^{-1} = x\alpha{\sim_{\mathbf{A}}}$ for every $x \in X$. □

A DR recognizer $\mathbf{A}$ is said to be *minimal* if no DR recognizer with fewer states recognizes $T(\mathbf{A})$. If $\mathbf{A}$ is minimal, then it is connected by Lemma 2.11.11. As $T(\mathbf{A}^*) = T(\mathbf{A})$ we may also assume that $\mathbf{A}$ is normalized. Then $T(\mathbf{A}) = T(\mathbf{A}/{\sim_{\mathbf{A}}})$ implies that $\mathbf{A}$ should be reduced, too. Conversely, if $\mathbf{A}$ is connected, normalized and reduced, then it is minimal and every normalized minimal DR recognizer is isomorphic to it (Theorem 2.11.12). These facts imply that the following three steps yield for any DR recognizer $\mathbf{A}$ an equivalent minimal DR recognizer $\mathbf{B}$. Moreover, this $\mathbf{B}$ is normalized.

Step 1. Form $\mathbf{A}^*$.

Step 2. Form $\mathbf{A}^{*c}$.

Step 3. Form $\sim$ for $\mathbf{A}^{*c}$, and put $\mathbf{B} = \mathbf{A}^{*c}/{\sim}$.

It is not hard to see that these steps are effectively realizable.

2.12 EXERCISES

1. Let leaf(t) denote the set of symbols which label the leaves of a given ΣX-tree t. Define leaf(t) by tree induction.

2. (a) Define the length $|t|$ of a ΣX-tree t (as a word) by tree induction.
   (b) For the sake of simplicity, let $\Sigma = \Sigma_2$. Derive an upper bound for $|t|$ in terms of hg(t). Give also a lower bound for $|t|$ in terms of hg(t).

3. Let $\Sigma = \Sigma_0 \cup \Sigma_2$, $\Sigma_0 = \{\omega\}$, $\Sigma_2 = \{\sigma\}$, and let $X = \{x, y\}$. Construct a CF grammar which generates the set $F_\Sigma(X)$ of all ΣX-trees (when these are viewed as words). Is the set of all ΣX-trees still a CF language if we use the Polish notation for ΣX-terms?

4. Let Σ and X be as in the previous exercise. Decide which ones of the ΣX-forests, R, S, and T are recognizable, when these are defined as follows:
   (i) $t \in R$ iff the number of σ's in t is odd.
   (ii) $t \in S$ iff all paths from the root to a leaf are of the same length.

(iii) $t \in T$ iff no leaf labelled by y appears to the left of a leaf labelled by x.

5. Let **A** be an NDF ΣX-recognizer and **B** an NDR ΣX-recognizer which are associated in the sense of Section 2.2. Prove the equality $\hat{\alpha} = \tilde{\beta}$ by tree induction.

6. Use regular tree grammars to prove directly that $\mathrm{Rec}(\Sigma, X)$ is closed under σ-products (Corollary 2.4.12).

7. Let us change the definition of the forest product $T(x \leftarrow T_x)$ (cf. Definition 2.4.3) in such a way that every occurrence of each letter $x \in X$ should be rewritten as the same tree $t_x \in T_x$. Then we get the new product

$$T[x \leftarrow T_x \mid x \in X] = \{t(x \leftarrow t_x \mid x \in X) \mid t \in T, t_x \in T_x \ (x \in X)\}.$$

Is $\mathrm{Rec}(\Sigma, X)$ closed under this product?

8. Let T be a ΣX-forest and let $x \in X$. Describe the forests $T \cdot_x \emptyset$ and $\emptyset \cdot_x T$.

9. Do the following laws hold for x-products?
   (a) $R \cdot_x (S \cup T) = (R \cdot_x S) \cup (R \cdot_x T)$.
   (b) $(R \cup S) \cdot_x T = (R \cdot_x T) \cup (S \cdot_x T)$.
   (c) $R \cdot_x (S \cdot_y T) = (R \cdot_x S) \cdot_y T$.

10. Let us change Definition 2.4.7 so that $T^{j+1,x} = T \cdot_x T^{j,x} \cup T^{j,x}$ for all $j \geq 0$. Does the new x-iteration coincide with the original one? If not, does it preserve recognizability?

11. Let $x \neq y$ $(x, y \in X)$. Is it possible that $(T^{*x})^{*y} \neq (T^{*y})^{*x}$ for some ΣX-forest T?

12. Show that the construction of the tree recognizer for the forest $S^{-x}T$ given in the proof of Theorem 2.4.10 is effective when S is recognizable (and given by a tree recognizer).

13. Prove Lemma 2.4.17.

14. Prove Corollary 2.4.20 directly without using Theorems 2.4.16 and 2.4.18.

15. Let $\varphi\colon \mathcal{F}_\Sigma(X) \to \mathcal{F}_\Sigma(X)$ be a homomorphism of Σ-algebras. Prove that if $T \in \mathrm{Rec}(\Sigma, X)$, then (a) $T\varphi \in \mathrm{Rec}(\Sigma, X)$ and (b) $T\varphi^{-1} \in \mathrm{Rec}(\Sigma, X)$.

16. The set of *atomic* ΣX-trees is defined as

$$A(\Sigma, X) = \{\sigma(x_{i_1}, \ldots, x_{i_m}) \mid m \geq 0,\ \sigma \in \Sigma_m,\ x_{i_1}, \ldots, x_{i_m} \in X\}.$$

For the sake of definiteness, let $X = \{x_1, \ldots, x_n\}$ $(n \geq 1)$. Prove that

$$(\ldots(A(\Sigma, X)^{*x_1})^{*x_2} \ldots)^{*x_n} = F_\Sigma(X)$$

(cf. Thatcher and Wright [241]).

17. Let $\Sigma = \Sigma_2 = \{\sigma\}$ and $X = \{x\}$. Write a regular expression for the forest of all ΣX-trees which contain an even number of σ's.

18. Let Σ and X be as in Exercise 3. Construct a ΣX-recognizer for the forest represented by the regular expression $\sigma(x, y) \cdot_z \sigma(\omega, \sigma(\omega, z))^{*z}$.

19. Prove Theorem 2.6.6.

20. If $\mathbf{A}$ is a ΣX-recognizer and $T(\mathbf{A}) = T$, then $\hat{\alpha}$ is a homomorphism from $\mathbf{F}_T$ to $\mathbf{A}$. Prove Lemma 2.6.2 using this observation.

21. Prove Lemma 2.6.14.

22. In Section 2.7 we noted that one may define recognizability for subsets of algebras. We call T $(\subseteq A)$ a *recognizable subset* of the Σ-algebra $\mathcal{A} = (A, \Sigma)$, if there exists a congruence θ of finite index which saturates T. Denote by $\operatorname{Rec} \mathcal{A}$ the set of all recognizable subsets of $\mathcal{A}$. Prove the following facts:
    (a) If $S, T \in \operatorname{Rec} \mathcal{A}$, then $S \cup T, S \cap T, S - T \in \operatorname{Rec} \mathcal{A}$.
    (b) If $\varphi\colon \mathcal{A} \to \mathcal{B}$ is a homomorphism and $T \in \operatorname{Rec} \mathcal{B}$, then $T\varphi^{-1} \in \operatorname{Rec} \mathcal{A}$.

    (Note. $T \in \operatorname{Rec} \mathcal{A}$ does not imply $T\varphi \in \operatorname{Rec} \mathcal{B}$. A counterexample where $\mathcal{A}$ and $\mathcal{B}$ are monoids can be found in Eilenberg's book (Vol. A) mentioned among the references of Chapter 1.)

23. Let $\Sigma = \Sigma_2 = \{\sigma\}$ and $X = \{x, y\}$, and let (U, V) be the least fixed-point of the system
$$u = x + \sigma(\sigma(u, v), y)$$
$$v = \sigma(y, u).$$
    Find a regular (Σ, X, k)-polynomial Π $(k \geq 2)$ such that U and V can be represented as unions of some components of $[\hat{\Pi}]$. (For a general treatment of such questions see Mezei and Wright [182].)

24. Show that every local ΣX-forest $\operatorname{Loc}(R, F)$ can be represented in terms of the elementary forests and the elementary operations intersection, union, and restriction. Note the resulting connection between the Theorems 2.8.6 and 2.9.5.

25. Show that the decidability of the equivalence problem of tree recognizers follows from the results of Section 2.6.

26. Prove Lemma 2.10.5.

27. Prove that it is decidable whether a recognizable forest can be recognized by a DR-recognizer.

28. Are all local forests recognizable by DR-recognizers?

29. Present algorithms for carrying out Steps 2 and 3 of the minimization algorithm for DR-recognizers which was outlined in Section 2.11.

2.13 NOTES AND REFERENCES

The observation (made about 1960) that finite automata may be defined as unary algebras is attributed to J. R. Büchi and J. B. Wright (see Mezei and Wright [182], Thatcher [239]). The generalization to tree automata was suggested independently by Doner [65, 66] and by Thatcher and Wright [240, 241]. Many of the basic results presented in this chapter were obtained in various forms by several authors, and often it would be hard to establish any priorities. Most of the important early contributions can be found in Mezei and Wright [182], Eilenberg and Wright [69], Thatcher and Wright [241], Doner [66], Thatcher [238], Pair and Quere [196], Brainerd [39, 40], Arbib and Give'on [5], and Magidor and Moran [166].

Already in many of these papers trees were defined as terms, and this formalism is now very common. However, most authors use no separate frontier alphabet. Also, often operators may have more than one rank. The original reason for our use of frontier alphabets was to keep the character of the algebras independent of the number of frontier symbols. Another popular formalism defines a tree as a pair (D, λ) consisting of a *"tree domain"* D and a labelling mapping λ. Each element d of D specifies a node of the tree and $\lambda(d)$ is the label of this node. This definition is quite convenient for discussing concepts and operations which involve specific occurrences of subtrees. Tree domains were introduced by S. Gorn in 1965 (for a reference, see Brainerd [40]).

Deterministic and nondeterministic frontier-to-root tree recognizers were defined, and their equivalence was established, by Thatcher and Wright [241], Doner [66], and Magidor and Moran [166]. Root-to-frontier tree recognizers were introduced by Rabin [204], and Magidor and Moran [166]. Magidor and Moran showed the equivalence of NDF and NDR recognizers, and they also studied DR recognizers.

Regular tree grammars and the results of Section 2.3 are due to Brainerd [40]. In Brainerd's grammars the form of the productions is quite general, but he shows that they can be reduced to, what we call, regular tree grammars.

The Boolean closure properties of $\mathrm{Rec}(\Sigma, X)$ were noted in many of the early papers mentioned above. The Kleene theorem (Theorem 2.5.8) was proved by Thatcher and Wright [241] and by Magidor and Moran [166]. A simplified proof was given by Arbib and Give'on [5]. Alphabetic tree homomorphisms (called there projections) and Corollary 2.4.20 appear in Thatcher and Wright [241]. General tree homomorphisms arose as special cases of finite-state tree transductions (see Thatcher [238, 239] and Engelfriet [75]). Tree transductions and tree homomorphisms will be considered in Chapter 4. Forest products (or *"substitutions"*) were also introduced in this context. Ito and Ando [127] present a complete axiom system for the equality of regular expressions (cf. also Ésik [91]).

Minimal tree recognizers and Nerode congruences are discussed in Brainerd [39], Arbib and Give'on [5], and Magidor and Moran [166].

The theory of equational forests is from Mezei and Wright [182]. We have simplified the exposition by considering only regular fixed-point equations. Mezei and Wright considered also equational and recognizable subsets of general algebras (cf. Exercise 22). They proved that the equational subsets of an algebra (of finite type) are the homomorphic images of the recognizable subsets of term algebras. Applied to term algebras this result gives our Theorem 2.7.9. EILENBERG and WRIGHT [69] present these results in a category theoretic form. For various classes of subsets in general algebras we refer also to WAGNER [249], LESCANNE [150], MARCHAND [175], SHEPARD [220], and STEINBY [227]. DUBINSKY [67] discusses equational and recognizable subsets of nondeterministic algebras. MAIBAUM [170], and ENGELFRIET and SCHMIDT [85] extend the subject into another direction by considering many-sorted algebras.

The material of Section 2.8 is from COSTICH [52]. Local forests, or similar concepts, and results related to Theorems 2.9.4 and 2.9.5 can be found in DONER [66], THATCHER [237, 238], and TAKAHASHI [234].

The characterization of the forests recognizable by DR recognizers is from VIRÁGH [248], although the basic idea is discernible already in MAGIDOR and MORAN [166] (cf. also THATCHER [239]). The minimization theory of DR recognizers appears in GÉCSEG and STEINBY [104].

We should also mention an alternative approach, originating with PAIR and QUERE [196] and popular among French writers, in which the basic objects are tuples of trees rather than trees. The usual tree operations are then augmented by operations which catenate tuples of trees or form a tree from an m-tuple by creating a new root labelled by an m-ary operator. As an abstract framework for their study Pair and Quere introduced *"binoids"*, the tuples of trees form such a binoid. Their results include the basic closure properties and a Kleene Theorem. This formalism has been developed further by ARNOLD and DAUCHET [21] to a theory of *"magmoids"* which also embodies many of the ideas of EILENBERG and WRIGHT [69]. ARNOLD [9, 10] discusses many topics relevant to this chapter within the framework of magmoids.

We shall now discuss briefly some topics and applications of the theory not covered by this book. The survey is by no means complete, and in many cases the choices were dictated by personal preference. Some more remarks will be made at the end of Chapters 3 and 4.

The category theoretic treatment of recognizable and equational subsets by EILENBERG and WRIGHT [69] was already mentioned. It is based on Lawvere's *"theories"*. This approach was developed further by GIVE'ON and ARBIB [111], and others. The theory of magmoids has also evolved from the same ideas. We have avoided the use of category theory altogether, but the bibliography contains a sample from the extensive and highly diversified literature on the subject. The items of interest include ALAGIĆ [3, 4], ARBIB and MANES [6], BOBROW and ARBIB [38], GOGUEN [113], GOGUEN et al [114, 115], HORVÁTH [122, 123], and TRNKOVÁ and ADÁMEK [244].

The *structure theory* of tree automata has received little attention although some initial steps were taken already by MAGIDOR and MORAN [166]. RICCI [209] considered cascade products of tree automata. Iterative realizations and general products of tree automata are studied in STEINBY [225]. Two sections of GÉCSEG and STEINBY [105] are devoted to the subject. It is evident that generalizations from the unary case will usually not be easy in this area.

Transition monoids have proved very useful in finite automaton theory and some equivalents of them for tree automata have been suggested. The *"m-ary monoids"* of

Give'on [110] and the *"substitution algebras"* of Yeh [253] are in fact special Menger algebras. The same idea reappears in the *"clone algebras"* of Turner [246]. Sommerhalder [223] develops the concept further and associates with an algebra a sequence $M_1, M_2, \dots$ of monoids. Here M_n consists of all n-tuples of n-ary polynomial functions of the algebra. It would be easy to define syntactic monoids of forests along these lines, but no such theory seems to have evolved yet. Another variant of the transition semigroup concept has been studied by Helton [120].

We shall mention some other algebraic topics of potential interest. A ΣX-forest T is said to be recognizable by a Σ-algebra $\mathcal{A} = (A, \Sigma)$ if one may choose $\alpha\colon X \to A$ and $A'(\subseteq A)$ in such a way that $(\mathcal{A}, \Sigma, A')$ recognizes T. Families of forests recognizable by algebras belonging to a given variety (equational class) were considered by Steinby [224] and by Gécseg and Horváth [103]. For a further study in this direction it would probably be advantageous to follow the example of Eilenberg's theory of M-varieties and varieties of recognizable languages and consider *"ω-varieties"* (usually called *pseudovarieties*) of algebras and the families of forests corresponding to them; an ω-variety is a class of finite algebras closed under the construction of subalgebras, homomorphic images and finite direct products. In Steinby [226] it was shown that Eilenberg's basic variety theorem can be extended to ω-varieties and varieties of recognizable subsets of free algebras (suitably defined). A specialization of this result to term algebras gives a correspondence between ω-varieties and varieties of recognizable forests. A ΣX-forest T is said to be *rationally represented* by an ΩX-recognizer $\mathbf{A}$ if there exists an embedding $\varphi\colon F_\Sigma(X) \to F_\Omega(X)$ of a certain kind such that $T\varphi = T(\mathbf{A})$. A variety $\mathcal{K}$ of algebras is said to be *rationally complete* if every recognizable forest can be rationally represented by a recognizer based on a finite algebra belonging to $\mathcal{K}$. Gécseg [101] studies the rational completeness of varieties and the equivalence of tree recognizers with respect to rational representation. Further results can be found in Maróti [176], and Marchand [173] also contains some related ideas.

We shall now list a few references to some more topics. *Probabilistic tree automata* and related topics have been discussed by Magidor and Moran [166, 167], Ellis [72] and Karpiński [141, 142]. Forests of *infinite trees* appear in Rabin [204], Engelfriet [73], Casteran [50] and Courcelle [54]. An alternative way to generate forests is provided by the *tree adjunct grammars* studied by Joshi, Levy and Takahashi [135, 136], Levy [155], and Levy and Joshi [157]. Also *Lindenmayer systems* (L-systems) for trees have been considered; see Čulik [56], Čulik and Maibaum [57], Engelfriet [76, 79], Karpiński [143], Steyart [230], and Szilard [231].

Although we present our subject as a part of pure automata and formal language theory, it should be clear that it has many connections to the more applied aspects of language specification, translation and semantics. As a conclusion we would like to point out some less obvious areas of application.

When Doner [65, 66] and Thatcher and Wright [240, 241] introduced tree automata their goal was to prove the decidability of the weak second order theory of multiple successors. Further applications to logic can be found in Rabin [204, 205].

In *syntactic pattern recognition* patterns are decomposed into simple basic elements which are represented by letters of an alphabet. A pattern is then represented, for example, as a word. However, essential information about the relations between the basic elements may be lost if the corresponding letters are simply concatenated to form a word. It is possible that these can be described adequately by representing the pattern as a tree, and then tree automata theory may be used. For example, the considered class of patterns may be generated by a tree grammar or recognized by a tree recognizer. One specific problem prompted by syntactic pattern recognition is the *inference of forests* from samples. The interested reader may consult the books by Fu [97] and Gonzalez and Thomason [117]. Some papers from this area are Berger and Pair [33], Brayer and Fu [42], Fu and Bhargava [98], Gonzalez, Edwards and Thomason [116], Lu and Fu [165], Pair [194], Tai [232], and Williams [251].

3 CONTEXT-FREE LANGUAGES AND TREE RECOGNIZERS

The words generated by a context-free grammar can be read from derivation trees. The connection between forests and languages implied by this fact is the subject matter of this chapter. In the first section we define the yield-function by means of which a word is extracted from a tree. In Section 3.2 the basic relations between recognizable forests and context-free grammars are established. The usual definition of derivation trees must be modified slightly as to make them "trees" in our sense of the term, but the difference is inessential. The forest of derivation trees of any CF grammar is shown to be recognizable. On the other hand, we shall see that the yield of any recognizable forest is a CF language. Hence tree recognizers may also be viewed as recognizers of CF languages. The section is concluded by showing that every CF language is the yield of a local forest recognizable by a deterministic R-recognizer.

The inverse image of a CF language under the yield-function is not always a recognizable forest, but we show in the beginning of Section 3.3 that the inverse image of a regular language is a recognizable forest. Also, a slightly restricted converse of this fact is presented. Then we show that every CF language can be obtained from a recognizable forest with a fixed and very simple ranked alphabet. Section 3.3 is concluded by some examples which show how facts about context-free languages can be proved using the theory of recognizable forests.

In Section 3.4 another, less well-known, way to obtain the context-free languages from recognizable forests is presented.

3.1 THE YIELD FUNCTION

We shall now formally define the function that extracts a word from the frontier of a tree. This will also give a function that associates a language with every forest.

Definition 3.1.1 The *yield* $\mathrm{yd}(t)$ of a ΣX-tree t is defined inductively as follows:

1° $\mathrm{yd}(x) = x$ for all $x \in X$.

2° If $t = \sigma(t_1, \ldots, t_m)$ ($m \geq 0$, $\sigma \in \Sigma_m$), then $\mathrm{yd}(t) = \mathrm{yd}(t_1) \ldots \mathrm{yd}(t_m)$.

The *yield* of a ΣX-forest T is the X-language $\mathrm{yd}(T) = \{\mathrm{yd}(t) \mid t \in T\}$.

To obtain the yield of a tree $\sigma(t_1, \ldots, t_m)$ one concatenates the yields of the subtrees $t_1, \ldots, t_m$. In particular, $\mathrm{yd}(\sigma) = e$ for all $\sigma \in \Sigma_0$. More generally, $\mathrm{yd}(t) = e$ iff

$t \in F_\Sigma(\emptyset)$. The mapping

$$\mathrm{yd}\colon F_\Sigma(X) \to X^*$$

is not injective; in general, a word is the yield of several trees.

We use the same symbol yd for its extension to forests. Of course, yd presupposes a Σ and an X although our notation does not show this.

Example 3.1.2 Let $\omega \in \Sigma_0$, $\sigma \in \Sigma_3$, and $x, y \in X$. For $s = \sigma(x, \sigma(y, \omega, y), \omega)$ and $t = \sigma(\omega, x, \sigma(y, y, \omega))$ we have $\mathrm{yd}(s) = \mathrm{yd}(t) = xyy$. □

Whether or not a given word $w \in X^*$ is the yield of some ΣX-tree depends on the length of w and the arities of the operators in Σ.

Lemma 3.1.3 *Let $r(\Sigma) = \{m_1, \ldots, m_k\}$. For a word $w \in X^*$ there exists a tree $t \in F_\Sigma(X)$ such that $\mathrm{yd}(t) = w$ iff the length of w can be expressed in the form*

$$|w| = h_1(m_1 - 1) + \cdots + h_k(m_k - 1) + 1$$

for some (integers) $h_1, \ldots, h_k \geq 0$. □

The proof of the lemma is an exercise. It is easy to see that $\mathrm{yd}(F_\Sigma(X)) = X^*$ iff $\Sigma_0 \neq \emptyset$ and $\Sigma - (\Sigma_1 \cup \Sigma_0) \neq \emptyset$. When this is the case, there exists for every X-language L a ΣX-forest T such that $\mathrm{yd}(T) = L$. The greatest among these is the forest

$$\mathrm{yd}^{-1}(L) = \{t \in F_\Sigma(X) \mid \mathrm{yd}(t) \in L\}.$$

In general, we know just that $\mathrm{yd}(\mathrm{yd}^{-1}(L)) \subseteq L$. From Lemma 3.1.3 one easily gets

Corollary 3.1.4 *For a given $L \subseteq X^*$, there exists a forest $T \subseteq F_\Sigma(X)$ such that $\mathrm{yd}(T) = L$ iff*

$$\{|w| \mid w \in L\} \subseteq \{h_1(m_1 - 1) + \cdots + h_k(m_k - 1) + 1 \mid h_1, \ldots, h_k \geq 0\},$$

where $\{m_1, \ldots, m_k\} = r(\Sigma)$. □

In the following lemma we list some obvious properties of yd and yd^{-1}.

Lemma 3.1.5 *Let S and T be ΣX-forests, and K and L X-languages. Then*

(a) $\mathrm{yd}(S \cup T) = \mathrm{yd}(S) \cup \mathrm{yd}(T)$,

(b) $\mathrm{yd}(S \cap T) \subseteq \mathrm{yd}(S) \cap \mathrm{yd}(T)$,

(c) $\mathrm{yd}^{-1}(K \cup L) = \mathrm{yd}^{-1}(K) \cup \mathrm{yd}^{-1}(L)$,

(d) $\mathrm{yd}^{-1}(K \cap L) = \mathrm{yd}^{-1}(K) \cap \mathrm{yd}^{-1}(L)$, *and*

(e) $\mathrm{yd}^{-1}(K - L) = \mathrm{yd}^{-1}(K) - \mathrm{yd}^{-1}(L)$. □

3.2 CONTEXT-FREE LANGUAGES AND RECOGNIZABLE FORESTS

In the customary definition of derivation trees the inner nodes are labelled by nonterminal symbols and a nonterminal may appear at nodes with different numbers of outgoing edges. Since we allowed a symbol of a ranked alphabet to have just one rank, the definition of derivation trees should be modified accordingly.

Let $G = (N, X, P, a_0)$ be a CF grammar as defined in Section 1.6. We associate with G a ranked alphabet Σ^G thus: for each $m \geq 0$,

$$\Sigma_m^G = \{(a, m) \mid (\exists a \to \eta \in P)|\eta| = m\}.$$

Definition 3.2.1 Let G and Σ^G be as above. For every $d \in N \cup X$ the set $D(G, d)$ of *derivation trees* with d as the root is defined by the following conditions:

1° $D(G, x) = \{x\}$ for each $x \in X$.

2° For $a \in N$, $(a, 0) \in D(G, a)$ iff $a \to e \in P$.

3° Suppose $a \to d_1 \dots d_m \in P$, with $m > 0$, $a \in N$ and $d_1, \dots, d_m \in N \cup X$. If $t_1 \in D(G, d_1), \dots, t_m \in D(G, d_m)$, then $(a, m)(t_1, \dots, t_m) \in D(G, a)$.

4° Nothing is in any $D(G, d)$ unless this follows from a finite number of applications of the rules 1°, 2° and 3°.

The *derivation forest* of G is the $\Sigma^G X$-forest $D(G) = D(G, a_0)$.

Exactly as in the case of conventional derivation trees, every t in $D(G, d)$ ($d \in N \cup X$) corresponds to a unique leftmost derivation in G of the word $\mathrm{yd}(t)$ from d. Also, every derivation

$$d \Rightarrow_G u_1 \Rightarrow_G \cdots \Rightarrow_G u_{k-1} \Rightarrow_G w,$$

with $d \in N \cup X$ and $w \in X^*$, can be described by a tree $t \in D(G, d)$ such that $\mathrm{yd}(t) = w$. This is easily shown by induction on the length of the derivation. Hence, $L(G) = \mathrm{yd}(D(G))$.

Theorem 3.2.2 *The derivation forests of CF grammars are local and, therefore, recognizable.*

Proof. Let $G = (N, X, P, a_0)$ be a CF grammar. It is obvious that $D(G)$ is the local $\Sigma^G X$-forest $L(R, F)$ (in the notation of Section 2.9), where

$$R = \{(a_0, m) \mid m \geq 0, (a_0, m) \in \Sigma_m^G\}$$

and the set F of the allowed forks is defined as follows. If $m > 0$ and $a \to d_1 \dots d_m \in P$, then we include in F every fork $(a, m)(c_1, \dots, c_m)$ such that for all $i = 1, \dots, m$,

$$c_i = \begin{cases} d_i & \text{if } d_i \in X, \\ (d_i, k) & \text{with } k \geq 0 \text{ and } (d_i, k) \in \Sigma_k^G, \text{ if } d_i \in N. \end{cases}$$

Nothing is in F unless this follows from the construction described above. □

It is also easy to see that $D(G)$ is generated by the regular $\Sigma^G X$-grammar $G_D = (N, \Sigma^G, X, P_D, a_0)$, where

$$P_D = \{a \to (a, m)(d_1, \ldots, d_m) \mid m \geq 0,\ a \to d_1 \ldots d_m \in P,\ d_1, \ldots, d_m \in N \cup X\}.$$

Example 3.2.3 Consider the CF grammar

$$G = (\{a_0, b\}, \{x, y\}, \{a_0 \to xa_0b, a_0 \to e, b \to xyb, b \to y\}, a_0).$$

In this case $\Sigma^G = \Sigma^G_0 \cup \Sigma^G_1 \cup \Sigma^G_3$, where $\Sigma^G_0 = \{(a_0, 0)\}$, $\Sigma^G_1 = \{(b, 1)\}$ and $\Sigma^G_3 = \{(a_0, 3), (b, 3)\}$. The productions of the grammar $G_D = (N, \Sigma^G, X, P_D, a_0)$ generating $D(G)$ are $a_0 \to (a_0, 3)(x, a_0, b)$, $a_0 \to (a_0, 0)$, $b \to (b, 3)(x, y, b)$ and $b \to (b, 1)(y)$. The allowed roots of the local forest $D(G)$ are $(a_0, 0)$ and $(a_0, 3)$, and the possible forks are $(a_0, 3)(x, (a_0, 0), (b, 1))$, $(a_0, 3)(x, (a_0, 0), (b, 3))$, $(a_0, 3)(x, (a_0, 3), (b, 1))$, $(a_0, 3)(x, (a_0, 3), (b, 3))$, $(b, 3)(x, y, (b, 1))$, $(b, 3)(x, y, (b, 3))$ and $(b, 1)(y)$. □

Theorem 3.2.2 yields immediately

Corollary 3.2.4 *Every CF language is the yield of a recognizable forest.* □

The converse is also true:

Theorem 3.2.5 *The yield of any recognizable forest is a context-free language.*

Proof. Let $G = (N, \Sigma, X, P, a_0)$ be a regular ΣX-grammar generating the given recognizable ΣX-forest T. To simplify matters we assume that G is in normal form. Now we construct the CF grammar $G_1 = (N, X, P_1, a_0)$ with

$$P_1 = \{a \to \mathrm{yd}'(p) \mid a \to p \in P\}.$$

Here yd' is the yield-function corresponding to the extended frontier alphabet $X \cup N$. Inductions on the lengths of the derivations show that

(1) $a \Rightarrow^*_G t$ implies $a \Rightarrow^*_{G_1} \mathrm{yd}(t)$, for all $a \in N$, $t \in F_\Sigma(X)$, and that

(2) for all $w \in X^*$ and $a \in N$, $a \Rightarrow^*_{G_1} w$ only in case there exists a tree $t \in F_\Sigma(X)$ such that $a \Rightarrow^*_G t$ and $\mathrm{yd}(t) = w$.

These two facts imply that $\mathrm{yd}(T) = L(G_1)$ is CF. □

In view of Theorem 3.2.5 any tree recognizer may be seen as a device which recognizes a CF language by checking the possible syntaxes of given words; a word is accepted iff it is the yield of at least one tree accepted by the tree recognizer.

Definition 3.2.6 The *language recognized* by a ΣX-recognizer $\mathbf{A}$ is the X-language $L(\mathbf{A}) = \mathrm{yd}(T(\mathbf{A}))$.

The previous results can now be expressed as follows.

Theorem 3.2.7 *A language is recognized by a tree recognizer iff it is context-free.* □

The equivalence expressed in Theorem 3.2.7 is effective both ways; for any CF language given by a CF grammar we can construct a tree recognizer, and for any tree recognizer $\mathbf{A}$ we can construct a CF grammar generating $L(\mathbf{A})$.

By Theorem 3.2.2 every CF language is the yield of a local forest. We shall now show that even a smaller class of forests will suffice. To this end we replace derivation trees by trees in which the inner nodes are labelled by complete productions.

With every CF grammar $G = (N, X, P, a_0)$ we associate another ranked alphabet Σ^P defined as follows. For each $m \geq 0$, let

$$\Sigma_m^P = \{(a \to \eta) \mid a \to \eta \text{ is in } P \text{ and } |\eta| = m\},$$

i.e., the m-ary symbols correspond to the productions with right-hand sides of length m.

Definition 3.2.8 Let G and Σ^P be as above. For every $d \in N \cup X$ the set $P(G, d)$ of *production trees* with d at the root is defined by the following conditions:

1° $P(G, x) = \{x\}$ for each $x \in X$.

2° For $a \in N$, $(a \to e) \in P(G, a)$ iff $a \to e \in P$.

3° Suppose $a \to d_1 \ldots d_m \in P$ ($m > 0$, $a \in N$ and $d_1, \ldots, d_m \in N \cup X$). If $p_1 \in P(G, d_1), \ldots, p_m \in P(G, d_m)$, then $(a \to d_1 \ldots d_m)(p_1, \ldots, p_m) \in P(G, a)$.

4° Nothing is in any $P(G, d)$ unless this follows from a finite number of applications of 1°, 2° and 3°.

The *production forest* of G is the $\Sigma^P X$-forest $P(G) = P(G, a_0)$.

In our previous discussion of DR-recognizers we excluded nullary symbols, but since the ranked alphabets Σ^P may contain such symbols, we now extend the definition of a DR ΣX-recognizer $\mathbf{A} = (\mathcal{A}, a_0, A')$ by setting $\sigma^{\mathcal{A}} \in A$ and $\sigma\tilde{\alpha} = \{\sigma^{\mathcal{A}}\}$ for any $\sigma \in \Sigma_0$.

Theorem 3.2.9 *The production forest $P(G)$ of any CF grammar G is local and it is also recognizable by a deterministic R-recognizer.*

Proof. Let $G = (N, X, P, a_0)$ be a CF grammar. The presentation of $P(G)$ as a local forest is similar to that of $D(G)$. We construct a DR $\Sigma^P X$-recognizer $\mathbf{A} = (A, \Sigma^P, X, A', \alpha)$ as follows. Put $A = N \cup X \cup \{d\}$ ($d \notin N \cup X$), $A' = \{a_0\}$, and for each $x \in X$, $x\alpha = \{x\}$. Next, the underlying root-to-frontier algebra $\mathcal{A} = (A, \Sigma^P)$ is defined. If $\sigma = (a \to e) \in \Sigma_0^P$, then $\sigma^{\mathcal{A}} = a$. Let $\sigma = (a \to c_1 \ldots c_m) \in \Sigma_m^P$ with $m > 0$. Then we put $\sigma^{\mathcal{A}}(a) = (c_1, \ldots, c_m)$, and $\sigma^{\mathcal{A}}(b) = (d, \ldots, d)$ for all $b \neq a$. It is easy to show by tree induction that for all $t \in F_{\Sigma^P}(X)$ and $a \in N \cup X$,

$$a \in t\tilde{\alpha} \quad \text{iff} \quad t \in P(G, a).$$

This implies that **A** recognizes $P(G)$. □

The language recognized by an R-recognizer is defined in the natural way. As it is obvious that $\mathrm{yd}(P(G)) = L(G)$ for every CF grammar G, we may state

Corollary 3.2.10 *Every CF language is recognized by a deterministic R-recognizer.* □

3.3 FURTHER RESULTS AND APPLICATIONS

Every CF language L is the yield of many different forests. Such a forest is not necessarily recognizable. In particular, the greatest of them (for a given Σ) $\mathrm{yd}^{-1}(L)$ may be nonrecognizable.

Example 3.3.1 Let $\Sigma = \Sigma_2 = \{\sigma\}$ and $X = \{x, y\}$. Consider the (minimal linear) CF language $L = \{x^n y^n \mid n \geq 1\}$. If $\mathrm{yd}^{-1}(L)$ were recognized by a ΣX-recognizer **A**, then **A** would accept all trees $\sigma(s_i, t_i)$ $(i \geq 1)$, where (i) $s_1 = x$, $t_1 = y$ and (ii) $s_{i+1} = \sigma(s_k, x)$ and $t_{k+1} = \sigma(y, t_k)$ for all $k \geq 1$. As **A** is finite, it would then also accept some tree $\sigma(s_i, t_j)$ with $i \neq j$. But this is a contradiction, because $\mathrm{yd}(\sigma(s_i, t_j)) = x^i y^j \notin L$. □

In contrast to Example 3.3.1 we have

Theorem 3.3.2 *If L is a regular X-language, then* $\mathrm{yd}^{-1}(L) \in \mathrm{Rec}(\Sigma, X)$ *for any ranked alphabet* Σ.

Proof. Let M be a finite monoid, $\varphi : X^* \to M$ a homomorphism and H a subset of M such that $L = H\varphi^{-1}$. Let $\mathcal{A} = (M, \Sigma)$ be the Σ-algebra defined so that

$$\sigma^{\mathcal{A}}(a_1, \ldots, a_m) = a_1 \cdot a_2 \cdot \ldots \cdot a_m \ \ \text{(product in } M\text{)}$$

for all $m \geq 0$, $\sigma \in \Sigma_m$ and $a_1, \ldots, a_m \in M$. In particular, $\sigma^{\mathcal{A}} = 1$ when $\sigma \in \Sigma_0$. If we put

$$\alpha = \varphi | X : X \to M,$$

then

$$t\hat{\alpha} = \mathrm{yd}(t)\varphi \ \ \text{for all } t \in F_\Sigma(X).$$

This implies that $\mathrm{yd}^{-1}(L) = T(\mathbf{A})$ for the ΣX-recognizer $\mathbf{A} = (\mathcal{A}, \alpha, H)$. Indeed, for all $t \in F_\Sigma(X)$,

$$\begin{aligned} t \in T(\mathbf{A}) \quad & \text{iff} \quad t\hat{\alpha} = \mathrm{yd}(t)\varphi \in H \\ & \text{iff} \quad \mathrm{yd}(t) \in L \\ & \text{iff} \quad t \in \mathrm{yd}^{-1}(L). \end{aligned}$$

□

The full converse of Theorem 3.3.2 is not valid, but the following result will be proven in Exercises 6 and 7.

Theorem 3.3.3 *Let L $(\subseteq X^*)$ be a language and Σ a ranked alphabet such that* $\mathrm{yd}(\mathrm{yd}^{-1}(L)) = L$. *Then* $\mathrm{yd}^{-1}(L) \in \mathrm{Rec}(\Sigma, X)$ *implies* $L \in \mathrm{Rec}X$. □

The ranked alphabets Σ^G and Σ^P depend on the given CF grammar. We shall now show that every CF language is the yield of a recognizable forest over a fixed ranked alphabet. In fact, a very simple alphabet will suffice.

Theorem 3.3.4 *Let Σ be a ranked alphabet which contains a binary operator and a nullary operator. Then every* CF *language is recognized by a Σ-recognizer. For e-free* CF *languages the binary symbol alone is sufficient.*

Proof. Let us consider the e-free case first. Every CF language $L \subseteq X^+$ is generated by a CF grammar $G = (N, X, P, a_0)$ in Chomsky normal form, where each production is of the form $a \to bc$ or $a \to x$ $(a, b, c \in N,\ x \in X)$. By Lemma 2.4.1 we may assume that $\Sigma = \Sigma_2 = \{\sigma\}$. Let $G_1 = (N, \Sigma, X, P_1, a_0)$ be the regular ΣX-grammar, where

$$P_1 = \{a \to \sigma(b, c) \mid a \to bc \in P\} \cup \{a \to x \mid a \to x \in P\}.$$

Adjoin N to the frontier alphabet and let

$$\mathrm{yd}' \colon F_\Sigma(X \cup N) \to (X \cup N)^*$$

be the corresponding yield-function. By induction on the length of the derivation one can verify that for every derivation

$$a \Rightarrow_G u_1 \Rightarrow_G \ldots \Rightarrow_G u_k \quad (a \in N, k \geq 1)$$

there is a derivation

$$a \Rightarrow_{G_1} p_1 \Rightarrow_{G_1} \ldots \Rightarrow_{G_1} p_k \quad (p_1, \ldots, p_k \in F_\Sigma(X \cup N)) \tag{*}$$

such that $\mathrm{yd}'(p_i) = u_i$ for $i = 1, \ldots, k$. This implies $L(G) \subseteq \mathrm{yd}(T(G_1))$ as $\mathrm{yd}'|F_\Sigma(X) = \mathrm{yd}$. The converse inclusion follows from the fact that for every derivation (*) we have a derivation

$$a \Rightarrow_G \mathrm{yd}'(p_1) \Rightarrow_G \ldots \Rightarrow_G \mathrm{yd}'(p_k).$$

If $L \subseteq X^*$ and $e \in L$, then we find, as above, a recognizable ΣX-forest T such that $\mathrm{yd}(T) = L - \{e\}$. Now add a nullary operator ω to Σ and let $T' = T \cup \omega$. Then T' is recognizable and $\mathrm{yd}(T') = L$. □

The connections established above suggest the possibility of developing, or just interpreting, the theory of context-free languages in terms of tree automata and recognizable forests. We shall illustrate this by a few examples. The results themselves are well known.

Theorem 3.3.5 *The intersection of a context-free language with a regular language is context-free.*

Proof. Consider a CF language $L \subseteq X^*$ and a regular language U over the same alphabet. Choose any ranked alphabet Σ and a recognizable ΣX-forest R such that $\mathrm{yd}(R) = L$. Then

$$L \cap U = \mathrm{yd}(R \cap \mathrm{yd}^{-1}(U)).$$

Since $R \cap \mathrm{yd}^{-1}(U) \in \mathrm{Rec}(\Sigma, X)$ by Theorem 3.3.2 and Theorem 2.4.2, this means that $L \cap U$ is context-free. □

The next example shows how the regular forest operations relate to language operations.

Definition 3.3.6 Let U and V be X-languages and $x \in X$. The *x-substitution* of U into V is the language $U \cdot_x V$ of all words

$$w_0 u_1 w_1 u_2 \ldots w_{k-1} u_k w_k,$$

where $k \geq 0$, $u_1, \ldots, u_k \in U$, $w_0 x w_1 x \ldots x w_{k-1} x w_k \in V$ and x does not appear in the word $w_0 w_1 \ldots w_k$.

The *x-substitution closure* of U is the language

$$U^{*x} = \bigcup (U^{i,x} \mid i \geq 0),$$

where $U^{0,x} = \{x\}$ and $U^{i,x} = U^{i-1,x} \cdot_x U \cup U^{i-1,x}$ for $i > 0$.

Consider two ΣX-forests S and T and a symbol $x \in X$. Every tree $p \in S \cdot_x T$ is obtained from some tree $t \in T$ by replacing each occurrence of x by some tree from S. Suppose x appears k times ($k \geq 0$) in t and that we get p by replacing these occurrences, from left to right, by the trees $s_1, \ldots, s_k \in S$. If

$$\mathrm{yd}(t) = w_0 x w_1 x \ldots x w_k,$$

then

$$\mathrm{yd}(p) = w_0 \mathrm{yd}(s_1) w_1 \mathrm{yd}(s_2) \ldots \mathrm{yd}(s_k) w_k \in \mathrm{yd}(S) \cdot_x \mathrm{yd}(T).$$

Conversely, if $w \in \mathrm{yd}(S) \cdot_x \mathrm{yd}(T)$, then we may write w in the form

$$w = w_0 u_1 w_1 u_2 \ldots w_{k-1} u_k w_k$$

so that $k \geq 0$, $w_0 x w_1 x \ldots x w_k \in \mathrm{yd}(T)$ and $u_1, \ldots, u_k \in \mathrm{yd}(S)$. Then there are trees $t \in T$ and $s_1, \ldots, s_k \in S$ such that $\mathrm{yd}(t) = w_0 x w_1 x \ldots x w_k$ and $\mathrm{yd}(s_1) = u_1, \ldots,$ $\mathrm{yd}(s_k) = u_k$. If we replace the occurrences of x in t by the trees $s_1, \ldots, s_k$, then we get a tree $p \in S \cdot_x T$ such that $\mathrm{yd}(p) = w$. An easy induction on i shows now that

$$\mathrm{yd}(T^{i,x}) = \mathrm{yd}(T)^{i,x} \quad \text{for all } i \geq 0.$$

Using these observations we get

Lemma 3.3.7 *For any two ΣX-forests S and T, and any letter $x \in X$,*

(a) $\mathrm{yd}(S \cdot_x T) = \mathrm{yd}(S) \cdot_x \mathrm{yd}(T)$

and

(b) $\mathrm{yd}(T^{*x}) = \mathrm{yd}(T)^{*x}$. □

Now we can derive the following well-known description of the family of context-free languages.

Theorem 3.3.8 *The context-free languages form the smallest family of languages which contains the finite languages and is closed under (finite) union, x-substitutions and x-substitution closures.*

Proof. Clearly, all finite languages are context-free. Let $U, V \subseteq X^*$ be CF and $x \in X$. There exist recognizable forests $S, T \subseteq F_\Sigma(X)$ such that $\mathrm{yd}(S) = U$, $\mathrm{yd}(T) = V$. Now $U \cup V = \mathrm{yd}(S \cup T)$, $U \cdot_x V = \mathrm{yd}(S \cdot_x T)$ and $V^{*x} = \mathrm{yd}(T^{*x})$ are all seen to be context-free. On the other hand, the Kleene theorem (Theorem 2.5.8) together with Corollary 3.2.4 and Lemma 3.3.7 shows that every CF language can be obtained from finite languages by forming unions, x-substitutions and x-substitution closures. □

Note that when a CF X-language is expressed in terms of finite languages, unions, substitutions and substitution closures, symbols not in X may be used as auxiliary symbols in substitutions.

As an example we consider the language $L = \{x^n y^n \mid n \geq 0\}$. Let $\omega \in \Sigma_0$ and $\sigma \in \Sigma_3$. Then L is the yield of, for example, the recognizable ΣX-forest

$$T = \{\omega, \sigma(x, \omega, y), \sigma(x, \sigma(x, \omega, y), y), \ldots\}$$

which has the regular expression $\omega \cdot_z \sigma(x, z, y)^{*z}$. From this we get for L the representation

$$L = \{e\} \cdot_z \{xzy\}^{*z}.$$

Here z is an auxiliary letter which does not appear in the language represented.

3.4 ANOTHER WAY TO RECOGNIZE CF LANGUAGES

If an ordinary finite automaton is viewed as a unary algebra, then its input symbols form a ranked alphabet. There is a way to interpret ΣX-trees as words over Σ in the general case, too. When this is done, recognizable forests become CF languages. Moreover, every CF language can be obtained this way as a recognizable forest once its alphabet is suitably ranked.

We consider the unary case as an introduction. The word

$$t\eta = \sigma_1 \ldots \sigma_k \in \Sigma^*$$

can be obtained from the corresponding $\Sigma\{x\}$-tree

$$t = \sigma_k(\ldots \sigma_1(x) \ldots)$$

recursively as follows:

1° $x\eta = e$ for all $x \in X$.

2° $t\eta = s\eta\sigma$ if $t = \sigma(s)$ $(\sigma \in \Sigma)$.

Another way to get $t\eta$ would be to erase the parentheses and x and then reverse the resulting word. Both of these constructions can serve as a basis for the generalization to the case of an arbitrary ranked alphabet. The reversing of the order of the word is an inessential step due to our way of writing trees, and it will be omitted in the generalization.

Let Σ be an arbitrary ranked alphabet and X any frontier alphabet. We shall treat Σ as an ordinary alphabet, too. We assume that Σ and X are disjoint and that they do not contain (,) or the comma. Let

$$Y = \Sigma \cup X \cup \{(,),,\}$$

and define

$$\eta : Y^* \to \Sigma^*$$

as the monoid homomorphism such that

$$y\eta = \begin{cases} y \text{ for } y \in \Sigma, \\ e \text{ for } y \in Y - \Sigma. \end{cases}$$

Applied to a ΣX-tree t η erases all frontier letters $x \in X$, the parentheses and the commas leaving the symbols $\sigma \in \Sigma$ intact. It is easy to see that this can be carried out as follows, too.

Lemma 3.4.1 *The words $t\eta$ $(t \in F_\Sigma(X))$ can be found recursively as follows:*

1° *$x\eta = e$ for all $x \in X$.*

2° *If $t = \sigma(t_1, \ldots, t_m)$ $(m \geq 0,\ \sigma \in \Sigma_m)$, then $t\eta = \sigma t_1\eta \ldots t_m\eta$.* □

We have already noted that every regular ΣX-grammar may also be viewed as a CF grammar generating a Y-language. Moreover, it is well-known that the family of context-free languages is closed under homomorphisms. Hence we have

Lemma 3.4.2 *If $T \in \mathrm{Rec}(\Sigma, X)$, then $T\eta \in \mathrm{CF}(\Sigma)$.* □

Next we prove the following converse of Lemma 3.4.2.

Lemma 3.4.3 *Let Σ and X be alphabets. If Σ is ranked so that $\Sigma_2 = \Sigma$, then there exists for each* CF *language $L \subseteq \Sigma^*$ a recognizable ΣX-forest T such that $T\eta = L$.*

Proof. First, let L be e-free. Then L is generated by a CF grammar $G = (N, \Sigma, P, a_0)$ in Greibach 2-form, where each production is of the form (i) $a \to \sigma bc$, (ii) $a \to \sigma b$ or (iii) $a \to \sigma$ ($a, b, c \in N$, $\sigma \in \Sigma$). We convert G into a regular ΣX-grammar $G_1 = (N, \Sigma, X, P_1, a_0)$, where the set P_1 of productions is defined as follows. Fix any $x \in X$ and put then

$$\begin{aligned} P_1 = \ & \{a \to \sigma(b, c) \mid a \to \sigma bc \in P\} \cup \{a \to \sigma(b, x) \mid a \to \sigma b \in P\} \cup \\ & \{a \to \sigma(x, x) \mid a \to \sigma \in P\}. \end{aligned}$$

In order to show that $T(G_1)$ is the required recognizable forest we extend η to a homomorphism

$$\eta_1 | (Y \cup N)^* \to (\Sigma \cup N)^*$$

so that $\eta_1 | Y = \eta$ and $\eta_1 | N = 1_N$. It is easy to see that to every derivation

$$a \Rightarrow_G u_1 \Rightarrow_G \ldots \Rightarrow_G u_k \quad (a \in N, k \geq 1)$$

there corresponds a derivation

$$a \Rightarrow_{G_1} v_1 \Rightarrow_{G_1} \ldots \Rightarrow_{G_1} v_k \tag{*}$$

such that $v_i\eta_1 = u_i$ ($i = 1, \ldots, k$). Conversely, every derivation (*) is matched by the derivation

$$a \Rightarrow_G v_1\eta_1 \Rightarrow_G \ldots \Rightarrow_G v_k\eta_1.$$

Since $\eta_1 | Y^* = \eta$, this implies $T(G_1)\eta = L(G) = L$. If $e \in L$, we apply this construction to $L - e$ and add then the tree x to $T(G_1)$. □

In the representation of Lemma 3.4.3 the frontier alphabet X can be fixed in advance independently of Σ and the language L. A one-element alphabet $X = \{x\}$ always suffices.

We say that a ΣX-recognizer $\mathbf{A}$ *η-accepts* a word $w \in \Sigma^*$, if it accepts at least one ΣX-tree t such that $t\eta = w$. The Σ-language $\eta(\mathbf{A})$ *η-recognized* by $\mathbf{A}$ is the set of all words η-accepted by $\mathbf{A}$. In this terminology the previous results may be summed up as follows.

Theorem 3.4.4 *A language is η-recognized by some tree recognizer iff it is a context-free language.* □

3.5 EXERCISES

1. Is is possible that $\mathrm{yd}^{-1}(w)$ is infinite for some word w?

2. Prove Lemma 3.1.3.

3. Find an example of a nonrecognizable forest T such that $\mathrm{yd}(T)$ is a recognizable language.

4. Show that for every CF grammar G, $D(G)$ is the image of $P(G)$ under an alphabetic tree homomorphism.

5. Recall that a *groupoid* is an algebra with one binary operation (and no other operations). For $\Sigma = \Sigma_2 = \{\sigma\}$, $F_\Sigma(X)$ is the free groupoid generated by X. Verify that $\mathrm{yd}\colon F_\Sigma(X) \to X^+$ is a groupoid epimorphism. Then prove that a language $L \subseteq X^+$ is context-free iff it is the homomorphic image of a recognizable subset of the free groupoid generated by X (cf. Exercise 2.22, and MEZEI and WRIGHT [182]).

6. The set $\mathrm{Comb}(\Sigma, X)$ of "comb-like" ΣX-trees is defined as the smallest set S satisfying the conditions 1° and 2°:

   1° $X \cup \Sigma_0 \subseteq S$.

   2° If $m > 0$, $\sigma \in \Sigma_m$, $x_1, \ldots, x_{m-1} \in X$ and $t \in S$, then $\sigma(x_1, \ldots, x_{m-1}, t) \in S$.

   (a) Prove that $\mathrm{Comb}(\Sigma, X) \in \mathrm{Rec}(\Sigma, X)$.

   (b) Let T be a recognizable forest such that $T \subseteq \mathrm{Comb}(\Sigma, X)$. Show that T is generated by a regular ΣX-grammar (N, Σ, X, P, a_0) in which each production has the form $a \to \sigma(x_1, \ldots, x_{m-1}, b)$, $a \to \omega$ or $a \to x$ ($a, b \in N$, $m > 0$, $\sigma \in \Sigma_m$, $x_1, \ldots, x_{m-1} \in X$, $\omega \in \Sigma_0$, $x \in X$).

   (c) Infer from (b) that $\mathrm{yd}(T) \in \mathrm{Rec}X$ for every recognizable $T \subseteq \mathrm{Comb}(\Sigma, X)$.

   (d) Prove that for every ΣX-tree t there exists a comb-like ΣX-tree s such that $\mathrm{yd}(s) = \mathrm{yd}(t)$. Deduce from this fact that if $\mathrm{yd}(\mathrm{yd}^{-1}(L)) = L$ for some $L \subseteq X^*$, then

$$\mathrm{yd}(\mathrm{yd}^{-1}(L) \cap \mathrm{Comb}(\Sigma, X)) = L.$$

7. Prove Theorem 3.3.3 using the results of the previous exercise.

8. Give another proof for Theorem 3.3.4 using the fact that every CF language can be generated by an invertible CF grammar in Chomsky normal form.

In Exercises 9–12 the theory of recognizable forests should be applied.

9. Prove that the languge $U - V$ is CF if U is CF and V is a regular language.

10. Let $\varphi\colon X^* \to Y^*$ be a homomorphism of monoids. Prove that $L\varphi^{-1} \in \mathrm{CF}(X)$ for every $L \in \mathrm{CF}(Y)$.

11. Let $h(t)$ denote the tree which is obtained from a given tree by rewriting every operator σ as its rank $r(\sigma)$. Obviously $\mathrm{yd}(h(t)) = \mathrm{yd}(t)$. Show that h can be defined, for any given Σ and X, as an alphabetic tree homomorphism. Two CF grammars G_1 and G_2 are said to be *structurally equivalent* if $h(D(G_1)) = h(D(G_2))$. Prove that there is an algorithm to determine whether or not two CF grammars are structurally equivalent.

12. Prove Bar-Hillel's pumping lemma (Lemma 1.6.13).

13. Let G be a regular ΣX-grammar. Construct a CF grammar G' such that $L(G') = T(G)\eta$. Note that Lemma 3.4.2 follows as a result.

3.6 NOTES AND REFERENCES

The basic connection between recognizable forests and context-free languages has been established in various ways. MEZEI and WRIGHT [182] proved that the equational subsets of an algebra of finite type (in the monoid X^* these are the CF languages) are the homomorphic images of the recognizable subsets of term algebras, i.e., recognizable forests. Applied to groupoids this theorem gives the result of Exercise 5 (credited to D. Muller). It also implies Theorem 3.3.4 which was explicitly formulated by MAGIDOR and MORAN [166]. The proof using derivation forests goes back to THATCHER [237, 238] and DONER [66]. Various forms of production trees have been used in this context by ENGELFRIET [74], and STEINBY [224]. Theorem 3.3.2 appears, for example, in ROUNDS [215]. It is a special instance of the fact that the inverse homomorphic images of recognizable subsets of algebras are recognizable (cf. Exercise 2.22). Theorem 3.3.3 appears to be well-known. The proof outlined in Exercises 6 and 7 is from STEYART [229]. The idea to use tree automata in the theory of CF languages was proposed by ROUNDS [214]. More examples of such applications can be found in THATCHER [239] and ENGELFRIET [74]. The results of Section 3.4 are due to FERENCI [93]. The interested reader may consult FERENCI [94] for further work in this direction.

As a conclusion we mention a few other topics. Using a ranked nonterminal alphabet it is possible to define *context-free tree grammars.* ROUNDS [213, 214, 215] shows that the yield-languages of CF forests are exactly the indexed languages. ARNOLD and DAUCHET [15, 17, 18], and ENGELFRIET and SCHMIDT [85] are some further references.

Possibilities to extend some of the results of this chapter to type 0 or context-sensitive languages by generalizing the tree-concept have been investigated by BENSON [32], BUTTELMAN [48, 49], HART [118, 119], and RÉVÉSZ [208]. Hierarchies of term languages obtained by iteration of the yield-forming process have been studied by MAIBAUM [170], ENGELFRIET and SCHMIDT [85], and TURNER [245, 246]. Families of languages defined by tree recognizers based on algebras belonging to a given variety of algebras were considered by STEINBY [224]. GÉCSEG and HORVÁTH [103] showed that a proper variety may be *complete* in the sense that every CF language is recognizable by a finite algebra of the variety (cf. the Notes and references section of Chapter 2).

4 TREE TRANSDUCERS AND TREE TRANSFORMATIONS

In this chapter we shall deal with systems transforming trees into trees similarly as generalized sequential machines transform strings into strings. There are two main categories of such systems: frontier-to-root tree transducers which process a tree from the leaves down towards the root, and root-to-frontier tree transducers which work in the opposite direction. Special classes of tree transducers will play a basic part in decomposing tree transformations into simpler ones.

4.1 BASIC CONCEPTS

Throughout this chapter, Σ, Ω and Δ will stand for ranked alphabets. It will be assumed that whenever an operator belongs to more than one ranked alphabet, then it has the same rank in all of them. Moreover, X, Y and Z will always stand for (finite, nonvoid) frontier alphabets.

Let us recall that $F_\Sigma(S)$ as defined in Section 2.1 denotes the set of Σ-trees over the frontier alphabet S. Here we shall allow S to be a possibly infinite set of trees and then use the notation $F_\Sigma[S]$ for $F_\Sigma(S)$. One can easily see that in such a case there always exist a ranked alphabet Ω and a frontier alphabet Y such that $F_\Sigma[S] \subseteq F_\Sigma(Y)$.

Binary relations $\tau \subseteq F_\Sigma(X) \times F_\Omega(Y)$ will be called *tree transformations.* An inclusion $(p, q) \in \tau$ is interpreted to mean that τ may transform p into q. Because tree transformations are binary relations, we can speak about *compositions, inverses, domains* and *ranges* of tree transformations as defined in Section 1.1.

With each tree transformation $\tau \subseteq F_\Sigma(X) \times F_\Omega(Y)$ we associate the *translation* $\{(\mathrm{yd}(p), \mathrm{yd}(q)) \mid (p, q) \in \tau\}$ from X^* into Y^*.

The important tree transformations are those which can be given in an effective way. Next we define two general systems (tree transducers) inducing such transformations. We shall need a countably infinite set

$$\Xi = \{\xi_1, \xi_2, \ldots\}$$

of auxiliary variables. The subset of Ξ consisting of its first $n \geq 0$ elements will be denoted by Ξ_n, i.e., $\Xi_n = \{\xi_1, \ldots, \xi_n\}$. The role of an auxiliary variable is to indicate an occurrence of a subtree in a tree.

If all variables occurring in a tree q are among $\xi_1, \ldots, \xi_n$, then the notation $q(\xi_1, \ldots, \xi_n)$ may be also used for q. Moreover, if $q_1, \ldots, q_n$ are arbitrary trees, then we generally write $q(q_1, \ldots, q_n)$ for $q(\xi_1 \leftarrow q_1, \ldots, \xi_n \leftarrow q_n)$.

Definition 4.1.1 A *frontier-to-root tree transducer* (F-*transducer*) is a system $\mathfrak{A} = (\Sigma, X, A, \Omega, Y, P, A')$ where

(1) Σ and Ω are ranked alphabets,

(2) X and Y are frontier alphabets,

(3) A is a ranked alphabet consisting of unary operators, the *state set* of $\mathfrak{A}$,

(It will be assumed that A is disjoint with all other sets in the definition of $\mathfrak{A}$, except A'.)

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

(5) P is a finite set of *productions* (or *rewriting rules*) of the following two types:

(i) $x \to a(q)$ ($x \in X$, $a \in A$, $q \in F_\Omega(Y)$),

(ii) $\sigma(a_1(\xi_1), \ldots, a_m(\xi_m)) \to a(q(\xi_1, \ldots, \xi_m))$ ($\sigma \in \Sigma_m$, $m \geq 0$, $a_1, \ldots, a_m, a \in A$, $q(\xi_1, \ldots, \xi_m) \in F_\Omega(Y \cup \Xi_m)$).

(In the sequel we shall write simply $\sigma(a_1, \ldots, a_m)$ for $\sigma(a_1(\xi_1), \ldots, a_m(\xi_m))$.)

We shall use also the notation (p, q) for a production $p \to q$. Moreover, if $a \in A$ is a state and t is a tree, then we generally write at for $a(t)$. Similarly, if T is a forest, then AT will denote the forest $\{at \mid a \in A, t \in T\}$. Furthermore, for any $a \in A$, we put $\mathfrak{A}(a) = (\Sigma, X, A, \Omega, Y, P, a)$.

Let us note that in the above definition it would be more exact to speak about production schemes instead of productions. Indeed, soon we shall see that they define patterns for rewriting trees.

Next we define the transformations induced by F-transducers. Consider the F-transducer $\mathfrak{A}$ of Definition 4.1.1 and, for every $p \in F_\Sigma[X \cup A\Xi]$, let $p\tau_\mathfrak{A}^*$ be the subset of $AF_\Omega(Y \cup \Xi)$ given as follows:

(1) if $p = a\xi$ ($a \in A$, $\xi \in \Xi$), then $a\xi \in p\tau_\mathfrak{A}^*$,

(2) if $p \in X \cup \Sigma_0$, then $aq \in p\tau_\mathfrak{A}^*$ for all $(p, aq) \in P$,

(3) if $p = \sigma(p_1, \ldots, p_m)$ ($\sigma \in \Sigma_m$, $m > 0$), then $aq(q_1, \ldots, q_m) \in p\tau_\mathfrak{A}^*$ for all $(\sigma(a_1, \ldots, a_m), aq) \in P$ and $a_i q_i \in p_i \tau_\mathfrak{A}^*$ ($a, a_i \in A$, $i = 1, \ldots, m$), and

(4) nothing is in any $p\tau_\mathfrak{A}^*$ unless this follows from (1)–(3).

Definition 4.1.2 Take an F-transducer $\mathfrak{A} = (\Sigma, X, A, \Omega, Y, P, A')$. Then the relation

$$\tau_\mathfrak{A} = \{(p, q) \mid p \in F_\Sigma(X), q \in F_\Omega(Y), aq \in p\tau_\mathfrak{A}^* \text{ for some } a \in A'\}$$

is called the *transformation* induced by $\mathfrak{A}$.

For Definition 4.1.2 it would be enough to apply $\tau_{\mathfrak{A}}^*$ to trees from $F_\Sigma(X)$. The above more general case will be needed later.

Sometimes in our proofs we should know how an input tree is transformed step by step into an output tree. Again, let $\mathfrak{A}$ be the F-transducer of Definition 4.1.1, and consider two trees $p, q \in F_\Sigma[X \cup AF_\Omega(Y \cup \Xi)]$. It is said that p *directly derives* q in $\mathfrak{A}$ if q can be obtained from p by

(i) replacing an occurrence of an $x \in X$ in p by the right side $a\overline{q}$ of a production $x \to a\overline{q}$ from P, or by

(ii) replacing an occurrence of a subtree $\sigma(a_1q_1, \ldots, a_mq_m)$ ($\sigma \in \Sigma_m$, $a_1, \ldots, a_m \in A$, $q_1, \ldots, q_m \in F_\Omega(Y \cup \Xi)$) in p by $a\overline{q}(q_1, \ldots, q_m)$, where $\sigma(a_1, \ldots, a_m) \to a\overline{q}$ is a production from P.

Each application of rule (i) or rule (ii) is called a *direct derivation* in $\mathfrak{A}$. If q is obtained from p by a direct derivation in $\mathfrak{A}$ (i.e., p directly derives q in $\mathfrak{A}$), then we write $p \Rightarrow_{\mathfrak{A}} q$. Therefore, $\Rightarrow_{\mathfrak{A}}$ is a binary relation in $F_\Sigma[X \cup AF_\Omega(Y \cup \Xi)]$. If there is no danger of confusion, we generally omit $\mathfrak{A}$ in $\Rightarrow_{\mathfrak{A}}$.

By finitely many consecutive applications of direct derivations we get derivations. Accordingly, for any two trees $p, q \in F_\Sigma[X \cup AF_\Omega(Y \cup \Xi)]$ we say that

$$p = p_0 \Rightarrow p_1 \Rightarrow \ldots \Rightarrow p_i \Rightarrow \ldots \Rightarrow p_j \Rightarrow \ldots \Rightarrow p_k = q \tag{1}$$

$$(k \geq 0, p_\ell \in F_\Sigma[X \cup AF_\Omega(Y \cup \Xi)],\ \ell = 1, \ldots, k,\ 0 \leq i < j \leq k)$$

is a *derivation* of q from p in $\mathfrak{A}$, k is the *length* of this derivation and $p_i \Rightarrow \ldots \Rightarrow p_j$ is a *subderivation* of (1). In this case we write $p \Rightarrow_{\mathfrak{A}}^* q$, or $p \Rightarrow^* q$ if $\mathfrak{A}$ is understood, and say that p *derives* q in $\mathfrak{A}$. Therefore, $\Rightarrow^*$ is the reflexive-transitive closure of $\Rightarrow$. Obviously, when $p \Rightarrow^* q$, there could be several (but finitely many) derivations of q from p. However, when we write $p \Rightarrow^* q$, we usually have in mind, at least implicitly, a certain well-defined derivation of q from p. Consequently, we may say that $p \Rightarrow^* q$ is a derivation.

Using the notation $\Rightarrow^*$ the transformation $\tau_{\mathfrak{A}}$ induced by an F-transducer $\mathfrak{A} = (\Sigma, X, A, \Omega, Y, P, A')$ can also be given thus:

$$\tau_{\mathfrak{A}} = \{(p, q) \mid p \in F_\Sigma(X), q \in F_\Omega(Y),\ p \Rightarrow^* aq \ \text{ for some } a \in A'\}.$$

As $\mathfrak{A}$ may have different productions with the same left side, there could be more than one $q \in F_\Omega(Y)$ such that $(p, q) \in \tau_{\mathfrak{A}}$ for a given $p \in F_\Sigma(X)$, i.e., $\mathfrak{A}$ is in general nondeterministic. However, at each step of a transformation we have only finitely many choices. Therefore, $p\tau_{\mathfrak{A}}$ is finite for every $p \in F_\Sigma(X)$.

A tree transformation is an F-*transformation* if it can be induced by an F-transducer. The class of all F-transformations will be denoted by $\mathcal{F}$.

Take an arbitrary set A. The ith component of a vector $\mathbf{a} \in A^n$ will be denote by a_i; i.e., $\mathbf{a} = (a_1, \ldots, a_n)$. If $a_1 = \ldots = a_n = a$ then for $\mathbf{a}$ we write a^n. If $\mathbf{a} \in A^n$ and $\mathbf{b} \in B^m$ are arbitrary two vectors, then $(\mathbf{a}, \mathbf{b})$ will stand for $(a_1, \ldots, a_n, b_1, \ldots, b_m)$. Assume that

$k = \min(m, n)$. Then $\mathbf{ab}$ stands for $(a_1b_1, \ldots, a_kb_k)$ or $((a_1, b_1), \ldots, (a_k, b_k))$, depending on the context.

Consider a $p \in F_\Sigma(X \cup \Xi_n)$, and let $\mathbf{p} = (p_1, \ldots, p_n)$ be a vector of trees. Then we shall write $p(\mathbf{p})$ for $p(p_1, \ldots, p_n)$. Moreover, if $\mathbf{p} \in F_\Sigma(X \cup \Xi_n)^m$ and $\mathbf{q} = (q_1, \ldots, q_n)$ is a vector of trees, then $\mathbf{p}(\mathbf{q})$ will stand for $(p_1(\mathbf{q}), \ldots, p_m(\mathbf{q}))$.

Consider the homomorphism $\varphi\colon (X \cup \Xi)^* \to \Xi^*$ given by $x\varphi = e$ $(x \in X)$ and $\xi\varphi = \xi$ $(\xi \in \Xi)$. Set

$$\hat{F}_\Sigma(X \cup \Xi_n) = \{p \in F_\Sigma(X \cup \Xi_n) \mid \mathrm{yd}(p)\varphi \text{ is a permutation of } \xi_1, \ldots, \xi_n\}$$

and

$$\hat{\hat{F}}_\Sigma(X \cup \Xi_n) = \{p \in F_\Sigma(X \cup \Xi_n) \mid \mathrm{yd}(p)\varphi = \xi_1 \ldots \xi_n\}.$$

Moreover, if $m > 0$ then let

$$\begin{aligned} \hat{F}^m_\Sigma(X \cup \Xi_n) \quad = \quad & \{p \in F_\Sigma(X \cup \Xi_n)^m \mid \mathrm{yd}(p_1)\varphi \ldots \mathrm{yd}(p_m)\varphi \text{ is a} \\ & \text{permutation of } \ \xi_1, \ldots, \xi_n\}. \end{aligned}$$

Now let $\mathfrak{A} = (\Sigma, X, A, \Omega, Y, P, A')$ be an F-transducer, and consider a derivation

$$\alpha\colon p \Rightarrow^* q \ \ (p, q \in F_\Sigma[X \cup AF_\Omega(Y)]).$$

Let

$$\begin{gathered} r(p_1, p_2) \Rightarrow r(p_{1_1}, p_2) \Rightarrow \ldots \Rightarrow r(p_{1_k}, p_2) \Rightarrow r(p_{1_k}, p_2') \\ (r \in \hat{F}_\Sigma[X \cup AF_\Omega(Y) \cup \Xi_2]) \end{gathered} \tag{2}$$

be a subderivation of α, where the first k direct derivation steps apply to the subtree p_1, and then the $(k+1)$th step concerns the subtree p_2. Replacing the subderivation (2) in α by

$$r(p_1, p_2) \Rightarrow r(p_1, p_2') \Rightarrow r(p_{1_1}, p_2') \Rightarrow \ldots \Rightarrow r(p_{1_k}, p_2') \tag{3}$$

we obviously get a new derivation

$$\beta\colon p \Rightarrow^* q.$$

The replacement of (2) in α by (3) is called an *inversion* of direct derivations. Finitely many inversions of direct derivations is a *reordering* of direct derivations.

In the sequel we do not distinguish between derivations obtained from each other by reorderings of direct derivations.

Again, consider the above F-transducer $\mathfrak{A}$ and a tree $p \in F_\Sigma(X)$. Then by

$$p = \overline{p}(p_1, \ldots, p_m) \Rightarrow^* \overline{p}(a_1q_1, \ldots, a_mq_m) \Rightarrow^* aq(q_1, \ldots, q_m)$$

$$(\overline{p} \in \hat{F}_\Sigma(X \cup \Xi_m),\ p_i \Rightarrow^* a_iq_i,\ i = 1, \ldots, m,\ \overline{p}(a_1\xi_1, \ldots, a_m\xi_m) \Rightarrow^* aq)$$

we mean the derivation

$$\overline{p}(p_1, \ldots, p_m) \Rightarrow \overline{p}(p_{1_1}, \ldots, p_m) \Rightarrow \ldots \Rightarrow \overline{p}(p_{1_{k_1}}, \ldots, p_m) \Rightarrow \ldots$$

$$\overline{p}(p_{1_{k_1}}, \ldots, p_{m_1}) \Rightarrow \ldots \Rightarrow \overline{p}(p_{1_{k_1}}, \ldots, p_{m_{k_m}}) =$$
$$\overline{p}(a_1q_1, \ldots, a_mq_m) \Rightarrow^* aq(q_1, \ldots, q_m)$$

if $p_i \Rightarrow^* a_iq_i$ is the derivation $p_i \Rightarrow p_{i_1} \Rightarrow \ldots \Rightarrow p_{i_{k_i}} = a_iq_i$ ($a_i \in A$, $q_i \in F_\Omega(Y)$, $i = 1, \ldots, m$), and $\overline{p}(a_1q_1, \ldots, a_mq_m) \Rightarrow^* aq(q_1, \ldots, q_m)$ is obtained by replacing ξ_i in $\overline{p}(a_1\xi_1, \ldots, a_m\xi_m) \Rightarrow^* aq$ by q_i ($i = 1, \ldots, m$).

If we say that we write the derivation

$$\alpha\colon p \Rightarrow^* aq \quad (a \in A, p \in F_\Sigma(X), q \in F_\Omega(Y))$$

in the (more detailed) form

$$\beta\colon p = \overline{p}(p_1, \ldots, p_m) \Rightarrow^* \overline{p}(a_1q_1, \ldots, a_mq_m) \Rightarrow^* a\overline{q}(q_1, \ldots, q_m)$$
$$(\overline{p} \in \hat{F}_\Sigma(X \cup \Xi_m),\ p_i \Rightarrow^* a_iq_i,\ i = 1, \ldots, m,\ \overline{p}(a_1\xi_1, \ldots, a_m\xi_m) \Rightarrow^* a\overline{q})$$

this also generally means that β is a reordering of α. Of course, such a reordering always exists.

In the special case $\overline{p} = \sigma(\xi_1, \ldots, \xi_m)$ ($\sigma \in \Sigma_m$) we write β in the form

$$\beta\colon \sigma(p_1, \ldots, p_m) \Rightarrow^* \sigma(a_1q_1, \ldots, a_mq_m) \Rightarrow a\overline{q}(q_1, \ldots, q_m)$$
$$(p_i \Rightarrow^* a_iq_i,\ i = 1, \ldots, m,\ (\sigma(a_1, \ldots, a_m), a\overline{q}) \in P).$$

We illustrate the concepts of F-transducers and F-transformations by

Example 4.1.3 Let $\mathfrak{A} = (\Sigma, \{x\}, \{a_0, a_1\}, \Omega, \{y\}, P, \{a_0\})$, where $\Sigma = \Sigma_2 = \{\sigma\}$, $\Omega = \Omega_1 = \{\omega\}$ and P consists of the productions $x \to a_1y$ and $\sigma(a_1, a_1) \to a_0\omega(\xi_1)$.

Consider the tree $\sigma(x, x)$. One of the possible derivations

$$\sigma(x, x) \Rightarrow \sigma(a_1y, x) \Rightarrow \sigma(a_1y, a_1y) \Rightarrow a_0\omega(y)$$

is illustrated by Fig. 4.1.

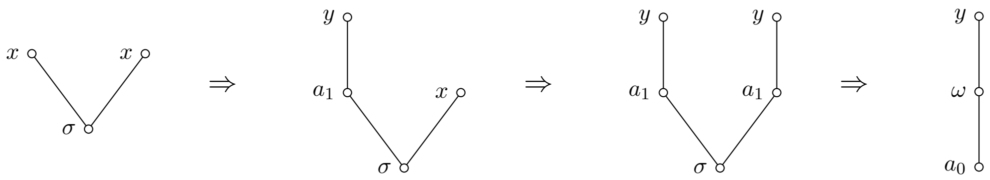

Figure 4.1

Thus $(\sigma(x, x), \omega(y))$ is in $\tau_{\mathfrak{A}}$. In fact, $\tau_{\mathfrak{A}}$ consists of this single pair $(\sigma(x, x), \omega(y))$. Indeed, the only ΣX-tree of height 0 is x, which obviously is not in $\mathrm{dom}(\tau_{\mathfrak{A}})$. If $p \in F_\Sigma(X)$ is a tree with height greater than 1, then it should contain at least one of the following trees as a subtree:

$$\sigma(\sigma(x, x), \sigma(x, x)), \quad \sigma(\sigma(x, x), x) \quad \text{and} \quad \sigma(x, \sigma(x, x)).$$

One can easily see that none of these subtrees can be transformed by $\mathfrak{A}$. $\square$

F-transducers transform a tree from the leaves of the tree towards the root of the tree. Now we define a system which works in the opposite direction.

Definition 4.1.4 A *root-to-frontier tree transducer* (R-*transducer*) is a system $\mathfrak{A} = (\Sigma, X, A, \Omega, Y, P, A')$, where

(1) Σ, X, A, Ω, Y and A' are specified the same way as in Definition 4.1.1, but here A' is called the set of *initial states,*

(2) P is a finite set of *productions* (or *rewriting rules*) of the following two types:
   (i) $ax \to q$ $(a \in A,\ x \in X,\ q \in F_\Omega(Y))$,
   (ii) $a\sigma(\xi_1, \ldots, \xi_m) \to q$ $(a \in A,\ \sigma \in \Sigma_m,\ m \geq 0,\ q \in F_\Omega[Y \cup A\Xi_m])$.

In the sequel we shall write simply $a\sigma$ for $a\sigma(\xi_1, \ldots, \xi_m)$. Moreover, for a production $p \to q$ we shall use the notation (p, q), too.

Obviously, a production of type (ii) in Definition 4.1.4 can be written in the form

$$a\sigma \to q(\mathbf{a}_1\xi_1^{n_1}, \ldots, \mathbf{a}_m\xi_m^{n_m})$$

where $\mathbf{a}_i \in A^{n_i}$, $n_i \geq 0$, $i = 1, \ldots, m$, $n_1 + \ldots + n_m = n$, and $q \in \hat{F}_\Omega(X \cup \Xi_n)$. In the sequel we shall assume that whenever $1 \leq i \leq m$ and $n_1 + \ldots + n_{i-1} + 1 \leq i_i < i_2 \leq n_1 + \ldots + n_i$, ξ_{i_1} precedes ξ_{i_2} in $\mathrm{yd}(q)\varphi$. Here φ is the homorphism defined on p. 134.

Next we define the transformations induced by R-transducers. Let $\mathfrak{A}$ be the R-transducer of Definition 4.1.4. For any $a \in A$ and $p \in F_\Sigma(X)$ we define the subsets $p\tau_{\mathfrak{A},a}$ as follows:

(i) if $p \in \Sigma_0 \cup X$ and $(ap, q) \in P$ then $q \in p\tau_{\mathfrak{A},a}$,

(ii) if $p = \sigma(p_1, \ldots, p_m)$ $(\sigma \in \Sigma_m,\ m > 0)$, then for any $(a\sigma, q(\mathbf{a}_1\xi_1^{n_1}, \ldots, \mathbf{a}_m\xi_m^{n_m})) \in P$ and $q_{i_j} \in p_i\tau_{\mathfrak{A},a_{i_j}}$ $(1 \leq i \leq m,\ 1 \leq j \leq n_i)$, $q(\mathbf{q}_1, \ldots, \mathbf{q}_m) \in p\tau_{\mathfrak{A},a}$ where $\mathbf{q}_i = (q_{i_1}, \ldots, q_{i_{n_i}})$ $(i = 1, \ldots, m)$,

(iii) nothing is in any $p\tau_{\mathfrak{A},a}$ unless this follows from (i) and (ii).

Definition 4.1.5 Let $\mathfrak{A} = (\Sigma, X, A, \Omega, Y, P, A')$ be an R-transducer. Then the *transformation induced by* $\mathfrak{A}$ is the relation

$$\tau_\mathfrak{A} = \{(p, q) \mid p \in F_\Sigma(X), q \in F_\Omega(Y),\ q \in p\tau_{\mathfrak{A},a} \text{ for some } a \in A'\}.$$

A tree transformation is an R-*transformation* if it can be induced by an R-transducer. The class of all R-transformations will be denoted by $\mathcal{R}$.

For R-transformations we also give another definition which shows how a transformation is carried out step by step.

Let $p, q \in F_\Omega[Y \cup AF_\Sigma(X \cup \Xi)]$ be trees, and consider the R-transducer of Definition 4.1.4. It is said that p *directly derives* q in $\mathfrak{A}$ if q can be obtained from p by

(i) replacing an occurrence of a subtree ax ($a \in A$, $x \in X$) in p by the right side $\overline{q}$ of a production $ax \to \overline{q}$ in P, or by

(ii) replacing an occurrence of a subtree $a\sigma(p_1, \ldots, p_m)$ ($a \in A$, $\sigma \in \Sigma_m$, $m \geq 0$, $p_1, \ldots, p_m \in F_\Sigma(X \cup \Xi)$) in p by $\overline{q}(p_1, \ldots, p_m)$ where $a\sigma \to \overline{q}$ is in P.

Each application of steps (i) and (ii) is called a *direct derivation* in $\mathfrak{A}$. The relation expressing the direct derivation will be denoted by $\Rightarrow_{\mathfrak{A}}$, i.e., we write $p \Rightarrow_{\mathfrak{A}} q$ if q is obtained from p by a direct derivation in $\mathfrak{A}$. Frequently, $\mathfrak{A}$ will be omitted in $\Rightarrow_{\mathfrak{A}}$. Any finite sequence of consecutive direct derivations defines a derivation. More precisely,

$$p = p_0 \Rightarrow p_1 \Rightarrow \ldots \Rightarrow p_i \Rightarrow \ldots \Rightarrow p_j \Rightarrow \ldots \Rightarrow p_k = q \tag{4}$$

$$(k \geq 0,\ p_\ell \in F_\Omega[Y \cup AF_\Sigma(X \cup \Xi)],\ \ell = 0, \ldots, k,\ 0 \leq i < j \leq k)$$

is a *derivation* of q from p in $\mathfrak{A}$, k is the *length* of this derivation and $p_i \Rightarrow \ldots \Rightarrow p_j$ is a *subderivation* of (4). If q can be obtained from p by a derivation, then we write $p \Rightarrow^*_{\mathfrak{A}} q$, or simply $p \Rightarrow^* q$ if $\mathfrak{A}$ is understood from the context. Thus, $\Rightarrow^*$ is the reflexive-transitive closure of $\Rightarrow$. Similarly as in the case of an F-transducer, we suppose that the notation $p \Rightarrow^* q$ implies a certain derivation of q from p in $\mathfrak{A}$.

Using the notation $\Rightarrow^*$, the transformation $\tau_{\mathfrak{A}}$ induced by an R-transducer $\mathfrak{A} = (\Sigma, X, A, \Omega, Y, P, A')$ can equivalently be defined thus:

$$\tau_{\mathfrak{A}} = \{(p, q) \mid p \in F_\Sigma(X), q \in F_\Omega(Y),\ ap \Rightarrow^* q \ \text{ for some } a \in A'\}.$$

Let us note that although an R-transducer $\mathfrak{A}$ is generally a nondeterministic system, $p\tau_{\mathfrak{A}}$ is finite for every input tree p of $\mathfrak{A}$.

Let $\mathfrak{A} = (\Sigma, X, A, \Omega, Y, P, A')$ be an R-transducer. Consider some $n > 0$, $\mathbf{a} \in A^n$, $\mathbf{p} \in F_\Sigma(X)^n$, $\mathbf{q} \in F_\Omega(Y)^n$ and derivations $a_i p_i \Rightarrow^* q_i$ ($i = 1, \ldots, n$). Then $\mathbf{ap} \Rightarrow^* \mathbf{q}$ will denote the vector of these derivations. Moreover, we assume that $\mathbf{ap} \Rightarrow^* \mathbf{q}$ implicitly expresses the n derivations $a_i p_i \Rightarrow^* q_i$ ($i = 1, \ldots, n$).

Take the above R-transducer $\mathfrak{A}$ and a derivation

$$\alpha\colon p \Rightarrow^* q \ \ (p, q \in F_\Omega[Y \cup AF_\Sigma(X)]).$$

Let

$$r(p_1, p_2) \Rightarrow r(p_{1_1}, p_2) \Rightarrow \ldots \Rightarrow r(p_{1_k}, p_2) \Rightarrow r(p_{1_k}, p_2') \tag{5}$$

$$(r \in \hat{F}_\Omega[Y \cup AF_\Sigma(X \cup \Xi_2)])$$

be a subderivation of α, where the first k direct derivation steps are carried out in the subtree p_1, and then in the $(k+1)$th step we apply a production in the subtree p_2. Replacing the subderivation (5) in α by

$$r(p_1, p_2) \Rightarrow r(p_1, p_2') \Rightarrow r(p_{1_1}, p_2') \Rightarrow \ldots \Rightarrow r(p_{1_k}, p_2') \tag{6}$$

we get a derivation

$$\beta\colon p \Rightarrow^* q.$$

The replacement of (5) in α by (6) is called an *inversion* of direct derivations. By finitely many applications of inversions we get a *reordering* of direct derivations. We shall not distinguish between derivations in an R-transducer if they are reorderings of each other.

Again, take the above R-transducer $\mathfrak{A}$, a state $a \in A$ and a tree $p \in F_\Sigma(X)$. Then by

$$ap = a\overline{p}(p_1, \ldots, p_m) \Rightarrow^* q(\mathbf{a}_1 p_1^{n_1}, \ldots, \mathbf{a}_m p_m^{n_m}) \Rightarrow^* q(\mathbf{q}_1, \ldots, \mathbf{q}_m)$$
$$(\overline{p} \in \hat{F}_\Sigma(X \cup \Xi_m),\ a\overline{p} \Rightarrow^* q(\mathbf{a}_1 \xi_1^{n_1}, \ldots, \mathbf{a}_m \xi_m^{n_m}),\ \mathbf{a}_i \in A^{n_i},$$
$$n_i \geq 0,\ i = 1, \ldots, m,\ n_1 + \ldots + n_m = n,\ q \in \hat{F}_\Omega(Y \cup \Xi_n),$$
$$\mathbf{a}_j p_j^{n_j} \Rightarrow \mathbf{q}_j,\ j = 1, \ldots, m)$$

we mean the derivation

$$a\overline{p}(p_1, \ldots, p_m) \Rightarrow^* q(\mathbf{a}_1 p_1^{n_1}, \ldots, \mathbf{a}_m p_m^{n_m}) \Rightarrow$$
$$q(p_{1_1(1)}, a_{1_2} p_1, \ldots, a_{1_{n_1}} p_1, \ldots, a_{m_1} p_m, \ldots, a_{m_{n_m}} p_m) \Rightarrow \ldots$$
$$q(p_{1_1(k_1)}, a_{1_2} p_1, \ldots, a_{1_{n_1}} p_1, \ldots, a_{m_1} p_m, \ldots, a_{m_{n_m}} p_m) \Rightarrow \ldots$$
$$q(p_{1_1(k_1)}, \ldots, p_{1_{n_1}(k_{1_{n_1}})}, \ldots, a_{m_1} p_m, \ldots, a_{m_{n_m}} p_m) \Rightarrow \ldots$$
$$q(p_{1_1(k_1)}, \ldots, p_{1_{n_1}(k_{1_{n_1}})}, \ldots, p_{m_1(k_{m_1})}, \ldots, a_{m_{n_m}} p_m) \Rightarrow \ldots$$
$$q(p_{1_1(k_1)}, \ldots, p_{1_{n_1}(k_{1_{n_1}})}, \ldots, p_{m_1(k_{m_1})}, \ldots, p_{m_{n_m}(k_{m_{n_m}})} =$$
$$q(q_{1_1}, \ldots, q_{1_{n_1}}, \ldots, q_{m_1}, \ldots, q_{m_{n_m}}), \quad \text{assuming}$$

that $\mathbf{a}_i p_i^{n_i} \Rightarrow^* \mathbf{q}_i$ $(1 \leq i \leq m)$ has its component derivations

$$a_{i_j} p_i \Rightarrow p_{i_j(1)} \Rightarrow \ldots \Rightarrow p_{i_j(k_{i_j})} = q_{i_j} \quad (q_{i_j} \in F_\Omega(Y),\ j = 1, \ldots, n_i),$$

and $a\overline{p}(p_1, \ldots, p_m) \Rightarrow^* q(\mathbf{a}_1 p_1^{n_1}, \ldots, \mathbf{a}_m p_m^{n_m})$ is obtained by replacing ξ_i $(i = 1, \ldots, m)$ in $a\overline{p} \Rightarrow^* q(\mathbf{a}_1 \xi_1^{n_1}, \ldots, \mathbf{a}_m \xi_m^{n_m})$ by p_i.

When we say that we write the derivation

$$\alpha\colon ap \Rightarrow^* q \quad (a \in A, p \in F_\Sigma(X), q \in F_\Omega(Y))$$

in the more detailed form

$$\beta\colon ap = a\overline{p}(p_1, \ldots, p_m) \Rightarrow^* \overline{q}(\mathbf{a}_1 p_1^{n_1}, \ldots, \mathbf{a}_m p_m^{n_m}) \Rightarrow^* \overline{q}(\mathbf{q}_1, \ldots, \mathbf{q}_m)$$
$$(\overline{p} \in \hat{F}_\Sigma(X \cup \Xi_m),\ a\overline{p} \Rightarrow^* \overline{q}(\mathbf{a}_1 \xi_1^{n_1}, \ldots, \mathbf{a}_m \xi_m^{n_m}),\ \mathbf{a}_i \in A^{n_i},\ n_i \geq 0,$$
$$i = 1, \ldots, m,\ n_1 + \ldots + n_m = n,\ \overline{q} \in \hat{F}_\Omega(Y \cup \Xi_n),\ \mathbf{a}_j p_j^{n_j} \Rightarrow \mathbf{q}_j,\ j = 1, \ldots, m),$$

it generally also means that β is a reordering of α. Obviously, such a reordering always exists.

In case $\overline{p} = \sigma(\xi_1, \ldots, \xi_m)$ $(\sigma \in \Sigma_m)$, we write β in the form

$$\beta\colon a\sigma(p_1, \ldots, p_m) \Rightarrow^* \overline{q}(\mathbf{a}_1 p_1^{n_1}, \ldots, \mathbf{a}_m p_m^{n_m}) \Rightarrow^* \overline{q}(\mathbf{q}_1, \ldots, \mathbf{q}_m)$$
$$((a\sigma, \overline{q}(\mathbf{a}_1 \xi_1^{n_1}, \ldots, \mathbf{a}_m \xi_m^{n_m})) \in P,\ \mathbf{a}_i \in A^{n_i},\ n_i \geq 0,\ i = 1, \ldots, m,\ n_1 + \ldots + n_m = n,$$
$$\overline{q} \in \hat{F}_\Omega(Y \cup \Xi_n),\ \mathbf{a}_j p_j^{n_j} \Rightarrow^* \mathbf{q}_j,\ j = 1, \ldots, m).$$

Example 4.1.6 Let $\mathfrak{A} = (\Sigma, \{x\}, \{a_0, a_1, a_2\}, \Omega, \{y_1, y_2\}, P, a_0)$ be the R-transducer, where $\Sigma = \Sigma_1 = \{\sigma\}$, $\Omega = \Omega_1 \cup \Omega_2$, $\Omega_1 = \{\omega_1\}$ and $\Omega_2 = \{\omega_2\}$ and P consists of the productions

$$a_0\sigma \to \omega_2(a_1\xi_1, a_2\xi_1),$$

$$a_1\sigma \to \omega_1(a_1\xi_1),\ a_2\sigma \to \omega_1(a_2\xi_1),$$

$$a_1x \to y_1,\ a_2x \to y_2.$$

Consider the trees $p = \sigma(\sigma(\sigma(x)))$ and $q = \omega_2(\omega_1(\omega_1(y_1)), \omega_1(\omega_1(y_2)))$. Then a derivation of q from a_0p is illustrated in Fig. 4.2.

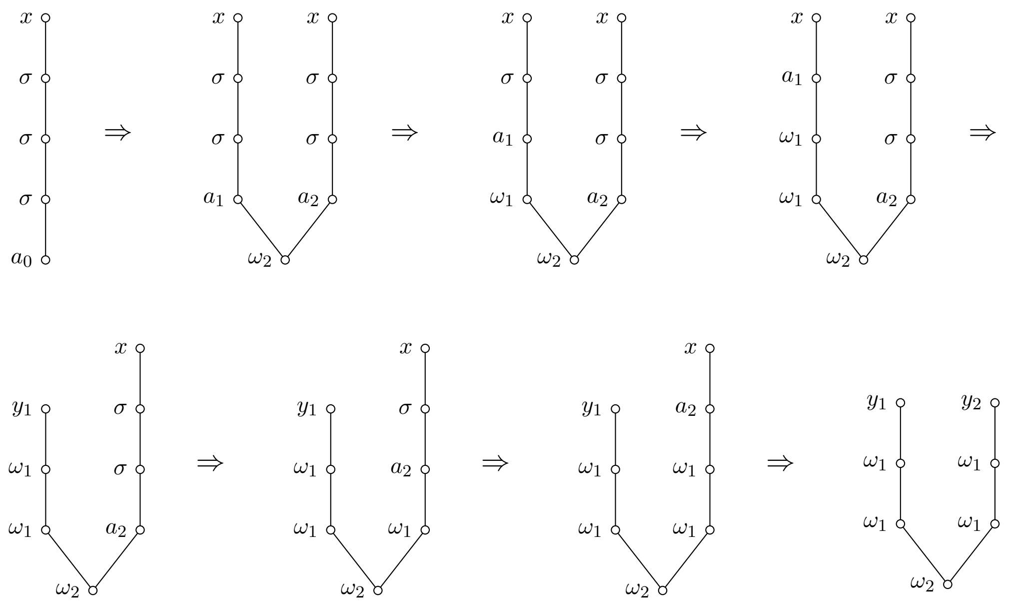

Figure 4.2

By induction on the heights of input trees one can easily prove that

$$\tau_{\mathfrak{A}} = \{(\sigma^n(x), \omega_2(\omega_1^{n-1}(y_1), \omega_1^{n-1}(y_2))) \mid n = 1, 2, \ldots\},$$

where $\sigma^0(\xi) = \xi$ and $\sigma^n(\xi) = \sigma(\sigma^{n-1}(\xi))$ if $n > 0$. $\square$

Both F-transducers and R-transducers generalize generalized sequential machines from strings to trees (or from unary polynomial symbols to polynomial symbols of arbitrary finite type if strings are interpreted as unary polynomial symbols, as we did in Section 2.2). At the same time there are the following main differences between F-transducers and R-transducers:

(1) An F-transducer first processes an input subtree nondeterministically and then makes copies of the resulting output subtree.

(2) An R-transducer can first make copies of an input subtree and then process each copy independently in a nondeterministic fashion.

(3) F-transducers should process even those subtrees which are deleted afterwards.

Before ending this section we state and prove some simple general results.

The concept of tree homomorphism was introduced in Section 2.4. It is easy to see that the tree homomorphism $h\colon F_\Sigma(X) \to F_\Omega(Y)$, given by the mappings

$$h_m\colon \Sigma_m \to F_\Omega(Y \cup \Xi_m) \ \ (m \geq 0)$$

and

$$h_X\colon X \to F_\Omega(Y),$$

can be induced by the one-state F-transducer $\mathfrak{A} = (\Sigma, X, \{a\}, \Omega, Y, P, a)$ where

$$P = \{x \to ah_X(x) \mid x \in X\} \cup \{\sigma(a, \dots, a) \to ah_m(\sigma) \mid \sigma \in \Sigma_m, m \geq 0\}.$$

Definition 4.1.7 A one-state F-transducer $\mathfrak{A} = (\Sigma, X, \{a\}, \Omega, Y, P, a)$ is an HF-*transducer* if for every $x \in X$, resp. $\sigma \in \Sigma$, in P there is exactly one production with left side x, resp. $\sigma(a, \dots, a)$.

We have seen that every tree homomorphism can be induced by an HF-transducer. The converse is also true: transformations induced by HF-transducers are tree homomorphisms.

We now introduce the R-transducer counterpart of HF-transducers.

Definition 4.1.8 A one-state R-transducer $\mathfrak{A} = (\Sigma, X, \{a\}, \Omega, Y, P, a)$ is an HR-*transducer* if for each $d \in X \cup \Sigma$ in P there is exactly one production with the left side ad.

Next we prove that the class of all tree homomorphisms coincides with the class of all transformations induced by HR-transducers.

Theorem 4.1.9 *The class of transformations induced by* HF*-transducers coincides with the class of all transformations induced by* HR*-transducers.*

Proof. Let $\mathfrak{A} = (\Sigma, X, \{a\}, \Omega, Y, P, a)$ be an HF-transducer. Consider the R-transducer $\mathfrak{B} = (\Sigma, X, \{a\}, \Omega, Y, P', a)$, where P' is given in the following way:

$$(ax, q) \in P' \iff (x, aq) \in P \ \ (x \in X)$$

and

$$(a\sigma, q(a\xi_1, \dots, a\xi_m)) \in P' \iff (\sigma(a, \dots, a), aq) \in P \ (\sigma \in \Sigma_m, m \geq 0, q \in F_\Omega(Y \cup \Xi_m)).$$

It is obvious that $\mathfrak{B}$ is an HR-transducer.

By induction on $\mathrm{hg}(p)$, we show that for an arbitrary $p \in F_\Sigma(X)$ and $q \in F_\Omega(Y)$ the equivalence

$$ap \Rightarrow^*_{\mathfrak{B}} q \iff p \Rightarrow^*_{\mathfrak{A}} aq \tag{7}$$

holds. This obviously implies $\tau_{\mathfrak{A}} = \tau_{\mathfrak{B}}$.

If $\mathrm{hg}(p) = 0$, then (7) holds by the definition of P'.

Let $p = \sigma(p_1, \ldots, p_m)$ ($\sigma \in \Sigma_m$, $m > 0$), and assume that (7) has been proved for all trees in $F_\Sigma(X)$ with heights less than $\mathrm{hg}(p)$.

Suppose that the left side of (7) holds, i.e., we have $ap = a\sigma(p_1, \ldots, p_m) \Rightarrow_{\mathfrak{B}} \overline{q}(ap_1, \ldots, ap_m) \Rightarrow^*_{\mathfrak{B}} \overline{q}(q_1, \ldots, q_m) = q$, where $(a\sigma, \overline{q}(a\xi_1, \ldots, a\xi_m)) \in P'$ and $ap_i \Rightarrow^*_{\mathfrak{B}} q_i$ ($i = 1, \ldots, m$). Then, by the definition of P', the production $\sigma(a, \ldots, a) \rightarrow a\overline{q}(\xi_1, \ldots, \xi_m)$ is in P. Moreover, by the induction hypothesis, $p_i \Rightarrow^*_{\mathfrak{A}} aq_i$ is valid for each i ($1 \leq i \leq m$). Therefore, we have a desired derivation

$$p = \sigma(p_1, \ldots, p_m) \Rightarrow^*_{\mathfrak{A}} \sigma(aq_1, \ldots, aq_m) \Rightarrow_{\mathfrak{A}} a\overline{q}(q_1, \ldots, q_m) = aq.$$

The fact that $p \Rightarrow^*_{\mathfrak{A}} aq$ implies $ap \Rightarrow^*_{\mathfrak{B}} q$ can be shown by reversing the above argument.

To see that every HR-transformation is induced by an HF-transducer, it suffices to observe that every HR-transducer $\mathfrak{B}$ arises from an HF-transducer $\mathfrak{A}$ by the above construction. Hence HR- and HF-transducers appear in equivalent "associated" pairs. □

We prove two more results.

Theorem 4.1.10 *The following statements hold.*

(i) *For every* F*-transformation* $\tau \subseteq F_\Sigma(X) \times F_\Omega(Y)$*,* $\mathrm{dom}(\tau) \in \mathrm{Rec}(\Sigma, X)$*.*

(ii) *There exists a tree homomorphism* $h\colon F_\Sigma(X) \rightarrow F_\Omega(Y)$ *such that* $\mathrm{range}(h) \notin \mathrm{Rec}(\Omega, Y)$*.*

Proof. In order to show (i) consider an F-transducer $\mathfrak{A} = (\Sigma, X, A, \Omega, Y, P, A')$. Construct an NDF ΣX-recognizer $\mathbf{B} = (\mathcal{B}, \beta, B')$, where $\mathcal{B} = (A, \Sigma)$, $B' = A'$, and, for all $m \geq 0$, $\sigma \in \Sigma_m$ and $a_1 \ldots, a_m \in A$,

$$\sigma^{\mathcal{B}}(a_1, \ldots, a_m) = \{a \mid (\exists q \in F_\Omega(Y \cup \Xi_m))((\sigma(a_1, \ldots, a_m), aq) \in P)\}.$$

Finally let

$$x\beta = \{a \in A \mid (\exists q \in F_\Omega(Y))((x, aq) \in P)\} \ \ (x \in X).$$

We end the proof of (i) by the observation that for all $a \in A$ and $p \in F_\Sigma(X)$ the equivalence

$$a \in p\hat{\beta} \iff (\exists q \in F_\Omega(Y))(p \Rightarrow^* aq)$$

holds. This can be shown by induction on $\mathrm{hg}(p)$.

For a proof of (ii), see Example 2.4.15. □

Example 2.4.15 shows also that the translation of a context-free language by a tree transducer is not always context-free. In fact, in this example the finite language $\{x\}$ is translated into the non-CF language $\{x^{2^n} \mid n \geq 0\}$.

Lemma 4.1.11 *For each* $T \in \mathrm{Rec}(\Sigma, X)$ *there exists an* F*-transducer* $\mathfrak{A}$ *such that* $\mathrm{dom}(\tau_{\mathfrak{A}}) = \mathrm{range}(\tau_{\mathfrak{A}}) = T$ *and* $\tau_{\mathfrak{A}}$ *is the identity mapping of* T*.*

Proof. Let $\mathbf{B} = (\mathcal{B}, \beta, B')$ be an F ΣX-recognizer with $\mathcal{B} = (B, \Sigma)$ and $T(\mathbf{B}) = T$. Take the F-transducer $\mathfrak{A} = (\Sigma, X, B, \Sigma, X, P, B')$ where

$$\begin{aligned} P \;\; = \;\; & \{x \to \beta(x)x \mid x \in X\} \cup \{\sigma(b_1, \ldots, b_m) \to b\sigma(\xi_1, \ldots, \xi_m) \mid \\ & m \geq 0,\ \sigma \in \Sigma_m,\ \ b, b_1, \ldots, b_m \in B,\ \sigma^{\mathcal{B}}(b_1, \ldots, b_m) = b\}. \end{aligned}$$

Obviously, $\mathfrak{A}$ has the desired properties. $\square$

We end off this Section with

Definition 4.1.12 Two R- or F-transducers $\mathfrak{A}$ and $\mathfrak{B}$ are *equivalent* if $\tau_{\mathfrak{A}} = \tau_{\mathfrak{B}}$ holds.

4.2 SOME CLASSES OF TREE TRANSFORMATIONS

In this section we shall define several classes of F- and R-transformations and then compare them with each other with respect to set theoretic inclusion. It will turn out that in most cases the classes to be investigated are incomparable.

Definition 4.2.1 Let $\mathfrak{A} = (\Sigma, X, A, \Omega, Y, P, A')$ be an F-transducer. Then:

(1) A production of $\mathfrak{A}$ is *linear* if each auxiliary variable occurs at most once in its right-hand side. Moreover, $\mathfrak{A}$ is a *linear* F*-transducer* (LF*-transducer*) if all of its productions are linear.

(2) $\mathfrak{A}$ is a *totally defined* F*-transducer* (TF*-transducer*) if

(i) for each $x \in X$ there is a production in P with left-hand side x and

(ii) for all $m \geq 0, \sigma \in \Sigma_m$ and $a_1, \ldots, a_m \in A$ there is a production in P with left-hand side $\sigma(a_1, \ldots, a_m)$.

(3) $\mathfrak{A}$ is a *nondeleting* F*-transducer* (NF*-transducer*) if for every production $\sigma(a_1, \ldots, a_m) \to aq$ $(\sigma \in \Sigma_m, m \geq 0)$ from P each $\xi_i \in \Xi_m$ occurs at least once in q.

(4) $\mathfrak{A}$ is a *deterministic* F*-transducer* (DF*-transducer*) if there are no two distinct productions in P with the same left-hand side.

(5) $\mathfrak{A}$ is an F*-relabeling* if each of its productions is of the form

(i) $x \to ay$ $(x \in X, a \in A, y \in Y)$ or

(ii) $\sigma(a_1, \ldots, a_m) \to a\omega(\xi_1, \ldots, \xi_m)$, where $\sigma \in \Sigma_m, a_1, \ldots, a_m, a \in A, \omega \in \Omega_m$.

Transformations induced by F-relabelings are also called F*-relabelings*.

To illustrate the above concepts, let us take the following example.

Example 4.2.2 Let $\mathfrak{A} = (\Sigma, \{x\}, \{a_0, a_1\}, \Omega, \{y\}, P, \{a_1\})$ be the F-transducer with $\Sigma = \Sigma_2 = \{\sigma\}$ and $\Omega = \Omega_2 = \{\omega\}$, where P consists of the productions

$$x \to a_0 y,$$
$$\sigma(a_0, a_0) \to a_1\omega(\xi_1, \xi_2), \sigma(a_0, a_1) \to a_0\omega(\xi_1, \xi_2), \sigma(a_1, a_0) \to a_1\omega(\xi_1, \xi_2),$$
$$\sigma(a_1, a_1) \to a_1\omega(\xi_1, \xi_2).$$

Then $\mathfrak{A}$ is a linear, totally defined, nondeleting and deterministic F-transducer. Moreover, $\mathfrak{A}$ is an F-relabeling. □

Example 4.1.3 gives an F-transducer which is linear and deterministic, but it is neither totally defined nor nondeleting.

Let us note that F-relabelings are always linear and nondeleting F-transducers.

We now define the R-transducer counterparts of the above classes of F-transducers.

Definition 4.2.3 Let $\mathfrak{A} = (\Sigma, X, A, \Omega, Y, P, A')$ be an R-transducer. Then:

(1) A production of $\mathfrak{A}$ is *linear* if each auxiliary variable occurs at most once in its right-hand side. Moreover, $\mathfrak{A}$ is a *linear* R-*transducer* (LR-*transducer*) if all of its productions are linear.

(2) $\mathfrak{A}$ is *a totally defined R-transducer* (TR-*transducer*) if

(i) for all $a \in A$ and $x \in X$ there is a production in P with left-hand side ax, and

(ii) for all $a \in A$ and $\sigma \in \Sigma_m$ $(m \geq 0)$ there is a production in P with left-hand side $a\sigma$.

(3) $\mathfrak{A}$ is a *nondeleting* R-*transducer* (NR-*transducer*) if for every production $a\sigma \to q$ $(\sigma \in \Sigma_m, m > 0)$ from P each $\xi_i \in \Xi_m$ occurs at least once in q.

(4) $\mathfrak{A}$ is a *deterministic* R-*transducer* (DR-*transducer*) if A' is a singleton and there are no distinct productions in P with the same left-hand side.

(5) $\mathfrak{A}$ is an R-*relabeling* if each of the productions of $\mathfrak{A}$ has the form

(i) $ax \to y$ $(a \in A, x \in X, y \in Y)$ or

(ii) $a\sigma \to \omega(a_1\xi_1, \ldots, a_m\xi_m)$, where $a, a_1, \ldots, a_m \in A, \sigma \in \Sigma_m, \omega \in \Omega_m$. Transformations induced by R-relabelings will also be called R-*relabelings*.

Example 4.2.4 Let $\mathfrak{A} = (\Sigma, \{x\}, \{a_0, a_1\}, \Omega, \{y_1, y_2\}, P, \{a_0\})$ be an R-transducer with $\Sigma = \Sigma_2 = \{\sigma\}$ and $\Omega = \Omega_2 = \{\omega\}$. Moreover, P consists of the productions

$$a_0x \to y_1, \;\; a_1x \to y_2,$$
$$a_0\sigma \to \omega(a_1\xi_1, a_1\xi_2), \;\; a_1\sigma \to \omega(a_0\xi_1, a_0\xi_2).$$

Then $\mathfrak{A}$ is a linear, totally defined, nondeleting and deterministic R-transducer. Moreover, $\mathfrak{A}$ is an R-relabeling. □

The R-transducer of 4.1.6 is deterministic and nondeleting, but it is neither linear nor totally defined.

Let us note that R-relabelings are linear and nondeleting R-transducers.

The abbreviations introduced above for classes of tree transducers can be combined to indicate further subclasses. For instance, an LNF-transducer is a linear nondeleting F-transducer. Moreover, a transformation is a K-*transformation* if it can be induced by a K-transducer. The class of all K-transformations will be denoted by $\mathcal{K}$. Thus, for example, $\mathcal{LNF}$ is the class of all LNF-transformations, i.e., the class of all transformations induced by linear nondeleting F-transducers. By Theorem 4.1.9, we shall write simply $\mathcal{H}$ instead of $\mathcal{HF}$ and $\mathcal{HR}$. Moreover, $\mathcal{F}$rel, resp. $\mathcal{R}$rel, will denote the class of F-relabelings, resp. R-relabelings.

We now prove

Theorem 4.2.5 *$\mathcal{F}$ and $\mathcal{R}$ are incomparable.*

Proof. In order to prove Theorem 4.2.5, we give (i) an F-transformation which is not in $\mathcal{R}$ and (ii) an R-transformation which cannot be induced by any F-transducer.

(i) Consider the LDF-transducer $\mathfrak{A}$ of Example 4.1.3. If for an R-transducer $\mathfrak{B} = (\Sigma, \{x\}, B, \Omega, \{y\}, P', B')$ we have $(\sigma(x,x), \omega(y)) \in \tau_{\mathfrak{B}}$, then at the first step of a derivation $b\sigma(x,x) \Rightarrow^*_{\mathfrak{B}} \omega(y)$ $(b \in B')$ we should apply a production of the form $b\sigma \to b'\xi_1, b\sigma \to b'\xi_2, b\sigma \to \omega(b'\xi_1), b\sigma \to \omega(b'\xi_2)$ or $b\sigma \to \omega(y)$, where $b' \in B$. In each of the above cases one of the auxiliary variables ξ_1 and ξ_2 is deleted. Therefore, $\mathrm{dom}(\tau_{\mathfrak{B}})$ is infinite.

(ii) Take the DR-transducer $\mathfrak{A}$ of Example 4.1.6. Assume that an F-transducer $\mathfrak{B} = (\Sigma, \{x\}, B, \Omega, \{y_1, y_2\}, P', B')$ induces $\tau_{\mathfrak{A}}$. Obviously, P' should then contain a production of the form

$$\sigma(b) \to b_1\omega_2(q_1, q_2) \ (b, b_1 \in B).$$

We may confine ourselves to the following cases:

(I) $q_1 = \sigma^k(y_1)$ and $q_2 = \sigma^k(y_2)$,

(II) $q_1 = \sigma^l(\xi_1)$ and $q_2 = \sigma^k(y_2)$,

(III) $q_1 = \sigma^k(y_1)$ and $q_2 = \sigma^l(\xi_1)$,

(IV) $q_1 = \sigma^m(\xi_1)$ and $q_2 = \sigma^n(\xi_1)$.

Obviously, in a derivation $\sigma^r(x) \Rightarrow^*_{\mathfrak{B}} b'\omega_2(\omega_1^{r-1}(y_1), \omega_1^{r-1}(y_2))$ $(r > 1, b' \in B')$ the last application of the above productions can be followed by applications of productions of the form $\sigma(\bar{b}) \to \bar{b}_1\xi_1$ $(\bar{b}, \bar{b}_1 \in B)$ only. Let t be the maximum of exponents in (I)–(IV). If $r > t+1$ and $\tau_{\mathfrak{B}}(\sigma^r(x)) = \omega_2(\omega_1^{r-1}(y_i), \omega_1^{r-1}(y_j))$ $(1 \le i, j \le 2)$ then $i = j$. □

From the proof of Theorem 4.2.5 we directly get

Corollary 4.2.6 *$\mathcal{DF}$ and $\mathcal{DR}$ are incomparable and so are $\mathcal{DF}$ and $\mathcal{R}$, and $\mathcal{F}$ and $\mathcal{DR}$.* □

As we have mentioned one of the main differences between F- and R-transducers is that while F-transducers first process an input subtree and then copy the resulting output subtree, R-transducers first copy an input subtree and then treat these copies independently. In the case of an LR-transducer none of the input subtrees of a tree is copied during the translation of the tree. This property leads to

Theorem 4.2.7 *$\mathcal{LR}$ is a proper subclass of $\mathcal{LF}$.*

Proof. By (i) in the proof of Theorem 4.2.5, $\mathcal{LF}$ is not a subclass of $\mathcal{LR}$. Thus, it is enough to show the validity of $\mathcal{LR} \subseteq \mathcal{LF}$.

Let $\mathfrak{A} = (\Sigma, X, A, \Omega, Y, P, A')$ be an LR-transducer. Then the productions from P can be written in the form

(i) $ax \to q$ $(a \in A, x \in X, q \in F_\Omega(Y))$, or

(ii) $a\sigma(\xi_1, \ldots, \xi_m) \to q(a_1\xi_1, \ldots, a_m\xi_m)$ $(a, a_1, \ldots, a_m \in A, m \geq 0, \sigma \in \Sigma_m, q \in F_\Omega[Y \cup A\Xi_m])$.

Now take the following R-transducer $\overline{\mathfrak{A}}$. If $\mathfrak{A}$ is nondeleting, then $\overline{\mathfrak{A}} = \mathfrak{A}$. In the opposite case $\overline{\mathfrak{A}} = (\Sigma, X, \overline{A}, \Omega, Y, \overline{P}, A')$ is given as follows. Let $\overline{A} = A \cup \{*\}$ $(* \notin A)$. Fix any $\overline{y} \in Y$ and enlarge P by all productions $*x \to \overline{y}$ $(x \in X)$ and $*\sigma \to \overline{y}$ $(m \geq 0, \sigma \in \Sigma_m)$. Denote by $\overline{P}$ the resulting set of productions. Obviously, $\overline{\mathfrak{A}}$ is linear and equivalent to $\mathfrak{A}$. The only difference between $\overline{\mathfrak{A}}$ and $\mathfrak{A}$ is that $\overline{\mathfrak{A}}$ transforms (in state $*$) even those subtrees of a tree $p \in F_\Sigma(X)$ which are deleted during the corresponding derivation of p in $\mathfrak{A}$.

Next, construct the F-transducer $\mathfrak{B} = (\Sigma, X, B, \Omega, Y, P', B')$, where $B = \overline{A}$ and $B' = A'$. Moreover, given any $x \in X, b \in B$ and $q \in F_\Omega(Y), x \to bq$ is in P' iff $bx \to q$ is in $\overline{P}$. Furthermore, the production

$$\sigma(b_1, \ldots, b_m) \to bq(\xi_1, \ldots, \xi_m) \ (\sigma \in \Sigma_m, m \geq 0, b_1, \ldots, b_m, b \in B, q \in F_\Omega(Y \cup \Xi_m))$$

is in P' iff $\overline{P}$ contains a production

$$b\sigma \to q(c_1\xi_1, \ldots, c_m\xi_m),$$

such that for each $i = 1, \ldots, m$,

$$b_i = \begin{cases} c_i & \text{if } \xi_i \text{ occurs in } q, \\ * & \text{otherwise.} \end{cases}$$

Obviously $\mathfrak{B}$ is linear.

In order to complete the proof of Theorem 4.2.7, it is enough to show that the equivalence

$$p \Rightarrow^*_{\mathfrak{B}} bq \iff bp \Rightarrow^*_{\mathfrak{A}} q \tag{1}$$

holds for all $b \in B, p \in F_\Sigma(X)$ and $q \in F_\Omega(Y)$. We shall proceed by induction on $\mathrm{hg}(p)$.

If $\mathrm{hg}(p) = 0$, then (1) obviously holds by the definition of P'.

Now let $p = \sigma(p_1, \ldots, p_m)$ ($\sigma \in \Sigma_m, m > 0$), and assume that (1) has been proved for all trees in $F_\Sigma(X)$ of lesser height.

(I) Let $p \Rightarrow_{\mathfrak{B}}^* bq$ hold. More in detail, let

$$p = \sigma(p_1, \ldots, p_m) \Rightarrow_{\mathfrak{B}}^* \sigma(b_1 q_1, \ldots, b_m q_m) \Rightarrow_{\mathfrak{B}} b\overline{q}(q_1, \ldots, q_m) = bq$$

where $p_i \Rightarrow_{\mathfrak{B}}^* b_i q_i$ ($i = 1, \ldots, m$). Then by the induction hypothesis, we have $b_i p_i \Rightarrow_{\overline{\mathfrak{A}}}^* q_i$ ($i = 1, \ldots, m$). Moreover, by the definition of P', $b\sigma \to \overline{q}(b_1\xi_1, \ldots, b_m\xi_m)$ is in $\overline{P}$. Therefore,

$$bp = b\sigma(p_1, \ldots, p_m) \Rightarrow \overline{q}(b_1 p_1, \ldots, b_m p_m) \Rightarrow^* \overline{q}(q_1, \ldots, q_m) = q$$

also exists in $\overline{\mathfrak{A}}$.

(II) Assume that in $\overline{\mathfrak{A}}$ we have a derivation

$$bp = b\sigma(p_1, \ldots, p_m) \Rightarrow \overline{q}(b_1 p_1, \ldots, b_m p_m) \Rightarrow^* \overline{q}(q_1, \ldots, q_m) = q$$

where each q_i ($i = 1, \ldots, m$) is obtained by a derivation $b_i p_i \Rightarrow^* q_i$ in $\overline{\mathfrak{A}}$. Moreover, let $b_i = *$ and $q_i = \overline{y}$ if ξ_i does not occur in $\overline{q}$. Then $\sigma(b_1, \ldots, b_m) \to b\overline{q}$ is in P'. Furthermore, by the induction hypothesis, there are derivations $p_i \Rightarrow_{\mathfrak{B}}^* b_i q_i$ ($i = 1, \ldots, m$). Therefore, the derivation

$$p = \sigma(p_1, \ldots, p_m) \Rightarrow_{\mathfrak{B}}^* \sigma(b_1 q_1, \ldots, b_m q_m) \Rightarrow_{\mathfrak{B}} b\overline{q}(q_1, \ldots, q_m) = bq$$

is also valid. □

For linear nondeleting tree transformations we have the following stronger result.

Theorem 4.2.8 *$\mathcal{LNR} = \mathcal{LNF}$.*

Proof. The LF-transducer $\mathfrak{B}$ constructed to the LNR-transducer $\mathfrak{A}$ in the proof of the previous Theorem is obviously nondeleting.

Conversely, let $\mathfrak{C} = (\Sigma, X, C, \Omega, Y, P'', C')$ be an arbitrary LNF-transducer. Construct the R-transducer $\mathfrak{A} = (\Sigma, X, C, \Omega, Y, P, C')$, where P is defined as follows:

$$(ax, q) \in P \iff (x, aq) \in P''$$

and

$$(a\sigma,\ q(a_1\xi_1, \ldots, a_m\xi_m)) \in P \iff (\sigma(a_1, \ldots, a_m), aq(\xi_1, \ldots, \xi_m)) \in P'',$$

where $x \in X, a, a_1, \ldots, a_m \in A, \sigma \in \Sigma_m$ ($m \geq 0$) and $q \in F_\Omega(Y \cup \Xi_m)$. Obviously, $\mathfrak{A}$ is an LNR-transducer. Now to $\mathfrak{A}$ construct the F-transducer $\mathfrak{B}$ as in the proof of Theorem 4.2.7. Then $\mathfrak{B} = \mathfrak{C}$. □

The LF-transducer $\mathfrak{B}$ constructed to an R-relabeling in the proof of Theorem 4.2.7 is obviously an F-relabeling. Moreover, the R-transducer $\mathfrak{A}$ given to an F-relabeling $\mathfrak{C}$ in the proof of Theorem 4.2.8 is an R-relabeling. Thus, we have

Corollary 4.2.9 $\mathcal{F}\text{rel} = \mathcal{R}\text{rel}$. □

According to Corollary 4.2.9, we may speak simply about relabelings.

One can easily show the existence of an LNF-transformation which is not a relabeling.

Our comparison results can be summarized by the diagram in Fig. 4.3.

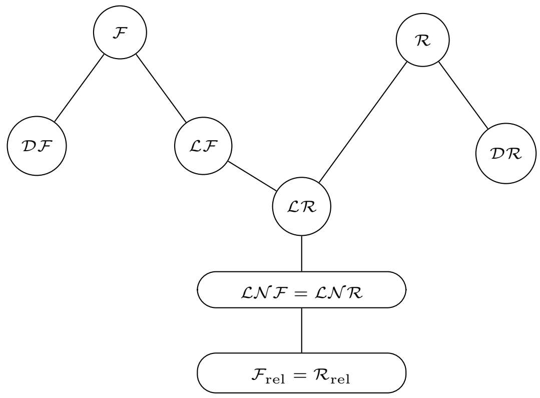

Figure 4.3.

4.3 COMPOSITIONS AND DECOMPOSITIONS OF TREE TRANSFORMATIONS

Let $\mathcal{K}$ be a class of tree transformations. We say that $\mathcal{K}$ is *closed under composition* if $\tau_1 \circ \tau_2 \in \mathcal{K}$ whenever $\tau_1, \tau_2 \in \mathcal{K}$. As we shall see, some of our classes of tree transformations are closed under composition while others are not. On the other hand, in many cases it is possible to decompose a tree transformation into a composition of simpler ones.

For any two classes $\mathcal{K}_1$ and $\mathcal{K}_2$ of tree transformations, we introduce the notation $\mathcal{K}_1 \circ \mathcal{K}_2 = \{\tau_1 \circ \tau_2 \mid \tau_1 \in \mathcal{K}_1, \tau_2 \in \mathcal{K}_2\}$. Using this notation, the closure of a class $\mathcal{K}$ of tree transformations under composition can be expressed by the inclusion $\mathcal{K} \circ \mathcal{K} \subseteq \mathcal{K}$. Similarly, the fact that all transformations in $\mathcal{K}$ can be given as compositions of a transformation in $\mathcal{K}_1$ by a transformation from $\mathcal{K}_2$ can be expressed by $\mathcal{K} \subseteq \mathcal{K}_1 \circ \mathcal{K}_2$. Finally, if $\mathcal{K}$ is a class of tree transformations, then let $\mathcal{K}^1 = \mathcal{K}$ and $\mathcal{K}^n = \mathcal{K} \circ \mathcal{K}^{n-1}$ $(n > 1)$. All of the classes defined in the previous section ($\mathcal{R}, \mathcal{F}, \mathcal{LF}, \mathcal{H}$ etc.) include all identity transformations $\{(t,\ t) \mid t \in F_\Sigma(X)\}$. Hence, if $\mathcal{K}$ is any one of these classes, then we know that

$$\mathcal{K} \subseteq \mathcal{K}^2 \subseteq \mathcal{K}^n \subseteq \ldots.$$

First we prove a decomposition theorem concerning F-transformations.

Lemma 4.3.1 *$\mathcal{F} \subseteq \mathcal{LF} \circ \mathcal{H}$ and $\mathcal{F} \subseteq \mathcal{LR} \circ \mathcal{H}$.*

Proof. Let $\mathfrak{A} = (\Sigma, X, A, \Delta, Z, P, A')$ be an arbitrary F-transducer. Arrange the productions from P in a fixed order and number them from 1 to $|P|$. For all $i(= 1, \ldots, |P|)$, if the left side of the ith production is $x \in X$, then let $x^{(i)}$ be a new letter. Denote by Y the set of all such $x^{(i)}$. Moreover, for all $i(= 1, \ldots, |P|)$, if the symbol $\sigma \in \Sigma_m$ $(m \geq 0)$ occurs in the left-hand side of the ith production, then $\sigma^{(i)}$ will be a new m-ary operator. The set of all such operators will be denoted by Ω.

Now we introduce the F-transducer $\mathfrak{B} = (\Sigma, X, A, \Omega, Y, P', A')$, where P' is defined as follows:

(i) $x \to ax^{(i)}$ $(x \in X, a \in A)$ is in P' iff the ith production in P is $x \to ar$ for some r,

(ii) $\sigma(a_1, \ldots, a_m) \to a\sigma^{(i)}(\xi_1, \ldots, \xi_m)$ $(\sigma \in \Sigma_m, m \geq 0,\ a_1, \ldots, a_m \in A)$ is in P' iff the ith production in P is $\sigma(a_1, \ldots, a_m) \to ar$ for some r.

Obviously, $\mathfrak{B}$ is linear and nondeleting. Thus, by Theorem 4.2.8, $\tau_{\mathfrak{B}}$ is a linear nondeleting R-transformation, as well.

Next define the F-transducer $\mathfrak{C} = (\Omega, Y, \{c_0\}, \Delta, Z, P'', c_0)$ in the following way:

(i) $x^{(i)} \to c_0 r$ is in P'' iff the ith production in P is $x \to ar$,

(ii) $\sigma^{(i)}(c_0, \ldots, c_0) \to c_0 r$ is in P'' iff the ith production in P is $\sigma(a_1, \ldots, a_m) \to ar$.

Then $\mathfrak{C}$ is an HF-transducer.

We prove that $\tau_{\mathfrak{A}} = \tau_{\mathfrak{B}} \circ \tau_{\mathfrak{C}}$. For this it is enough to show that, for all $p \in F_{\Sigma}(X)$, $r \in F_{\Delta}(Z)$ and $a \in A$, the equivalence

$$p \Rightarrow^*_{\mathfrak{A}} ar \iff (\exists q \in F_{\Omega}(Y))(p \Rightarrow^*_{\mathfrak{B}} aq \wedge q \Rightarrow^*_{\mathfrak{C}} c_0 r) \tag{1}$$

holds. We proceed by induction on $\mathrm{hg}(p)$.

If $\mathrm{hg}(p) = 0$, then (1) obviously holds.

Assume that $p = \sigma(p_1, \ldots, p_m)$ $(\sigma \in \Sigma_m, m > 0)$ and that (1) has been proved for all trees from $F_{\Sigma}(X)$ of lesser height.

(I) Let

$$p \Rightarrow^*_{\mathfrak{A}} \sigma(a_1 r_1, \ldots, a_m r_m) \Rightarrow_{\mathfrak{A}} a\overline{r}(r_1, \ldots, r_m) = ar, \tag{2}$$

where $p_i \Rightarrow^*_{\mathfrak{A}} a_i r_i$ $(r_i \in F_{\Delta}(Z))$ holds for each $i(= 1, \ldots, m)$. Then, by the induction hypothesis, there are trees $q_i \in F_{\Omega}(Y)$ $(i = 1, \ldots, m)$ such that $p_i \Rightarrow^*_{\mathfrak{B}} a_i q_i$ and $q_i \Rightarrow^*_{\mathfrak{C}} c_0 r_i$ hold. Assume that the production $\sigma(a_1, \ldots, a_m) \to a\overline{r}$ last applied in (2) is the ith one in P. Then

$$(\sigma(a_1, \ldots, a_m), a\sigma^{(i)}(\xi_1, \ldots, \xi_m)) \in P' \text{ and } (\sigma^{(i)}(c_0, \ldots, c_0), c_0\overline{r}) \in P''.$$

Therefore, taking $q = \sigma^{(i)}(q_1, \ldots, q_m)$, we have the desired derivations

$$p \Rightarrow^*_{\mathfrak{B}} \sigma(a_1 q_1, \ldots, a_m q_m) \Rightarrow_{\mathfrak{B}} a\sigma^{(i)}(q_1, \ldots, q_m) = aq$$

and

$$q \Rightarrow^*_{\mathfrak{C}} \sigma^{(i)}(c_0 r_1, \ldots, c_0 r_m) \Rightarrow_{\mathfrak{C}} c_0\overline{r}(r_1, \ldots, r_m) = c_0 r.$$

(II) The fact that the right side of (1) implies its left side can be proved by inverting the above computation. □

Lemma 4.3.2 $\mathcal{F} \circ \mathcal{H} \subseteq \mathcal{F}$.

Proof. Let $\mathfrak{A} = (\Sigma, X, A, \Omega, Y, P, A')$ be an F-transducer and $\mathfrak{B} = (\Omega, Y, \{b_0\}, \Delta, Z, P', b_0)$ an HF-transducer. We shall construct an F-transducer $\mathfrak{C}$ whose productions will be composed of productions of $\mathfrak{A}$ and derivations in $\mathfrak{B}$. For this, using the fact that derivations in $\mathfrak{B}$ can be started from trees in $F_\Omega[Y \cup b_0\Xi]$ (see p. 133), we define derivations in $\mathfrak{B}$ for trees in $F_\Omega(Y \cup \Xi)$. Take two trees $q \in F_\Omega(Y \cup \Xi_m)$ and $r \in F_\Delta(Z \cup \Xi_m)$. We write $q \Rightarrow^*_{\mathfrak{B}} b_0 r$ if

$$q(b_0\xi_1, \dots, b_0\xi_m) \Rightarrow^*_{\mathfrak{B}} b_0 r$$

holds. Now define an F-transducer $\mathfrak{C} = (\Sigma, X, A, \Delta, Z, P'', A')$, where P'' is given as follows:

(i) $x \to ar$ $(x \in X, a \in A, r \in F_\Delta(Z))$ is in P'' iff there is a production $x \to aq$ in P such that $q \Rightarrow^*_{\mathfrak{B}} b_0 r$ holds,

(ii) $\sigma(a_1, \dots, a_m) \to ar$ $(\sigma \in \Sigma_m, m \geq 0, a_1, \dots, a_m, a \in A, r \in F_\Delta(Z \cup \Xi_m))$ is in P'' iff there is a production $\sigma(a_1, \dots, a_m) \to aq$ in P such that $q \Rightarrow^*_{\mathfrak{B}} b_0 r$ holds. Since at each step of the transformation of a tree the number of applications is finite, P'' is finite.

We prove that for all $a \in A, p \in F_\Sigma(X)$ and $r \in F_\Delta(Z)$ the equivalence

$$p \Rightarrow^*_{\mathfrak{C}} ar \iff (\exists q \in F_\Omega(Y))(p \Rightarrow^*_{\mathfrak{A}} aq \wedge q \Rightarrow^*_{\mathfrak{B}} b_0 r) \tag{3}$$

holds. We proceed by induction on $\mathrm{hg}(p)$.

If hg $(p) = 0$ then (3) obviously holds.

Assume that $p = \sigma(p_1, \dots, p_m)$ $(\sigma \in \Sigma_m, m > 0)$ and that (3) has been proved for all trees from $F_\Sigma(X)$ of lesser height.

(I) First we show that the right side of (3) implies its left side. For this assume that the derivations

$$p \Rightarrow^*_{\mathfrak{A}} \sigma(a_1q_1, \dots, a_mq_m) \Rightarrow_{\mathfrak{A}} a\overline{q}(q_1, \dots, q_m) = aq$$

$$(p_i \Rightarrow^*_{\mathfrak{A}} a_iq_i,\ i = 1, \dots, m)$$

and

$$q \Rightarrow^*_{\mathfrak{B}} \overline{q}(b_0r_1, \dots, b_0r_m) \Rightarrow^*_{\mathfrak{B}} b_0\overline{r}(r_1, \dots, r_m) = b_0 r$$

$$(q_i \Rightarrow^*_{\mathfrak{B}} b_0r_i, i = 1, \dots,\ m)$$

are given. Then, by the induction hypothesis, the relations $p_i \Rightarrow^*_{\mathfrak{C}} a_ir_i$ $(i = 1, \dots, m)$ also hold. Moreover, by the definition of P'', $\sigma(a_1, \dots, a_m) \to a\overline{r}$ is in P''. Thus, we have the derivation

$$p \Rightarrow^*_{\mathfrak{C}} \sigma(a_1r_1, \dots, a_mr_m) \Rightarrow_{\mathfrak{C}} a\overline{r}(r_1, \dots, r_m) = ar. \tag{4}$$

(II) Suppose that (4) and the derivations $p_i \Rightarrow^*_{\mathfrak{C}} a_ir_i$ $(i = 1, \dots, m)$ are valid. Then, by the induction hypothesis, there are trees $q_i \in F_\Omega(Y)$ $(i = 1, \dots, m)$ such that $p_i \Rightarrow^*_{\mathfrak{A}} a_iq_i$

and $q_i \Rightarrow^*_{\mathfrak{B}} b_0 r_i$ hold. Moreover, by the definition of P'', there exists a $\overline{q} \in F_\Omega(Y \cup \Xi_m)$ with $(\sigma(a_1, \ldots, a_m), a\overline{q}) \in P$ and $\overline{q} \Rightarrow^*_{\mathfrak{B}} b_0 \overline{r}$. Therefore, for $q = \overline{q}(q_1, \ldots, q_m)$

$$p \Rightarrow^*_{\mathfrak{A}} \sigma(a_1 q_1, \ldots, a_m q_m) \Rightarrow_{\mathfrak{A}} a\overline{q}(q_1, \ldots, q_m) = aq$$

and

$$q \Rightarrow^*_{\mathfrak{B}} \overline{q}(b_0 r_1, \ldots, b_0 r_m) \Rightarrow^*_{\mathfrak{B}} b_0 \overline{r}(r_1, \ldots, r_m) = b_0 r$$

hold. □

From Theorem 4.2.7 and the Lemmas 4.3.1 and 4.3.2 we directly obtain

Theorem 4.3.3 $\mathcal{F} = \mathcal{LF} \circ \mathcal{H} = \mathcal{LR} \circ \mathcal{H}$. □

The constructions in the proofs of Lemma 4.3.1 and 4.3.2 preserve determinism. Thus, we have

Corollary 4.3.4 $\mathcal{DF} = \mathcal{LDF} \circ \mathcal{H}$. □

Now we investigate some special classes of F-transformations for closure under composition.

Lemma 4.3.5 *Let* $\mathfrak{A} = (\Sigma, X, A, \Omega, Y, P, A')$ *be an* F*-transducer. Then there exists a totally defined* F*-transducer* $\mathfrak{B} = (\Sigma, X, B, \Omega, Y, P', B')$ *such that* $\tau_{\mathfrak{A}} = \tau_{\mathfrak{B}}$. *Moreover, if* $\mathfrak{A}$ *is linear, then* $\mathfrak{B}$ *can be chosen linear, too.*

Proof. Let $B = A \cup \{*\}$ and $B' = A'$. The required $\mathfrak{B}$ results if we put

$$P' = P \cup \{x \to *y \mid x \in X, y \in Y\} \cup \{\sigma(b_1, \ldots, b_m) \to *y \mid \sigma \in \Sigma_m,$$

$$m \geq 0, b_1, \ldots, b_m \in B, y \in Y\}.$$

If $\mathfrak{A}$ is linear, then so is $\mathfrak{B}$. □

Theorem 4.3.6 *The following equalities hold:*

(i) $\mathcal{LF} \circ \mathcal{LF} = \mathcal{LF}$,

(ii) $\mathcal{LR} \circ \mathcal{LR} = \mathcal{LF}$.

Proof. In order to show (i), take two LF-transducers $\mathfrak{A} = (\Sigma, X, A, \Omega, Y, P, A')$ and $\mathfrak{B} = (\Omega, Y, B, \Delta, Z, P', B')$. In view of Lemma 4.3.5, we may assume that $\mathfrak{B}$ is totally defined. Construct an F-transducer $\mathfrak{C} = (\Sigma, X, C, \Delta, Z, P'', C')$ with $C = A \times B$ and $C' = A' \times B'$. Furthermore, P'' is defined as follows:

(I) $x \to (a, b)r$ ($x \in X, (a, b) \in C, r \in F_\Delta(Z)$) is in P'' iff there is a production $x \to aq$ in P such that $q \Rightarrow^*_{\mathfrak{B}} br$ holds,

(II) $\sigma((a_1, b_1), \ldots, (a_m, b_m)) \to (a, b)r$

$$(\sigma \in \Sigma_m, m \geq 0, (a_1, b_1), \ldots, (a_m, b_m), (a, b) \in C, r \in F_\Delta(Z \cup \Xi_m))$$

is in P'' iff there is a production $\sigma(a_1, \ldots, a_m) \to aq$ in P such that $q(b_1\xi_1, \ldots, b_m\xi_m) \Rightarrow_{\mathfrak{B}}^* br$ holds.

We shall prove that for arbitrary $p \in F_\Sigma(X), r \in F_\Delta(Z)$ and $(a, b) \in C$ the equivalence

$$p \Rightarrow_{\mathfrak{C}}^* (a, b)r \iff (\exists q \in F_\Omega(Y))(p \Rightarrow_{\mathfrak{A}}^* aq \wedge q \Rightarrow_{\mathfrak{B}}^* br) \tag{5}$$

holds. We proceed by induction on $\mathrm{hg}(p)$.

If $\mathrm{hg}(p) = 0$, then (5) obviously holds.

Now let $p = \sigma(p_1, \ldots, p_m)$ $(\sigma \in \Sigma_m, m > 0)$, and assume that (5) has been proved for all trees of lesser height.

First we show that the right side of (5) implies the left side. Suppose we are given derivations

$$p \Rightarrow_{\mathfrak{A}}^* \sigma(a_1q_1, \ldots, a_mq_m) \Rightarrow_{\mathfrak{A}} a\overline{q}(q_1, \ldots, q_m) = aq$$

and

$$q \Rightarrow_{\mathfrak{B}}^* \overline{q}(b_1r_1, \ldots, b_mr_m) \Rightarrow_{\mathfrak{B}}^* b\overline{r}(r_1, \ldots, r_m) = br$$

where $p_i \Rightarrow_{\mathfrak{A}}^* a_iq_i$ and $q_i \Rightarrow_{\mathfrak{B}}^* b_ir_i$ $(i = 1, \ldots, m)$. (Observe that for each i $(1 \leq i \leq m)$ there exists an r_i such that $q_i \Rightarrow_{\mathfrak{B}}^* b_ir_i$ holds since $\mathfrak{B}$ is totally defined.) Then, by the induction hypothesis, the derivations $p_i \Rightarrow_{\mathfrak{C}}^* (a_i, b_i)r_i$ $(i = 1, \ldots, m)$ are also valid. Furthermore, by the definition of P'', the production

$$\sigma((a_1, b_1), \ldots, (a_m, b_m)) \to (a, b)\overline{r}$$

is in P''. Therefore, we get the derivation

$$p \Rightarrow_{\mathfrak{C}}^* \sigma((a_1, b_1)r_1, \ldots, (a_m, b_m)r_m) \Rightarrow_{\mathfrak{C}} (a, b)\overline{r}(r_1, \ldots, r_m) = (a, b)r.$$

The fact that the left side of (5) implies its right side can be shown by reversing the above argument.

In order to prove (ii) it is enough to note that the HF-transducer $\mathfrak{C}$ constructed to the LF-transducer $\mathfrak{A}$ in the proof of Lemma 4.3.1 is also linear. Moreover, by Theorem 4.2.7, the inclusion $\mathcal{LR} \subseteq \mathcal{LF}$ holds. □

Using an argument similar to that used in the proof of Theorem 4.3.6 (i), one can prove

Theorem 4.3.7 *The classes $\mathcal{DF}$ and $\mathcal{H}$ are closed under composition.* □

From Theorem 4.3.7, by Theorem 4.3.6 (i), we get

Corollary 4.3.8 *The class $\mathcal{LDF}$ is closed under composition.* □

Using our decomposition results, one can prove

Theorem 4.3.9 $\mathcal{F} \circ \mathcal{DF} = \mathcal{F}$. □

Now we turn to decomposition of R-transducers.

Lemma 4.3.10 $\mathcal{R} \subseteq \mathcal{H} \circ \mathcal{LR}$.

Proof. Let $\mathfrak{A} = (\Sigma, X, A, \Delta, Z, P, A')$ be an arbitrary R-transducer. Let n be the greatest integer with $\Sigma_n \neq \emptyset$. For any production $d \in P$ and natural number i $(1 \leq i \leq n)$, denote by $k(d, i)$ the number of occurrences of ξ_i in the right-hand side of d. Set $k = \max\{k(d, i) \mid d \in P, i = 1, \ldots, n\}$. Furthermore, take the ranked alphabet Ω given by $\Omega = \bigcup(\Omega_{m \cdot k} \mid m \geq 0)$ and $\Omega_{m \cdot k} = \{\sigma' \mid \sigma \in \Sigma_m\}$ $(m \geq 0)$.

Let $\mathfrak{B} = (\Sigma, X, \{b_0\}, \Omega, X, P', b_0)$ be the HR-transducer where P' consists of all productions

$$b_0 x \to x \ (x \in X)$$

and

$$b_0 \sigma \to \sigma'(b_0^k \xi_1^k, \ldots, b_0^k \xi_m^k) \ (\sigma \in \Sigma_m, m \geq 0).$$

Next define an LR-transducer $\mathfrak{C} = (\Omega, X, A, \Delta, Z, P'', A')$, where P'' is given as follows:

(i) $ax \to r$ $(x \in X)$ is in P'' iff it is in P.

(ii) Let $\sigma \in \Sigma_m$ $(m \geq 0)$ and $\boldsymbol{\xi}_i \in \Xi^k$ with $\xi_{i_j} = \xi_{(i-1)k+j}$ $(i = 1, \ldots, m, j = 1, \ldots, k)$. Then $a\sigma' \to r(\mathbf{a}_1 \boldsymbol{\xi}_1, \ldots, \mathbf{a}_m \boldsymbol{\xi}_m)$ is in P'' iff $a\sigma \to r(\mathbf{a}_1 \xi_1^{n_1}, \ldots, \mathbf{a}_m \xi_m^{n_m})$ is in P (for some $n_1, \ldots, n_m$).

For each $p \in F_\Sigma(X)$ let us denote by $p' \in F_\Omega(X)$ the tree given as follows:

(I) if $p = x \in X$, then $p' = x$,

(II) if $p = \sigma(p_1, \ldots, p_m)$ $(\sigma \in \Sigma_m, m \geq 0)$, then $p' = \sigma'(p_1'^k, \ldots, p_m'^k)$.

It is easy to show that the transformation $\tau_{\mathfrak{B}}$ is exactly the mapping $p \to p'$ $(p \in F_\Sigma(X))$.

In order to prove $\tau_{\mathfrak{A}} = \tau_{\mathfrak{B}} \circ \tau_{\mathfrak{C}}$ it is enough to show that for all $a \in A, p \in F_\Sigma(X)$ and $r \in F_\Delta(Z)$ the equivalence

$$ap \Rightarrow_{\mathfrak{A}}^* r \iff ap' \Rightarrow_{\mathfrak{C}}^* r \tag{6}$$

holds. We proceed by induction on $\mathrm{hg}(p)$.

If $\mathrm{hg}(p) = 0$ then, by the choice of P'', (6) is obviously valid.

Now let $p = \sigma(p_1, \ \ldots, p_m)$ $(\sigma \in \Sigma_m, m > 0)$, and assume that (6) has been proved for all trees of lesser height.

First we prove that the left side of (6) implies its right side. Assume that

$$ap \Rightarrow_{\mathfrak{A}} \overline{r}(\mathbf{a}_1 p_1^{n_1}, \ldots, \mathbf{a}_m p_m^{n_m}) \Rightarrow_{\mathfrak{A}}^* \overline{r}(\mathbf{r}_1, \ldots, \mathbf{r}_m) = r$$

where $\mathbf{a}_i p_i^{n_i} \Rightarrow_{\mathfrak{A}}^* \mathbf{r}_i$ $(i = 1, \ldots, m)$. Then, by the definition of P'', the production $a\sigma' \to \overline{r}(\mathbf{a}_1\boldsymbol{\xi}_1, \ldots, \mathbf{a}_m\boldsymbol{\xi}_m)$ is in P''. Moreover, by the induction hypothesis, there are derivations $\mathbf{a}_i p_i'^{n_i} \Rightarrow_{\mathfrak{C}}^* \mathbf{r}_i$ for all $i(= 1, \ldots, m)$. Therefore, we have the desired derivation

$$ap' \Rightarrow_{\mathfrak{C}} \overline{r}(\mathbf{a}_1 p_1'^{n_1}, \ldots, \mathbf{a}_m p_m'^{n_m}) \Rightarrow_{\mathfrak{C}}^* \overline{r}(\mathbf{r}_1, \ldots, \mathbf{r}_n) = r.$$

The fact that the right side of (6) implies its left side can be proved by the converse of the computation above. □

Lemma 4.3.11 $\mathcal{H} \circ \mathcal{R} \subseteq \mathcal{R}$.

Proof. Let $\mathfrak{A} = (\Sigma, X, \{a_0\}, \Omega, Y, P, a_0)$ be an HR-transducer and $\mathfrak{B} = (\Omega, Y, B, \Delta, Z, P', B')$ an arbitrary R-transducer. Take the R-transducer $\mathfrak{C} = (\Sigma, X, B, \Delta, Z, P'', B')$, where P'' is given in the following way:

(i) $bx \to r$ $(b \in B, x \in X, r \in F_\Delta(Z))$ is in P'' iff there is a production $a_0 x \to q$ in P such that $bq \Rightarrow_{\mathfrak{B}}^* r$ holds;

(ii) $b\sigma \to r$ $(b \in B, \sigma \in \Sigma_m, m \geq 0, r \in F_\Delta[Z \cup B\Xi_m])$ is in P'' iff there is a production $a_0\sigma \to q(a_0\xi_1, \ldots, a_0\xi_m)$ $(q \in F_\Omega(Y \cup \Xi_m))$ in P such that $bq \Rightarrow_{\mathfrak{B}}^* r$ holds.

To show $\tau_{\mathfrak{A}} \circ \tau_{\mathfrak{B}} = \tau_{\mathfrak{C}}$ it is enough to prove that for arbitrary $b \in B, p \in F_\Sigma(X)$ and $r \in F_\Delta(Z)$ the equivalence

$$bp \Rightarrow_{\mathfrak{C}}^* r \iff (\exists q \in F_\Omega(Y))(a_0 p \Rightarrow_{\mathfrak{A}}^* q \wedge bq \Rightarrow_{\mathfrak{B}}^* r)$$

holds. This can be carried out by induction on $\mathrm{hg}(p)$. □

From Lemmas 4.3.10 and 4.3.11 we directly get

Theorem 4.3.12 $\mathcal{R} = \mathcal{H} \circ \mathcal{LR}$. □

Using Theorems 4.3.3 and 4.3.12 we obtain

Theorem 4.3.13 *For each $n \geq 1$ the inclusions $\mathcal{F}^n \subseteq \mathcal{R}^{n+1}$ and $\mathcal{R}^n \subseteq \mathcal{F}^{n+1}$ hold.* □

Taking $n = 1$ in Theorem 4.3.13, we see that every F-transformation can be given as the composition of two R-transformations, and each R-transformation can be obtained as the composition of two F-transformations. Thus, taking Theorem 4.2.5 into account, we get

Corollary 4.3.14 *Neither $\mathcal{F}$ nor $\mathcal{R}$ is closed under composition.* □

One can show that $\mathcal{F}$ is not closed under composition by LNF-transformations either. For $\mathcal{R}$, we have

Theorem 4.3.15 $\mathcal{R} \circ \mathcal{LNR} = \mathcal{R}$.

Proof. By Theorem 4.3.12, it suffices to show that $\mathcal{LR}$ is closed under compositions by LNR-transformations.

Let $\mathfrak{A} = (\Sigma, X, A, \Omega, Y, P, A')$ be an LR-transducer and $\mathfrak{B} = (\Omega, Y, B, \Delta, Z, P', B')$ an LNR-transducer. Take the R-transducer $\mathfrak{C} = (\Sigma, X, C, \Delta, Y, P'', C')$ with $C = A \times B$ and $C' = A' \times B'$. Moreover, P'' is given as follows:

(i) $(a, b)x \to r$ $((a, b) \in C, x \in X, r \in F_\Delta(Z))$ is in P'' iff there is a production $ax \to q$ in P such that $bq \Rightarrow^*_{\mathfrak{B}} r$ holds.

(ii) $(a, b)\sigma \to r((a_1, b_1)\xi_1, \ldots, (a_m, b_m)\xi_m)$

$$((a, b), (a_1, b_1), \ldots, (a_m, b_m) \in C, \sigma \in \Sigma_m, m \geq 0, r \in F_\Delta[Z \cup C\Xi_m])$$

is in P'' iff there is a production $a\sigma \to q(a_1\xi_1, \ldots, a_m\xi_m)$ $(q \in F_\Omega(Y \cup \Xi_m))$ in P such that $bq \Rightarrow^*_{\mathfrak{B}} r(b_1\xi_1, \ldots, b_m\xi_m)$ holds.

In order to show $\tau_{\mathfrak{C}} = \tau_{\mathfrak{A}} \circ \tau_{\mathfrak{B}}$ it is enough to prove that for arbitrary $(a, b) \in C, p \in F_\Sigma(X)$ and $q \in F_\Delta(Z)$ the equivalence

$$(a, b)p \Rightarrow^*_{\mathfrak{C}} r \iff (\exists q \in F_\Omega(Y))(ap \Rightarrow^*_{\mathfrak{A}} q \wedge bq \Rightarrow^*_{\mathfrak{B}} r)$$

holds. This can be done by induction on $\mathrm{hg}(p)$. □

Later on we need the following results.

Lemma 4.3.16 *Let $\tau \subseteq F_\Sigma(X) \times F_\Omega(Y)$ be an arbitrary F-transformation and $T \in \mathrm{Rec}(\Omega, Y)$. Then $T\tau^{-1} \in \mathrm{Rec}(\Sigma, X)$.*

Proof. By Lemma 4.1.11, there exists an F-transducer $\mathfrak{A}$ with $\mathrm{dom}(\tau_{\mathfrak{A}}) = \mathrm{range}(\tau_{\mathfrak{A}}) = T$ and $\tau_{\mathfrak{A}}$ is the identity mapping on T. Moreover, by the proof of Lemma 4.1.11, we may suppose that $\mathfrak{A}$ is deterministic. Furthermore, by Theorem 4.3.9, $\mathcal{F} \circ \mathcal{DF} = \mathcal{F}$. Thus, since $T\tau^{-1} = \mathrm{dom}(\tau \circ \tau_{\mathfrak{A}})$, in order to prove Lemma 4.3.16, it is enough to show that the domain of an F-transformation is recognizable. But this is true by (i) of Theorem 4.1.10. □

From Theorem 4.1.10 and Lemma 4.3.16, using the inclusion $\mathcal{R} \subseteq \mathcal{F}^2$ (see Theorem 4.3.13), we get

Corollary 4.3.17 *Let $\tau \subseteq F_\Sigma(X) \times F_\Omega(Y)$ be an arbitrary* R*-transformation. If $T \in \mathrm{Rec}(\Omega, Y)$, then $T\tau^{-1} \in \mathrm{Rec}(\Sigma, X)$. In particular,* $\mathrm{dom}(\tau) \in \mathrm{Rec}(\Sigma, X)$. □

4.4 TREE TRANSDUCERS WITH REGULAR LOOK-AHEAD

Consider an F-transducer $\mathfrak{A} = (\Sigma, X, A, \Omega, Y, P, A')$. Take a tree $p = \sigma(p_1, \ldots, p_m) \in F_\Sigma(X)$ $(\sigma \in \Sigma_m, m > 0)$ and a derivation $\sigma(p_1, \ldots, p_m) \Rightarrow^* \sigma(a_1q_1, \ldots, a_mq_m)$ $(a_i \in A, q_i \in F_\Omega(Y), p_i \Rightarrow^* a_iq_i, i = 1, \ldots, m)$. Then, knowing the states $a_1, \ldots, a_m$, our transducer can decide which production $\sigma(a_1,\ldots,a_m) \to q$ to apply next. In other words, after inspecting the properties of the subtrees $p_1,\ldots,p_m$, the F-transducer $\mathfrak{A}$ can select the production to be applied in the next step of the translation of p. Moreover, these properties of subtrees are regular in the sense that $\mathrm{dom}(\tau_{\mathfrak{A}(a_i)})$ is a regular forest for each $i(=1,\ldots,m)$. Obviously, R-transducers lack this possibility. This observation leads to the idea to provide R-transducers with regular look-ahead as follows.

Definition 4.4.1 A *root-to-frontier tree transducer with regular look-ahead* ($\mathrm{R_R}$-*transducer*) is a system $\mathfrak{A} = (\Sigma, X, A, \Omega, Y, P, A')$, where

(1) Σ, X, A, Ω, Y and A' have the same meanings as in Definition 4.1.4,

(2) P is a finite set of *productions* (or *rewriting rules*) of the form $(p \to q, D)$, where $p \to q$ is an R-transducer production and D is a mapping of the set of all auxiliary variables occurring in p into $\mathrm{Rec}(\Sigma, X)$.

If p is of the form ax ($x \in X$) or $a\sigma$ with $\sigma \in \Sigma_0$, then the domain of D is empty. We write such rules generally as $ax \to q$ and $a\sigma \to q$, respectively. Moreover, for any $a \in A$, we put $\mathfrak{A}(a) = (\Sigma, X, A, \Omega, Y, P, a)$.

Definition 4.4.2 Let $\mathfrak{A}$ be the $\mathrm{R_R}$-transducer of Definition 4.4.1. $\mathfrak{A}$ is called *deterministic* if the following conditions are satisfied:

(i) A' is a singleton.

(ii) If $(p_1 \to q_1, D_1)$ and $(p_2 \to q_2, D_2)$ are two productions in P with $p_1 = p_2$, and $q_1 \neq q_2$, then there exists an i ($1 \le i \le m$) such that $D_1(\xi_i) \cap D_2(\xi_i) = \emptyset$, where m is the number of auxiliary variables in $p_1 (= p_2)$.

Linear and *nondeleting* $\mathrm{R_R}$-transducers are defined in the same way as their R-transducer counterparts.

Definition 4.4.3 Take an $\mathrm{R_R}$-transducer $\mathfrak{A} = (\Sigma, X, A, \Omega, Y, P, A')$, and let $p, q \in F_\Omega[Y \cup AF_\Sigma(X)]$ be two trees. It is said that p *directly derives* q in $\mathfrak{A}$ (in notation, $p \Rightarrow_{\mathfrak{A}} q$) if q can be obtained from p

(i) by replacing an occurrence of an ax ($a \in A, x \in X$) in p by the right side $\overline{q}$ of a production $ax \to \overline{q}$ in P, or

(ii) by replacing an occurrence of a subtree $a\sigma(p_1,\ldots,p_m)$ ($a \in A, \sigma \in \Sigma_m, m \ge 0, p_1,\ldots,p_m \in F_\Sigma(X)$) in p by $\overline{q}(p_1,\ldots,p_m)$, where $(a\sigma \to \overline{q}, D)$ is in P and $p_i \in D(\xi_i)$ for each $i(=1,\ldots,\ m)$.

A sequence

$$p = p_0 \Rightarrow_{\mathfrak{A}} p_1 \Rightarrow_{\mathfrak{A}} \ldots \Rightarrow_{\mathfrak{A}} p_k = q \ (k \ge 0)$$

obtained by consecutive applications of direct derivations is a *derivation* of q from p in $\mathfrak{A}$. When such a derivation exists, we write $p \Rightarrow^*_{\mathfrak{A}} q$. Again, this notation will also be used to indicate a certain derivation.

If there is no danger of confusion, then we generally omit $\mathfrak{A}$ in $\Rightarrow_{\mathfrak{A}}$ and $\Rightarrow^*_{\mathfrak{A}}$.

According to Definition 4.4.3, the difference between derivations in R-transducers and R_R-transducers is that in case of an R_R-transducer $\mathfrak{A}$ a production $a\sigma \to q$ can be applied to a tree $a\sigma(p_1, \dots, p_m)$ if and only if there is a production $(a\sigma \to q, D)$ of $\mathfrak{A}$ such that each subtree p_i $(1 \le i \le m)$ is in the recognizable forest $D(\xi_i)$.

Definition 4.4.4 Let $\mathfrak{A} = (\Sigma, X, A, \Omega, Y, P, A')$ be an R_R-transducer. Then the relation

$$\tau_{\mathfrak{A}} = \{(p, q) \mid p \in F_\Sigma(X), q \in F_\Omega(Y),\ ap \Rightarrow^* q \text{ for some } a \in A'\}$$

is called the *transformation induced* by $\mathfrak{A}$.

A relation τ is an R_R-*transformation* if there exists an R_R-transducer $\mathfrak{A}$ such that $\tau = \tau_{\mathfrak{A}}$.

Linear, nondeleting and *deterministic* R_R-*transformations* are defined in an obvious way.

The class of all R_R-transformations will be denoted by $\mathcal{R}_R$.

Let us note that there exists a recursive definition of transformations induced by R_R-transducers. This can be obtained by an obvious modification of the corresponding definition of transformations induced by R-transducers.

Moreover, for R_R-transducers the notion of a reordering of direct derivations can be defined in the same way as in the case of R-transducers. Furthermore, the remarks concerning different forms of derivations in R-transducers are valid for R_R-transducers, too.

To illustrate the concepts of R_R-transducers and R_R-transformations, consider

Example 4.4.5 Let $X = \{x\}$ and $\Sigma = \Sigma_1 \cup \Sigma_2$, where $\Sigma_i = \{\sigma_i\}$ $(i = 1, 2)$. Take the forests $T_1 = \{\sigma_1(x)\}^{*x}$ and $T_2 = \{\sigma_1(x)\}$. Let $\mathfrak{A} = (\Sigma, X, \{a_0, a_1\}, \Omega, Y, P, a_0)$ be the R_R-transducer where $\Omega = \Omega_1 = \{\omega\}, Y = \{y\}$ and P consists of the productions

$$(a_0\sigma_2 \to \omega(a_1\xi_1), D_1) \ \ (D_1(\xi_1) = T_1, D_1(\xi_2) = T_2),$$

$$(a_1\sigma_1 \to \omega(a_1\xi_1), D_2) \ \ (D_2(\xi_1) = T_1),$$

$$a_1 x \to y.$$

Then $\tau_{\mathfrak{A}} = \{(\sigma_2(\sigma_1^n(x), \sigma_1(x)), \omega^{n+1}(y)) \mid n = 0, 1, \dots\}$. Observe that (without regular look-ahead) the corresponding R-transducer would induce the transformation $\{(\sigma_2(\sigma_1^n(x), p), \omega^{n+1}(y)) \mid p \in F_\Sigma(X), n = 0, 1, \dots\}$. □

Obviously R-transducers are special cases of R_R-transducers. On the other hand, R_R-transducers can restrict the domain of possible subtrees of input trees even if these are deleted. In fact, no R-transducer could induce the $\tau_{\mathfrak{A}}$ considered in the above example. Assume that such an R-transducer

$$\mathfrak{B} = (\Sigma, X, B, \Omega, Y, P', B')$$

exists. Then for every $n (\ge 0)$, the production applied first in a derivation $b_0\sigma_2(\sigma_1^n(x), \sigma_1(x)) \Rightarrow_{\mathfrak{B}}^* \omega^{n+1}(y)$ $(b_0 \in B')$ should be of the form

(i) $b_0\sigma_2 \to q(b\xi_1)$ or

(ii) $b_0\sigma_2 \to q(b\xi_2)$ $(b \in B, q = \omega^m(\xi_1), m \geq 0)$.

Let k be the maximum of the heights of right sides of productions from P' and $n \geq 3k$. Then the considered production should be of the form (i). But in this case all pairs $(\sigma_2(\sigma_1^n(x), p), \omega^{n+1}(y))$ $(p \in F_\Sigma(X))$ are in $\tau_{\mathfrak{B}}$, which is a contradiction. □

Theorem 4.4.6 *The following inclusions hold:*

(i) $\mathcal{R}_R \subseteq \mathcal{DF}\mathrm{rel} \circ \mathcal{R}$,

(ii) $\mathcal{LR}_R \subseteq \mathcal{DF}\mathrm{rel} \circ \mathcal{LR}$,

(iii) $\mathcal{DR}_R \subseteq \mathcal{DF}\mathrm{rel} \circ \mathcal{DR}$,

(iv) $\mathcal{LDR}_R \subseteq \mathcal{DF}\mathrm{rel} \circ \mathcal{LDR}$.

Proof. Let $\mathfrak{A} = (\Sigma, X, A, \Delta, Y, P, A')$ be an arbitrary $\mathrm{R_R}$-transducer. Let $T_1, \ldots, T_k$ $(\subseteq F_\Sigma(X))$ be all regular forests which appear as images in the D-mappings of the productions in P. Denote by V the set of all k-dimensional vectors with components 0 or 1. Now take a ranked alphabet Ω, where $\Omega_0 = \Sigma_0$, and for each $m > 0, \Omega_m = \Sigma_m \times V^m$. Thus, the elements from Ω_m $(m > 0)$ can be given in the form $(\sigma, (\mathbf{v}_1, \ldots, \mathbf{v}_m))$, where $\sigma \in \Sigma_m$ and $\mathbf{v}_1, \ldots, \mathbf{v}_m \in V$.

Let $\mathbf{A}_i = (\mathcal{A}_i, \alpha_i, A_i')$ be ΣX-recognizers with $\mathcal{A}_i = (A_i, \Sigma)$ and $T(\mathbf{A}_i) = T_i$ $(i = 1, \ldots, k)$. We introduce the F-transducer $\mathfrak{B} = (\Sigma, X, B, \Omega, X, P', B')$ where $B = B' = A_1 \times \ldots \times A_k$ and P' consists of the following productions:

(I) $x \to (x\alpha_1, \ldots, x\alpha_k)x$ $(x \in X)$,

(II) $\sigma \to (\sigma^{\mathcal{A}_1}, \ldots, \sigma^{\mathcal{A}_k})\sigma$ $(\sigma \in \Sigma_0)$,

(III) $\sigma(\mathbf{a}_1, \ldots, \mathbf{a}_m) \to \mathbf{a}(\sigma, (\mathbf{v}_1, \ldots, \mathbf{v}_m))(\xi_1, \ldots, \xi_m)$

$$(\sigma \in \Sigma_m, m > 0; \mathbf{a}, \mathbf{a}_i \in B, \mathbf{v}_i \in V, i = 1, \ldots, m),$$

where
$$\mathbf{a} = (\sigma^{\mathcal{A}_1}(a_{1_1}, \ldots, a_{m_1}), \ldots, \sigma^{\mathcal{A}_k}(a_{1_k}, \ldots, a_{m_k}))$$
and $v_{i_j} = 1$ iff $a_{i_j} \in A_j'$. Obviously, $\mathfrak{B}$ is a deterministic F-relabeling.

One can easily show that $\mathfrak{B}$ relabels every ΣX-tree p in the following way:

(α) if $p \in X \cup \Sigma_0$, then $\tau_{\mathfrak{B}}(p) = p$,

(β) if $p = \sigma(p_1, \ldots, p_m)$ $(\sigma \in \Sigma_m, m > 0)$ then $\tau_{\mathfrak{B}}(p) = (\sigma, (\mathbf{v}_1, \ldots, \mathbf{v}_m))(\tau_{\mathfrak{B}}(p_1), \ldots, \tau_{\mathfrak{B}}(p_m))$, where $v_{i_j} = 1$ iff $p_i \in T_j$ $(1 \leq i \leq m, 1 \leq j \leq k)$.

Next construct the R-transducer $\mathfrak{C} = (\Omega, X, A, \Delta, Y, P'', A')$ where P'' consists of the productions below:

(α') $ap \to r$ ($a \in A, p \in X \cup \Omega_0, r \in F_\Delta(Y)$) is in P'' iff it is in P,

(β') $a(\sigma, (\mathbf{v}_1, \ldots, \mathbf{v}_m)) \to r$ ($a \in A; \sigma \in \Sigma_m, m > 0; \mathbf{v}_i \in V, i = 1, \ldots, m; r \in F_\Delta[Y \cup A\Xi_m]$) is in P'' iff $(\sigma, (\mathbf{v}_1, \ldots, \mathbf{v}_m))$ occurs in a tree $\tau_{\mathfrak{B}}(p)$ ($p \in F_\Sigma(X)$) and P contains a production $(a\sigma \to r, D)$ such that $v_{i_j} = 1$ whenever $D(\xi_i) = T_j$ ($1 \leq i \leq m, 1 \leq j \leq k$).

In order to prove $\tau_{\mathfrak{A}} = \tau_{\mathfrak{B}} \circ \tau_{\mathfrak{C}}$ it is enough to show that for arbitrary $a \in A, p \in F_\Sigma(X)$ and $r \in F_\Delta(Y)$ the equivalence

$$ap \Rightarrow^*_{\mathfrak{A}} r \iff a\tau_{\mathfrak{B}}(p) \Rightarrow^*_{\mathfrak{C}} r$$

holds. This can be carried out by induction on $\mathrm{hg}(p)$.

It is also easy to show that $\mathfrak{C}$ is deterministic (linear) if $\mathfrak{A}$ is deterministic (linear). □

Theorem 4.4.6 (iii) shows that $\mathrm{DR_R}$-transducers induce (partial) mappings.

Next we show that $\mathcal{R}_R$ is closed under certain special F-transformations.

Theorem 4.4.7 *The following inclusions hold:*

(i) $\mathcal{R}_R \circ \mathcal{LF} \subseteq \mathcal{R}_R$,

(ii) $\mathcal{DR}_R \circ \mathcal{DLF} \subseteq \mathcal{DR}_R$,

(iii) $\mathcal{DR}_R \circ \mathcal{DLR} \subseteq \mathcal{DR}_R$,

(iv) $\mathcal{DR}_R \circ \mathcal{H} \subseteq \mathcal{DR}_R$.

Proof. Let $\mathfrak{A} = (\Sigma, X, A, \Omega, Y, P, A')$ be an $\mathrm{R_R}$-transducer, and take an LF- transducer $\mathfrak{B} = (\Omega, Y, B, \Delta, Z, P', B')$.

We want to treat cases (i) and (ii) together. Since the set of initial states of a $\mathrm{DR_R}$-transducer should be a singleton we shall use the LF-transducer $\overline{\mathfrak{B}} = (\Omega, Y, \overline{B}, \Delta, Z, \overline{P}', b_0)$ instead of $\mathfrak{B}$, where $\overline{B} = B \cup b_0$ ($b_0 \notin B$) and $\overline{P}'$ is obtained by enlarging P' by the following productions: if $y \to bq$ ($y \in Y$), is in P' and $b \in B'$, then $y \to b_0 q$ is in $\overline{P}'$. Similarly, if $\sigma(b_1, \ldots, b_m) \to bq$ ($\sigma \in \Sigma_m, m \geq 0$) is in P' and $b \in B'$ then the production $\sigma(b_1, \ldots, b_m) \to b_0 q$ is in $\overline{P}'$. It is obvious that $\tau_{\overline{\mathfrak{B}}} = \tau_{\mathfrak{B}}$.

Construct the $\mathrm{R_R}$-transducer $\mathfrak{C} = (\Sigma, X, A \times \overline{B}, \Delta, Z, P'', A' \times \{b_0\})$, where P'' is given as follows:

(I) $(a, b)p \to r$ ($a \in A, b \in \overline{B}, p \in X \cup \Sigma_0, r \in F_\Delta(Z)$) is in P'' iff there exists a production $ap \to q$ in P such that $q \Rightarrow^*_{\overline{\mathfrak{B}}} br$ holds.

(II) Assume that the production $(a\sigma \to q(\mathbf{a}_1\xi_1^{n_1}, \ldots, \mathbf{a}_m\xi_m^{n_m}), D)$ ($a \in A$; $\sigma \in \Sigma_m, m > 0; \mathbf{a}_i \in A^{n_i}, i = 1, \ldots, m; n_1 + \ldots + n_m = n, q \in \hat{F}_\Omega(Y \cup \Xi_n)$) is in P and that there is a derivation $q(\mathbf{b}_1\boldsymbol{\xi}_1, \ldots, \mathbf{b}_m\boldsymbol{\xi}_m) \Rightarrow^*_{\overline{\mathfrak{B}}} br(\boldsymbol{\xi}_1, \ldots, \boldsymbol{\xi}_m)$ with $b \in B; \mathbf{b}_i \in B^{n_i}, \boldsymbol{\xi}_i \in \Xi^{n_i}, \xi_{i_j} = \xi_{n_1+\ldots+n_{i-1}+j}, 1 \leq j \leq n_i, i = 1, \ldots, m$ and $r \in F_\Delta(Z \cup \Xi_n)$. Then P'' contains the production $((a, b)\sigma \to r(\mathbf{a}_1\mathbf{b}_1\xi_1^{n_1}, \ldots, \mathbf{a}_m\mathbf{b}_m\xi_m^{n_m}), D')$, where $D'(\xi_i) = \bigcap(\tau^{-1}_{\mathfrak{A}(a_{i_j})}(\mathrm{dom}(\tau_{\mathfrak{B}(b_{i_j})})) \mid j = 1, \ldots, n_i) \cap D(\xi_i)$ ($i = 1, \ldots, m$). If $b \in B'$, then $((a, b_0)\sigma \to r(\mathbf{a}_1\mathbf{b}_1\xi_1^{n_1}, \ldots, \mathbf{a}_m\mathbf{b}_m\xi_m^{n_m}), D')$ is also in P''.

By Corollary 4.3.17, the domain of an R-transformation is regular. Moreover, also by Corollary 4.3.17, the inverse of an R-transformation preserves regularity. Thus, by Corollary 4.2.9 and Theorems 4.4.6 and 2.4.2, $D'(\xi_i)$ $(1 \le i \le m)$ is regular.

In order to show $\tau_{\mathfrak{A}} \circ \tau_{\overline{\mathfrak{B}}} = \tau_{\mathfrak{C}}$ it is enough to prove that for all $(a, b) \in A \times \overline{B}, p \in F_\Sigma(X)$ and $r \in F_\Delta(Z)$ the equivalence

$$(a, b)p \Rightarrow^*_{\mathfrak{C}} r \iff (\exists q \in F_\Omega(Y))(ap \Rightarrow^*_{\overline{\mathfrak{A}}} q \wedge q \Rightarrow^*_{\mathfrak{B}} br)$$

holds. This can be done by induction on $\mathrm{hg}(p)$.

One can easily check that if $\mathfrak{A}$ and $\mathfrak{B}$ are deterministic, then so is $\mathfrak{C}$. Thus, (i) and (ii) are valid.

For (iii), take a $\mathrm{DR_R}$-transducer $\mathfrak{A} = (\Sigma, X, A, \Omega, Y, P, a_0)$ and a DLR-transducer $\mathfrak{B} = (\Omega, Y, B, \Delta, Z, P', b_0)$.

Consider the $\mathrm{R_R}$-transducer $\mathfrak{C} = (\Sigma, X, A \times B, \Omega, Y, P'', (a_0, b_0))$, where P'' is given in the following way:

(I) If $ap \to q$ $(a \in A, p \in X \cup \Sigma_0, q \in F_\Omega(Y))$ is in P and $bq \Rightarrow^*_{\mathfrak{B}} r$ $(b \in B, r \in F_\Delta(Z))$ holds, then $(a, b)p \to r$ is in P''.

(II) Suppose that $(a\sigma \to q(\mathbf{a}_1\xi_1^{n_1}, \ldots, \mathbf{a}_m\xi_m^{n_m}), D)$ $(a \in A, \sigma \in \Sigma_m, m > 0, \mathbf{a}_i \in A^{n_i}, i = 1, \ldots, m, n_1 + \ldots + n_m = n, q \in \hat{F}_\Omega(Y \cup \Xi_n))$ is in P and there is a derivation $bq \Rightarrow^*_{\mathfrak{B}} r(\mathbf{b}_1\boldsymbol{\xi}_1, \ldots, \mathbf{b}_m\boldsymbol{\xi}_m)$ with $b \in B, \mathbf{b}_i \in B^{n_i}, \boldsymbol{\xi}_i \in \Xi^{n_i}, \xi_{i_j} = \xi_{n_1+\ldots+n_{i-1}+j}, 1 \le j \le n_i, i = 1, \ldots, m$ and $r \in F_\Delta(Z \cup \Xi_n)$. Then the production

$$((a, b) \to r(\mathbf{a}_1\mathbf{b}_1\xi_1^{n_1}, \ldots, \mathbf{a}_m\mathbf{b}_m\xi_m^{n_m}), D')$$

is in P'', where for every $i(= 1, \ldots, m)$,

$$D'(\xi_i) = \bigcap(\mathrm{dom}(\tau_{\mathfrak{A}(a_{i_j})}) \mid \xi_{i_j}\ (1 \le j \le n_i) \text{ does not occur in } r) \cap D(\xi_i).$$

Obviously, $\mathfrak{C}$ is a $\mathrm{DR_R}$-transducer. Moreover, for all $a \in A, b \in B, p \in F_\Sigma(X)$ and $r \in F_\Delta(Z)$ the equivalence

$$(a, b)p \Rightarrow^*_{\mathfrak{C}} r \iff (\exists q \in F_\Omega(Y))(ap \Rightarrow^*_{\mathfrak{A}} q \wedge bq \Rightarrow^*_{\mathfrak{B}} r)$$

holds. This can be proved by induction on $\mathrm{hg}(p)$. Therefore, $\tau_{\mathfrak{C}} = \tau_{\mathfrak{A}} \circ \tau_{\mathfrak{B}}$. Thus we have shown that $\mathcal{DR}_R \circ \mathcal{DLR} \subseteq \mathcal{DR}_R$.

To show (iv), let $\mathfrak{A} = (\Sigma, X, A, \Omega, Y, P, a_0)$ be a $\mathrm{DR_R}$-transducer and $\mathfrak{B} = (\Omega, Y, \{b_0\}, \Delta, Z, P', b_0)$ an HF-transducer.

Construct an $\mathrm{R_R}$-transducer $\mathfrak{C} = (\Sigma, X, A, \Delta, Z, P'', a_0)$, where P'' is given as follows:

(I) $ap \to r$ $(a \in A, p \in \Sigma_0 \cup X, r \in F_\Delta(Z))$ is in P'' iff there is a production $ap \to q$ in P such that $q \Rightarrow^*_{\mathfrak{B}} b_0r$ holds.

(II) Suppose that the production $(a\sigma \to q(\mathbf{a}_1\xi_1^{n_1}, \ldots, \mathbf{a}_m\xi_m^{n_m}), D)$ $(a \in A, \sigma \in \Sigma_m, m > 0, \mathbf{a}_i \in A^{n_i}, i = 1, \ldots, m, n_1 + \ldots + n_m = n, q \in \hat{F}_\Omega(Y \cup \Xi_n))$ is in P and there is a derivation $q(b_0^{n_1}\boldsymbol{\xi}_1, \dots, b_0^{n_m}\boldsymbol{\xi}_m) \Rightarrow_{\mathfrak{B}}^* b_0 r(\xi_{1_1}^{k_{11}}, \dots, \xi_{1_{n_1}}^{k_{1n_1}}, \dots, \xi_{m_1}^{k_{m1}}, \dots, \xi_{m_{n_m}}^{k_{mn_m}})$ where $\boldsymbol{\xi}_i \in \Xi^{n_i}, \xi_{i_j} = \xi_{n_1+\dots+n_{i-1}+j}, 1 \le j \le n_i, i = 1, \dots, m, k_{11} + \dots + k_{1n_1} + \dots + k_{m1} + \dots + k_{mn_m} = k, r \in \hat{F}_\Delta(Z \cup \Xi_k)$. Then the production

$$(a\sigma \to r(a_{1_1}^{k_{11}}\xi_1^{k_{11}}, \dots, a_{1n_1}^{k_{1n_1}}\xi_1^{k_{1n_1}}, \dots, a_{m_1}^{k_{m1}}\xi_m^{k_{m1}}, \dots, a_{mn_m}^{k_{mn_m}}\xi_m^{k_{mn_m}}), D')$$

is in P'', where for every $i(= 1, \dots, m), D'(\xi_i) = \bigcap(\mathrm{dom}(\tau_{\mathfrak{A}(a_{i_j})}) \mid \xi_{i_j}$ occurs in q but it does not occur in $r) \cap D(\xi_i)$.

Using a similar argument as in the proof of (ii), we get that $D'(\xi_i)$ is a regular forest. It is obvious that $\mathfrak{C}$ is deterministic.

Finally, to show $\tau_{\mathfrak{A}} \circ \tau_{\mathfrak{B}} = \tau_{\mathfrak{C}}$ it is enough to prove that for all $a \in A, p \in F_\Sigma(X)$ and $r \in F_\Delta(Z)$ the equivalence

$$ap \Rightarrow_{\mathfrak{C}}^* r \iff p\tau_{\mathfrak{A}(a)} \Rightarrow_{\mathfrak{B}}^* b_0 r$$

holds. This can be done by induction on $\mathrm{hg}(p)$. □

From Theorem 4.4.7 we get

Corollary 4.4.8 *The inclusions*

(i) $\mathcal{R}_R \circ \mathcal{F}\mathrm{rel} \subseteq \mathcal{R}_R$,

(ii) $\mathcal{DR}_R \circ \mathcal{DF}\mathrm{rel} \subseteq \mathcal{DR}_R$, *and*

(iii) $\mathcal{DR}_R \circ \mathcal{DR}\mathrm{rel} \subseteq \mathcal{DR}_R$

hold. □

Next we show that the classes of LF-transformations and $\mathrm{LR_R}$-transformations coincide.

Theorem 4.4.9 $\mathcal{LR}_R = \mathcal{LF}$.

Proof. Since $\mathcal{DF}\mathrm{rel} \subseteq \mathcal{LNF}$, the inclusion $\mathcal{LR}_R \subseteq \mathcal{LF}$ is implied by Theorems 4.4.6 (ii), 4.2.8 and 4.3.6 (ii).

In order to prove $\mathcal{LF} \subseteq \mathcal{LR}_R$, take an LF-transducer $\mathfrak{A} = (\Sigma, X, A, \Omega, Y, P, A')$. Consider the $\mathrm{R_R}$-transducer $\mathfrak{B} = (\Sigma, X, A, \Omega, Y, P', A')$, where P' is given as follows:

(i) If $x \to aq$ ($x \in X, a \in A, q \in F_\Omega(Y)$) is in P, then $ax \to q$ is in P'.

(ii) If $\sigma(a_1, \dots, a_m) \to aq$ ($\sigma \in \Sigma_m, m \ge 0, a_1, \dots, a_m, a \in A, q \in F_\Omega(Y \cup \Xi_m)$) is in P, then $(a\sigma \to q(a_1\xi_1, \dots, a_m\xi_m), D)$ is in P', where for every $i(= 1, \dots, m)$,

$$D(\xi_i) = \begin{cases} \mathrm{dom}(\tau_{\mathfrak{A}(a_i)}) \text{ if } \xi_i \text{ does not occur in } q, \\ F_\Sigma(X) \text{ otherwise.} \end{cases}$$

Obviously, $\mathfrak{B}$ is an $\mathrm{LR_R}$-transducer. To prove $\tau_{\mathfrak{A}} = \tau_{\mathfrak{B}}$ it is enough to show that for each $a \in A, p \in F_\Sigma(X)$ and $q \in F_\Omega(Y)$ the equivalence

$$p \Rightarrow_{\mathfrak{A}}^* aq \iff ap \Rightarrow_{\mathfrak{B}}^* q$$

holds. Again, we omit the straightforward inductive proof. □

In the proof of the above theorem we used look-ahead to ensure that the $\mathrm{LR_R}$-transducer will not transform any tree which contains a subtree for which the LF-transducer has no transform but which it would later delete.

From Theorem 4.4.9, by Theorem 4.3.6 (i), we get

Corollary 4.4.10 *$\mathcal{LR}_R$ is closed under composition.* □

Next we show that $\mathcal{R}_R$ is closed under $\mathrm{LR_R}$-transformations and $\mathcal{DR}_R$ is closed under composition.

Theorem 4.4.11 *The following equations hold:*

(i) $\mathcal{R}_R \circ \mathcal{LR}_R = \mathcal{R}_R$,

(ii) $\mathcal{DR}_R \circ \mathcal{DR}_R = \mathcal{DR}_R$.

Proof. $\mathcal{R}_R \circ \mathcal{LR}_R = \mathcal{R}_R$ follows from Theorem 4.4.7 by Theorem 4.4.9.

Since, for each Σ and X, the identity mapping on $F_\Sigma(X)$ is in $\mathcal{DR}_R$, in order to prove (ii) it is enough to show the validity of the inclusion $\mathcal{DR}_R \circ \mathcal{DR}_R \subseteq \mathcal{DR}_R$.

By Theorem 4.4.6 (iii), the inclusion $\mathcal{DR}_R \circ \mathcal{DR}_R \subseteq \mathcal{DR}_R \circ \mathcal{DF}\mathrm{rel} \circ \mathcal{DR}$ holds from which, using Corollary 4.4.8 (ii), we get $\mathcal{DR}_R \circ \mathcal{DR}_R \subseteq \mathcal{DR}_R \circ \mathcal{DR}$. This latter inclusion, by the proof of Lemma 4.3.10, implies $\mathcal{DR}_R \circ \mathcal{DR}_R \subseteq \mathcal{DR}_R \circ \mathcal{H} \circ \mathcal{DLR}$. Now, using Theorem 4.4.7 (iv), we get $\mathcal{DR}_R \circ \mathcal{DR}_R \subseteq \mathcal{DR}_R \circ \mathcal{DLR}$, from which by Theorem 4.4.7 (iii), we arrive at the desired inclusion $\mathcal{DR}_R \circ \mathcal{DR}_R \subseteq \mathcal{DR}_R$. □

To end this section we prove the analogue of Theorem 4.3.12.

Theorem 4.4.12 $\mathcal{R}_R = \mathcal{H} \circ \mathcal{LR}_R$.

Proof. The inclusion $\mathcal{H} \circ \mathcal{LR}_R \subseteq \mathcal{R}_R$ directly follows from Theorem 4.4.11 (i). To show $\mathcal{R}_R \subseteq \mathcal{H} \circ \mathcal{LR}_R$, consider an $\mathrm{R_R}$-transducer $\mathfrak{A} = (\Sigma, X, A, \Delta, Z, P, A')$. Omit regular look-ahead in $\mathfrak{A}$ and for the resulting R-transducer consider the H-transducer $\mathfrak{B}$ and LR-transducer $\mathfrak{C}$ given in the proof of Lemma 4.3.10. Now it is impossible to provide $\mathfrak{C}$ with a suitable regular look-ahead in an obvious way since H-transducers do not preserve regularity. We shall solve this problem in the following way.

Take the tree homomorphism $h : F_\Omega(X) \to F_\Sigma(X)$ given as follows:

(i) $h_X(x) = x \ (x \in X)$,

(ii) $h_{mk}(\sigma') = \sigma(\xi_1, \xi_{k+1}, \ldots, \xi_{(m-1)k+1}) \ (\sigma \in \Sigma_m, m \geq 0)$.

One can easily verify that for every $p \in F_\Sigma(X)$ the equality $h(\tau_{\mathfrak{B}}(p)) = p$ holds, i.e., $h(p') = p$ (for p', see the proof of Lemma 4.3.10).

Now replacing each production $a\sigma' \to r(\mathbf{a}_1\boldsymbol{\xi}_1, \ldots, \mathbf{a}_m\boldsymbol{\xi}_m)$ $(\sigma \in \Sigma_m, m > 0, (a\sigma \to r(\mathbf{a}_1\xi_1^{n_1}, \ldots, \mathbf{a}_m\xi_m^{n_m}), D) \in P)$ in P'' by $(a\sigma' \to r(\mathbf{a}_1\boldsymbol{\xi}_1, \ldots, \mathbf{a}_m\boldsymbol{\xi}_m), D')$, where $D'(\xi_{i_j}) = h^{-1}(D(\xi_i))$ $(i = 1, \ldots, m, j = 1, \ldots, k)$, from $\mathfrak{C}$ we get an $\mathrm{LR_R}$-transducer since, by Theorem 2.4.18, h^{-1} preserves recognizability. Let us denote the resulting $\mathrm{LR_R}$-transducer also by $\mathfrak{C}$.

Using tree induction, it is easy to prove that $\tau_{\mathfrak{A}} = \tau_{\mathfrak{B}} \circ \tau_{\mathfrak{C}}$. □

4.5 GENERALIZED SYNTAX DIRECTED TRANSLATORS

In studying certain properties of tree transformations it is technically useful to consider systems that translate trees into strings. Such systems are also of interest as mathematical models of syntax directed translations of context-free languages.

Definition 4.5.1 A *generalized syntax directed translator* (GSDT) is a system $\mathfrak{A} = (\Sigma, X, A, Y, P, A')$, where

(1) Σ is a ranked alphabet,

(2) A is a unary ranked alphabet (the *state set*),

(3) X and Y are alphabets,

(4) $A' \subseteq A$ is the set of *initial states*, and

(5) P is a finite set of *productions* (or *rewriting rules*) of the following two types:

  (i) $ax \to w$ $(a \in A, x \in X, w \in Y^*)$,

  (ii) $a\sigma \to w$ $(a \in A, \sigma \in \Sigma_m, m \geq 0, w \in (Y \cup A\Xi_m)^*)$. (Here $A\Xi_m$ is treated as an alphabet; the elements of it are the trees of the form $a\xi_i$ with $a \in A$ and $\xi_i \in \Xi_m$.)

For $ap \to w$ we shall use the notation (ap, w), too. Moreover, for any $a \in A$, we put $\mathfrak{A}(a) = (\Sigma, X, A, Y, P, a)$.

Next we define translations induced by a GSDT $\mathfrak{A}$. To this end, we associate with each $a \in A$ and $p \in F_\Sigma(X)$ a subset $p\tau_{\mathfrak{A},a}$ as follows:

(i) if $p \in (X \cup \Sigma_0)$, then $p\tau_{\mathfrak{A},a} = \{w \mid (ap, w) \in P\}$;

(ii) if $p = \sigma(p_1, \ldots, p_m)$ $(\sigma \in \Sigma_m, m > 0)$, then for all

$$(a\sigma, w_1 a_{i(1)}\xi_{i_1} w_2 \ldots w_k a_{i(k)}\xi_{i_k} w_{k+1}) \in P$$

$(1 \leq i_j \leq m, j = 1, \ldots, k, w_1, \ldots, w_{k+1} \in Y^*)$ and $v_{i_j} \in p_{i_j}\tau_{\mathfrak{A},a_{i(j)}}$ $(j = 1, \ldots, k)$ the word $w_1 v_{i_1} w_2 \ldots w_k v_{i_k} w_{k+1}$ is in $p\tau_{\mathfrak{A},a}$, and

(iii) nothing is in any $p\tau_{\mathfrak{A},a}$ unless this follows from (i) and (ii).

Definition 4.5.2 Let $\mathfrak{A} = (\Sigma, X, A, Y, P, A')$ be a GSDT. Then the *translation induced* by $\mathfrak{A}$ is the relation $\tau_{\mathfrak{A}} = \{(p, w) \mid p \in F_\Sigma(X), w \in Y^*, w \in p\tau_{\mathfrak{A},a}$ for some $a \in A'\}$.

The class of all translations induced by GSDTs will be denoted by $\mathcal{G}$.

For translations induced by GSDTs we give another definition showing how a translation is carried out step by step.

Let $\mathfrak{A}$ be the GSDT of Definition 4.5.1. Take two words $v, w \in (Y \cup AF_\Sigma(X \cup \Xi))^*$. (Here again each element of $AF_\Sigma(X \cup \Xi)$ is considered a symbol, i.e., we ignore the fact that these elements are composed of simpler objects.) We say that v *directly derives* w in $\mathfrak{A}$, and write $v \Rightarrow_{\mathfrak{A}} w$, if w can be obtained from v by

(i) replacing an occurrence of ax $(a \in A, x \in X)$ in v by the right side $\overline{w}$ of a production $ax \to \overline{w}$ from P, or

(ii) replacing an occurrence of an $a\sigma(p_1, \ldots, p_m)$ $(a \in A, \sigma \in \Sigma_m, m \geq 0, p_1, \ldots, p_m \in F_\Sigma(X \cup \Xi))$ in v by $w_1 a_{i(1)} p_{i_1} w_2 \ldots w_k a_{i(k)} p_{i_k} w_{k+1}$ where

$$a\sigma \to w_1 a_{i(1)} \xi_{i_1} w_2 \ldots w_k a_{i(k)} \xi_{i_k} w_{k+1} \ (1 \leq i_j \leq m, j = 1, \ldots, k, w_1, \ldots, w_{k+1} \in Y^*)$$

is a production in P.

Each application of a step (i) or (ii) is called a *direct derivation* in $\mathfrak{A}$. A sequence

$$v = v_0 \Rightarrow_{\mathfrak{A}} v_1 \Rightarrow_{\mathfrak{A}} \ldots \Rightarrow_{\mathfrak{A}} v_k = w \quad (k \geq 0, v_i \in (Y \cup AF_\Sigma(X \cup \Xi))^*, i = 0, \ldots, k)$$

of consecutive direct derivations is a *derivation* of w from v in $\mathfrak{A}$, and k is the *length* of this derivation. If w can be obtained from v by a derivation in $\mathfrak{A}$, then we write $v \Rightarrow_{\mathfrak{A}}^* w$. Thus $\Rightarrow_{\mathfrak{A}}^*$ is the reflexive-transitive closure of $\Rightarrow_{\mathfrak{A}}$. Again, we suppose that the notation $v \Rightarrow_{\mathfrak{A}}^* w$ implicitly includes a given derivation of w from v.

Using the notation $\Rightarrow_{\mathfrak{A}}^*$, the translation $\tau_{\mathfrak{A}}$ induced by a GSDT $\mathfrak{A} = (\Sigma, X, A, Y, P, A')$ can be given by

$$\tau_{\mathfrak{A}} = \{(p, w) \mid p \in F_\Sigma(X), w \in Y^*, ap \Rightarrow_{\mathfrak{A}}^* w \text{ for some } a \in A'\}.$$

The concept of a reordering of direct derivations in GSDTs can be defined in a similar way as in the case of an R-transducer. Moreover, different forms of derivations can be introduced in an obvious manner.

Deterministic, linear, totally defined and *nondeleting* GSDTs are defined in a natural way. Moreover, a one-state totally defined deterministic GSDT is a GSDH-*translator.* The translation induced by a GSDH-translator is called a *generalized syntax directed homomorphism* (GSD *homomorphism*). The class of all GSD homomorphisms will be denoted by $\mathcal{G}_{\text{hom}}$.

Example 4.5.3 Let $\mathfrak{B} = (\Sigma, \{x\}, \{b_0, b_1, b_2\}, \{y_1, y_2\}, P', b_0)$ be a GSDT, where $\Sigma = \Sigma_1 = \{\sigma\}$ and P' consists of the productions

$$b_0\sigma \to b_1\xi_1 b_2\xi_1,$$
$$b_1\sigma \to b_1\xi_1, \quad b_2\sigma \to b_2\xi_1,$$
$$b_1 x \to y_1, \quad b_2 x \to y_2.$$

Then $\mathfrak{B}$ is deterministic, totally defined and nondeleting, but it is not linear.

Take the tree $p = \sigma(\sigma(\sigma(x)))$ and the word $w = y_1y_2$. Moreover, consider the derivation

$$p \Rightarrow_{\mathfrak{B}} b_1\sigma(\sigma(x))b_2\sigma(\sigma(x)) \Rightarrow_{\mathfrak{B}} b_1\sigma(x)b_2\sigma(\sigma(x)) \Rightarrow_{\mathfrak{B}} b_1xb_2\sigma(\sigma(x)) \Rightarrow_{\mathfrak{B}}$$
$$y_1b_2\sigma(\sigma(x)) \Rightarrow_{\mathfrak{B}} y_1b_2\sigma(x) \Rightarrow_{\mathfrak{B}} y_1b_2x \Rightarrow_{\mathfrak{B}} y_1y_2 = w,$$

i.e., $\tau_{\mathfrak{B}}(p) = \mathrm{yd}(\tau_{\mathfrak{A}}(p))$, where $\mathfrak{A}$ is the R-transducer of Example 4.1.6. One can easily show that the previous equality holds for every $p \in F_\Sigma(\{x\})$. □

The above relation generally holds between GSDTs and R-transducers as it is shown by

Theorem 4.5.4 *For each* GSDT $\mathfrak{A} = (\Sigma, X, A, Y, P, A')$ *there exist a ranked alphabet* Ω *and an* R*-transducer* $\mathfrak{B} = (\Sigma, X, A, \Omega, Y, P', A')$ *such that* $\tau_{\mathfrak{A}} = \{(p, \mathrm{yd}(q)) \mid (p,q) \in \tau_{\mathfrak{B}}\}$. *Moreover, if* $\mathfrak{A}$ *is linear, deterministic, nondeleting or a* GSDH*-transducer, then* $\mathfrak{B}$ *can also be chosen, correspondingly, as a linear, deterministic, nondeleting or an* RH*-transducer.*

Conversely, for every R*-transducer* $\mathfrak{B}$ *there exists a* GSDT $\mathfrak{A}$ *such that* $\{(p, \mathrm{yd}(q)) \mid (p,q) \in \tau_{\mathfrak{B}}\} = \tau_{\mathfrak{A}}$. *If* $\mathfrak{B}$ *is, respectively linear, deterministic, nondeleting or an* RH*-transducer, then* $\mathfrak{A}$ *is linear, deterministic, nondeleting or a* GSDH*-translator.*

Proof. Let $\mathfrak{A} = (\Sigma, X, A, Y, P, A')$ be a GSDT. To define $\mathfrak{B}$, for each production $ap \to w$ ($a \in A, p \in X \cup \Sigma, w \in (Y \cup A\Xi)^*$) in P, let $\omega_{(ap,w)}$ be an operator with rank $|w|$. Let Ω be the resulting ranked alphabet. Moreover, P' is defined as follows:

(i) If $ap \to w$ ($a \in A, p \in X \cup \Sigma_0, w \in Y^*$) is in P and $|w| = k$, then the production $ap \to \omega_{(ap,w)}(q_1, \ldots, q_k)$ ($q_i \in Y, i = 1, \ldots, k$) with $\mathrm{yd}(\omega_{(ap,w)}(q_1, \ldots, q_k)) = w$ is in P'.

(ii) If $a\sigma \to w$ ($a \in A, \sigma \in \Sigma_m, m > 0, w \in (Y \cup A\Xi_m)^*$) is in P with $|w| = k$, then the production $a\sigma \to \omega_{(a\sigma,w)}(q_1, \ldots, q_k)$ ($q_i \in Y \cup A\Xi_m, i = 1, \ldots, k$) satisfying $\mathrm{yd}(\omega_{(a\sigma,w)}(q_1, \ldots, q_k)) = w$ is in P', where yd is taken over the frontier alphabet $Y \cup A\Xi_m$.

In order to prove $\tau_{\mathfrak{A}} = \{(p, \mathrm{yd}(q)) \mid (p,q) \in \tau_{\mathfrak{B}}\}$ it is enough to show that, for all $a \in A, p \in F_\Sigma(X)$ and $w \in Y^*$, the equivalence

$$ap \Rightarrow_{\mathfrak{A}}^* w \iff (\exists q \in F_\Omega(Y))(ap \Rightarrow_{\mathfrak{B}}^* q \wedge \mathrm{yd}(q) = w)$$

holds. This can be done in an obvious way by induction on $\mathrm{hg}(p)$.

It is also obvious from the construction of $\mathfrak{B}$ that the remaining conclusions of the first part of Theorem 4.5.4 hold, too.

Conversely, consider an R-transducer $\mathfrak{B} = (\Sigma, X, B, \Omega, Y, P', B')$. The productions of the desired GSDT $\mathfrak{A} = (\Sigma, X, B, Y, P, B')$ are given as follows:

(I) For all $b \in B, p \in X \cup \Sigma_0$ and $q \in F_\Omega(Y)$, if $bp \to q$ is in P', then $bp \to \mathrm{yd}(q)$ is in P.

(II) For all $b \in B, \sigma \in \Sigma_m$ $(m > 0)$ and $q \in F_\Omega(Y \cup B\Xi_m)$, if $b\sigma \to q$ is in P', then $b\sigma \to \mathrm{yd}(q)$ is in P, where yd is again taken over the alphabet $Y \cup B\Xi_m$.

To prove $\tau_{\mathfrak{A}} = \{(p, \mathrm{yd}(q)) \mid (p, q) \in \tau_{\mathfrak{B}}\}$ it is enough to show that the equivalence

$$bp \Rightarrow_{\mathfrak{A}}^* w \iff (\exists q \in F_\Omega(Y))(bp \Rightarrow_{\mathfrak{B}}^* q \wedge \mathrm{yd}(q) = w)$$

holds for arbitrary $b \in B, p \in F_\Sigma(X)$ and $w \in Y^*$. This can be carried out by induction on $\mathrm{hg}(p)$. Moreover, the remaining conclusions of the second part of Theorem 4.5.4 are obviously valid. □

4.6 SURFACE FORESTS

The images of regular forests under tree transformations are called surface forests. In this section we compare classes of surface forests belonging to different classes of tree transformations.

Definition 4.6.1 Let $\mathcal{K}$ be a class of tree transformations. A forest $S \subseteq F_\Omega(Y)$ is called a *$\mathcal{K}$-surface forest* if there exist a ranked alphabet Σ, a frontier alphabet X, a forest $R \in \mathrm{Rec}(\Sigma, X)$, and a $\mathcal{K}$-transformation $\tau \subseteq F_\Sigma(X) \times F_\Omega(Y)$ such that $S = R\tau$. The class of all $\mathcal{K}$-surface forests is denoted by $\mathrm{Surf}(\mathcal{K})$.

The following lemma is obvious.

Lemma 4.6.2 *If $\mathcal{K}$ is a class of tree transformations which contains all identity transformations, then* Rec *is included as a subclass in* $\mathrm{Surf}(\mathcal{K})$*.* □

Of course, this lemma applies to all of the classes of tree transformations which we have considered ($\mathcal{F}, \mathcal{R}, \mathcal{LF}, \mathcal{H}$ etc.).

Next we characterize F-transformations preserving regularity. For this we should introduce some more terminology.

Definition 4.6.3 A tree transformation $\tau \subseteq F_\Sigma(X) \times F_\Omega(Y)$ is said to *preserve regularity* if $R\tau \in \mathrm{Rec}(\Omega, Y)$ whenever $R \in \mathrm{Rec}(\Sigma, X)$. Moreover, a class $\mathcal{K}$ of tree transformations *preserves regularity* if every τ in $\mathcal{K}$ preserves regularity.

We say that an F-transducer $\mathfrak{A} = (\Sigma, X, A, \Omega, Y, P, A')$ is *connected* if for each $a \in A$ there are $p \in F_\Sigma(X)$ and $q \in F_\Omega(Y)$ such that $p \Rightarrow^* aq$ holds.

Definition 4.6.4 For each $p \in F_\Sigma(X \cup \Xi_n)$, $\mathrm{path}_i(p)$ $(1 \le i \le n)$ is given in the following way:

(i) if $p \in \Sigma_0 \cup X$, then $\mathrm{path}_i(p) = \emptyset$,

(ii) if $p = \xi_i$, then $\mathrm{path}_i(p) = \{e\}$,

(iii) if $p = \xi_j$ $(j \neq i)$, then $\mathrm{path}_i(p) = \emptyset$,

(iv) if $p = \sigma(p_1, \ldots, p_m)$ $(\sigma \in \Sigma_m, m > 0)$, then

$$\text{path}_i(p) = \{jw_j \mid w_j \in \text{path}_i(p_j), j = 1, \ldots, m\}.$$

Thus, $\text{path}_i(p)$ is a language over the alphabet $\{1, \ldots, m\}$, where m is the maximal integer with $\Sigma_m \neq \emptyset$.

Obviously, the elements of $\text{path}_i(p)$ describe paths leading from the root of p to a leaf labelled by ξ_i.

If $\text{path}_i(p)$ consists of a single word, then $l\big(\text{path}_i(p)\big)$ denotes the length of this word.

Lemma 4.6.5 *$\mathcal{LF}$ preserves regularity.*

Proof. Since the F-transducer given in the proof of Lemma 4.1.11 is linear, by Theorem 4.3.6 (i), it is enough to show that for each LF-transducer $\mathfrak{A} = (\Sigma, X, A, \Omega, Y, P, A')$, $\text{range}(\tau_{\mathfrak{A}})$ is regular. Without loss of generality, we may assume that $\mathfrak{A}$ is connected.

Consider the regular ΩY-grammar $G = (A, \Omega, Y, P', A')$, where P' is given as follows:

(i) if $x \to aq$ $\big(x \in X, a \in A, q \in F_\Omega(Y)\big)$ is in P, then $a \to q$ is in P',

(ii) if $\sigma(a_1, \ldots, a_m) \to aq$ $\big(\sigma \in \Sigma_m, m \geq 0, a_1, \ldots, a_m, a \in A, q \in F_\Omega(Y \cup \Xi_m)\big)$ is in P, then $a \to q(a_1, \ldots, a_m)$ is in P'.

In order to prove the lemma it is enough to show that the equivalence

$$a \Rightarrow_G^* q \iff \big(\exists p \in F_\Sigma(X)\big)(p \Rightarrow_{\mathfrak{A}}^* aq) \tag{1}$$

holds for all $a \in A$ and $q \in F_\Omega(Y)$.

(I) First we prove that the left side of (1) implies its right side. For this, assume that $a \Rightarrow_G^* q$ is valid. We shall proceed by induction on the length l of $a \Rightarrow_G^* q$.

Let $l = 1$. Then $a \to q$ is in P', and the following two cases are possible:

(Ia) There is a production $x \to aq$ $\big(x \in X, a \in A, q \in F_\Omega(Y)\big)$.

(Ib) There is a production $\sigma(a_1, \ldots, a_m) \to aq$ $(\sigma \in \Sigma_m, m \geq 0, a_1, \ldots, a_m, a \in A)$ such that in q no auxiliary variables occur, i.e., $q \in F_\Omega(Y)$.

In case (Ia) take $p = x$.

In case (Ib), since $\mathfrak{A}$ is connected, there are $p_i \in F_\Sigma(X)$ and $q_i \in F_\Omega(Y)$ $(i = 1, \ldots, m)$ such that $p_i \Rightarrow_{\mathfrak{A}}^* a_i q_i$ hold. Now taking $p = \sigma(p_1, \ldots, p_m)$ we have $p = \sigma(p_1, \ldots, p_m) \Rightarrow_{\mathfrak{A}}^* \sigma(a_1 q_1, \ldots, a_m q_m) \Rightarrow_{\mathfrak{A}} aq(q_1, \ldots, q_m) = aq$.

Next, assume that $l > 1$ and that our statement has been proved for derivations of length less than l. Then $a \Rightarrow_G^* q$ can be written in the form $a \Rightarrow_G \overline{q}(a_1, \ldots, a_m) \Rightarrow_G^* \overline{q}(q_1, \ldots, q_m) = q$, where $\sigma(a_1, \ldots, a_m) \to a\overline{q}$ is in P for some $\sigma \in \Sigma_m$ $(m > 0)$ and $a_i \Rightarrow_G^* q_i$ $(1 \leq i \leq m)$ if ξ_i occurs in $\overline{q}$. By the induction hypothesis, for all such i there exists a $p_i \in F_\Sigma(X)$ with $p_i \Rightarrow_{\mathfrak{A}}^* a_i q_i$. In the remaining cases, i.e., if ξ_i does not occur in $\overline{q}$, let $p_i \in F_\Sigma(X)$ and $q_i \in F_\Omega(Y)$ $(1 \leq i \leq m)$ be arbitrary such that $p_i \Rightarrow_{\mathfrak{A}}^* a_i q_i$. Then $p = \sigma(p_1, \ldots, p_m)$ satisfies $p \Rightarrow_{\mathfrak{A}}^* aq$.

(II) Assume that $p \Rightarrow_{\mathfrak{A}}^* aq$ holds. We shall show by induction on $\mathrm{hg}(p)$ that the left side of (1) is also valid. If $\mathrm{hg}(p) = 0$, then, by the choice of P', the right side of (1) obviously implies its left side.

Now let $p = \sigma(p_1, \ldots, p_m)$ ($\sigma \in \Sigma_m, m > 0$), and assume that our statement has been proved for all trees from $F_\Sigma(X)$ with height less than $\mathrm{hg}(p)$. Moreover, let us write $p \Rightarrow_{\mathfrak{A}}^* aq$ in the form $p \Rightarrow_{\mathfrak{A}}^* \sigma(a_1 q_1, \ldots, a_m q_m) \Rightarrow_{\mathfrak{A}} a\overline{q}(q_1, \ldots, q_m)$, where $\sigma(a_1, \ldots, a_m) \to a\overline{q}$ is in P and $p_i \Rightarrow_{\mathfrak{A}}^* a_i q_i$ ($i = 1, \ldots, m$). Then, by the definition of P' and the induction hypothesis, we have $a \Rightarrow_G \overline{q}(a_1, \ldots, a_m) \Rightarrow_G^* \overline{q}(q_1, \ldots, q_m) = q$. □

From Lemma 4.6.5, using Theorems 4.2.7 and 4.4.9, respectively, we get the following results.

Corollary 4.6.6 *$\mathcal{LR}$ preserves regularity.* □

Corollary 4.6.7 *$\mathcal{LR}_R$ preserves regularity.* □

A state $a \in A$ of an F-transducer $\mathfrak{A} = (\Sigma, X, A, \Omega, Y, P, A')$ is *nondeleting* if there exist two trees $p \in \hat{F}_\Sigma(X \cup \Xi_1)$ and $q \in F_\Omega(Y \cup \Xi_1)$ such that $p(a\xi_1) \Rightarrow^* a'q$ for some $a' \in A'$ and ξ_1 occurs in q. Otherwise a is *deleting*. The state a is *copying* if there are two trees $p \in \hat{F}_\Sigma(X \cup \Xi_1)$ and $q \in F_\Omega(Y \cup \Xi_1)$ such that $p(a\xi_1) \Rightarrow^* a'q$ for some $a' \in A'$ and ξ_1 occurs at least twice in q.

Lemma 4.6.8 *Let $\mathfrak{A} = (\Sigma, X, A, \Omega, Y, P, A')$ be a connected F-transducer. If $\tau_{\mathfrak{A}}$ preserves regularity and $a \in A$ is copying, then* $\mathrm{range}(\tau_{\mathfrak{A}(a)})$ *is finite.*

Proof. Assume that $\tau_{\mathfrak{A}}$ preserves regularity. Let $a \in A$ be a copying state, and take two trees $p \in \hat{F}_\Sigma(X \cup \Xi_1)$ and $q \in \hat{F}_\Omega(Y \cup \Xi_n)$ such that $p(a\xi_1) \Rightarrow^* a'q(\xi_1^n)$ where $a' \in A'$ and $n > 1$. Suppose that $\mathrm{range}(\tau_{\mathfrak{A}(a)})$ is infinite. Then there is an $s \in \mathrm{range}(\tau_{\mathfrak{A}(a)})$ with $\mathrm{hg}(s) > k \cdot |A|$, where k is the maximum of the heights of the right-hand sides of the productions in P. Let $r \in F_\Sigma(X)$ be a tree such that $r \Rightarrow^* as$. Since $\mathrm{hg}(s) > k \cdot |A|$, there are trees $r_1, r_2 \in \hat{F}_\Sigma(X \cup \Xi_1)$ and $r_3 \in F_\Sigma(X)$ such that the following conditions are satisfied:

(i) $r_1\big(r_2(r_3)\big) = r$,

(ii) $r_3 \Rightarrow^* bs_3$, $r_2(b\xi_1) \Rightarrow^* bs_2$ and $r_1(b\xi_1) \Rightarrow^* as_1$ for some $b \in A$, $s_1, s_2 \in F_\Omega(Y \cup \Xi_1)$ and $s_3 \in F_\Omega(Y)$,

(iii) $\mathrm{hg}(s_2) > 0$, and ξ_1 occurs in s_1 and s_2,

(iv) $s_1(s_2(s_3)) = s$.

Therefore, for each $i(=0,1,\ldots)$, there is a derivation $p_i = p\big(r_1(r_2^i(r_3))\big) \Rightarrow^* a'q(t_i^n) = q_i$ where $t_i = s_1\big(s_2^i(s_3)\big)$ (the powers t^i of any tree $t \in F_\Sigma(X \cup \Xi_1)$ are defined thus: $t^0 = \xi_1$, and $t^{i+1} = t(t^i)$ for each $i \geq 0$). Obviously, $\mathrm{hg}(q_i)$ increases with i when i is large enough.

Now consider the forest $T = \{p_i \mid i = 0,1,\ldots\}$. Obviously, T is regular. Since $\tau_{\mathfrak{A}}$ preserves regularity, this implies that $T' = T\tau_{\mathfrak{A}}$ is also regular. Take an ΩY-recognizer $\mathbf{B} = (B, \Omega, Y, \beta, B')$ with $T' = T(\mathbf{B})$. Choose an

$$i \geq \big(2k(\mathrm{hg}(p(r)) + 1) + 2|B|\big)k\big(\mathrm{hg}(p(r)) + 1\big).$$

Then there exists a tree $t \in F_\Omega(Y)$ with $k\big(\mathrm{hg}(p(r)) + 1\big) + |B| \leq \mathrm{hg}(t) < k\big(\mathrm{hg}(p(r)) + 1\big) + 2|B|$ such that

$$\overline{q} = q(t, t_i^{n-1}) \tag{2}$$

is also in T'. To prove the lemma it is enough to show that there exist no j and $a'' \in A'$ such that $p_j \Rightarrow^* a''\overline{q}$. Suppose

$$p_j \Rightarrow^* a''q'(t'^m) = a''\overline{q}$$

holds, where $a'' \in A'$, $q' \in \hat{F}_\Omega(Y \cup \Xi_m)$, $r_3 \Rightarrow^* b_1 s_1'$, $r_2(b_l\xi_1) \Rightarrow^* b_{l+1}s_{l+1}'$ $\big(b_1, b_{l+1} \in A$, $s_{l+1} \in F_\Omega(Y \cup \Xi_1)$, $l = 1,\ldots,j$, $s_1' \in F_\Omega(Y)\big)$, $r_1(b_{j+1}\xi_1) \Rightarrow^* b_{j+2}s_{j+2}'$ $\big(b_{j+2} \in A$, $s_{j+2}' \in F_\Omega(Y \cup \Xi_1)\big)$, $p(b_{j+2}\xi_1) \Rightarrow^* a''s_{j+3}' = a''q'$ and $t' = s_{j+2}'\big(s_{j+1}'(\ldots(s_1')\ldots)\big)$.

By the choice of i, there exists a u $(2 \leq u \leq j+3)$ such that ξ_1 occurs in $s_u', s_{u+1}', \ldots, s_{j+3}'$ but ξ_1 does not occur in s_{u-1}'. Moreover, let $u - 1 \leq u_1 < \ldots < u_v \leq j+3$ be a maximal sequence with $1 \leq \mathrm{hg}(\overline{s}_{u_1}) < \ldots < \mathrm{hg}(\overline{s}_{u_v})$, where $\overline{s}_l = s_l'\big(s_{l-1}'(\ldots(s_1')\ldots)\big)$ $(l = 1,\ldots,j+3)$. Then $v \geq 2k\big(\mathrm{hg}(p(r))+1\big)+2|B|$. Taking into consideration that $\mathrm{hg}(t) \geq k(\mathrm{hg}(p(r))+1)+|B|$ (and $|B| \geq 1$), for an l $(2 \leq l \leq v)$, the word w forming $\mathrm{path}_1(q)$ is a subword of a word in $\mathrm{path}_1\big(s_{j+3}'(s_{j+2}'(\ldots(s_{u_l}')\ldots))\big)$. (Informally speaking, this means that there is a word in $\mathrm{path}_1\big(s_{j+3}'(s_{j+2}'(\ldots(s_{u_l}')\ldots))\big)$ going through the root of t.) Therefore, we have $l\big(\mathrm{path}_1(q)\big) + \mathrm{hg}(t) \geq 2k\big(\mathrm{hg}(p(r))+1\big)+2|B|$. But, by (2) and the choice of t, $l\big(\mathrm{path}_1(q)\big) + \mathrm{hg}(t) < 2k\big(\mathrm{hg}(p(r))+1\big)+2|B|$, which is a contradiction. $\square$

Lemma 4.6.9 *Let $\mathfrak{A} = (\Sigma, X, A, \Omega, Y, P, A')$ be a connected F-transducer such that for every copying state $a \in A$,* $\mathrm{range}(\tau_{\mathfrak{A}(a)})$ *is finite. Then $\mathfrak{A}$ is equivalent to a linear F-transducer.*

Proof. Suppose that $a_1, \ldots, a_k$ are all the copying states of $\mathfrak{A}$. Let $T_i = \mathrm{range}(\tau_{\mathfrak{A}(a_i)})$ $(i = 1,\ldots,k)$. Moreover, set $T = \bigcup(T_i \mid i = 1,\ldots,k)$. By our assumptions, T is finite.

Define an F-transducer $\mathfrak{B} = (\Sigma, X, B, \Omega, Y, P', B')$, where

$$B = (A - \{a_i \mid i = 1,\ldots,k\}) \cup \bigcup(\{a_i\} \times T_i \mid i = 1,\ldots,k)$$

and

$$B' = (A' \cup A' \times T) \cap B.$$

Moreover, P' is given as follows:

(i) If $p \to aq$ ($p \in \Sigma_0 \cup X$) is in P and $a = a_i$ for some i ($1 \leq i \leq k$), then $p \to (a, q)q$ is in P'. If $a \notin \{a_1, \ldots, a_k\}$, then $p \to aq$ itself is in P'.

(ii) Let

$$\sigma(b_1, \ldots, b_m) \to aq(\xi_1, \ldots, \xi_m)$$

($\sigma \in \Sigma_m, m > 0, b_1, \ldots, b_m, a \in A, q \in F_\Omega(Y \cup \Xi_m)$) be in P. We distinguish the following cases:

(iia) The state a is deleting. Fix any $\overline{q} \in F_\Omega(Y \cup \Xi_m)$ such that every ξ_i occurs at most once in $\overline{q}$. Then P contains every linear production $\sigma(c_1, \ldots, c_m) \to a\overline{q}(\xi_1, \ldots, \xi_m)$ such that

$$c_j = \begin{cases} (b_j, q_j)(q_j \in T_l) & \text{if } b_j \text{ is copying and } b_j = a_l, \\ b_j & \text{otherwise.} \end{cases}$$

(iib) The state a is nondeleting but not copying. Then all productions

$$\sigma(c_1, \ldots, c_m) \to aq(\eta_1, \ldots, \eta_m)$$

are in P' where for each $j(= 1, \ldots, m)$,

$$c_j = \begin{cases} (b_j, q_j) \;\; (q_j \in T_l) & \text{if } b_j \text{ is copying and } b_j = a_l, \\ b_j & \text{otherwise} \end{cases}$$

and

$$\eta_j = \begin{cases} \xi_j & \text{if } \xi_j \text{ occurs at most once in } q, \\ q_j (= \pi_2(c_j)) & \text{otherwise.} \end{cases}$$

(Observe that if ξ_j occurs at least twice in q, then b_j is copying.)

(iic) The state a is copying. Then P' contains all productions

$$\sigma(c_1, \ldots, c_m) \to (a, \overline{q})\overline{q}$$

where $\overline{q} = q(\eta_1, \ldots, \eta_m)$ and for each $j(= 1, \ldots, m)$,

$$c_j = \begin{cases} (b_j, q_j) \;\; (q_j \in T_l) & \text{if } b_j \text{ is copying and } b_j = a_l, \\ b_j & \text{otherwise} \end{cases}$$

and

$$\eta_j = \begin{cases} q_j & \text{if } \xi_j \text{ occurs in } q, \\ \text{any fixed tree from } F_\Omega(Y) & \text{otherwise.} \end{cases}$$

(Note that b_j is copying if ξ_j occurs in q.)

This ends the construction of P'. Obviously, $\mathfrak{B}$ is an LF-transducer.

We show that $\mathfrak{A}$ is equivalent to $\mathfrak{B}$.

(I) Assume that $p \Rightarrow^*_{\mathfrak{A}} aq$ ($p \in F_\Sigma(X), q \in F_\Omega(Y), a \in A$) holds. We prove that

(Ia) $p \Rightarrow^*_{\mathfrak{B}} aq$ if a is nondeleting but not copying,

(Ib) $p \Rightarrow^*_{\mathfrak{B}} (a,q)q$ if a is copying,

(Ic) $p \Rightarrow^*_{\mathfrak{B}} a\overline{q}$ for some $\overline{q} \in F_\Omega(Y)$ if a is deleting.

We shall proceed by induction on $\mathrm{hg}(p)$. If $\mathrm{hg}(p) = 0$ then, by (i), (Ia), (Ib) and (Ic) obviously hold.

Next let $p = \sigma(p_1, \dots, p_m)$ $(\sigma \in \Sigma_m, m > 0)$, and write $p \Rightarrow^*_{\mathfrak{A}} aq$ in the more detailed form

$$\sigma(p_1, \dots, p_m) \Rightarrow^*_{\mathfrak{A}} \sigma(b_1 q'_1, \dots, b_m q'_m) \Rightarrow_{\mathfrak{A}} aq'(q'_1, \dots, q'_m) = aq$$

where $\sigma(b_1, \dots, b_m) \to aq'$ is in P and for each j $(1 \le j \le m)$, $p_j \Rightarrow^*_{\mathfrak{A}} b_j q'_j$. Then, by the induction hypothesis, for all $j(= 1, \dots, m)$, we have $p_j \Rightarrow^*_{\mathfrak{B}} c_j q_j$, where

(Ia′) $c_j = b_j$ and $q_j = q'_j$ if b_j is nondeleting and not copying,

(Ib′) $c_j = (b_j, q_j)$ and $q_j = q'_j$ if b_j is copying,

(Ic′) $c_j = b_j$ and $q_j = \overline{q}_j$ for some $\overline{q}_j \in F_\Omega(Y)$ if b_j is deleting.

Therefore:

(Ia″) If a is nondeleting but not copying, then the production

$$\sigma(c_1, \dots, c_m) \to aq'(\eta_1, \dots, \eta_m)$$

is in P', were η_j $(j = 1, \dots, m)$ is given by (iib).

(Ib″) If a is copying then the production

$$\sigma(c_1, \dots, c_m) \to (a, \overline{q}')\overline{q}'$$

with $\overline{q}' = q'(\eta_1, \dots, \eta_m)$ is in P', were η_j $(j = 1, \dots, m)$ is given by (iic).

(Ic″) If a is deleting then the production

$$\sigma(c_1, \dots, c_m) \to a\overline{q}'$$

given by (iia) is in P'.

Thus, in all three cases the required derivations in $\mathfrak{B}$ exist.

(II) Assume that one of the following relations hold:

(IIa) $p \Rightarrow^*_{\mathfrak{B}} aq$ or

(IIb) $p \Rightarrow^*_{\mathfrak{B}} (a,q)q$

where $p \in F_\Sigma(X)$, $q \in F_\Omega(Y)$ and $a \in A$.

Then, by reversing the above computation, one can show that the desired derivations

(IIc) $p \Rightarrow_{\mathfrak{A}}^* aq$ if a is nondeleting,

(IId) $p \Rightarrow_{\mathfrak{A}}^* a\overline{q}$ for some $\overline{q} \in F_\Omega(Y)$ if a is deleting

exist. Since the final states are nondeleting, this ends the proof of the lemma. $\square$

We can now state and prove

Theorem 4.6.10 *Let $\mathfrak{A} = (\Sigma, X, A, \Omega, Y, P, A')$ be an arbitrary F-transducer. Then $\tau_{\mathfrak{A}}$ preserves regularity iff $\mathfrak{A}$ is equivalent to an LF-transducer.*

Proof. If $\mathfrak{A}$ is equivalent to an LF-transducer then, by Lemma 4.6.5, $\tau_{\mathfrak{A}}$ preserves regularity.

Conversely, let $\tau_{\mathfrak{A}}$ preserve regularity. We may assume that $\mathfrak{A}$ is connected. Then by Lemmas 4.6.8 and 4.6.9, $\mathfrak{A}$ is equivalent to an LF-transducer. $\square$

From Example 2.4.15, we directly obtain

Theorem 4.6.11 *Neither $\mathcal{F}$ nor $\mathcal{R}$ preserves regularity.* $\square$

The following result shows that $\mathrm{Surf}(\mathcal{F}) \subset \mathrm{Surf}(\mathcal{R})$. More precisely, we have

Theorem 4.6.12 $\mathrm{Surf}(\mathcal{F}) = \mathrm{Surf}(\mathcal{H})$ *and* $\mathrm{Surf}(\mathcal{H})$ *is a proper subclass of* $\mathrm{Surf}(\mathcal{R})$.

Proof. The first statement of Theorem 4.6.12 follows from Theorem 4.3.3 and Lemma 4.6.5.

It is obvious that $\mathrm{Surf}(\mathcal{H}) \subseteq \mathrm{Surf}(\mathcal{R})$. We show that the inclusion is proper. For this, consider Example 4.1.6. Moreover, let $S = \{\omega_2(\omega_1^n(y_1), \omega_1^n(y_2)) \mid n = 0, 1, \ldots\}$. If R denotes the regular forest $\{\sigma(x)\} \cdot_x \{\sigma(x)\}^{*x}$, then $R\tau_{\mathfrak{A}} = S$. Therefore, $S \in \mathrm{Surf}(\mathcal{R})$.

Assume that for an HR-transducer $\mathfrak{B} = (\Delta, Z, \{b_0\}, \Omega, Y, P', b_0)$ and regular forest $T \subseteq F_\Delta(Z)$, we have $S = T\tau_{\mathfrak{B}}$. Then $\mathfrak{B}$ can be chosen linear since in the opposite case in $T\tau_{\mathfrak{B}}$ there is a tree with at least two occurrences of a subtree. Therefore, by Theorem 2.4.16, S is regular. But one can show similarly as in Example 2.4.15 that S is not regular. $\square$

Next we show some closure properties of surface forests which will be needed also in Section 4.7.

Theorem 4.6.13 *Let $S \in \mathrm{Surf}(\mathcal{F})$ and let T be a recognizable forest. Then $S \cap T \in \mathrm{Surf}(\mathcal{F})$.*

Proof. Let $\tau_1 \subseteq F_\Sigma(X) \times F_\Omega(Y)$ be an F-transformation and $S = R\tau_1$ where $R \in \mathrm{Rec}(\Sigma, X)$. Take an arbitrary regular forest $T \subseteq F_\Omega(Y)$. Denote by $\tau_2 \subseteq F_\Omega(Y) \times F_\Omega(Y)$ the DF-transformation given in the proof of Lemma 4.1.11 which corresponds to T. Then $R\tau_1 \circ \tau_2 = S \cap T$. But, by Theorem 4.3.9, $\tau_1 \circ \tau_2$ is an F-transformation. $\square$

For R-surface forests we have a similar result.

Theorem 4.6.14 *The intersection of an* R*-surface forest with a regular forest is again an* R*-surface forest.*

Proof. The proof is similar to that of the previous theorem, but now we shall use the fact that the transformation given in the proof of Lemma 4.1.11 is an LNR-transformation. Moreover, by Theorem 4.3.15, the composition of an R-transformation by an LNR-transformation is again an R-transformation. □

By Theorem 4.3.7, $\mathcal{DF}$ is closed under composition. Therefore, Surf($\mathcal{DF}$) is closed under DF-transformations. Although $\mathcal{DR}$ is not closed under composition, we shall show that Surf($\mathcal{DR}$) is closed under DR-transformations. For this, we need

Theorem 4.6.15 *Let* $\mathfrak{A} = (\Sigma, X, A, \Omega, Y, P, a_0)$ *and* $\mathfrak{B} = (\Omega, Y, B, \Delta, Z, P', b_0)$ *be any* DR*-transducers. Then there exists a* DR*-transducer* $\mathfrak{C} = (\Sigma, X, C, \Delta, Z, P'', c_0)$ *such that for every* $R \subseteq F_\Sigma(X)$, $S\tau_{\mathfrak{C}} = R\tau_{\mathfrak{A}} \circ \tau_{\mathfrak{B}}$, *where* $S = R \cap \mathrm{dom}(\tau_{\mathfrak{A}} \circ \tau_{\mathfrak{B}})$.

Proof. Let $C = A \times B$ and $c_0 = (a_0, b_0)$. We want to define P'' in such a way that whenever $ap \Rightarrow^*_{\mathfrak{A}} q$ $\big(a \in A, p \in F_\Sigma(X), q \in F_\Omega(Y)\big)$ and $bq \Rightarrow^*_{\mathfrak{B}} r$ $\big(b \in B, r \in F_\Delta(Z)\big)$ hold, then $(a,b)p \Rightarrow^*_{\mathfrak{C}} r$. If $p \in \Sigma_0 \cup X$, then $(ap, q) \in P$. If we put the production $(a,b)p \to r$ in P'', $\mathfrak{C}$ will have the desired property for these a, b, p, q and r.

Now let $p = \sigma(p_1, \ldots, p_m)$ $(\sigma \in \Sigma_m, m > 0)$ and suppose

$$ap = a\sigma(p_1, \ldots, p_m) \Rightarrow_{\mathfrak{A}} \overline{q}(\ldots, a_{ij}p_i, \ldots) \Rightarrow^*_{\mathfrak{A}} \overline{q}(\ldots, q_{ij}, \ldots) = q,$$

where $(a\sigma, \overline{q}(\ldots, a_{ij}\xi_i, \ldots)) \in P$ $(\overline{q} \in \hat{F}_\Omega(Y \cup \Xi_n)$ for some $n)$ and $a_{ij}p_i \Rightarrow^*_{\mathfrak{A}} q_{ij}$, i.e., the considered copy of p_i is translated by $\mathfrak{A}$ starting in state a_{ij} into q_{ij}. Furthermore, suppose that applying to q the transducer $\mathfrak{B}$ starting in b, we get

$$bq = b\overline{q}(\ldots, q_{ij}, \ldots) \Rightarrow^*_{\mathfrak{B}} \overline{r}(\ldots, b_{ij1}q_{ij}, \ldots, b_{ijk}q_{ij}, \ldots) \Rightarrow^*_{\mathfrak{B}} \overline{r}(\ldots, r_{ij1}, \ldots, r_{ijk}, \ldots) = r$$

$$(b\overline{q} \Rightarrow^*_{\mathfrak{B}} \overline{r}, \overline{r} \in F_\Delta(Z \cup \Xi_n), b_{ijl}q_{ij} \Rightarrow^*_{\mathfrak{B}} r_{ijl}, l = 1, \ldots, k)$$

(meaning that the given occurrence of q_{ij} in $\overline{q}$ has k translations by $\mathfrak{B}$ starting the translations in states $b_{ij1}, \ldots, b_{ijk}$). Thus, if we have the production

$$(a,b)\sigma \to \overline{r}\big(\ldots, (a_{ij}, b_{ij1})\xi_i, \ldots, (a_{ij}, b_{ijk})\xi_i, \ldots\big)$$

in P'' and suppose that $\mathfrak{C}$ has the required property for trees with height less than hg(p), then $(a,b)p \Rightarrow^*_{\mathfrak{C}} r$ also holds. Accordingly, the formal definition of P'' reads as follows:

(i) The production $(a,b)x \to r$ $\big((a,b) \in C, x \in X, r \in F_\Delta(Z)\big)$ is in P'' if there is an $ax \to q$ in P such that $bq \Rightarrow^*_{\mathfrak{B}} r$.

(ii) If the production $a\sigma \to q(\mathbf{a}_1\xi_1^{n_1}, \ldots, \mathbf{a}_m\xi_m^{n_m})$ $\big(a \in A, \sigma \in \Sigma_m, m \geq 0, \mathbf{a}_i \in A^{n_i}, i = 1, \ldots, m,$ $n_1 + \ldots + n_m = n, q \in \hat{F}_\Omega(Y \cup \Xi_n)\big)$ is in P and

$$bq \Rightarrow^*_{\mathfrak{B}} r\big(\mathbf{b}_{11}\xi_{11}^{n'_{11}}, \ldots, \mathbf{b}_{1n_1}\xi_{1n_1}^{n'_{1n_1}}, \ldots, \mathbf{b}_{m1}\xi_{m1}^{n'_{m1}}, \ldots, \mathbf{b}_{mn_m}\xi_{mn_m}^{n'_{mn_m}}\big)$$

$\big(\mathbf{b}_{ij} \in B^{n'_{ij}}$, $\xi_{ij} = \xi_{n_1+\ldots+n_{i-1}+j}$, $i = 1, \ldots, m$, $j = 1, \ldots, n_i$, $n'_{11} + \ldots + n'_{mn_m} = n'$, $r \in \hat{F}_\Delta(Z \cup \Xi_{n'})\big)$ holds, then the production

$$(a,b)\sigma \to r\big((a_{1_1}^{n'_{11}}\mathbf{b}_{11}, \ldots, a_{1_{n_1}}^{n'_{1n_1}}\mathbf{b}_{1n_1})\xi_1^{k_1}, \ldots, (a_{m_1}^{n'_{m1}}\mathbf{b}_{m1}, \ldots, a_{m_{n_m}}^{n'_{mn_m}}\mathbf{b}_{mn_m})\xi_m^{k_m}\big)$$

in in P'', where $k_i = n'_{i1} + \ldots + n'_{in_i}$ $(i = 1, \ldots, m)$.

Obviously, $\mathfrak{C}$ is a DR-transducer. Moreover, to prove the theorem it is enough to show that for arbitrary $(a,b) \in C, p \in F_\Sigma(X), q \in F_\Omega(Y)$ and $r \in F_\Delta(Z)$, $ap \Rightarrow^*_{\mathfrak{A}} q$ and $bq \Rightarrow^*_{\mathfrak{B}} r$ jointly imply $(a,b)p \Rightarrow^*_{\mathfrak{C}} r$. This can be proved by induction on $\mathrm{hg}(p)$. □

Let us note that the $\mathfrak{C}$ constructed above may delete certain subtrees of input trees so that $\mathrm{dom}(\tau_{\mathfrak{C}})$ becomes larger than $\mathrm{dom}(\tau_{\mathfrak{A}} \circ \tau_{\mathfrak{B}})$.

If R in Theorem 4.6.15 is regular then, by Corollary 4.3.17 and Theorem 2.4.2, S is also regular. Thus we have

Corollary 4.6.16 $\mathrm{Surf}(\mathcal{DR})$ *is closed under* DR-*transformations.* □

4.7 AUXILIARY CONCEPTS AND RESULTS

In Section 4.3 it has been shown that neither $\mathcal{F}$ nor $\mathcal{R}$ is closed under composition. In the next section we shall prove that compositions of n F-transformations or n R-transformations lead to proper hierarchies when n assumes the values $0, 1, 2, \ldots$.

The purpose of this section is to introduce concepts and present results needed in Section 4.8.

Let K be a class of forests and $\mathcal{S}$ a class of tree transformations. Then $\mathcal{S}(K)$ denotes the class $\{T\tau \mid T \in K, \tau \in \mathcal{S}\}$. Moreover, $\mathrm{yd}\,\mathcal{S}(K)$ will stand for $\{\mathrm{yd}(T) \mid T \in \mathcal{S}(K)\}$.

Definition 4.7.1 Let Σ be a ranked alphabet and X an alphabet. Let f be a mapping which associates with each $d \in \Sigma \cup X$ a nonvoid recognizable forest $T_d \subseteq F_{\Omega(d)}(\Xi_1)$ where $\Omega(d)$ is a ranked alphabet consisting of unary operational symbols only. It is also supposed that $\Omega(d)$ is disjoint with Σ.

Now define the mapping $\overline{f}$ from the set of all ΣX-forests into the set of subsets of $F_{\Sigma\cup\Omega}(X)$ $\big(\Omega = \bigcup(\Omega(d) \mid d \in \Sigma \cup X)\big)$ in the following way:

(i) if $p \in \Sigma_0 \cup X$, then $\overline{f}(p) = \{q(p) \mid q \in T_p\}$,

(ii) if $p = \sigma(p_1, \ldots, p_m)$ $(\sigma \in \Sigma_m, m > 0, p_1, \ldots, p_m \in F_\Sigma(X))$, then

$$\overline{f}(p) = \{q(\sigma(q'_1, \ldots, q'_m)) \mid q \in T_\sigma, q'_i \in \overline{f}(p_i), i = 1, \ldots, m\}, \quad \text{and}$$

(iii) if $T \subseteq F_\Sigma(X)$, then $\overline{f}(T) = \bigcup\big(\overline{f}(p) \mid p \in T\big)$.

The mapping $\overline{f}$ is called a *regular insertion*.

In the sequel we shall write simply f for $\overline{f}$.

The above regular insertion can be interpreted as follows: f inserts directly below each node of a tree $p \in F_\Sigma(X)$ a unary tree from the regular forest T_d if the label of the node in question is d. The insertion of ξ_1 means that the given node is unchanged. The name "regular insertion" is more expressive if trees are given in Polish prefix form. In this case f inserts a word from T_d directly before an occurrence of d in the word p.

Lemma 4.7.2 Rec *is closed under regular insertion.*

Proof. Let $T \subseteq F_\Sigma(X)$ be a regular forest and f a regular insertion given by $f(d) = T_d$ $(d \in \Sigma \cup X, T_d \subseteq F_\Omega(\Xi_1))$. Consider a regular tree grammar $G = (N, \Sigma, X, P, a_0)$ given in normal form such that $T(G) = T$. Moreover, for every T_d $(d \in \Sigma \cup X)$ let $G^d = (N^d, \Omega, \Xi_1, P^d, a_0^d)$ be a regular tree grammar in normal form generating T_d. For each $d \in \Sigma \cup X$ and $a \in N$ consider the tree grammar $G_a^d = (N_a^d, \Omega, \Xi_1, P_a^d, (a_0^d, a))$, where $N_a^d = N^d \times \{a\}$ and

$$P_a^d = \{(a^d, a) \to \omega((b^d, a)) \mid a^d \to \omega(b^d) \in P^d\} \cup \{(a^d, a) \to \xi_1 \mid a^d \to \xi_1 \in P^d\}.$$

Obviously, $T(G_a^d) = T_d$ holds for each $d \in \Sigma \cup X$ and $a \in N$.

Assume that the sets of nonterminal symbols of the grammar G^d $(d \in \Sigma \cup X)$ are pairwise disjoint and also disjoint with N and $N \times (\Sigma \cup X)$. Construct the tree grammar $G' = (N', \Sigma \cup \Omega, X, P', a_0)$, where $N' = \bigcup(N_a^d \mid d \in \Sigma \cup X, a \in N) \cup N \cup N \times (\Sigma \cup X)$ and P' is given as follows:

$$\begin{aligned}
P' = {} & \{a \to (a_0^d, a) \mid a \in N, d \in \Sigma \cup X\} \\
& \cup \bigcup \big(P_a^d - \{(a^d, a) \to \xi_1 \mid a^d \in N^d\} \mid a \in N, d \in \Sigma \cup X\big) \\
& \cup \{(a^d, a) \to (a, d) \mid a^d \to \xi_1 \in P_a^d, a^d \in N^d, a \in N, d \in \Sigma \cup X\} \\
& \cup \{(a, \sigma) \to \sigma(a_1, \ldots, a_m) \mid a \to \sigma(a_1, \ldots, a_m) \in P, \sigma \in \Sigma_m, m > 0, a, a_1, \ldots, a_m \in N\} \\
& \cup \{(a, d) \to d \mid a \to d \in P, a \in N, d \in \Sigma_0 \cup X\}.
\end{aligned}$$

From the construction of G' it is obvious that the following statements are valid:

(ia) For any production $a \to \sigma(a_1, \ldots, a_m) \in P$ $(\sigma \in \Sigma_m, m > 0)$ and tree $q \in T_\sigma$ there exists a derivation in G'

$$a \Rightarrow (a_0^\sigma, a) \Rightarrow^* q((a^\sigma, a)) \Rightarrow q((a, \sigma)) \Rightarrow q(\sigma(a_1, \ldots, a_m)) \qquad (a^\sigma \in N^\sigma).$$

(ib) For any production $a \to d \in P$ $(d \in \Sigma_0 \cup X)$ and tree $q \in T_d$ there exists a derivation in G', $a \Rightarrow (a_0^d, a) \Rightarrow^* q((a^d, a)) \Rightarrow q((a, d)) \Rightarrow d \quad (a^d \in N^d)$.

Conversely,

(ii) for any $a \in N$ and $p \in F_{\Sigma \cup \Omega}(X)$ each derivation $a \Rightarrow_{G'}^* p$ should have the form (iia) $a \Rightarrow (a_0^\sigma, a) \Rightarrow q_1((a_1^\sigma, a)) \Rightarrow \ldots \Rightarrow q_n((a_n^\sigma, a)) \Rightarrow q_n((a,\sigma)) \Rightarrow$

$$q_n(\sigma(a_1,\ldots,a_m)) \Rightarrow^* p$$

for some $a \to \sigma(a_1,\ldots,a_m) \in P, q_n \in T_\sigma, \sigma \in \Sigma_m, m > 0$ and $a_0^\sigma,\ldots,a_n^\sigma \in N^\sigma$, or the form

(iib) $a \Rightarrow (a_0^d, a) \Rightarrow q_1((a_1^d, a)) \Rightarrow \ldots \Rightarrow q_n((a_n^d, a)) \Rightarrow q_n((a,d)) \Rightarrow q_n(d)$ for some $a \to d \in P, q_n \in T, d \in \Sigma_0 \cup X$ and $a_0^d,\ldots,a_n^d \in N^d$.

Properties (ia), (ib), and (ii) obviously imply that $T(G') = f(T)$. □

Lemma 4.7.3 *Let K be a class of forests closed under regular insertion. Then $\mathcal{R}(K)$ is also closed under regular insertion.*

Proof. Let $R \in K$ be an arbitrary ΣX-forest and take an R-transducer $\mathfrak{A} = (\Sigma, X, A, \Omega, Y, P, A')$. Set $S = R\tau_{\mathfrak{A}}$. Moreover, for every $d \in \Sigma \cup X$ take a unary operator $\#_d$, and let f be the regular insertion given by $f(d) = \{\#_d(\xi_1)\}^{*\xi_1}$.

First we shall show that if g is a regular insertion for which $g(d) = \{\#(\xi_1)\}^{*\xi_1}$ ($d \in \Omega \cup Y$), then $g(S) \in \mathcal{R}(K)$.

Construct the R-transducer $\mathfrak{B} = (\Sigma \cup \{\#_d \mid d \in \Sigma \cup X\}, X, B, \Omega \cup \{\#\}, Y, P', A')$ with $B = A \cup C$, where $C = \{\overline{p} \mid p \in \big(\bigcup(\mathrm{sub}(q) \mid q$ is the right-hand side of a rule in $P) - \Xi\big)\}$. Moreover, P' is the union of the following ten sets of productions:

$$P_1 = \{a\#_d \to \#(a\xi_1), a\#_d \to a\xi_1 \mid a \in A, d \in \Sigma \cup X\},$$

$$P_2 = \{a\#_d \to \omega(\overline{q}_1\xi_1,\ldots,\overline{q}_m\xi_1) \mid ad \to q \text{ is in } P \text{ for some } d \in \Sigma \cup X, a \in A, q = \omega(q_1,\ldots,q_m), \omega \in \Omega_m,\ m > 0\},$$

$$P_3 = \{a\#_d \to \overline{q}\xi_1 \mid ad \to q \text{ is in } P \text{ for some } d \in \Sigma, q = a'\xi_i, a, a' \in A\},$$

$$P_4 = \{a\#_d \to \overline{q}\xi_1 \mid ad \to q \text{ is in } P \text{ for some } d \in \Sigma \cup X, a \in A, q = \omega \in \Omega_0\},$$

$$P_5 = \{a\#_d \to \overline{q}\xi_1 \mid ad \to q \text{ is in } P \text{ for some } d \in \Sigma \cup X, a \in A, q = y \in Y\},$$

$$P_6 = \{\overline{q}\#_d \to \#(\overline{q}\xi_1),\ \overline{q}\#_d \to \omega(\overline{q}_1\xi_1,\ldots,\overline{q}_m\xi_1),\ \overline{q}\#_d \to \overline{q}\xi_1 \mid q = \omega(q_1,\ldots,q_m), \omega \in \Omega_m,\ m > 0\},$$

$$P_7 = \{\overline{a\xi_i}\#_d \to \#(\overline{a\xi_i}\xi_1),\ \overline{a\xi_i}\#_d \to \overline{a\xi_i}\xi_1,\ \overline{\omega}\#_d \to \#(\overline{\omega}\xi_1),\ \overline{\omega}\#_d \to \overline{\omega}\xi_1,\ \overline{y}\#_d \to \#(\overline{y}\xi_1),\ \overline{y}\#_d \to \overline{y}\xi_1 \mid 1 \le i \le r(P), r(P) \text{ is the maximum of ranks of the operators appearing in the left-hand sides of productions from } P, a \in A, \omega \in \Omega_0, y \in Y\},$$

$$P_8 = \{\overline{a\xi_i}\sigma \to a\xi_i \mid a \in A, \sigma \in \Sigma_m, m > 0, 1 \le i \le m\},$$

$$P_9 = \{\overline{\omega}d \to \omega \mid \omega \in \Omega_0, d \in \Sigma \cup X\} \text{ and}$$

$$P_{10} = \{\overline{y}d \to y \mid y \in Y,\ d \in \Sigma \cup X\}.$$

One can easily see that $\mathfrak{B}$ works as follows: assume that for some $a \in A, p \in F_\Sigma(X)$ and $q \in F_\Omega(Y)$ a derivation $ap \Rightarrow^*_{\mathfrak{A}} q$ exists. Let q' be a tree obtained by inserting in q arbitrary trees from $\{\#(\xi_1)\}^{*\xi_1}$ below symbols from $\Omega \cup Y$. Then for a $p' \in f(p)$, $ap' \Rightarrow^*_{\mathfrak{B}} q'$ holds. Conversely, if for some $a \in A, p \in F_\Sigma(X), p' \in f(p)$ and $q' \in F_{\Omega\cup\{\#\}}(Y)$ a derivation $ap' \Rightarrow^*_{\mathfrak{B}} q'$ holds then there is a $q \in F_\Omega(Y)$ such that $q' \in g(q)$ and $ap \Rightarrow^*_{\mathfrak{A}} q$.

Now, consider an arbitrary regular insertion h (into ΩY-trees). For each $d \in \Omega \cup Y$, there is a regular tree grammar $G_d = (N_d, \Omega(d), \Xi_1, P_d, \{a_{d_0}\})$ such that $h(d) = T(G_d)$. We may assume that every G_d is in normal form. Since $\Omega(d)$ is unary, this means that the productions of G_d are of the form $a_d \to \omega_d(a'_d)$ or $a_d \to \xi_1$ $(a_d, a'_d \in N_d, \omega_d \in \Omega(d))$. Furthermore we may assume that the sets N_d are pairwise disjoint. Now construct the R-transducer

$$\mathfrak{C} = (\Omega \cup \{\#\}, Y, C, \Delta, Y, P'', C')$$

with

$$C = \bigcup (N_d \mid d \in \Omega \cup Y), \qquad C' = \{a_{d_0} \mid d \in \Omega \cup Y\}$$

and

$$\Delta = \bigcup \big(\Omega(d) \mid d \in \Omega \cup Y\big) \cup \Omega \quad \big(\Delta_1 = \bigcup \big(\Omega(d) \mid d \in \Omega \cup Y\big) \cup \Omega_1, \Delta_m = \Omega_m \ (m \neq 1)\big).$$

Furthermore, P'' is given as follows:

(I) $a_d\# \to \omega_d(a'_d\xi_1)$ $(a_d, a'_d \in N_d, \omega_d \in \Omega(d), d \in \Omega \cup Y)$ is in P'' if $a_d \to \omega_d(a'_d)$ is in P_d.

(II) $a_\omega\omega \to \omega(a_{d_{1_0}}\xi_1, \ldots, a_{d_{m_0}}\xi_m)$ is in P'' for $\omega \in \Omega_m, m \geq 0, d_1, \ldots, d_m \in \Omega \cup Y$ and $a_\omega \in N_\omega$ if $a_\omega \to \zeta_1$ is in P_ω.

(III) For each $y \in Y$ and $a_y \in N_y, a_y y \to y$ is in P'' if $a_y \to \xi_1$ is in P_y.

Obviously, $\mathfrak{C}$ is an R-relabeling. Therefore, by Theorem 4.3.15, $\tau_{\mathfrak{B}} \circ \tau_{\mathfrak{C}} = \tau$ is an R-transformation. Moreover, by the constructions of $\mathfrak{B}$ and $\mathfrak{C}$, it is clear that the equality $h(S) = f(R)\tau$ holds. □

In the next section we shall need

Theorem 4.7.4 *Let $\tau : X^* \to Y^*$ be a mapping induced by a deterministic* gsm *and Σ a ranked alphabet. Then there exist a ranked alphabet Ω and a* $\mathrm{DR_R}$*-transducer $\mathfrak{B} = (\Sigma, X, B, \Omega, Y, P', b_0)$ such that the equality* $\mathrm{yd}(T)\tau = \mathrm{yd}(T\tau_{\mathfrak{B}})$ *holds for every $T \subseteq F_\Sigma(X)$.*

Proof. Consider the deterministic gsm $\mathbf{A} = (X, A, Y, a_0, P, A')$ inducing τ. We shall show the existence of a ranked alphabet Ω and that of a $\mathrm{DR_R}$-transducer $\mathfrak{B} = (\Sigma, X, B, \Omega, Y, P', b_0)$ such that for any $p \in F_\Sigma(X)$,

(i) $\mathrm{yd}(p\tau_{\mathfrak{B}}) = \mathrm{yd}(p)\tau$ if $\mathrm{yd}(p) \in \mathrm{dom}(\tau)$, and

(ii) $p \in \mathrm{dom}(\tau_{\mathfrak{B}})$ implies $\mathrm{yd}(p) \in \mathrm{dom}(\tau)$.

These obviously will imply the validity of Theorem 4.7.4.

For each $a_1, a_2 \in A$, let $T(a_1, a_2)$ denote the set of all such trees $p \in F_\Sigma(X)$ that $a_1\mathrm{yd}(p) \Rightarrow^*_{\mathbf{A}} wa_2$ holds for some $w \in Y^*$. By Lemma 1.7.4 and Theorem 3.3.2, every $T(a_1, a_2) = \mathrm{yd}^{-1}\big(L(a_1, a_2)\big)$ is a regular forest. Now let $B = (A \times A) \cup \{b_0\}$ ($b_0 \notin A$) and $\Omega = \Sigma \cup \{\omega_{ax} \mid a \in A, x \in X\}$, where $r(\omega_{ax})$ equals the length of the word w obtained from the production $ax \to wa' \in P$ ($a \in A'$). (The ranks of symbols from Σ are unchanged.) Moreover, P' is given as follows:

(I) For arbitrary $m > 0, \sigma \in \Sigma_m$ and $a_1, a_2, \ldots, a_{m+1} \in A$, P' contains the production $\big((a_1, a_{m+1})\sigma \to \sigma((a_1, a_2)\xi_1, \ldots, (a_m, a_{m+1})\xi_m), D\big)$ where $D(\xi_i) = T(a_i, a_{i+1})$ ($i = 1, \ldots, m$).

(II) If $\sigma \in \Sigma_0$ and $a \in A$, then the production $(a, a)\sigma \to \sigma$ is in P'.

(III) For arbitrary $x \in X$ and $(a_1, a_2) \in A \times A$, P' contains the production $(a_1, a_2)x \to q$, where $a_1 x \Rightarrow_{\mathbf{A}} wa_2$ ($w \in Y^*$) and $q \in F_\Omega(Y)$ is a fixed tree with $\mathrm{yd}(q) = w$ (such q exists by the definition of $\omega_{a_1 x}$).

(IV) For arbitrary $m > 0, \sigma \in \Sigma_m$ and $a_1, \ldots, a_{m+1} \in A$, if $a_1 = a_0$ and $a_{m+1} \in A'$, then the production $\big(b_0\sigma \to \sigma((a_1, a_2)\xi_1, \ldots, (a_m, a_{m+1})\xi_m), D\big)$ is in P', where $D(\xi_i) = T(a_i, a_{i+1})$ ($i = 1, \ldots, m$).

(V) For arbitrary $x \in X$, if $a_0 x \Rightarrow_{\mathbf{A}} wa_1$ ($w \in Y^*$) and $a_1 \in A'$, then the production $b_0 x \to q$ is in P', where $q \in F_\Omega(Y)$ is a fixed tree with $\mathrm{yd}(q) = w$ (again, by the definition of $\omega_{a_0 x}$, such q exists).

(VI) If $a_0 \in A'$ and $\sigma \in \Sigma_0$, then the production $b_0\sigma \to \sigma$ is in P'.

In order to prove Theorem 4.7.4 it is enough to show that for arbitrary $(a_1, a_2) \in A \times A, p \in F_\Sigma(X)$ and $q \in F_\Omega(Y)$ the implication

$$(a_1, a_2)p \Rightarrow^*_{\mathfrak{B}} q \quad \Longrightarrow \quad a_1\mathrm{yd}(p) \Rightarrow^*_{\mathbf{A}} \mathrm{yd}(q)a_2$$

holds. This can be carried out by induction on $\mathrm{hg}(p)$. □

We shall now introduce some more concepts that will be needed in the next section.

Let $\mathfrak{A} = (\Sigma, X, A, \Omega, Y, P, A')$ be an R-transducer. Take a tree $p \in F_\Sigma(X)$ and a node d of p. Denote by s the subtree of p at this node d. Consider a state a and a derivation $\alpha : ap \Rightarrow^* q$ ($q \in F_\Omega(Y)$). Suppose exactly k copies of this occurrence of s are created during α and that these are translated into the trees $t_1, \ldots, t_k$ ($\in F_\Omega(Y)$) starting the translations, respectively, in states $a_1, \ldots, a_k$. In the next definition we distinguish a sequence of these states which will be called the state-sequence of α at d.

Definition 4.7.5 Let $\mathfrak{A} = (\Sigma, X, A, \Omega, Y, P, A')$ be an R-transducer. Take a derivation

$$\alpha : ap \Rightarrow^* q \qquad (a \in A, p \in F_\Sigma(X), q \in F_\Omega(Y)).$$

Let d be a node of p and s the subtree at this node d. Replace the given occurrence of s in p by ξ_1 and denote by r the resulting tree. Write α in the form

$$ap = ar(s) \Rightarrow^* \overline{q}(\mathbf{a}s^n) \Rightarrow^* \overline{q}(\mathbf{t}),$$

where $\overline{q} \in \hat{\hat{F}}_\Omega(Y \cup \Xi_n), \mathbf{a} \in A^n, ar \Rightarrow^* \overline{q}(\mathbf{a}\xi_1^n), \mathbf{a}s^n \Rightarrow^* \mathbf{t}$ and $\mathbf{t} \in F_\Omega(Y)^n$. Denote by $a_i d_i \to q_i$ $(a_i \in A, d_i \in \Sigma \cup X)$ the production applied first in the derivation $a_i s \Rightarrow^* t_i$ $(i = 1, \ldots, n)$. Then $\mathbf{a} = (a_1, \ldots, a_n)$ is the *state-sequence* and

$$(a_1 d_1 \to q_1, \ldots, a_n d_n \to q_n)$$

is the *production-sequence* of α at d.

Often we shall speak about the state-sequence and production-sequence of α at a subtree s. In such cases the node to which the given occurrence of s belongs will be clear from the context.

We now define state-sequences for derivations in GSDTs.

Definition 4.7.6 Let $\mathfrak{A} = (\Sigma, X, A, Y, P, A')$ be a GSDT. Take a derivation

$$\alpha : ap \Rightarrow^* w \qquad (a \in A, p \in F_\Sigma(X), w \in Y^*).$$

Let d be a node of p and s the subtree of p at d. Replace the given occurrence of s in p by ξ_1 and denote by r the resulting tree. Write α in the form

$$ap = ar(s) \Rightarrow^* w_1 a_1 s w_2 \ldots w_n a_n s w_{n+1} \Rightarrow^* w_1 v_1 w_2 \ldots w_n v_n w_{n+1},$$

where $ar \Rightarrow^* w_1 a_1 \xi_1 w_2 \ldots w_n a_n \xi_1 w_{n+1}$ $(w_i \in Y^*, i = 1, \ldots, n+1, a_1, \ldots, a_n \in A)$ and $a_i s \Rightarrow^* v_i$ $(v_i \in Y^*, i = 1, \ldots, n)$. Then $\mathbf{a} = (a_1, \ldots, a_n)$ is the *state-sequence of* α at d.

Like in the case of R-transducers, we shall also speak about the state-sequence of α at the subtree s.

Definition 4.7.7 Let $\mathfrak{A}$ be an R-transducer $\mathfrak{A} = (\Sigma, X, A, \Omega, Y, P, A')$ [a GSDT $\mathfrak{A} = (\Sigma, X, A, Y, P, A')$]. Then a derivation $\alpha : ap \Rightarrow^* q$ $(a \in A, p \in F_\Sigma(X), q \in F_\Omega(Y))$ [$\beta : ap \Rightarrow^* w$ $(a \in A, p \in F_\Sigma(X), w \in Y^*)$] is *k-copying* if for every node d of p the length of the state sequence of α [β] at d is at most k. Moreover, $\mathfrak{A}$ is *k-copying* if every derivation $\alpha : ap \Rightarrow^* q$ $(p \in F_\Sigma(X), q \in F_\Omega(Y))$ [$\beta : ap \Rightarrow^* w$ $(p \in F_\Sigma(X), w \in Y^*)$] with $a \in A'$ is k-copying. Finally, $\mathfrak{A}$ is *finite-copying* if it is k-copying for some k.

We shall use the notation $\mathcal{R}_k$ for the class of all transformations induced by k-copying R-transducers. Similarly, $\mathcal{G}_k$ denotes the class of all transformations induced by k-copying GSDT's. Moreover, $\mathcal{R}_f$ and $\mathcal{G}_f$ will stand for the classes of transformations induced by finite-copying R-transducers and finite copying GSDT's, respectively. Corresponding notations will be used for the classes $\mathcal{DR}, \mathcal{DG}$ etc.

The next result shows that R-transformational languages can be studied through generalized syntax directed translations.

Theorem 4.7.8 *For every k-copying* GSDT $\mathfrak{A} = (\Sigma, X, A, Y, P, A')$ *there exist a ranked alphabet* Ω *and a k-copying* R*-transducer* $\mathfrak{B} = (\Sigma, X, A, \Omega, Y, P', A')$ *such that* $\tau_{\mathfrak{A}} = \{(p, \mathrm{yd}(q)) \mid (p, q) \in \tau_{\mathfrak{B}}\}$.

Conversely, for every k-copying R*-transducer* $\mathfrak{B}$ *there exists a k-copying* GSDT $\mathfrak{A}$ *such that* $\tau_{\mathfrak{A}} = \{(p, \mathrm{yd}(q)) \mid (p, q) \in \tau_{\mathfrak{B}}\}$.

Proof. The R-transducer and GSDT constructed in the proof of Theorem 4.5.4 obviously have the required properties. □

The following theorem gives sufficient conditions under which $\mathcal{R}_k(K) = \mathcal{DR}_k(K)$ holds for a given class K of forests.

Theorem 4.7.9 *Let K be a class of forests closed under relabeling and regular insertion. Take an* R*-transducer* $\mathfrak{A} = (\Sigma, X, A, \Omega, Y, P, A')$, *an* $R \in K$ *and a positive integer k. Then*

$$S = \{q \in F_\Omega(Y) \mid \textit{there is a } k\textit{-copying derivation } ap \Rightarrow^* q \textit{ for some } a \in A' \textit{ and } p \in R\}$$

is in $\mathcal{DR}_k(K)$.

Proof. Since K is closed under regular insertion, we may assume that A' is a singleton. Indeed, in the opposite case enlarge A by a new state a_0, Σ by a new unary operational symbol σ and P by all productions $a_0\sigma \to a\xi_1$ ($a \in A'$). Let $\overline{\mathfrak{A}}$ be the resulting R-transducer with initial state a_0, and let $\overline{R} = f(R)$, where f is a regular insertion given by $f(d) = \{\sigma(\xi_1)\}$ ($d \in X \cup \Sigma$). Then $\overline{R} \in K$ and $\tau_{\overline{\mathfrak{A}}}(\overline{R}) = \tau_{\mathfrak{A}}(R)$. Furthermore, a derivation $ap \Rightarrow^*_{\mathfrak{A}} q$ ($a \in A', p \in R, q \in F_\Omega(Y)$) is k-copying if the corresponding derivation $a_0\sigma(p) \Rightarrow^*_{\overline{\mathfrak{A}}} q$ is k-copying, and conversely. Thus, we shall assume that $A' = \{a_0\}$. Now we introduce the alphabet

$$\overline{X} = \{((a_1x, q_1), \ldots, (a_tx, q_t)) \mid t \le k, x \in X, a_ix \to q_i \in P \quad (i = 1, \ldots, t)\}$$

and the ranked alphabet Δ with

$$\Delta_m = \{((a_1\sigma, q_1), \ldots, (a_t\sigma, q_t)) \mid t \le k, \sigma \in \Sigma_m, a_i\sigma \to q_i \in P \quad (i = 1, \ldots, t)\}$$

($m = 0, 1, \ldots$). Consider the R-transducer $\mathfrak{B} = (\Sigma, X, \{b_0\}, \Delta, \overline{X}, P', b_0)$ where P' consists of the productions

$$b_0x \to ((a_1x, q_1), \ldots, (a_tx, q_t)) \qquad (x \in X, ((a_1x, q_1), \ldots, (a_tx, q_t)) \in \overline{X})$$

and

$$b_0\sigma \to ((a_1\sigma, q_1), \ldots, (a_t\sigma, q_t))(b_0\xi_1, \ldots, b_0\xi_m)$$

$$(\sigma \in \Sigma_m, ((a_1\sigma, q_1), \ldots, (a_t\sigma, q_t)) \in \Delta_m, m = 0, 1, \ldots).$$

Obviously, $\mathfrak{B}$ is an R-relabeling which relabels trees in the following way: if $\sigma \in \Sigma$ [resp. $x \in X$] is a label at a node d of a tree $p \in F_\Sigma(X)$, then $\mathfrak{B}$ relabels d by a sequence of productions $((a_1\sigma, q_1), \ldots, (a_t\sigma, q_t))$ [resp. $((a_1x, q_1), \ldots, (a_tx, q_t))$] from P with length at most k.

Next define an R-transducer $\mathfrak{C} = (\Delta, \overline{X}, C, \Omega, Y, P'', c_0)$ with

$$C = \{(u; a_1, \ldots, a_t) \mid 1 \le u \le t \le k, a_i \in A \ (i = 1, \ldots, t)\}$$

and $c_0 = (1; a_0)$. Moreover, P'' is defined as follows:

(i) For each $(u; a_1, \ldots, a_t) \in C$ and $((a_1p, q_1), \ldots, (a_tp, q_t)) \in \Delta_0 \cup \overline{X}$, $(u; a_1, \ldots, a_t)((a_1p, q_1), \ldots, (a_tp, q_t)) \to q_u$ is in P''.

(ii) Let $(u; a_1, \ldots, a_t) \in C$ and $((a_1\sigma, q_1), \ldots, (a_t\sigma, q_t)) \in \Delta_m$ $(m > 0)$. Write $(a_i\sigma, q_i)$ in the more detailed form $a_i\sigma \to q_i(\mathbf{a}_{i1}\xi_1^{n_{i1}}, \ldots, \mathbf{a}_{im}\xi_m^{n_{im}})$ $(\mathbf{a}_{ij} \in A^{n_{ij}}, j = 1, \ldots, m, n_{i1} + \ldots + n_{im} = n_i, q_i \in \hat{F}_\Omega(Y \cup \Xi_{n_i}), i = 1, \ldots, t)$. Then the production

$$(u; a_1, \ldots, a_t)((a_1\sigma, q_1), \ldots, (a_t\sigma, q_t)) \to$$

$$q_u\big(((u_{11}; \mathbf{b}_1), \ldots, (u_{1n_{u1}}; \mathbf{b}_1))\xi_1^{n_{u1}}, \ldots, ((u_{m1}; \mathbf{b}_m), \ldots, (u_{mn_{um}}; \mathbf{b}_m))\xi_m^{n_{um}}\big)$$

in is P', provided that $n_{1j} + \ldots + n_{tj} \le k$ $(j = 1, \ldots, m)$, where $u_{jl} = n_{1j} + \ldots + n_{u-1j} + l, \mathbf{b}_j = (\mathbf{a}_{1j}, \ldots, \mathbf{a}_{tj})$ and $j = 1, \ldots, m$.

Obviously, $\mathfrak{C}$ is a deterministic R-transducer. Furthermore, one can easily see the following connection between derivations in $\mathfrak{A}$ and $\mathfrak{C}$:

Let $p \in F_\Sigma(X)$ and $q \in F_\Omega(Y)$ be arbitrary trees, and take a k-copying derivation

$$\alpha : \ a_0p \Rightarrow^*_{\mathfrak{A}} q.$$

Consider the tree $\overline{p}$ with $(p, \overline{p}) \in \tau_{\mathfrak{B}}$ which is the result of relabeling each node d of p by the production-sequence of α at d. Then in $\mathfrak{C}$ we have a derivation

$$\beta : (1; a_0)\overline{p} \Rightarrow^*_{\mathfrak{C}} q$$

such that if $\mathbf{a} = (a_1, \ldots, a_n)$ $(n \le k)$ is the state-sequence of α at d then $((1; \mathbf{a}), \ldots, (n; \mathbf{a}))$ is the state-sequence of β at d. Conversely, if for a $\overline{p}' \in F_\Delta(\overline{X})$ and $q' \in F_\Omega(Y)$ there is a derivation

$$\beta' : (1; a_0)\overline{p}' \Rightarrow^*_{\mathfrak{C}} q',$$

then for the (uniquely determined) tree $p' \in F_\Sigma(X)$ with $(p', \overline{p}') \in \tau_{\mathfrak{B}}$ we have the derivation

$$\alpha' : a_0p' \Rightarrow^*_{\mathfrak{A}} q'.$$

Moreover, the state-sequence of β' at a node d of $\overline{p}'$ is of the form $((1; \mathbf{a}'), \ldots, (m; \mathbf{a}'))$ $(\mathbf{a}' = (a'_1, \ldots, a'_m))$ with $m \le k$, and $\mathbf{a}'$ is the state-sequence of α' at d. Therefore, $\mathfrak{C}$ is k-copying and $S = R\tau_{\mathfrak{B}} \circ \tau_{\mathfrak{C}}$ holds. Since K is closed under relabelings, this implies $S \in \mathcal{DR}_k(K)$. □

From Theorem 4.7.9, by Theorem 4.7.8, we get

Corollary 4.7.10 *Let K be a class of forests closed under relabeling and regular insertion. Take a* GSDT $\mathfrak{A} = (\Sigma, X, A, Y, P, A')$*, a $T \in K$ and a positive integer k. Then the language*

$$L = \{w \in Y^* \mid \text{there is a } k\text{-copying derivation } ap \Rightarrow^* w \text{ for some } a \in A' \text{ and } p \in T\}$$

is in $\mathcal{DG}_k(K)$. □

Three more language operations will be needed.

Definition 4.7.11 Let X be an alphabet and $\# \notin X$ a symbol. For each $L \subseteq X^*$, $\mathrm{res}(L, \#)$ (*re*gular *s*ubstitution) denotes the language defined as follows:

(i) if $L = \{e\}$, then $\mathrm{res}(L, \#) = \#^*$,

(ii) if $L = \{x\}$ $(x \in X)$, then $\mathrm{res}(L, \#) = \#^* x \#^*$,

(iii) if $L = \{ux\}$ $(u \in X^*, x \in X)$, then $\mathrm{res}(L, \#) = \mathrm{res}(u, \#)\mathrm{res}(x, \#)$,

(iv) if L is arbitrary, then $\mathrm{res}(L, \#) = \bigcup(\mathrm{res}(w, \#) \mid w \in L)$.

Theorem 4.7.12 *Let K be a class of forests closed under regular insertion. For each $R \in K$ there exist a linear nondeleting* GSDT $\mathfrak{A}$ *and a forest $S \in K$ such that* $\mathrm{res}(\mathrm{yd}(R), \#) = S\tau_{\mathfrak{A}}$.

Proof. Let $R \subseteq F_\Sigma(X)$, $R \in K$, and denote $\mathrm{yd}(R)$ by L. Let $\Delta = \Delta_1 = \{\overline{d} \mid d \in \Sigma \cup X\}$ and let f be the regular insertion defined by $f(d) = \{\overline{d}(\xi_1)\}^{*\xi_1}$ $(d \in \Sigma \cup X)$. Define the GSDT $\mathfrak{A} = (\Omega, X, \{a_0\}, X \cup \{\#\}, P, a_0)$ with $\Omega = \Sigma \cup \Delta$ $(\Omega_1 = \Sigma_1 \cup \Delta, \Omega_m = \Sigma_m, m \neq 1)$ so that

$$P = \{a_0\overline{x} \to \#a_0\xi_1, a_0\overline{x} \to a_0\xi_1\# \mid x \in X\} \cup \{a_0\overline{\sigma} \to \#a_0\xi_1 \mid \sigma \in \Sigma_0\} \cup$$

$$\{a_0 x \to x \mid x \in X\} \cup \{a_0\sigma \to a_0\xi_1 \dots a_0\xi_m \mid \sigma \in \Sigma_m, m \geq 0\}.$$

Obviously, $\mathfrak{A}$ is a linear nondeleting GSDT satisfying $\mathrm{res}(L, \#) = f(R)\tau_{\mathfrak{A}}$. Moreover, by our assumptions, $f(R) = S \in K$. □

Theorem 4.7.13 *Let Y be an alphabet and $\# \notin Y$ a symbol. Take a language $L \subseteq Y^*$ and a class K of forests closed under relabeling and regular insertion. If* $\mathrm{res}(L, \#) \in \mathcal{DG}(K)$*, then $L \in \mathcal{DG}_f(K)$.*

Proof. Let $\mathrm{res}(L, \#) = T\tau_{\mathfrak{A}}$ where $\mathfrak{A} = (\Sigma, X, A, Y \cup \{\#\}, P, a_0)$ is a deterministic GSDT and $T \subseteq F_\Sigma(X)$ is a forest from K. Moreover, let $A = \{a_1, \dots, a_k\}$. A word $y_{i_1}\#^{n_1}y_{i_2}\#^{n_2}\dots y_{i_{r-1}}\#^{n_{r-1}}y_{i_r}$ $(\in \mathrm{res}(L, \#), y_{i_1}, \dots, y_{i_r} \in Y)$ is called *proper* if $n_1, n_2, \dots, n_{r-1}$ are pairwise distinct.

Consider a derivation

$$\alpha : a_0 p \Rightarrow^* w_1 b_1 p_1 w_2 b_2 p_1 w_3 \dots w_s b_s p_1 w_{s+1} \Rightarrow^* w_1 v_1 w_2 v_2 w_3 \dots w_s v_s w_{s+1} = w,$$

where $p \in T$, p_1 is a subtree of p, $(b_1, b_2, \dots, b_s)$ is the state-sequence of α at p_1, $b_i p_1 \Rightarrow^* v_i$ $(i = 1, \dots, s)$ and $w_1, \dots, w_{s+1}, v_1, \dots, v_s \in (Y \cup \{\#\})^*$. If w is proper and $b_i = b_j$ $(i \neq j)$, then in v_i (and thus in v_j) at most one symbol from Y may occur.

Now for each $\sigma \in \Sigma_m$ $(m > 0)$ take all pairs (σ, M), where M is a matrix of type $k \times m$ whose elements are from $Y \cup A\Xi_m \cup \{e\}$. Moreover, let Ω be a ranked alphabet with $\Omega_0 = \Sigma_0$ and $\Omega_m = \{(\sigma, M) \mid \sigma \in \Sigma_m\}$ $(m > 0)$.

Let $Y = \{y_1, \dots, y_l\}$ and denote by T_{ij} $(i = 1, \dots, k, j = 1, \dots, l)$ the set of all trees $p \in F_\Sigma(X)$ for which $v \in \#^* y_j \#^*$, where v is the word obtained from the derivation $a_i p \Rightarrow^* v$. Moreover, let T_{il+1} $(i = 1, \dots, k)$ be the forest of all trees $p \in F_\Sigma(X)$ satisfying $v \in \#^*$, where v is obtained again by the derivation $a_i p \Rightarrow^* v$.

By Theorems 4.5.4 and 3.3.2 and Corollary 4.3.17, the T_{ij} $(i = 1, \dots, k, j = 1, \dots, l+1)$ are recognizable forests. Therefore, there are ΣX-recognizers $\mathbf{A}_{ij} = (\mathcal{A}_{ij}, \alpha_{ij}, A'_{ij})$ $(i = 1, \dots, k, j = 1, \dots, l+1)$ with $\mathcal{A}_{ij} = (A_{ij}, \Sigma)$ such that $T(\mathbf{A}_{ij}) = T_{ij}$. Consider the DF-relabeling $\mathfrak{B} = (\Sigma, X, B, \Omega, X, P', B)$ where

$$B = \{(p\hat{\alpha}_{11}, \dots, p\hat{\alpha}_{1l+1}, \dots, p\hat{\alpha}_{k1}, \dots, p\hat{\alpha}_{kl+1}) \mid p \in F_\Sigma(X)\},$$

and P' is given as follows:

(i) For each $x \in X$, the production

$$x \to (x\alpha_{11}, \dots, x\alpha_{1l+1}, \dots, x\alpha_{k1}, \dots, x\alpha_{kl+1})x$$

is in P'.

(ii) For every $\sigma \in \Sigma_0$, the production

$$\sigma \to (\sigma^{\mathcal{A}_{11}}, \dots, \sigma^{\mathcal{A}_{1l+1}}, \dots, \sigma^{\mathcal{A}_{k1}}, \dots, \sigma^{\mathcal{A}_{kl+1}})\sigma$$

is in P'.

(iii) For each $\sigma \in \Sigma_m$ $(m > 0)$ the productions

$$\sigma(\mathbf{b}_1, \dots, \mathbf{b}_m) \to \mathbf{b}(\sigma, M)(\xi_1, \dots, \xi_m)$$

are in P', where $\mathbf{b}_t = (b_{11}^{(t)}, \dots, b_{1l+1}^{(t)}, \dots, b_{k1}^{(t)}, \dots, b_{kl+1}^{(t)})$, $\mathbf{b} = (b_{11}, \dots, b_{1l+1}, \dots, b_{k1}, \dots, b_{kl+1}) \in B$ $(t = 1, \dots, m)$, $b_{ij} = \sigma^{\mathcal{A}_{ij}}(b_{ij}^{(1)}, \dots, b_{ij}^{(m)})$ $(i = 1, \dots, k, j = 1, \dots, l+1)$ and the element m_{it} $(i = 1, \dots, k, t = 1, \dots, m)$ of matrix M is given by

$$m_{it} = \begin{cases} e & \text{if } b_{il+1}^{(t)} \in A'_{il+1}, \\ y_u & \text{if } b_{iu}^{(t)} \in A'_{iu} \ (1 \leq u \leq l), \\ a_i\xi_t & \text{otherwise.} \end{cases}$$

Obviously, m_{it} is well-defined since there are no two components $b_{ij_1}^{(t)}$ and $b_{ij_2}^{(t)}$ $(1 \leq i \leq k, 1 \leq j_1, j_2 \leq l+1, j_1 \neq j_2)$ such that $b_{ij_1}^{(t)} \in A'_{ij_1}$ and $b_{ij_2}^{(t)} \in A'_{ij_2}$ both hold.

By the definition of $\mathfrak{B}$, it relabels trees in the following way: take a tree $p \in F_\Sigma(X)$, and let $\sigma(p_1, \ldots, p_m)$ $(m > 0)$ be the subtree of p at a node d. The $\mathfrak{B}$ provides us with the information about which of the subtrees $p_1, \ldots, p_m$ is translated by $\mathfrak{A}(a_i)$ $(i = 1, \ldots, k)$ into a word from $(Y \cup \{\#\})^*$ with

(I) no occurrence of letters from Y,

(II) exactly one occurrence of letters from Y,

(IIIa) at least two occurrences of letters from Y, or

(IIIb) the given subtree is not in $\mathrm{dom}(\tau_{\mathfrak{A}(a_i)})$.

Next take the GSDT $\mathfrak{C} = (\Omega, X, A, Y, P'', a_0)$ where P'' is given as follows:

(a) If $ap \to w$ $\big(a \in A, p \in X \cup \Sigma_0, w \in (Y \cup \{\#\})^*\big)$ is in P, then the production obtained from $ap \to w$ by replacing all occurrences of $\#$ in w by e will be in P''.

(b) Let $a\sigma \to w$ $\big(a \in A, \sigma \in \Sigma_m, m > 0, w \in (Y \cup \{\#\} \cup A\Xi_m)^*\big)$ be in P. Then all productions $a(\sigma, M) \to w'$ are in P'' where w' is the result of replacing all occurrences of $a_i\xi_j$ in w by m_{ij} $(1 \le i \le k, 1 \le j \le m)$ and all occurrences of $\#$ by e.

It is clear that $\mathfrak{C}$ is deterministic. Moreover, one can show by induction on $\mathrm{hg}(p)$ for arbitrary $a \in A, p \in F_\Sigma(X)$ and $w \in (Y \cup \{\#\})^*$ the implication

$$ap \Rightarrow^*_{\mathfrak{A}} w \quad \Longrightarrow \quad a\tau_{\mathfrak{B}}(p) \Rightarrow^*_{\mathfrak{C}} \varphi(w)$$

holds, where $\varphi : (Y \cup \{\#\})^* \to Y^*$ is the homomorphism given by $\varphi(y) = y$ $(y \in Y)$ and $\varphi(\#) = e$. Thus

$$\begin{aligned} L = \{w' \in Y^* \mid a_0\tau_{\mathfrak{B}}(p) \Rightarrow^*_{\mathfrak{C}} w', a_0 p \Rightarrow^*_{\mathfrak{A}} w, \\ p \in T, w \in (Y \cup \{\#\})^* \text{ and } w \text{ is proper if } |w'| > 2\}. \end{aligned} \tag{1}$$

Furthermore, by our remark concerning state-sequences of derivations yielding proper words and the construction of $\mathfrak{C}$, the elements of a state-sequence of a derivation $a_0\tau_{\mathfrak{B}}(p) \Rightarrow^*_{\mathfrak{C}} w'$ from (1) are different at any node of $\tau_{\mathfrak{B}}(p)$. Therefore, since $\mathfrak{C}$ has k elements, each element of L can be obtained by a k-copying derivation in $\mathfrak{C}$. Finally, since by our assumptions $T\tau_{\mathfrak{B}} \in K$, using Corollary 4.7.10 we get $L \in \mathcal{DG}_k(K)$. □

Definition 4.7.14 Let X be an alphabet and $\# \notin X$ a symbol. For each language $L \subseteq X^*$, the language $c_*(L, \#)$ is defined by

$$c_*(L, \#) = \{(w\#)^n \mid w \in L,\ n = 1, 2, \ldots\}.$$

Theorem 4.7.15 *Let K be a class of forests closed under regular insertion. For each $R \in K$ there exist a* DGSDT $\mathfrak{A}$ *and a forest $S \in K$ such that* $c_*\big(\mathrm{yd}(R), \#\big) = S\tau_{\mathfrak{A}}$.

Proof. Suppose $R \subseteq F_\Sigma(X)$ and let $L = \mathrm{yd}(R)$. We introduce the ranked alphabet $\Delta = \Delta_1 = \{\overline{d} \mid d \in \Sigma \cup X\}$ and define a regular insertion f by $f(d) = \{\overline{d}(\xi_1)\}^{*\xi_1}$ $(d \in \Sigma \cup X)$. Moreover, let Ω be the ranked alphabet for which $\Omega_1 = \Sigma_1 \cup \Delta$ and $\Omega_m = \Sigma_m$ $(m \geq 0,\ m \neq 1)$. Consider the GSDT

$$\mathfrak{A} = (\Omega, X, \{a_1, a_2\}, X \cup \{\#\}, P, a_1)$$

where

$$\begin{aligned} P = &\{a_1\overline{d} \to a_1\xi_1 a_2\xi_1\# \mid d \in \Sigma \cup X\} \\ &\cup \{a_2\overline{d} \to a_2\xi_1 \mid d \in \Sigma \cup X\} \\ &\cup \{a_1 x \to e \mid x \in X\} \cup \{a_1\sigma \to e \mid \sigma \in \Sigma_m,\ m \geq 0\} \\ &\cup \{a_2 x \to x \mid x \in X\} \cup \{a_2\sigma \to a_2\xi_1 \ldots a_2\xi_m \mid \sigma \in \Sigma_m,\ m \geq 0\}. \end{aligned}$$

It is obvious that $\mathfrak{A}$ is a deterministic GSDT satisfying $c_*(L, \#) = S\tau_{\mathfrak{A}}$, where $S = f(R)$. Moreover, by our assumptions $S \in K$. $\square$

Theorem 4.7.16 *Let $U \subseteq c_*(L, \#)$ $(L \subseteq Z^*,\ \# \notin Z)$ be a language containing infinitely many words $(w\#)^n$ for each $w \in L$. Furthermore, let K be a class of forests closed under relabeling and regular insertion. If $U \in \mathcal{DG}_f\big(\mathcal{R}(K)\big)$, then $L \in \mathcal{DG}(K)$.*

Proof. Let $\mathfrak{A} = (\Sigma, X, A, \Omega, Y, P, A')$ be an R-transducer and $\mathfrak{B} = (\Omega, Y, B, Z \cup \{\#\}, P', b_0)$ a k-copying deterministic GSDT. Moreover, take a forest $R \subseteq F_\Sigma(X)$ from K satisfying $U = (R\tau_{\mathfrak{A}})\tau_{\mathfrak{B}}$. Since K is closed under regular insertion, we may, without any loss of generality, assume that A' is a singleton, say $A' = \{a_0\}$. First we shall construct an R-transducer $\overline{\mathfrak{A}} = (\Sigma, X, \overline{A}, \Omega, Y, \overline{P}, \overline{a}_0)$ which translates every $p \in F_\Sigma(X)$ into a tree $q \in F_\Omega(Y)$ in the same way as $\mathfrak{A}$ provided that $q \in \mathrm{dom}(\tau_{\mathfrak{B}})$. In addition, if during the translation of p into q by $\mathfrak{A}$, an occurrence of a subtree p' in p is translated starting in a state a into a tree q', then during the corresponding translation of p by $\overline{\mathfrak{A}}$, p' will be translated starting in a state consisting of a and the state-sequence of the derivation of q in $\mathfrak{B}$ at the subtree q'. Thus, $\overline{\mathfrak{A}}$ will have the property that if during the above translation of p by $\overline{\mathfrak{A}}$, two copies of an occurrence of p' are translated starting in states $\overline{a}_1$ and $\overline{a}_2$, respectively, into the trees q_1 and q_2 such that $\overline{a}_1 = \overline{a}_2$, then the state-sequences of the derivation of q in $\mathfrak{B}$ at q_1 and q_2 coincide.

Let $\tau_{\mathfrak{B}}(q) = (w\#)^m$ $(w \in Z^*)$. If m is large enough, then the properties of $\overline{\mathfrak{A}}$ will make it possible to replace in a derivation $\overline{a}_0 p \Rightarrow^*_{\overline{\mathfrak{A}}} q$ different derivations of p' starting from the same state by one of them such that for the resulting output tree $\overline{q}$ we shall have $\tau_{\mathfrak{B}}(\overline{q}) = (w\#)^{m'}$ with $m' \geq m$. By prescribing the applications of productions of $\overline{\mathfrak{A}}$ in this manner we shall arrive at a DR-transducer $\mathfrak{A}_1$ such that $(S\tau_{\mathfrak{A}_1})\tau_{\mathfrak{B}}$ contains infinitely many words $(w\#)^m$ for each $w \in L$ and S is obtained from R by a relabeling. Afterwards applying a deterministic gsm to $(S\tau_{\mathfrak{A}_1})\tau_{\mathfrak{B}}$, we shall get L.

Thus construct the R-transducer $\overline{\mathfrak{A}} = (\Sigma, X, \overline{A}, \Omega, Y, \overline{P}, \overline{a}_0)$ where

$$\overline{A} = \{(a, \mathbf{b}) \mid a \in A,\ \mathbf{b} \in B^n,\ n = 0, 1, \ldots, k\}$$

and $\overline{a}_0 = \big(a_0, (b_0)\big)$. Moreover, $\overline{P}$ is given in the following way:

(i) Let $ap \to q$ $\big(a \in A,\ p \in X \cup \Sigma_0,\ q \in F_\Omega(Y)\big)$ be in P and take a vector $\mathbf{b} \in B^n$ $(0 \le n \le k)$. Then the production $(a, \mathbf{b})p \to q$ is in $\overline{P}$.

(ii) Let $a\sigma \to q(\mathbf{a}_1\xi_1^{n_1}, \ldots, \mathbf{a}_m\xi_m^{n_m})$ $\big(a \in A,\ \sigma \in \Sigma_m,\ m > 0,\ \mathbf{a}_i \in A^{n_i},\ i = 1, \ldots, m,$ $n_1 + \ldots + n_m = n,\ q \in \hat{F}_\Omega(Y \cup \Xi_n)\big)$ be in P and $\mathbf{b} = (b_1, \ldots, b_s) \in B^s$. Moreover, for every u $(1 \le u \le s)$, and every j $(1 \le j \le n)$ take the derivation

$$b_u q \Rightarrow_{\mathfrak{B}}^* w_{uj_1} b_{uj_1} \xi_j w_{uj_2} \ldots w_{uj_{u_j}} b_{uj_{u_j}} \xi_j w_{uj_{u_j+1}}$$
$$\big(w_{uj_1}, \ldots, w_{uj_{u_j+1}} \in (Z \cup \{\#\} \cup B(\Xi_n - \{\xi_j\}))^*,\ b_{uj_1}, \ldots, b_{uj_{u_j}} \in B\big).$$

Set $\mathbf{b}_j = (b_{1j_1}, \ldots, b_{1j_{1_j}}, \ldots, b_{sj_1}, \ldots, b_{sj_{s_j}})$ $(j = 1, \ldots, n)$. Then the production

$$(a, \mathbf{b})\sigma \to q\big(((a_{11}, \mathbf{b}_1), \ldots, (a_{1n_1}, \mathbf{b}_{n_1}))\xi_1^{n_1}, ((a_{21}, \mathbf{b}_{n_1+1}), \ldots$$
$$\ldots, (a_{2n_2}, \mathbf{b}_{n_1+n_2}))\xi_2^{n_2}, \ldots, ((a_{m_1}, \mathbf{b}_{n_1+\ldots+n_{m-1}+1}), \ldots, (a_{mn_m}, \mathbf{b}_n))\xi_m^{n_m}\big)$$

is in $\overline{P}$, provided that for each $j = 1, \ldots, n$ the length of the sequence $\mathbf{b}_j$ is not greater than k.

From the construction of $\overline{\mathfrak{A}}$, one can easily see the following connection between $\mathfrak{A}$ and $\overline{\mathfrak{A}}$. Take a tree $p \in F_\Sigma(X)$, a node d of p and let p' be the subtree of p at d. Moreover, write $p = r(p')$ $\big(r \in \hat{F}_\Sigma(X \cup \Xi_1)\big)$, and consider a derivation

$$\alpha:\ a_0 r(p') \Rightarrow_{\mathfrak{A}}^* \overline{q}(\mathbf{a}p'^n) \Rightarrow_{\mathfrak{A}}^* \overline{q}(\mathbf{t}) = q$$
$$\big(q \in F_\Omega(Y),\ a_0 r \Rightarrow_{\mathfrak{A}}^* \overline{q}(\mathbf{a}\xi_1^n),\ \overline{q} \in \hat{\hat{F}}_\Omega(Y \cup \Xi_n),\ \mathbf{a}p'^n \Rightarrow_{\mathfrak{A}}^* \mathbf{t},\ \mathbf{t} \in F_\Omega(Y)^n\big)$$

with $q \in \mathrm{dom}(\tau_{\mathfrak{B}})$. Then in $\overline{\mathfrak{A}}$ we have a derivation

$$\beta:\ \big(a_0, (b_0)\big)r(p') \Rightarrow^* \overline{q}\big(((a_1, \mathbf{b}_1), \ldots, (a_n, \mathbf{b}_n))p'^n\big) \Rightarrow^* \overline{q}(\mathbf{t}) = q,$$

where $\mathbf{b}_i$ $(1 \le i \le n)$ is the state-sequence of the derivation

$$\gamma:\ b_0 q \Rightarrow_{\mathfrak{B}}^* w\ \big(\in (Z \cup \{\#\})^*\big)$$

at the subtree t_i. Therefore, if $(a_i, \mathbf{b}_i) = (a_j, \mathbf{b}_j)$ $(1 \le i, j \le n)$, then the state-sequences of γ at the subtrees t_i and t_j coincide. We can assume that $\mathfrak{A}$ itself has this property, because the equality $\tau_{\mathfrak{A}} \circ \tau_{\mathfrak{B}} = \tau_{\overline{\mathfrak{A}}} \circ \tau_{\mathfrak{B}}$ obviously holds.

Consider a word $(w\#)^m \in (R\tau_{\mathfrak{A}})\tau_{\mathfrak{B}}$ with $m > 2k + 1$. More exactly, let $p \in R$ be a tree for which under the derivation $a_0 p \Rightarrow_{\mathfrak{A}}^* q$ $\big(\in F_\Omega(Y)\big)$ the equality $\tau_{\mathfrak{B}}(q) = (w\#)^m$ holds. Let $r \in \hat{F}_\Sigma(X \cup \Xi_1)$ and $p' \in F_\Sigma(X)$ with $r(p') = p$. Moreover, write the above derivation in the form

$$\alpha':\ a_0 r(p') \Rightarrow_{\mathfrak{A}}^* \overline{q}(\mathbf{a}p'^n) \Rightarrow_{\mathfrak{A}}^* \overline{q}(\mathbf{t}) = q$$
$$\big(q \in F_\Omega(Y),\ a_0 r \Rightarrow_{\mathfrak{A}}^* \overline{q}(\mathbf{a}\xi_1^n),\ \overline{q} \in \hat{\hat{F}}_\Omega(Y \cup \Xi_n),\ \mathbf{a}p'^n \Rightarrow_{\mathfrak{A}}^* \mathbf{t},\ \mathbf{t} \in F_\Omega(Y)^n\big).$$

Assume that a state $a \in A$ occurs more than once in $\mathbf{a}$, and let $a_{i_1}, \ldots, a_{i_j}$ $(1 \le i_1 < \ldots < i_j \le n)$ be all occurrences of a in $\mathbf{a}$. Then the state-sequences of

$$\beta':\ b_0 q \Rightarrow_{\mathfrak{A}}^* (w\#)^m\ \big(\in (Z \cup \{\#\})^*\big)$$

at the subtrees $t_{i_1}, \ldots, t_{i_j}$ coincide. Let $(b_1, \ldots, b_s)$ be this common state-sequence.

Among $t_{i_1}, \ldots, t_{i_j}$ let t_{i_1} be the tree for which $\tau_{\mathfrak{B}(b_1)}(t_{i_1}) \ldots \tau_{\mathfrak{B}(b_s)}(t_{i_1})$ has a maximal number of occurrences of $\#$. Replace the considered occurrences of $t_{i_1}, \ldots, t_{i_j}$ in q by t_{i_1}, and denote by q' the resulting tree. We claim that for q' we have $\tau_{\mathfrak{B}}(q') = (w\#)^{m'}$ with $m' \geq m$. To prove it let us distinguish the following two cases:

(I) There exists an r $(1 \leq r \leq s)$ such that $\#$ occurs at least twice in the word $\tau_{\mathfrak{B}(b_r)}(t_{i_1})$. Then our claim obviously holds.

(II) $\#$ occurs at most once in each word $\tau_{\mathfrak{B}(b_1)}(t_{i_1}), \ldots, \tau_{\mathfrak{B}(b_s)}(t_{i_1})$. Take a fixed r $(1 < r \leq j)$, and write β' in the form

$$b_0 q \Rightarrow^*_{\mathfrak{B}} w_1 b_1 t_{i_r} w_2 \ldots w_s b_s t_{i_r} w_{s+1} \Rightarrow^*_{\mathfrak{B}} \\ w_1 v_1 w_2 \ldots w_s v_s w_{s+1} = (w\#)^m.$$

Since $m > 2k+1$ and $s \leq k$, there exists a w_u $(1 \leq u \leq s+1)$ such that $\#$ occurs at least twice in w_u. This also implies our claim.

Thus we have got the following result. If we replace in α' every subderivation $a_r p' \Rightarrow^*_{\mathfrak{A}} t_r$ $(a_r = a, r = i_1, \ldots, i_j)$ by $ap' \Rightarrow^*_{\mathfrak{A}} t_{i_1}$, then $b_0 q' \Rightarrow^*_{\mathfrak{B}} (w\#)^{m'}$ with $m' \geq m$ holds for the resulting output tree q'. Therefore, prescribing the applications of the productions of $\mathfrak{A}$ in this way, we arrive at a deterministic R-transformation whose composition by $\tau_{\mathfrak{B}}$, applied to a suitable forest from K, for each $w \in L$ yields infinitely many words $(w\#)^m$ $(m \geq 1)$, and only such words. Next we show how this can be carried out. First we define a deterministic R-transducer $\mathfrak{A}_1$.

Let $A = \{a_1 \ldots, a_s\}$, and define a set $\overline{X}$ of variables by

$$\overline{X} = \{(x, (c_1, \ldots, c_s)) \mid x \in X,\ c_i = (a_i x, q_i) \in P \text{ or } c_i = *,\ i = 1, \ldots, s\}$$

where $*$ is a new symbol. Moreover, define the ranked alphabet Δ, where for each m (≥ 0),

$$\Delta_m = \{(\sigma, (c_1, \ldots, c_s)) \mid \sigma \in \Sigma_m,\ c_i = (a_i \sigma, q_i) \in P \text{ or } c_i = *,\ i = 1, \ldots, s\}.$$

Now take the R-transducer $\mathfrak{A}_1 = (\Delta, \overline{X}, A, \Omega, Y, P_1, a_0)$ for which P_1 is given as follows:

(α) For each $a_i \in A$ and $(x, (c_1, \ldots, c_s)) \in \overline{X}$, if $c_i = (a_i x, q_i)$, then the production

$$a_i(x, (c_1, \ldots, c_s)) \to q_i$$

is in P_1.

(β) For each $a_i \in A$ and $(\sigma, (c_1, \ldots, c_s)) \in \Delta_m$, if $c_i = (a_i \sigma, q_i)$, then the production

$$a_i(\sigma, (c_1, \ldots, c_s)) \to q_i$$

is in P_1.

Obviously, $\mathfrak{A}_1$ is a deterministic R-transducer.

Next, let $\mathfrak{D} = (\Sigma, X, \{d_0\}, \Delta, \overline{X}, P'', d_0)$ be the F-relabeling where

$$P'' = \{x \to d_0\big(x, (c_1, \dots, c_s)\big) \mid x \in X,\ \big(x, (c_1, \dots, c_s)\big) \in \overline{X}\} \cup$$
$$\{\sigma(d_0, \dots, d_0) \to d_0\big(\sigma, (c_1, \dots, c_s)\big)(\xi_1, \dots, \xi_m) \mid \sigma \in \Sigma_m,$$
$$\big(\sigma, (c_1, \dots, c_s)\big) \in \Delta_m,\ m \geq 0\}.$$

Put $S = R\tau_{\mathfrak{D}}$. Since K is closed under relabeling, $S \in K$. Moreover, taking into consideration the remarks preceding the construction of $\mathfrak{A}_1$, one can easily see that, for each $w \in L$, $(S\tau_{\mathfrak{A}_1})\tau_{\mathfrak{B}}$ contains infinitely many words of the form $(w\#)^m$ $(m \geq 1)$, and only such words.

Finally, take the deterministic gsm $\mathbf{C} = (Z \cup \{\#\}, \{c_0, c_1\}, Z, c_0, P_{\mathbf{C}}, \{c_1\})$ where

$$P_{\mathbf{C}} = \{c_0 z \to z c_0 \mid z \in Z\} \cup \{c_0\# \to e c_1\} \cup \{c_1 \overline{z} \to e c_1 \mid \overline{z} \in Z \cup \{\#\}\}.$$

Obviously, $(w\#)^m \tau_{\mathbf{C}} = w$ for all $w \in Z^*$ and $m \geq 1$.

Denote by $\mathfrak{B}_1$ the deterministic k-copying R-transducer obtained from $\mathfrak{B}$ by Theorems 4.5.4 and 4.7.8. Moreover, let $\mathfrak{C}_1$ be the $\mathrm{DR_R}$-transducer given to $\mathbf{C}$ by Theorem 4.7.4. Then the equality $L = \mathrm{yd}(S\tau_{\mathfrak{A}_1} \circ \tau_{\mathfrak{B}_1} \circ \tau_{\mathfrak{C}_1})$ holds. Thus, by a repeated application of Theorem 4.4.6 (iii) and Corollary 4.4.8 (ii) and using Theorem 4.6.15 and Corollary 4.3.17, we get for a suitable deterministic R-transformation τ and a suitable $T \in K$ the equality $T\tau = S\tau_{\mathfrak{A}_1} \circ \tau_{\mathfrak{B}_1} \circ \tau_{\mathfrak{C}_1}$. (Observe that the F-transducer $\mathfrak{A}$ given in Lemma 4.1.11 is an F-relabeling. Hence, closure under relabeling implies closure under intersection with regular forests.) Finally, again by Theorem 4.5.4, we have $L \in \mathcal{DG}(T)$. □

Definition 4.7.17 Let X be an alphabet and $\# \notin X$ a symbol. Then for $L \subseteq X^*$ the language $c_2(L, \#)$ is defined by $c_2(L, \#) = \{w\#w \mid w \in L\}$.

Theorem 4.7.18 *Let K be a class of forests closed under relabeling and regular insertion. If $R \in K$, then there exist a 2-copying* GSDH*-transducer $\mathfrak{A}$ and a forest $T \in K$ such that $c_2(\mathrm{yd}(R), \#) = T\tau_{\mathfrak{A}}$.*

Proof. Suppose $R \subseteq F_\Sigma(X)$ and let $L = \mathrm{yd}(R)$. Moreover, take the ranked alphabet $\Delta = \Delta_1 = \{\overline{d} \mid d \in \Sigma \cup X\}$, and consider the regular insertion defined by $f(d) = \{\overline{d}(\xi_1)\}^{*\xi_1}$ $(d \in \Sigma \cup X)$, and set $S = f(R)$. Then $S \in K$. Finally, let $\Omega = \Sigma \cup \Delta$ be the ranked alphabet with $\Omega_1 = \Sigma_1 \cup \Delta$ and $\Omega_m = \Sigma_m$ $(m \geq 0,\ m \neq 1)$.

Now consider the R-relabeling $\mathfrak{B} = (\Omega, X, \{b_0, b_1\}, \Omega, X, P, b_0)$, where

$$P = \{b_0\overline{d} \to \overline{d}(b_1\xi_1) \mid d \in \Sigma \cup X\} \cup$$
$$\{b_1\sigma \to \sigma(b_1\xi_1, \dots, b_1\xi_m) \mid \sigma \in \Sigma_m,\ m \geq 0\} \cup$$
$$\{b_1 x \to x \mid x \in X\}.$$

Obviously, $T = S\tau_{\mathfrak{B}}$ consists of all trees of the form $\overline{d}(r)$, where $r \in R$ and $d = \mathrm{root}(r)$. Since $\mathfrak{B}$ is a relabeling, $T \in K$. Now we construct the required GSDT $\mathfrak{A} = (\Omega, X, \{a_0\}, X \cup \{\#\}, P', a_0)$, where

$$P' = \{a_0\overline{d} \to a_0\xi_1\#a_0\xi_1 \mid d \in \Sigma \cup X\} \cup$$
$$\{a_0\sigma \to a_0\xi_1 \dots a_0\xi_m \mid \sigma \in \Sigma_m,\ m \geq 0\} \cup \{a_0 x \to x \mid x \in X\}.$$

It is clear that $\mathfrak{A}$ is a 2-copying GSDH-transducer and that $c_2(L, \#) = T\tau_{\mathfrak{A}}$ holds. □

Theorem 4.7.19 *Let Y be an alphabet and $\# \notin Y$ a symbol. Take a language $L \subseteq Y^*$ and a class K of forests closed under relabeling and regular insertion. If $c_2(L,\#) \in \mathcal{G}(K)$, then $L \in \mathcal{DG}(K)$.*

Proof. The idea behind the proof is similar to that of Theorem 4.7.16, but this is much simpler.

Let $\mathfrak{A} = (\Sigma, X, A, Y \cup \{\#\}, P, A')$ be a GSDT and $R \in K$ a ΣX-forest such that $R\tau_{\mathfrak{A}} = c_2(L,\#)$. Since K is closed under regular insertion, we may assume that A' is a singleton, say $A' = \{a_0\}$.

Take a tree $p \in R$, a subtree p' of p and let $p = r(p')$ $\big(r \in \hat{F}_\Sigma(X \cup \Xi_1)\big)$. Consider a derivation

$$\alpha: \; a_0 r(p') \Rightarrow^* w_1 a_1 p' w_2 \dots w_k a_k p' w_{k+1} \Rightarrow^* w_1 v_1 w_2 \dots w_k v_k w_{k+1} = w\#w,$$

where $a_0 r(\xi_1) \Rightarrow^* w_1 a_1 \xi_1 w_2 \dots w_k a_k \xi_1 w_{k+1}$, $w_1, \dots, w_{k+1}, v_1, \dots, v_k \in (Y \cup \{\#\})^*$ and $a_i p' \Rightarrow^* v_i$ $(i = 1, \dots, k)$. Then $(a_1, \dots, a_k)$ is the state-sequence of α at p'. Assume that a state $a \in A$ occurs at least twice in $(a_1, \dots, a_k)$, and let a_{i_1} and a_{i_2} $(1 \le i_1 < i_2 \le k)$ be two such occurrences of a. Then, taking the relevant occurrences of v_{i_1} and v_{i_2} in $w\#w$, we have the decomposition $w\#w = u_1 v_{i_1} u_2 v_{i_2} u_3$. On the other hand the words $u_1 v_{i_j} u_2 v_{i_j} u_3$ $(j = 1, 2)$ are also in $R\tau_{\mathfrak{A}}$. Hence, $v_{i_1} = v_{i_2}$ must hold. This implies that if we replace for each t $(1 \le t \le k)$ such that $a_t = a$, $a_t p' \Rightarrow^* v_t$ by $a_t p' \Rightarrow^* v_{i_1}$, we get the same word $w\#w$. Therefore, prescribing accordingly the applications of productions from P, we arrive at a deterministic GSDT yielding $c_2(L,\#)$. This can be carried out in the same way as in the proof of Theorem 4.7.16, but here the resulting $\mathfrak{A}_1$ is a DGSDT. Thus, taking the F-relabeling $\mathfrak{D}$ defined in the proof of Theorem 4.7.16, for $S = R\tau_{\mathfrak{D}}$, we have $S \in K$ and $S\tau_{\mathfrak{A}_1} = c_2(L,\#)$. Moreover, by Theorem 4.5.4, there exists a DR-transducer $\mathfrak{B}_1$ with $c_2(L,\#) = \mathrm{yd}(S\tau_{\mathfrak{B}_1})$. Finally, consider the deterministic gsm $\mathbf{C}$ of the proof of Theorem 4.7.16 with Y instead of Z, and let $\mathfrak{C}_1$ be the corresponding $\mathrm{DR_R}$-transducer. Then the equality $L = \mathrm{yd}(S\tau_{\mathfrak{B}_1} \circ \tau_{\mathfrak{C}_1})$ holds. Thus, by Theorem 4.4.6 (iii), Corollary 4.4.8 (ii), Theorem 4.6.15 and Corollary 4.3.17, for suitable DR-transformation τ and a $T \in K$, we get $T\tau = S\tau_{\mathfrak{B}_1} \circ \tau_{\mathfrak{C}_1}$. This, by Theorem 4.5.4, implies $L \in \mathcal{DG}(T)$. □

4.8 THE HIERARCHIES OF TREE TRANSFORMATIONS, SURFACE FORESTS AND TRANSFORMATIONAL LANGUAGES

In this section we prove that the compositions of n F-transformations or n R-transformations form proper hierarchies when $n = 0, 1, 2, \dots$. Similar results will be shown for the classes of forests (n-surface forests) which can be obtained from regular forests by compositions of n F- or n R-transformations. All these results will follow from the fact that the classes of languages (n-transformational languages) obtained by taking the yields of n-surface forests form a proper hierarchy.

Definition 4.8.1 A forest T is an (n, R)-*surface forest* if $T \in \mathrm{Surf}(\mathcal{R}^n)$. (n, F)- and $(n, \mathrm{R_R})$-*surface forests* are defined in a similar way.

Definition 4.8.2 A (string) language L is an (n, R)-*transformational language* if $L = \mathrm{yd}(T)$ for some (n, R)-surface forest T. (n, F)- and $(n, \mathrm{R_R})$-*transformational languages* are defined similarly.

If $n = 1$ then we shall speak about R-, F- and $\mathrm{R_R}$-*transformational languages*, as well.

The following results show that in studying (n, R)-surface forests and (n, R)-transformational languages we can use $\mathrm{R_R}$-transformations, too.

Theorem 4.8.3 *For each natural number n, the equality* $\mathrm{Surf}(\mathcal{R}^n) = \mathrm{Surf}(\mathcal{R}^n_{\mathrm{R}})$ *holds.*

Proof. This follows from Theorems 4.4.6 (i) and 4.3.15 and Lemma 4.6.5. □

From Theorem 4.8.3 we directly get

Corollary 4.8.4 *For every natural number n, the class of (n, R)-transformational languages coincides with the class of $(n, \mathrm{R_R})$-transformational languages.* □

Using Theorems 4.4.7 (i) and 4.2.7, from Theorem 4.8.3 we obtain

Corollary 4.8.5 *For every natural number n,* $\mathrm{Surf}(\mathcal{R}^n)$ *is closed under* LF-*transformations and* LR-*transformations.* □

Now we can state and prove a result giving a recursive procedure by which the hierarchy theorems can be proved easily. The procedure will be based on the "bridge theorems" of the previous section which concern the operations res, c_2 and c_*. These associate with each language which is not in a given class another language which is not in another, larger class.

Theorem 4.8.6 *Let K be a class of forests closed under relabeling and regular insertion. If* $\mathrm{yd}\mathcal{DR}_f(K) \subset \mathrm{yd}\mathcal{R}(K)$*, then for each integer $n \geq 1$,*

$$\mathrm{yd}\mathcal{R}^n(K) \subset \mathrm{yd}\mathcal{DR}_f\big(\mathcal{R}^n(K)\big) \subset \mathrm{yd}\mathcal{DR}\big(\mathcal{R}^n(K)\big) \subset \mathrm{yd}\mathcal{R}^{n+1}(K).$$

Proof. By Theorem 4.3.15 and Lemma 4.7.3, $\mathcal{R}^n(K)$ is closed under relabeling and regular insertion, for every $n \geq 1$. In the sequel these facts will be used without further mention.

We shall proceed by induction on n. Let $n = 1$. Take a forest R such that $R \in \mathcal{R}(K)$ and $\mathrm{yd}(R) \notin \mathrm{yd}\mathcal{DR}_f(K)$. Then by Theorems 4.7.12, 4.5.4 and 4.2.8 there exist an LNF-transformation τ and a forest $S \in \mathcal{R}(K)$ such that $\mathrm{res}\big(\mathrm{yd}(R), \#\big) = \mathrm{yd}(S\tau)$. Moreover, by Theorem 4.3.15, $S\tau \in \mathcal{R}(K)$. On the other hand, since $\mathrm{yd}(R) \notin \mathrm{yd}\mathcal{DR}_f(K)$, by Theorems 4.7.13 and 4.5.4, $\mathrm{res}\big(\mathrm{yd}(R), \#\big) \notin \mathrm{yd}\mathcal{DR}(K)$. Thus, the proper inclusion $\mathrm{yd}\mathcal{DR}(K) \subset \mathrm{yd}\mathcal{R}(K)$ holds.

Next take an $R \in \mathcal{R}(K)$ with $\mathrm{yd}(R) \notin \mathrm{yd}\mathcal{DR}(K)$. Then, by Theorems 4.7.18 and 4.7.8, there exist a 2-copying homomorphism τ and a forest $S \in \mathcal{R}(K)$ such that $c_2\big(\mathrm{yd}(R), \#\big) = \mathrm{yd}(S\tau)$. On the other hand, since $\mathrm{yd}(R) \notin \mathrm{yd}\mathcal{DR}(K)$, by Theorems 4.5.4

and 4.7.19, $c_2\big(\mathrm{yd}(R), \#\big) \notin \mathrm{yd}\mathcal{R}(K)$. Therefore, the inclusion $\mathrm{yd}\mathcal{R}(K) \subset \mathrm{yd}\mathcal{DR}_f(\mathcal{R}(K))$ is valid.

Again take an $R \in \mathcal{R}(K)$ with $\mathrm{yd}(R) \notin \mathrm{yd}\mathcal{DR}(K)$. By Theorems 4.7.15 and 4.5.4 there exist a DR-transformation τ and a forest $S \in \mathcal{R}(K)$ such that, $c_*\big(\mathrm{yd}(R), \#\big) = \mathrm{yd}(S\tau)$. Moreover, since $\mathrm{yd}(R) \notin \mathrm{yd}\mathcal{DR}(K)$, by Theorems 4.7.16 and 4.7.8, $c_*\big(\mathrm{yd}(R), \#\big) \notin \mathrm{yd}\mathcal{DR}_f\big(\mathcal{R}(K)\big)$. Thus we have got that

$$\mathrm{yd}\mathcal{DR}_f\big(\mathcal{R}(K)\big) \subset \mathrm{yd}\mathcal{DR}\big(\mathcal{R}(K)\big).$$

Finally, take an $R \in \mathcal{R}^2(K)$ with $\mathrm{yd}(R) \notin \mathrm{yd}\mathcal{DR}_f\big(\mathcal{R}(K)\big)$. Then again by Theorems 4.7.12 and 4.5.4, there exist an LNF-transformation τ and a forest $S \in \mathcal{R}^2(K)$ such that $\mathrm{res}\big(\mathrm{yd}(R), \#\big) = \mathrm{yd}(S\tau)$. Moreover, by Theorem 4.3.15, $S\tau \in \mathcal{R}^2(K)$. On the other hand, since $\mathrm{yd}(R) \notin \mathrm{yd}\mathcal{DR}_f\big(\mathcal{R}(K)\big)$, by Theorems 4.7.13 and 4.7.8, $\mathrm{res}\big(\mathrm{yd}(R), \#\big) \notin \mathrm{yd}\mathcal{DR}\big(\mathcal{R}(K)\big)$. Therefore, $\mathrm{yd}\mathcal{DR}\big(\mathcal{R}(K)\big) \subset \mathrm{yd}\mathcal{R}^2(K)$.

Summarizing our results, we have

$$\mathrm{yd}\mathcal{R}(K) \subset \mathrm{yd}\mathcal{DR}_f\big(\mathcal{R}(K)\big) \subset \mathrm{yd}\mathcal{DR}\big(\mathcal{R}(K)\big) \subset \mathrm{yd}\mathcal{R}^2(K)$$

which completes the proof for $n = 1$.

The transition from n to $n+1$ is illustrated by Fig. 4.4. □

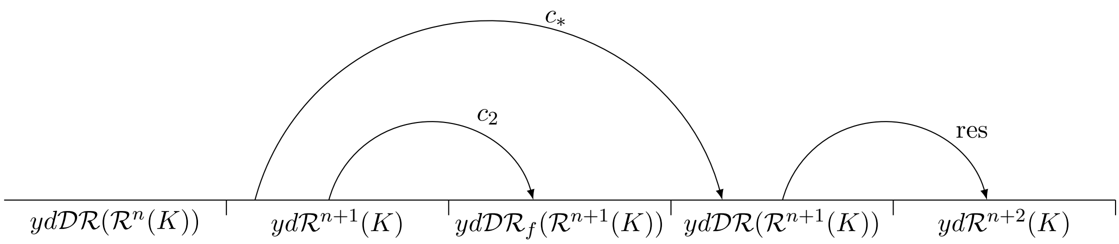

Figure 4.4.

According to Theorem 4.8.6, to show that the classes of (n,R)-transformational languages form a proper hierarchy it is enough to prove the properness of the inclusion $\mathrm{yd}\mathcal{DR}_f(\mathrm{Rec}) \subset \mathrm{yd}\mathcal{R}(\mathrm{Rec})$. For this we need

Lemma 4.8.7 *For each k-copying* DGSDT $\mathfrak{A} = (\Sigma, X, A, Y, P, a_0)$ *there exists a linear* DGSDT $\mathfrak{B} = (\Sigma, X, B, Y, P', b_0)$ *such that* $\mathrm{Par}(T\tau_{\mathfrak{B}}) = \mathrm{Par}(T\tau_{\mathfrak{A}})$, *for every forest* $T \subseteq F_\Sigma(X)$.

Proof. For each $w \in (Y \cup A\Xi)^*$, let $\overline{w}$ denote the word obtained from w by erasing all $a\xi$'s ($a \in A,\ \xi \in \Xi$).

Let $B = \{(a_1, \ldots, a_n) \mid n \le k,\ a_i \in A(i = 1, \ldots, n)\}$ and $b_0 = (a_0)$. Moreover, P' is defined in the following way:

(i) Let $\mathbf{a} = (a_1, \ldots, a_n) \in B$ and $x \in X$ be arbitrary. Assume that the productions $a_i x \to v_i$ ($a_i \in A,\ v_i \in Y^*,\ i = 1, \ldots, n$) are in P. Then the production $\mathbf{a}x \to v_1 \ldots v_n$ is in P'.

(ii) Take an arbitrary $\mathbf{a} = (a_1, \ldots, a_n) \in B$ and $\sigma \in \Sigma_m$ ($m \geq 0$). Suppose P contains, for each $i = 1, \ldots, n$, a production

$$a_i\sigma \to w_{ij_1} a_{ij_1} \xi_j w_{ij_2} \ldots w_{ij_{i_j}} a_{ij_{i_j}} \xi_j w_{ij_{i_j+1}} = w_i$$
$$\big(w_{ij_1}, \ldots, w_{ij_{i_j+1}} \in (Y \cup A(\Xi_m - \{\xi_j\}))^*,\ a_{ij_1}, \ldots, a_{ij_{i_j}} \in A,\ 1 \leq j \leq m\big).$$

Then the production

$$\mathbf{a}\sigma \to (a_{11_1}, \ldots, a_{11_{1_1}}, \ldots, a_{n1_1}, \ldots, a_{n1_{n_1}})\xi_1 \ldots$$
$$\ldots (a_{1m_1}, \ldots, a_{1m_{1_m}}, \ldots, a_{nm_1}, \ldots, a_{nm_{n_m}})\xi_m \overline{w}_1 \ldots \overline{w}_n$$

is in P', provided that $1_j + \ldots + n_j \leq k$ ($j = 1, \ldots, m$).

Obviously, $\mathfrak{B}$ is a linear DGSDT. Moreover, the derivations in $\mathfrak{A}$ and in $\mathfrak{B}$ are related as follows. Take a vector $\mathbf{a} \in A^n$ ($n \leq k$) and a tree $p \in F_\Sigma(X)$. Consider the derivations $\alpha:\ \mathbf{a}p^n \Longrightarrow^*_{\mathfrak{A}} w$, where $w = w_1 \ldots w_n \in Y^*$ and $\alpha_i:\ a_i p \Longrightarrow^*_{\mathfrak{A}} w_i$ ($i = 1, \ldots, n$). By the state-sequence of α at a node d of p we mean $(\mathbf{a}_1, \ldots, \mathbf{a}_n)$, where $\mathbf{a}_i$ ($1 \leq i \leq n$) is the state-sequence of α_i at d. Furthermore, we say that α is k-copying if the length of the state-sequence of α at any node of p is at most k. Assume that α is k-copying. Then for some $w' \in Y^*$, $\beta\colon \mathbf{a}p^n \Longrightarrow^*_{\mathfrak{B}} w'$ exists. One can easily show by induction on $\mathrm{hg}(p)$ that the state-sequence of β at any node d of p is of length one (if it exists) and coincides, as a sequence of states of $\mathfrak{A}$, with the state-sequence of α at d. Finally, w is a permutation of w'. Therefore, the equality $\mathrm{Par}(T\tau_{\mathfrak{A}}) = \mathrm{Par}(T\tau_{\mathfrak{B}})$ holds. □

From Lemma 4.8.7, by Theorems 1.6.17 and 4.5.4 and Corollary 4.6.6, we get

Corollary 4.8.8 *Let $T \subseteq F_\Sigma(X)$ be a recognizable forest and $\mathfrak{A} = (\Sigma, X, A, Y, P, a_0)$ a finite-copying* DGSDT. *Then* $\mathrm{Par}(T\tau_{\mathfrak{A}})$ *is semilinear.* □

We now can state and prove that the hierarchy of (n,R)-transformational languages is infinite.

Theorem 4.8.9 *For every natural number n, the inclusions*

$$\mathrm{yd}\mathcal{R}^n(\mathrm{Rec}) \subset \mathrm{yd}\mathcal{DR}_f\big(\mathcal{R}^n(\mathrm{Rec})\big) \subset \mathrm{yd}\mathcal{DR}\big(\mathcal{R}^n(\mathrm{Rec})\big) \subset \mathrm{yd}\mathcal{R}^{n+1}(\mathrm{Rec})$$

hold.

Proof. By Lemma 4.7.2 and Corollary 4.6.6, Rec is closed under regular insertion and relabeling. Thus, by Theorems 4.8.6, 4.5.4, and 4.7.8, and Corollary 4.8.8, it is enough to show that there exist a regular forest $T \subseteq F_\Sigma(X)$ and a GSDT $\mathfrak{A} = (\Sigma, X, A, Y, P, a_0)$ such that $\mathrm{Par}(T\tau_{\mathfrak{A}})$ is not semilinear. For this let $\Sigma = \Sigma_1 = \{\sigma\}$, $A = \{a_0\}$, $X = \{x\}$,

$Y = \{y\}$ and $P = \{a_0\sigma \to a_0\xi_1 a_0\xi_1,\ a_0 x \to y\}$. Moreover, let $T = \{\sigma(x)\}^{*x}$. Then $T\tau_{\mathfrak{A}} = \{y^{2^n} \mid n = 0, 1, \ldots\}$. Thus, $\mathrm{Par}(T\tau_{\mathfrak{A}}) = \{\langle 2^n \rangle \mid n = 0, 1, \ldots\}$, which is obviously not semilinear. □

From Theorem 4.8.9 we directly get

Corollary 4.8.10 *For every natural number n the inclusions*

(i) $\mathrm{yd}\mathcal{R}^n(\mathrm{Rec}) \subset \mathrm{yd}\mathcal{R}^{n+1}(\mathrm{Rec})$,

(ii) $\mathcal{R}^n(\mathrm{Rec}) \subset \mathcal{R}^{n+1}(\mathrm{Rec})$,

(iii) $\mathcal{R}^n \subset \mathcal{R}^{n+1}$

hold. □

Finally, we give two more hierarchies of transformational languages, surface forests and tree transformations.

Theorem 4.8.11 *For every natural number n the inclusions*

$$\mathrm{yd}\mathcal{R}^n(\mathrm{Rec}) \subset \mathrm{yd}\,\mathcal{F}^{n+1}(\mathrm{Rec}) \subset \mathrm{yd}\mathcal{R}^{n+1}(\mathrm{Rec})$$

are valid.

Proof. By Theorems 4.3.3 and 4.3.12 and Corollary 4.6.6, the inclusions $\mathrm{yd}\mathcal{R}^n(\mathrm{Rec}) \subseteq \mathrm{yd}\,\mathcal{F}^{n+1}(\mathrm{Rec}) \subseteq \mathrm{yd}\mathcal{R}^{n+1}(\mathrm{Rec})$ hold. By the proofs of Theorems 4.8.6 and 4.8.9, $\mathrm{yd}\mathcal{R}^n(\mathrm{Rec})$ is a proper subclass of $\mathrm{yd}\mathcal{H}\big(\mathcal{R}^n(\mathrm{Rec})\big)$. Moreover, by Theorems 4.3.3 and 4.3.12 and Corollary 4.6.6, the equality $\mathcal{H}\big(\mathcal{R}^n(\mathrm{Rec})\big) = \mathcal{F}^{n+1}(\mathrm{Rec})$ holds. Thus, the inclusion $\mathrm{yd}\mathcal{R}^n(\mathrm{Rec}) \subset \mathrm{yd}\mathcal{F}^{n+1}(\mathrm{Rec})$ is valid. Finally, by Theorem 4.8.9, $\mathrm{yd}\mathcal{H}\big(\mathcal{R}^n(\mathrm{Rec})\big) \subseteq \mathrm{yd}\mathcal{DR}\big(\mathcal{R}^n(\mathrm{Rec})\big) \subset \mathrm{yd}\mathcal{R}^{n+1}(\mathrm{Rec})$. Therefore, the inclusion $\mathrm{yd}\,\mathcal{F}^{n+1}(\mathrm{Rec}) \subset \mathrm{yd}\mathcal{R}^{n+1}(\mathrm{Rec})$ is also valid. □

From Theorem 4.8.11, using Theorems 4.3.3 and 4.3.12 and Corollary 4.6.6, we get the following results.

Corollary 4.8.12 *For every natural number n the inclusions*

$$\mathcal{R}^n(\mathrm{Rec}) \subset \mathcal{F}^{n+1}(\mathrm{Rec}) \subset \mathcal{R}^{n+1}(\mathrm{Rec})$$

hold. □

Corollary 4.8.13 *For every natural number n the inclusions*

(i) $\mathrm{yd}\mathcal{F}^n(\mathrm{Rec}) \subset \mathrm{yd}\mathcal{F}^{n+1}(\mathrm{Rec})$,

(ii) $\mathcal{F}^n(\mathrm{Rec}) \subset \mathcal{F}^{n+1}(\mathrm{Rec})$,

(iii) $\mathcal{F}^n \subset \mathcal{F}^{n+1}$

are valid. □

4.9 THE EQUIVALENCE OF TREE TRANSDUCERS

Since the equivalence problem for (nondeterministic) generalized sequential machines is undecidable, there exists no algorithm to decide for arbitrary two tree transducers whether or not they are equivalent. In this section we show that there is an algorithm for deciding the equivalence of two tree transducers when at least one of them induces a partial mapping. Moreover, we shall prove that it is decidable whether the tree transformation induced by a given tree transducer is a partial mapping when restricted to a given recognizable forest.

We start by introducing a concept.

Definition 4.9.1 Let $p \in F_\Sigma(X)$. A tree $p' \in \hat{\hat{F}}_\Sigma(X \cup \Xi^n)$ is called a *supertree* of p if there are trees $p_1, \ldots, p_n \in F_\Sigma(X)$ such that $p = p'(p_1, \ldots, p_n)$.

To prove the decidability results we shall give five reduction rules formulated in the following five lemmas. In these lemmas $\mathfrak{A} = (\Sigma, X, A, \Omega, Y, P, A')$ will be a fixed R-transducer and $\mathbf{B} = (\mathcal{B}, \beta, B')$ will be a fixed ΣX-recognizer with $\mathcal{B} = (B, \Sigma)$ and $T(\mathbf{B}) = T$. Furthermore, set $Q = \{p \in T \mid |p\tau_{\mathfrak{A}}| \geq 2\}$, i.e., Q consists of all trees from T which are translated into at least two different output trees by $\mathfrak{A}$.

Lemma 4.9.2 *Let* $p_1, p_2 \in \hat{F}_\Sigma(X \cup \Xi_1)$, $p_3 \in F_\Sigma(X)$, $n_1, n_1', n_2, n_2' \geq 0$, $q_1 \in \hat{F}_\Omega(Y \cup \Xi_{n_1})$, $q_1' \in \hat{F}_\Omega(Y \cup \Xi_{n_1'})$, $\mathbf{q}_2 \in \hat{F}_\Omega^{n_1}(Y \cup \Xi_{n_2})$, $\mathbf{q}_2' \in \hat{F}_\Omega^{n_1'}(Y \cup \Xi_{n_2'})$, $\mathbf{q}_3 \in F_\Omega(Y)^{n_2}$, $\mathbf{q}_3' \in F_\Omega(Y)^{n_2'}$, $a_0, a_0' \in A'$ *and* $\mathbf{a}_i \in A^{n_i}$, $\mathbf{a}_i' \in A^{n_i'}$ $(i = 1, 2)$. *Moreover, set* $A_i = \{a_{i_j} \mid j = 1, \ldots, n_i\}$ *and* $A_i' = \{a_{i_j}' \mid j = 1, \ldots, n_i'\}$ $(i = 1, 2)$. *Assume that the following conditions are satisfied:*

(i) $p_1\big(p_2(p_3)\big) \in T$,

(ii) $a_0 p_1 \Rightarrow^* q_1(\mathbf{a}_1 \xi_1^{n_1})$, $a_0' p_1 \Rightarrow^* q_1'(\mathbf{a}_1' \xi_1^{n_1'})$,

(iii) $\mathbf{a}_1 p_2^{n_1} \Rightarrow^* \mathbf{q}_2(\mathbf{a}_2 \xi_1^{n_2})$, $\mathbf{a}_1' p_2^{n_1'} \Rightarrow^* \mathbf{q}_2'(\mathbf{a}_2' \xi_1^{n_2'})$,

(iv) $\mathbf{a}_2 p_3^{n_2} \Rightarrow^* \mathbf{q}_3$, $\mathbf{a}_2' p_3^{n_2'} \Rightarrow^* \mathbf{q}_3'$,

(v) $p_3\hat{\beta} = p_2(p_3)\hat{\beta}$, $A_1 \subseteq A_2$, $A_1' \subseteq A_2'$,

(vi) *for all* $\mathbf{r} \in F_\Omega(Y)^{n_1}$ *and* $\mathbf{r}' \in F_\Omega(Y)^{n_1'}$, $q_1(\mathbf{r}) \neq q_1'(\mathbf{r}')$.

Then $p_1(p_3) \in Q$.

Proof. First let us note that the conditions of Lemma 4.9.2 imply $p_1\big(p_2(p_3)\big) \in Q$.

Next take two mappings $f \colon \{1, \ldots, n_1\} \to \{1, \ldots, n_2\}$ and $g \colon \{1, \ldots, n_1'\} \to \{1, \ldots, n_2'\}$ such that $a_{1_i} = a_{2_{f(i)}}$ $(i = 1, \ldots, n_1)$ and $a_{1_i}' = a_{2_{g(i)}}'$ $(i = 1, \ldots, n_1')$. By (v), there are such mappings f and g. Thus, by (iv), we have $\mathbf{a}_1 p_3^{n_1} \Rightarrow^* \mathbf{r}$ and $\mathbf{a}_1' p_3^{n_1'} \Rightarrow^* \mathbf{r}'$ with $\mathbf{r} = (q_{3_{f(1)}}, \ldots, q_{3_{f(n_1)}})$ and $\mathbf{r}' = (q_{3_{g(1)}}', \ldots, q_{3_{g(n_1')}}')$. This, by (ii) implies $a_0 p_1(p_3) \Rightarrow^* q_1(\mathbf{r})$ and $a_0' p_1(p_3) \Rightarrow^* q_1'(\mathbf{r}')$. By (vi), $q_1(\mathbf{r}) \neq q_1'(\mathbf{r}')$. Moreover by (v), $p_1(p_3) \in T$. Therefore, $p_1(p_3) \in Q$. $\square$

Lemma 4.9.3 *Let* $p_1 \in \hat{F}_\Sigma(X \cup \Xi_1)$, $p_2 \in F_\Sigma(X)$, $n, n' > 0$, $q_1 \in \hat{F}_\Omega(Y \cup \Xi_n)$, $q_1' \in \hat{F}_\Omega(Y \cup \Xi_{n'})$, $\mathbf{q}_2 \in F_\Omega(Y)^n$, $\mathbf{q}_2' \in F_\Omega(Y)^{n'}$, $a_0, a_0' \in A'$, $\mathbf{a} \in A^n$ *and* $\mathbf{a}' \in A^{n'}$. *Furthermore, let* K *be the maximum of the heights of the right-hand sides of the productions from* P. *Assume that the following conditions are satisfied:*

(i) $p_1(p_2) \in T$,

(ii) $a_0 p_1 \Rightarrow^* q_1(\mathbf{a}\xi_1^n)$, $a_0' p_1 \Rightarrow^* q_1'(\mathbf{a}'\xi_1^{n'})$,

(iii) $\mathbf{a}p_2^n \Rightarrow^* \mathbf{q}_2$, $\mathbf{a}'p_2^{n'} \Rightarrow^* \mathbf{q}_2'$,

(iv) $\mathrm{path}_1(q_1)$ *is an initial segment of* $\mathrm{path}_1(q_1')$, *and*
$l\big(\mathrm{path}_1(q_1')\big) - l\big(\mathrm{path}_1(q_1)\big) > |\mathfrak{p}A|^2|B|K$, $\mathrm{hg}(p_2) \geq |\mathfrak{p}A|^2|B|$.

Then there exists an $r \in F_\Sigma(X)$ *with* $|r| < |p_2|$ *such that* $p_1(r) \in Q$.

Proof. Set $R = \{r \in F_\Sigma(X) \mid p_1(r) \in T,\ |r| \leq |p_2|,\ \mathbf{a}r^n \Rightarrow^* \mathbf{s},\ \mathbf{a}'r^{n'} \Rightarrow^* \mathbf{s}'$ for some $\mathbf{s} \in F_\Omega(Y)^n$ and $\mathbf{s}' \in F_\Omega(Y)^{n'}\}$. Obviously, R is nonvoid. Denote by r an element from R with minimal length. We prove that $p_1(r) \in Q$ and $\mathrm{hg}(r) < |\mathfrak{p}A|^2|B|$.

First assume that $\mathrm{hg}(r) \geq |\mathfrak{p}A|^2|B|$. Then there are

$$r_1, r_2 \in \hat{F}_\Sigma(X \cup \Xi_1),\ r_3 \in F_\Sigma(X),\ m_1, m_1', m_2, m_2' \geq 0,\ \mathbf{s}_1 \in \hat{F}_\Omega^n(Y \cup \Xi_{m_1}),$$
$$\mathbf{s}_1' \in \hat{F}_\Omega^{n'}(Y \cup \Xi_{m_1'}),\ \mathbf{s}_2 \in \hat{F}_\Omega^{m_1}(Y \cup \Xi_{m_2}),\ \mathbf{s}_2' \in \hat{F}_\Omega^{m_1'}(Y \cup \Xi_{m_2'}),$$
$$\mathbf{s}_3 \in F_\Omega(Y)^{m_2},\ \mathbf{s}_3' \in F_\Omega(Y)^{m_2'},\ \mathbf{b}_i \in A^{m_i},\ \mathbf{b}_i' \in A^{m_i'}\ (i = 1, 2) \text{ such}$$

that

(I) $r = r_1\big(r_2(r_3)\big)$, $r_2 \neq \xi_1$,

(II) $\mathbf{a}r_1^n \Rightarrow^* \mathbf{s}_1(\mathbf{b}_1\xi_1^{m_1})$, $\mathbf{a}'r_1^{n'} \Rightarrow^* \mathbf{s}_1'(\mathbf{b}_1'\xi^{m_1'})$,

(III) $\mathbf{b}_1 r_2^{m_1} \Rightarrow^* \mathbf{s}_2(\mathbf{b}_2\xi_1^{m_2})$, $\mathbf{b}_1' r_2^{m_1'} \Rightarrow^* \mathbf{s}_2'(\mathbf{b}_2'\xi_1^{m_2'})$,

(IV) $\mathbf{b}_2 r_3^{m_2} \Rightarrow^* \mathbf{s}_3$, $\mathbf{b}_2' r_3^{m_2'} \Rightarrow^* \mathbf{s}_3'$,

(V) $r_3\hat{\beta} = r_2(r_3)\hat{\beta}$, $B_1 \subseteq B_2$ and $B_1' \subseteq B_2'$, where $B_i = \{b_{i_j} \mid 1 \leq j \leq m_i\}$,
$B_i' = \{b_{i_j}' \mid 1 \leq i \leq m_i'\}$ $(i = 1, 2)$.

Take two mappings $f\colon \{1, \ldots, m_1\} \to \{1, \ldots, m_2\}$ and $g\colon \{1, \ldots, m_1'\} \to \{1, \ldots, m_2'\}$ such that $b_{1_i} = b_{2_{f(i)}}$ $(1 \leq i \leq m_1)$ and $b_{1_i}' = b_{2_{g(i)}}'$ $(1 \leq i \leq m_1')$. Obviously, $\mathbf{a}t^n \Rightarrow^* \mathbf{s}_1(s_{3_{f(1)}}, \ldots, s_{3_{f(m_1)}})$ and $\mathbf{a}'t^{n'} \Rightarrow^* \mathbf{s}_1'(s_{3_{g(1)}}', \ldots, s_{3_{g(m_1')}}')$, where $t = r_1(r_3)$. Moreover, $r_1(r_3)\hat{\beta} = r\hat{\beta}$ also holds. Therefore, $r_1(r_3) \in R$, which is a contradiction since $|r_1(r_3)| < |r|$.

Thus, we got that $\mathrm{hg}(r) < |\mathfrak{p}A|^2|B|$. Therefore, for arbitrary vectors $\mathbf{s} \in F_\Omega(Y)^n$ and $\mathbf{s}' \in F_\Omega(Y)^{n'}$ satisfying $\mathbf{a}r^n \Rightarrow^* \mathbf{s}$ and $\mathbf{a}'r^{n'} \Rightarrow^* \mathbf{s}'$, the inequalities $\mathrm{hg}(s_1), \mathrm{hg}(s_1') \leq |\mathfrak{p}A|^2|B|K$ hold. This, by (iv), obviously implies the conclusion of Lemma 4.9.3. □

Lemma 4.9.4 *Let* $p_1, p_2, p_3 \in \hat{F}_\Sigma(X \cup \Xi_1)$, $p_4 \in F_\Sigma(X)$, $n_i, n_i', m_i \geq 0$ $(i = 1, 2, 3)$,

$$\begin{gathered} q_1 \in \hat{F}_\Omega(Y \cup \Xi_{n_1+1}),\ q_1' \in \hat{F}_\Omega(Y \cup \Xi_{n_1'+1}),\ r_1 \in \hat{F}_\Omega(Y \cup \Xi_{m_1}), \\ \mathbf{q}_2 \in \hat{F}_\Omega^{n_1}(Y \cup \Xi_{n_2}),\ \mathbf{q}_2' \in \hat{F}_\Omega^{n_1'}(Y \cup \Xi_{n_2'}),\ \mathbf{r}_2 \in \hat{F}_\Omega^{m_1}(Y \cup \Xi_{m_2}), \\ \mathbf{q}_3 \in \hat{F}_\Omega^{n_2}(Y \cup \Xi_{n_3}),\ \mathbf{q}_3' \in \hat{F}_\Omega^{n_2'}(Y \cup \Xi_{n_3'}),\ \mathbf{r}_3 \in \hat{F}_\Omega^{m_2}(Y \cup \Xi_{m_3}), \\ \mathbf{q}_4 \in F_\Omega(Y)^{n_3},\ \mathbf{q}_4' \in F_\Omega(Y)^{n_3'},\ \mathbf{r}_4 \in F_\Omega(Y)^{m_3}, \\ a_0, a_0' \in A',\ a \in A,\ \mathbf{a}_i \in A^{n_i},\ \mathbf{a}_i' \in A^{n_i'},\ \mathbf{b}_i \in A^{m_i}\ (i = 1, 2, 3). \end{gathered}$$

Moreover, take an $r \in F_\Omega(Y)$, *and let* $r' = r_1\big(\mathbf{r}_2(\mathbf{r}_3(\mathbf{r}_4))\big)$. *Finally, set* $A_i = \{a_{i_j} \mid j = 1, \ldots, n_i\}$, $A_i' = \{a_{i_j}' \mid j = 1, \ldots, n_i'\}$ *and* $B_i = \{b_{i_j}' \mid j = 1, \ldots, m_i\}$ $(i = 1, 2, 3)$. *Assume that the following conditions are satisfied:*

(i) $p_1\big(p_2(p_3(p_4))\big) \in T$,

(ii) $a_0 p_1 \Rightarrow^* q_1(a\xi_1, \mathbf{a}_1\xi_1^{n_1})$, $a_0' p_1 \Rightarrow^* q_1'\big(r_1(\mathbf{b}_1\xi_1^{m_1}), \mathbf{a}_1'\xi_1^{n_1'}\big)$,

(iii) $\mathbf{a}_1 p_2^{n_1} \Rightarrow^* \mathbf{q}_2(\mathbf{a}_2\xi_1^{n_2})$, $\mathbf{a}_1' p_2^{n_1'} \Rightarrow^* \mathbf{q}_2'(\mathbf{a}_2'\xi_1^{n_2'})$,
$ap_2 \Rightarrow^* a\xi_1$, $\mathbf{b}_1 p_2^{m_1} \Rightarrow^* \mathbf{r}_2(\mathbf{b}_2\xi_1^{m_2})$,

(iv) $\mathbf{a}_2 p_3^{n_2} \Rightarrow^* \mathbf{q}_3(\mathbf{a}_3\xi_1^{n_3})$, $\mathbf{a}_2' p_3^{n_2'} \Rightarrow^* \mathbf{q}_3'(\mathbf{a}_3'\xi_1^{n_3'})$,
$ap_3 \Rightarrow^* a\xi_1$, $\mathbf{b}_2 p_3^{m_2} \Rightarrow^* \mathbf{r}_3(\mathbf{b}_3\xi_1^{m_3})$,

(v) $\mathbf{a}_3 p_4^{n_3} \Rightarrow^* \mathbf{q}_4$, $\mathbf{a}_3' p_4^{n_3'} \Rightarrow^* \mathbf{q}_4'$, $ap_4 \Rightarrow^* r$, $\mathbf{b}_3 p_4^{m_3} \Rightarrow^* \mathbf{r}_4$,

(vi) $p_4\hat{\beta} = p_3(p_4)\hat{\beta} = p_2\big(p_3(p_4)\big)\hat{\beta}$, $A_1 \subseteq A_2 \subseteq A_3$,
$A_1' \subseteq A_2' \subseteq A_3'$, $B_1 = B_2 \subseteq B_3$,

(vii) $r \neq r'$, $\mathrm{path}_1(q_1) = \mathrm{path}_1(q_1')$.

Then at least one of the trees $p_1\big(p_2(p_4)\big)$, $p_1\big(p_3(p_4)\big)$ *and* $p_1(p_4)$ *is in* Q.

Proof. First note that the conditions of Lemma 4.9.4 imply $p_1\big(p_2(p_3(p_4))\big) \in Q$. Indeed, let $t = q_1\big(\xi_1, \mathbf{q}_2(\mathbf{q}_3(\mathbf{q}_4))\big)$ and $t' = q_1'\big(\xi_1, \mathbf{q}_2'(\mathbf{q}_3'(\mathbf{q}_4'))\big)$. Then

$$a_0 p_1\big(p_2(p_3(p_4))\big) \Rightarrow^* t(r), \quad a_0' p_1\big(p_2(p_3(p_4))\big) \Rightarrow^* t'(r')$$

and $t(r) \neq t(r')$.

Take six mappings $f_i \colon \{1, \ldots, n_i\} \to \{1, \ldots, n_{i+1}\}$, $g_i \colon \{1, \ldots, n_i'\} \to \{1, \ldots, n_{i+1}'\}$ and

$$h_i \colon \{1, \ldots, m_i\} \to \{1, \ldots, m_{i+1}\}\ (i = 1, 2)$$

such that

$$\begin{gathered} a_{i_j} = a_{i+1_{f_i(j)}}\ (i = 1, 2,\ 1 \leq j \leq n_i)\ a_{i_j}' = a_{i+1_{g_i(j)}}'\ (i = 1, 2,\ 1 \leq j \leq n_i'), \\ b_{i_j} = b_{i+1_{h_i(j)}}\ (i = 1, 2,\ 1 \leq j \leq m_i). \end{gathered}$$

Furthermore, set $f_3 = f_1 \circ f_2$, $g_3 = g_1 \circ g_2$ and $h_3 = h_1 \circ h_2$. Moreover, introduce the notations

$$\begin{aligned}
&\mathbf{s}_1 = (q_{3_{f_1(1)}}, \ldots, q_{3_{f_1(n_1)}})(\mathbf{q}_4),\ \mathbf{s}'_1 = (q'_{3_{g_1(1)}}, \ldots, q'_{3_{g_1(n'_1)}})(\mathbf{q}'_4),\\
&\mathbf{t}_1 = (r_{3_{h_1(1)}}, \ldots, r_{3_{h_1(m_1)}})(\mathbf{r}_4),\\
&\mathbf{s}_2 = \mathbf{q}_2(q_{4_{f_2(1)}}, \ldots, q_{4_{f_2(n_2)}}),\ \mathbf{s}'_2 = \mathbf{q}'_2(q'_{4_{g_2(1)}}, \ldots, q'_{4_{g_2(n'_2)}}),\\
&\mathbf{t}_2 = \mathbf{r}_2(r_{4_{h_2(1)}}, \ldots, r_{4_{h_2(m_2)}}),\\
&\mathbf{s}_3 = (q_{4_{f_3(1)}}, \ldots, q_{4_{f_3(n_1)}}),\ \mathbf{s}'_3 = (q'_{4_{g_3(1)}}, \ldots, q'_{4_{g_3(n'_1)}}),\\
&\mathbf{t}_3 = (r_{4_{h_3(1)}}, \ldots, r_{4_{h_3(m_1)}}).
\end{aligned}$$

Then the following derivations obviously hold:

$$\begin{gathered}
a_0 p_1\big(p_3(p_4)\big) \Rightarrow^* q_1(r, \mathbf{s}_1),\ a'_0 p_1\big(p_3(p_4)\big) \Rightarrow^* q'_1\big(r_1(\mathbf{t}_1), \mathbf{s}'_1\big),\\
a_0 p_1\big(p_2(p_4)\big) \Rightarrow^* q_1(r, \mathbf{s}_2),\ a'_0 p_1\big(p_2(p_4)\big) \Rightarrow^* q'_1\big(r_1(\mathbf{t}_2), \mathbf{s}'_2\big),\\
a_0 p_1(p_4) \Rightarrow^* q_1(r, \mathbf{s}_3),\ a'_0 p_1(p_4) \Rightarrow^* q'_1\big(r_1(\mathbf{t}_3), \mathbf{s}'_3\big).
\end{gathered}$$

It is also obvious that $p_1\big(p_3(p_4)\big), p_1\big(p_2(p_4)\big), p_1(p_4) \in T$.

Now assume that $p_1\big(p_2(p_4)\big) \notin Q$. Then, by (vi) and (vii), $m_1, m_2, m_3 > 0$ and there exists an i $(1 \le i \le m_2)$ such that $r_{3_i}(\mathbf{r}_4) \ne r_{4_{h_2(i)}}$. We can choose h_1 in such a way that for some j $(1 \le j \le m_1)$ $h_1(j) = i$ holds. Now assume that, under the latter choice of h_1, none of $p_1\big(p_3(p_4)\big)$ and $p_1(p_4)$ are in Q. Then we get $r_1(\mathbf{t}_1) = r_1(\mathbf{t}_3) = r$. But this is impossible since $t_{1_j} \ne t_{3_j}$. □

Lemma 4.9.5 *Let* $p_1, p_2, p_3 \in \hat{F}_\Sigma(X \cup \Xi_1)$, $p_4 \in F_\Sigma(X)$, $n_i, n'_i, m_i \ge 0$ $(i = 1, 2, 3)$,

$$\begin{gathered}
q_1 \in \hat{F}_\Omega(Y \cup \Xi_{n_1+1}),\ q'_1 \in \hat{F}_\Omega(Y \cup \Xi_{n'_1+1}),\ r_1 \in \hat{F}_\Omega(Y \cup \Xi_{m_1}),\\
\mathbf{q}_2 \in \hat{F}_\Omega^{n_1}(Y \cup \Xi_{n_2}),\ \mathbf{q}'_2 \in \hat{F}_\Omega^{n'_1}(Y \cup \Xi_{n'_2}),\ \mathbf{r}_2 \in \hat{F}_\Omega^{m_1}(Y \cup \Xi_{m_2}),\\
\mathbf{q}_3 \in \hat{F}_\Omega^{n_2}(Y \cup \Xi_{n_3}),\ \mathbf{q}'_3 \in \hat{F}_\Omega^{n'_2}(Y \cup \Xi_{n'_3}),\ \mathbf{r}_3 \in \hat{F}_\Omega^{m_2}(Y \cup \Xi_{m_3}),\\
\mathbf{q}_4 \in F_\Omega(Y)^{n_3},\ \mathbf{q}'_4 \in F_\Omega(Y)^{n'_3},\ \mathbf{r}_4 \in F_\Omega(Y)^{m_3},\\
a_0, a'_0 \in A',\ \mathbf{a}_i \in A^{n_i},\ \mathbf{a}'_i \in A^{n'_i},\ \mathbf{b}_i \in A^{m_i}\ (i = 1, 2, 3).
\end{gathered}$$

Moreover, take an $r' \in F_\Omega(Y)$*, and let* $r = r_1\big(\mathbf{r}_2(\mathbf{r}_3(\mathbf{r}_4))\big)$*. Finally, set* $A_i = \{a_{i_j} \mid j = 1, \ldots, n_i\}$, $A'_i = \{a'_{i_j} \mid j = 1, \ldots, n'_i\}$ *and* $B = \{b_{i_j} \mid j = 1, \ldots, m_i\}$ $(i = 1, 2, 3)$*. Assume that the following conditions are satisfied:*

(i) $p_1\big(p_2(p_3(p_4))\big) \in T$,

(ii) $a_0 p_1 \Rightarrow^* q_1\big(r_1(\mathbf{b}_1 \xi_1^{m_1}), \mathbf{a}_1 \xi_1^{n_1}\big)$, $a'_0 p_1 \Rightarrow^* q'_1(r', \mathbf{a}'_1 \xi_1^{n'_1})$,

(iii) $\mathbf{a}_1 p_2^{n_1} \Rightarrow^* \mathbf{q}_2(\mathbf{a}_2 \xi_1^{n_2})$, $\mathbf{a}'_1 p_2^{n'_1} \Rightarrow^* \mathbf{q}'_2(\mathbf{a}'_2 \xi_1^{n'_2})$, $\mathbf{b}_1 p_2^{m_1} \Rightarrow^* \mathbf{r}_2(\mathbf{b}_2 \xi_1^{m_2})$,

(iv) $\mathbf{a}_2 p_3^{n_2} \Rightarrow^* \mathbf{q}_3(\mathbf{a}_3 \xi_1^{n_3})$, $\mathbf{a}'_2 p_3^{n'_2} \Rightarrow^* \mathbf{q}'_3(\mathbf{a}'_3 \xi_1^{n'_3})$, $\mathbf{b}_2 p_3^{m_2} \Rightarrow^* \mathbf{r}_3(\mathbf{b}_3 \xi_1^{m_3})$,

(v) $\mathbf{a}_3 p_4^{n_3} \Rightarrow^* \mathbf{q}_4$, $\mathbf{a}'_3 p_4^{n'_3} \Rightarrow^* \mathbf{q}'_4$, $\mathbf{b}_3 p_4^{m_3} \Rightarrow^* \mathbf{r}_4$,

(vi) $p_4\hat{\beta} = p_3(p_4)\hat{\beta} = p_2\big(p_3(p_4)\big)\hat{\beta}$,
$A_1 \subseteq A_2 \subseteq A_3$, $A'_1 \subseteq A'_2 \subseteq A'_3$, $B_1 = B_2 \subseteq B_3$,

(vii) $r \neq r'$, $\mathrm{path}_1(q_1) = \mathrm{path}_1(q'_1)$.

Then at least one of the trees $p_1\big(p_2(p_4)\big)$, $p_1\big(p_3(p_4)\big)$ *and* $p_1(p_4)$ *is in* Q.

Proof. The proof of this lemma is similar to that of Lemma 4.9.4. □

Lemma 4.9.6 *Let*

$$p_1, p_2 \in \hat{F}_\Sigma(X \cup \Xi_1),\ p_3 \in F_\Sigma(X),\ k, l, m, k', l', m' \geq 0,$$
$$q_1 \in \hat{F}_\Omega(Y \cup \Xi_{k+1}),\ q'_1 \in \hat{F}_\Omega(Y \cup \Xi_{k'+1}),\ q_2 \in \hat{F}_\Omega(Y \cup \Xi_{l+1}),\ q'_2 \in \hat{F}_\Omega(Y \cup \Xi_{l'+1}),$$
$$\mathbf{r} \in \hat{F}_\Omega^k(Y \cup \Xi_m),\ \mathbf{r}' \in \hat{F}_\Omega^{k'}(Y \cup \Xi_{m'}),\ q_3 \in \hat{F}_\Omega(Y \cup \Xi_1),\ q'_3, r \in F_\Omega(Y),$$
$$\mathbf{s} \in F_\Omega(Y)^l,\ \mathbf{s}' \in F_\Omega(Y)^{l'},\ \mathbf{t} \in F_\Omega(Y)^m,\ \mathbf{t}' \in F_\Omega(Y)^{m'},\ a_0, a'_0 \in A',\ a, a' \in A,$$
$$\mathbf{a} \in A^k,\ \mathbf{a}' \in A^{k'},\ \mathbf{b} \in A^l,\ \mathbf{b}' \in A^{l'},\ \mathbf{c} \in A^m \text{ and } \mathbf{c}' \in A^{m'}.$$

Moreover, set $A_1 = \{a_i \mid i = 1, \ldots, k\}$, $B_1 = \{b_i \mid i = 1, \ldots, l\}$, $C_1 = \{c_i \mid i = 1, \ldots, m\}$, $A'_1 = \{a'_i \mid i = 1, \ldots, k'\}$, $B'_1 = \{b'_i \mid i = 1, \ldots, l'\}$ *and* $C'_1 = \{c'_i \mid i = 1, \ldots m'\}$. *Assume that the following conditions are satisfied:*

(i) $p_1\big(p_2(p_3)\big) \in T$,

(ii) $a_0 p_1 \Rightarrow^* q_1(a\xi_1, \mathbf{a}\xi_1^k)$, $a'_0 p_1 \Rightarrow^* q'_1(a'\xi_1, \mathbf{a}'\xi_1^{k'})$,

(iii) $a p_2 \Rightarrow^* q_2(a\xi_1, \mathbf{b}\xi_1^l)$, $a' p_2 \Rightarrow^* q'_2(a'\xi_1, \mathbf{b}'\xi_1^{l'})$,
$\mathbf{a} p_2^k \Rightarrow^* \mathbf{r}(\mathbf{c}\xi_1^m)$, $\mathbf{a}' p_2^{k'} \Rightarrow^* \mathbf{r}'(\mathbf{c}'\xi_1^{m'})$,

(iv) $a p_3 \Rightarrow^* q_3(r)$, $a' p_3 \Rightarrow^* q'_3$, $\mathbf{b} p_3^l \Rightarrow^* \mathbf{s}$, $\mathbf{b}' p_3^{l'} \Rightarrow^* \mathbf{s}'$,
$\mathbf{c} p_3^m \Rightarrow^* \mathbf{t}$, $\mathbf{c}' p_3^{m'} \Rightarrow^* \mathbf{t}'$,

(v) $A_1 \subseteq B_1 \cup C_1$, $A'_1 \subseteq B'_1 \cup C'_1$, $p_3\hat{\beta} = p_2(p_3)\hat{\beta}$,

(vi) $\mathrm{path}_1(q'_1) = \mathrm{path}_1(q_1)\mathrm{path}_1(q_3)$ *and* $r \neq q'_3$.

Then $p_1(p_3) \in Q$.

Proof. Introduce the notation $\mathbf{d} = (\mathbf{b}, \mathbf{c})$, $\mathbf{d}' = (\mathbf{b}', \mathbf{c}')$, $\mathbf{u} = (\mathbf{s}, \mathbf{t})$ and $\mathbf{u}' = (\mathbf{s}', \mathbf{t}')$. Moreover, take two mappings $f\colon \{1, \ldots, k\} \to \{1, \ldots, l+m\}$ and $g\colon \{1, \ldots, k'\} \to \{1, \ldots, l'+m'\}$ satisfying the equalities $a_i = d_{f(i)}$ $(1 \leq i \leq k)$ and $a'_i = u_{g(i)}$ $(1 \leq i \leq k')$. Obviously, there are derivations $a_0 p_1(p_3) \Rightarrow^* q_1\big(q_3(r), u_{f(1)}, \ldots, u_{f(k)}\big)$ and $a'_0 p_1(p_3) \Rightarrow^* q'_1\big(q'_3, u'_{g(1)}, \ldots, u'_{g(k')}\big)$. Moreover, $p_1(p_3) \in T$. Since

$$\mathrm{path}_1\big(q_1(q_3(\xi_1), u_{f(1)}, \ldots, u_{f(k)})\big) = \mathrm{path}_1\big(q'_1(\xi_1, u'_{g(1)}, \ldots, u'_{g(k')})\big)$$

and $q_3' \neq r, q_1\big(q_3(r),\, u_{f(1)}, \ldots, u_{f(k)}\big) \neq q_1'(q_3', u'_{g(1)}, \ldots, u'_{g(k')})$. Hence, $p_1(p_3) \in Q$. $\square$

Now we are ready to state a theorem from which the main decidability results of this section easily follow.

Theorem 4.9.7 *There exists an algorithm to decide whether Q is empty.*

Proof. Let K denote the maximum of the heights of the right-hand sides of the productions from P, $\|A\| = 2^{|A|}$ and let L be the number of all words over $\{1, \ldots, r_\Sigma\}$ with length at most $\|A\|^2|B|K$, where r_Σ is the maximal m for which $\Sigma_m \neq \emptyset$. Moreover, let $k = \|A\|^2|A|^2|B|2L + 1$, $l = k + (2\|A\|^3|A||B|)(\|A\|^2|B|K + 1)$ and $m = l + 2\|A\|^3|B|$.

We shall show that Q is nonvoid iff it contains a tree with height less than m. The case $K = 0$ being obvious, we assume that $K \neq 0$.

Let p be an element of Q with minimal length, and $q, q' \in F_\Omega(Y)$ trees such that $q \neq q'$ and $(p, q), (p, q') \in \tau_{\mathfrak{A}}$. Assume that $\mathrm{hg}(p) \geq m$. Then there are $a_0, a_0' \in A'$, $p_0, \ldots, p_m \in \hat{F}_\Sigma(X \cup \Xi_1)$, $p_{m+1} \in F_\Sigma(X)$, $n_i, n_i' \geq 0$ $(i = 0, \ldots, m)$, $q_0 \in \hat{F}_\Omega(Y \cup \Xi_{n_0})$, $q_0' \in \hat{F}_\Omega(Y \cup \Xi_{n_0'})$, $\mathbf{q}_i \in \hat{F}_\Omega^{n_{i-1}}(Y \cup \Xi_{n_i})$, $\mathbf{q}_i' \in \hat{F}_\Omega^{n_{i-1}'}(Y \cup \Xi_{n_i'})$ $(i = 1, \ldots, m)$, $\mathbf{q}_{m+1} \in F_\Omega(Y)^{n_m}$, $\mathbf{q}'_{m+1} \in F_\Omega(Y)^{n_m'}$, $\mathbf{a}_i \in A^{n_i}$, $\mathbf{a}_i' \in A^{n_i'}$ $(i = 0, \ldots, m)$ such that the following conditions are satisfied:

(1) $p = p_0\big(p_1(\ldots(p_{m+1})\ldots)\big),\ p_i \neq \xi_1\ (i = 1, \ldots, m)$,

(2) $q = q_0\big(\mathbf{q}_1(\ldots(\mathbf{q}_{m+1})\ldots)\big),\ q' = q_0'\big(\mathbf{q}_1'(\ldots(\mathbf{q}'_{m+1})\ldots)\big)$,

(3) $a_0 p_0 \Rightarrow^* q_0(\mathbf{a}_0 \xi_1^{n_0}),\ a_0' p_0 \Rightarrow^* q_0'(\mathbf{a}_0' \xi_1^{n_0'})$,
$\mathbf{a}_i p_{i+1}^{n_i} \Rightarrow^* \mathbf{q}_{i+1}(\mathbf{a}_{i+1} \xi_1^{n_{i+1}}),\ \mathbf{a}_i' p_{i+1}^{n_i'} \Rightarrow^* \mathbf{q}'_{i+1}(\mathbf{a}'_{i+1} \xi_1^{n'_{i+1}})$
$(i = 0, \ldots, m-1),\ \mathbf{a}_m p_{m+1}^{n_m} \Rightarrow^* \mathbf{q}_{m+1},\ \mathbf{a}_m' p_{m+1}^{n_m'} \Rightarrow^* \mathbf{q}'_{m+1}$.

For $i = 0, \ldots, m$, introduce the notations $\check{p}_i = p_0\big(p_1(\ldots(p_i)\ldots)\big)$, $\check{q}_i = q_0\big(\mathbf{q}_1(\ldots(\mathbf{q}_i)\ldots)\big)$ and $\check{q}_i' = q_0'\big(\mathbf{q}_1'(\ldots(\mathbf{q}_i')\ldots)\big)$. Moreover, let $\hat{p}_i = p_{i+1}\big(\ldots(p_{m+1})\ldots\big)$, $\hat{q}_i = \mathbf{q}_{i+1}\big(\ldots(\mathbf{q}_{m+1})\ldots\big)$ and $\hat{q}_i' = \mathbf{q}'_{i+1}\big(\ldots(\mathbf{q}'_{m+1})\ldots\big)$ $(i = 0, \ldots, m)$. Finally, set $A_i = \{a_{i_j} \mid 1 \leq j \leq n_i\}$ and $A_i' = \{a'_{i_j} \mid 1 \leq j \leq n_i'\}$ $(i = 0, \ldots, m)$.

If $\check{q}_l(\mathbf{r}) \neq \check{q}_l'(\mathbf{r}')$ holds for all $\mathbf{r} \in F_\Omega(Y)^{n_l}$ and $\mathbf{r}' \in F_\Omega(Y)^{n_l'}$, then the fact that $m - l + 1 > |\mathfrak{p}A|^2|B|$ makes Lemma 4.9.2 applicable and hence there are i and j with $l \leq i < j \leq m$ such that $\check{p}_i(\hat{p}_j) \in Q$. This is obviously a contradiction since $|\check{p}_i(\hat{p}_j)| < |p|$.

Thus, we may assume that at least one of n_l and n_l', say n_l, is greater than 0. Moreover, it can also be supposed that there are an i_l $(1 \leq i_l \leq n_l)$, an $r' \in \hat{F}_\Omega(Y \cup \Xi_1)$ and an $s' \in F_\Omega(Y)$ such that $q' = r'(s')$, $\mathrm{path}_1(r') = \mathrm{path}_{i_l}(\check{q}_l)$ and $s' \neq \hat{q}_{l_{i_l}}$. Then for each $j < l$, $n_j > 0$. Now let i_j $(0 \leq j < l,\ 1 \leq i_j \leq n_j)$ be those uniquely determined integers for which $\mathrm{path}_{i_j}(\check{q}_j)$ are initial segments of $\mathrm{path}_{i_l}(\check{q}_l)$. Without loss of generality, we may assume that $i_0 = \ldots = i_l = 1$.

Now suppose that there exists no $w \in \{\mathrm{path}_i(\check{q}_l') \mid 1 \leq i \leq n_l'\}$ such that $\mathrm{path}_1(\check{q}_l)$ is an initial segment of w or w is an initial segment of $\mathrm{path}_1(\check{q}_l)$. Then for each i $(l \leq i \leq m)$, set

$$B_i = \{a_{i_j} \mid \mathrm{path}_1(\check{q}_l) \text{ is an initial segment of } \mathrm{path}_j(\check{q}_i)\}$$

and

$$C_i = \{a_{i_j} \mid \mathrm{path}_1(\check{q}_l) \text{ is not an initial segment of } \mathrm{path}_j(\check{q}_i)\}.$$

Since the cardinality of $\{l, \ldots, m\}$ is $2\|A\|^3|B|+1$, there are i_1, i_2, i_3 ($l \le i_1 < i_2 < i_3 \le m$) such that the following conditions are satisfied: $\hat{p}_{i_1}\hat{\beta} = \hat{p}_{i_2}\hat{\beta} = \hat{p}_{i_3}\hat{\beta}$, $B_{i_1} = B_{i_2} \subseteq B_{i_3}$, $C_{i_1} \subseteq C_{i_2} \subseteq C_{i_3}$ and $A'_{i_1} \subseteq A'_{i_2} \subseteq A'_{i_3}$. From this, by Lemma 4.9.5 we get that at least one of the trees $\check{p}_{i_2}(\hat{p}_{i_3}), \check{p}_{i_1}(\hat{p}_{i_2})$ and $\check{p}_{i_1}(\hat{p}_{i_3})$ is in Q, which is again a contradiction.

Therefore, for an i_l ($1 \le i_l \le n'_l$), $\mathrm{path}_{i_l}(\check{q}'_l)$ is an initial segment of $\mathrm{path}_1(\check{q}_l)$ or $\mathrm{path}_1(\check{q}_l)$ is an initial segment of $\mathrm{path}_{i_l}(\check{q}'_l)$. Let i_j ($0 \le j < l$, $1 \le i_j \le n'_j$) be those uniquely determined integers for which $\mathrm{path}_{i_j}(\check{q}'_j)$ are initial segments of $\mathrm{path}_{i_j}(\check{q}'_l)$. Without loss of generality we may assume that $i_0 = \ldots = i_l = 1$. We can also assume that $\mathrm{path}_1(\check{q}_l)$ is an initial segment of $\mathrm{path}_1(\check{q}'_l)$.

Now let us distinguish the following two cases:

a) $\mathrm{path}_1(\check{q}'_k)$ is an initial segment of $\mathrm{path}_1(\check{q}_l)$. If in addition for some i ($0 \le i \le k$), $\mathrm{abs}\big(l(\mathrm{path}_1(\check{q}_i)) - l(\mathrm{path}_1(\check{q}'_i))\big) > \|A\|^2|B|K$ then, by Lemma 4.9.3, there exists an $r \in F_\Omega(Y)$ such that $\check{q}_i(r) \in Q$ and $|r| < |\hat{p}_i|$. (Here abs stands for absolute value.) This obviously is a contradiction. Therefore, for each i ($0 \le i \le k$), $\mathrm{abs}\big(l(\mathrm{path}_1(\check{q}_i)) - l(\mathrm{path}_1(\check{q}'_i))\big) \le \|A\|^2|B|K$. Then, since the cardinality of $\{1, \ldots, k\}$ is $\|A\|^2|A|^2|B|2L + 1$, for some integers i and j ($1 \le i < j \le k$), we have:

   (I) $\mathrm{path}_1(\check{q}_i)$ is an initial segment of $\mathrm{path}_1(\check{q}'_i)$, $\mathrm{path}_1(\check{q}_j)$ is an initial segment of $\mathrm{path}_1(\check{q}'_j)$, $\mathrm{path}_1(\check{q}'_i)/\mathrm{path}_1(\check{q}_i) = \mathrm{path}_1(\check{q}'_j)/\mathrm{path}_1(\check{q}_j)$, or

   (II) $\mathrm{path}_1(\check{q}'_i)$ is an initial segment of $\mathrm{path}_1(\check{q}_i)$, $\mathrm{path}_1(\check{q}'_j)$ is an initial segment of $\mathrm{path}_1(\check{q}_j)$, $\mathrm{path}_1(\check{q}_i)/\mathrm{path}_1(\check{q}'_i) = \mathrm{path}_1(\check{q}_j)/\mathrm{path}_1(\check{q}'_j)$. (Here $uv/u = v$ for any two words u and v.) Moreover, $\hat{p}_j\hat{\beta} = \hat{p}_i\hat{\beta}$, $a_{i_1} = a_{j_1}$, $a'_{i_1} = a'_{j_1}$, $B_i \subseteq B_j$ and $B'_i \subseteq B'_j$, where $B_s = \{a_{s_t} \mid 2 \le t \le n_s\}$ and $B'_s = \{a'_{s_t} \mid 2 \le t \le n'_s\}$ ($s = i, j$). Then, by Lemma 4.9.6, $\check{p}_i(\hat{p}_j) \in Q$, which is a contradiction since $|\check{p}(\hat{p}_{i_j})| < |p|$.

b) $\mathrm{path}_1(\check{q}_l)$ is an initial segment of $\mathrm{path}_1(\check{q}'_k)$. We shall show that

$$l\big(\mathrm{path}_1(\check{q}_l)\big) - l\big(\mathrm{path}_1(\check{q}_k)\big) > \|A\|^2|B|K.$$

Then $l\big(\mathrm{path}_1(\check{q}'_k)\big) - l\big(\mathrm{path}_1(\check{q}_k)\big) > \|A\|^2|B|K$ will also hold, which, by Lemma 4.9.3, will be a contradiction.

Thus, assume that $l\big(\mathrm{path}_1(\check{q}_l)\big) - l\big(\mathrm{path}_1(\check{q}_k)\big) \le \|A\|^2|B|K$. Then, since the cardinality of $\{k+1, \ldots, l\}$ is $(2\|A\|^3|A||B|)(\|A\|^2|B|K + 1)$, there are i_1 and i_2 ($k \le i_1 < i_2 \le l$) such that $i_2 - i_1 = 2\|A\|^3|A||B|$ and $\mathrm{path}_1(\check{q}_{i_1}) = \ldots = \mathrm{path}_1(\check{q}_{i_2})$, i.e., $q_{(i_1+1)_1} = \ldots = q_{i_{2_1}} = \xi_1$. Now for each j ($i_1 \le j \le i_2$) set

$$B_j = \{a'_{j_t} \mid 1 \le t \le n'_j,\ \mathrm{path}_1(\check{q}'_{i_1}) \text{ is an initial segment of } \mathrm{path}_1(\check{q}'_j)\}$$

and

$$C_j = \{a'_{j_t} \mid 1 \le t \le n'_j,\ \mathrm{path}_1(\check{q}'_{i_1}) \text{ is not an initial segment of } \mathrm{path}_1(\check{q}'_j)\}.$$

Since the cardinality of $\{i_1, \ldots, i_2\}$ is $2\|A\|^3|A||B|+1$, there are integers j_1, j_2 and j_3 ($i_1 \le j_1 < j_2 < j_3 \le i_2$) such that $\hat{p}_{j_1}\hat{\beta} = \hat{p}_{j_2}\hat{\beta} = \hat{p}_{j_3}\hat{\beta}$, $a_{j_{1_1}} = a_{j_{2_1}} = a_{j_{3_1}}$, $\overline{A}_{j_1} \subseteq \overline{A}_{j_2} \subseteq \overline{A}_{j_3}$, $B_{j_1} = B_{j_2} \subseteq B_{j_3}$ and $C_{j_1} \subseteq C_{j_2} \subseteq C_{j_3}$, where $\overline{A}_{j_t} = \{a_{j_{t_s}} \mid 2 \le s \le n_{j_t}\}$ $(t = 1, 2, 3)$. Therefore, by Lemma 4.9.4, at least one of the trees $\check{p}_{j_2}(\hat{p}_{j_3})$, $\check{p}_{j_1}(\hat{p}_{j_2})$ and $\check{p}_{j_1}(\hat{p}_{j_3})$ is in Q which is again a contradiction. □

Now we are ready to prove

Theorem 4.9.8 *For any two* R*-transducers* $\mathfrak{A} = (\Sigma, X, A, \Omega, Y, P, A')$ *and* $\mathfrak{B} = (\Sigma, X, B, \Omega, Y, P', B')$ *and any recognizable* ΣX*-forest* T *it is decidable*

(i) *whether* $\tau_{\mathfrak{A}}|T$ *is a (partial) mapping,*

(ii) *whether* $\tau_{\mathfrak{A}}|T \subseteq \tau_{\mathfrak{B}}|T$*, provided that* $\tau_{\mathfrak{B}}|T$ *is a (partial) mapping,*

(iii) *whether* $\mathfrak{A}$ *is equivalent to* $\mathfrak{B}$*, provided that* $\tau_{\mathfrak{A}}$ *or* $\tau_{\mathfrak{B}}$ *is a (partial) mapping,*

and

(iv) *whether* $\mathfrak{A}$ *is equivalent to* $\mathfrak{B}$*, provided that at least one of them is deterministic.*

Proof. By Theorem 4.9.7, (i) is true. Moreover, (iii) and (iv) follow from (ii) since the domain of an R-transformation is regular and, by Theorem 2.10.3, it is decidable for two regular forests whether one of them contains the other one. Therefore, it is enough to prove (ii).

We may assume that $A \cap B = \emptyset$. Let us construct an R-transducer $\mathfrak{C} = (\Sigma, X, C, \Omega, Y, P'', C')$ with $C = A \cup B$, $C' = A' \cup B'$ and $P'' = P \cup P'$. Obviously, $\tau_{\mathfrak{C}}|T = \tau_{\mathfrak{A}}|T \cup \tau_{\mathfrak{B}}|T$. Thus $\tau_{\mathfrak{A}}|T \subseteq \tau_{\mathfrak{B}}|T$ holds iff $\mathrm{dom}(\tau_{\mathfrak{A}}) \cap T \subseteq \mathrm{dom}(\tau_{\mathfrak{B}}) \cap T$ and $\tau_{\mathfrak{C}}|T$ is a partial mapping. □

Before stating the analogous result for F-transducers we prove a lemma.

Lemma 4.9.9 *For any* F*-transducer* $\mathfrak{A} = (\Sigma, X, A, \Delta, Y, P, A')$ *and* $R \in \mathrm{Rec}(\Sigma, X)$ *one can effectively give an* R*-transducer* $\mathfrak{B} = (\Omega, X, B, \Delta, Y, P', B')$ *and a forest* $S \in \mathrm{Rec}(\Omega, X)$ *such that* $\tau_{\mathfrak{A}}|R$ *is a partial mapping iff* $\tau_{\mathfrak{B}}|S$ *is a partial mapping.*

Proof. Construct an $\mathrm{R_R}$-transducer $\overline{\mathfrak{A}} = (\Sigma, X, A, \Delta, Y, \overline{P}, A')$ where $\overline{P}$ is given as follows:

(i) If $x \to ar$ $\big(x \in X,\ a \in A,\ r \in F_\Delta(Y)\big)$ is in P, then $ax \to r$ is in $\overline{P}$.

(ii) If $\sigma(a_1, \ldots, a_m) \to ar$ $\big(\sigma \in \Sigma_m,\ m \ge 0,\ a_1, \ldots, a_m, a \in A,\ r \in F_\Delta(Y \cup \Xi_m)\big)$ is in P, then $\big(a\sigma \to r(a_1\xi_1, \ldots, a_m\xi_m), D\big)$ is in $\overline{P}$, where $D(\xi_i) = \mathrm{dom}(\tau_{\mathfrak{A}(a_i)})$ $(i = 1, \ldots, m)$. Since, by Theorem 4.1.10 (i), $\mathrm{dom}(\tau_{\mathfrak{A}(a)})$ $(a \in A)$ is regular, $\overline{\mathfrak{A}}$ is an $\mathrm{R_R}$-transducer. Observe that $\tau_{\mathfrak{A}(a)} \subseteq \tau_{\overline{\mathfrak{A}}(a)}$ holds for every $a \in A$.

We shall show that for all $\{a, a'\} \subseteq A$ and $p \in F_\Sigma(X)$ the equivalence

$$|\tau_{\mathfrak{A}(a)}(p) \cup \tau_{\mathfrak{A}(a')}(p)| > 1 \iff |\tau_{\overline{\mathfrak{A}}(a)}(p) \cup \tau_{\overline{\mathfrak{A}}(a')}(p)| > 1 \tag{1}$$

holds. (Note that a and a' are not necessarily distinct.)

Since $\tau_{\mathfrak{A}(a)} \subseteq \tau_{\overline{\mathfrak{A}}(a)}$, the left side of (1) implies its right side.

The converse will be proved by induction on $\mathrm{hg}(p)$. If $\mathrm{hg}(p) = 0$, then our statement obviously holds. Now let $p = \sigma(p_1, \ldots, p_m)$ $\big(\sigma \in \Sigma_m, m > 0, p \in F_\Sigma(X)\big)$ and $r, r' \in F_\Delta(Y)$ be such that $ap \Rightarrow^*_{\overline{\mathfrak{A}}} r$, $a'(p) \Rightarrow^*_{\overline{\mathfrak{A}}} r'$ and $r \neq r'$. Moreover, assume that the right side of (1) implies its left side for every state and every ΣX-tree of height less than $\mathrm{hg}(p)$.

Let us write the above derivations in the form

$$a\sigma \Rightarrow_{\overline{\mathfrak{A}}} \overline{r}(a_1^{n_1}\xi_1^{n_1}, \ldots, a_m^{n_m}\xi_m^{n_m}),\ a_i^{n_i}p_i^{n_i} \Rightarrow_{\overline{\mathfrak{A}}} \mathbf{r}_i\ (i = 1, \ldots, m)$$

and

$$a'\sigma \Rightarrow_{\overline{\mathfrak{A}}} \overline{r}'(b_1^{n'_1}\xi_1^{n'_1}, \ldots, b_m^{n'_m}\xi_m^{n'_m}),\ b_i^{n'_i}p_i^{n'_i} \Rightarrow_{\overline{\mathfrak{A}}} \mathbf{r}'_i\ (i = 1, \ldots, m),$$

where $a, a', a_i, b_i \in A,\ i = 1, \ldots, m,\ n_1 + \ldots + n_m = n,\ n'_1 + \ldots + n'_m = n'$, $\overline{r} \in \hat{F}_\Delta(Y \cup \Xi_n),\ \overline{r}' \in \hat{F}_\Delta(Y \cup \Xi_{n'}),\ \overline{r}(\mathbf{r}_1, \ldots, \mathbf{r}_m) = r$ and $\overline{r}'(\mathbf{r}_1, \ldots, \mathbf{r}_m) = r'$. Moreover, $\big(\sigma(a_1, \ldots, a_m),\ a\overline{r}(\xi_1^{n_1}, \ldots, \xi_m^{n_m})\big)$, $\big(\sigma(b_1, \ldots, b_m),\ a'\overline{r}'(\xi_1^{n'_1}, \ldots, \xi_m^{n'_m})\big) \in P$.

Now distinguish the following two cases:

(I) There exists an i $(1 \leq i \leq m)$ with $n_i > 0$ and $|\tau_{\overline{\mathfrak{A}}(a_i)}(p_i)| > 1$ or there exists a j $(1 \leq j \leq m)$ with $n'_j > 0$ and $|\tau_{\overline{\mathfrak{A}}(b_j)}(p_j)| > 1$. Then, by the induction hypothesis, $|\tau_{\mathfrak{A}(a_i)}(p_i)| > 1$ or $|\tau_{\mathfrak{A}(b_j)}(p_j)| > 1$. Therefore, by the definition of $\overline{P}$, $|\tau_{\mathfrak{A}(a)}(p)| > 1$ or $|\tau_{\mathfrak{A}(a')}(p)| > 1$ also holds.

(II) Assume that there are no i and j satisfying (I). Then, $r_{i_1} = \ldots = r_{i_{n_i}} = r_i$ $(1 \leq i \leq m)$ if $n_i > 0$. For all such i, by $\tau_{\mathfrak{A}(a_i)} \subseteq \tau_{\overline{\mathfrak{A}}(a_i)}$ and the choice of D, we have $p_i \Rightarrow^*_{\mathfrak{A}} a_i r_i$. Moreover, again by the choice of D, if $n_i = 0$ then also there exists an $r_i \in F_\Delta(Y)$ such that $p_i \Rightarrow^*_{\mathfrak{A}} a_i r_i$ holds. Thus, we have the derivation $p \Rightarrow^*_{\mathfrak{A}} ar$. Using similar arguments, one can show that $p \Rightarrow^*_{\mathfrak{A}} a'r'$ is also valid. Therefore, $|\tau_{\mathfrak{A}(a)}(p) \cup \tau_{\mathfrak{A}(a')}(p)| > 1$.

Thus, we have proved that $\tau_{\mathfrak{A}}|R$ is a partial mapping iff $\tau_{\overline{\mathfrak{A}}}|R$ is a partial mapping. By Theorem 4.4.6 (i), there exist a deterministic F-relabeling $\tau\colon F_\Sigma(X) \to F_\Omega(X)$ and an R-transducer $\mathfrak{B} = (\Omega, X, B, \Delta, Y, P'', B')$ such that $\tau_{\overline{\mathfrak{A}}} = \tau \circ \tau_{\mathfrak{B}}$. Moreover, by Lemma 4.6.5, $R\tau = S$ is in $\mathrm{Rec}(\Omega, X)$ and S can be obtained effectively from R. Therefore, $\tau_{\overline{\mathfrak{A}}}|R$ is a partial mapping iff $\tau_{\mathfrak{B}}|S$ is a partial mapping. □

Now we state and prove

Theorem 4.9.10 *For any two* F*-transducers* $\mathfrak{A} = (\Sigma, X, A, \Omega, Y, P, A')$ *and* $\mathfrak{B} = (\Sigma, X, B, \Omega, Y, P', B')$ *and recognizable* ΣX*-forest* T*, it is decidable*

(i) *whether* $\tau_{\mathfrak{A}}|T$ *is a partial mapping,*

(ii) *whether* $\tau_{\mathfrak{A}}|T \subseteq \tau_{\mathfrak{B}}|T$*, provided that* $\tau_{\mathfrak{B}}|T$ *is a partial mapping,*

(iii) *whether* $\mathfrak{A}$ *is equivalent to* $\mathfrak{B}$*, provided that* $\tau_{\mathfrak{A}}$ *or* $\tau_{\mathfrak{B}}$ *is a partial mapping, and*

(iv) *whether* $\mathfrak{A}$ *is equivalent to* $\mathfrak{B}$*, provided that at least one of them is deterministic.*

Proof. Obviously, (i) follows from Theorem 4.9.8 by Lemma 4.9.9. Moreover, (ii) implies (iii) and (iv) since, by Theorem 4.1.10 (i), the domain of an F-transformation is recognizable. Thus, it suffices to prove (ii).

Assume that $A \cap B = \emptyset$, and construct the F-transducer

$$\mathfrak{C} = (\Sigma, X, C, \Omega, Y, P'', C')$$

with $C = A \cup B$, $C' = A' \cup B'$ and $P'' = P \cup P'$. Obviously, $\tau_{\mathfrak{C}} = \tau_{\mathfrak{A}} \cup \tau_{\mathfrak{B}}$. Therefore, $\tau_{\mathfrak{A}}|T \subseteq \tau_{\mathfrak{B}}|T$ iff $\mathrm{dom}(\tau_{\mathfrak{A}}) \cap T \subseteq \mathrm{dom}(\tau_{\mathfrak{B}}) \cap T$ and $\tau_{\mathfrak{C}}|T$ is a partial mapping. $\square$

4.10 EXERCISES

1. Define generalized sequential machines as tree transducers when strings are interpreted as unary trees in the usual way.

2. Let τ be a DR-transformation. Then $\mathrm{dom}(\tau)$ can be recognized by a DR-recognizer.

3. Show that the classes $\mathcal{LDF}$ and $\mathcal{LDR}$, and similarly the classes $\mathcal{LNDF}$ and $\mathcal{LNDR}$, are incomparable.

4. Let us call a DR-transducer $\mathfrak{A} = (\Sigma, X, A, \Omega, Y, P, A')$ *simple*, if for every $a\sigma \to q \in P$, whenever $a_1\xi_i$ and $a_2\xi_i$ occur in q, then $a_1 = a_2$. If $\mathfrak{A}$ is a simple DR-transducer, then $\tau_{\mathfrak{A}}$ can be induced by an F-transducer.

5. Prove that $\mathcal{DR}$ is not closed under composition.

6. The composition of a totally defined DR-transformation by an R-transformation is an R-transformation.

7. Is $\mathcal{R}$ closed under composition with LR-transformations from the right?

8. Show that $\mathcal{F}$ is not closed under composition with LNF-transformations from the right.

9. Prove Theorems 4.3.7 and 4.3.9.

10. Find two R-transformations τ_1 and τ_2 such that $\tau_1 \circ \tau_2$ is the F-transformation given in Example 4.1.3.

11. Give two F-transformations whose composition is the R-transformation of Example 4.1.6.

12. Show that $\mathcal{F}$ and $\mathcal{R}_R$ are incomparable.

13. Prove that $\mathcal{DR}_R$ is closed under DF-transformations.

14. An F-transformation (or an R-transformation) is a partial mapping iff it can be induced by a $\mathrm{DR_R}$-transducer.

15. Find a $\mathrm{DR_R}$-transducer which is not equivalent to any DR-transducer.

16. The equivalence problem of two $\mathrm{R_R}$-transducers is decidable, provided that at least one of them induces a partial mapping.

17. Find an algorithm to decide for an F-transducer whether it is equivalent to an LF-transducer.

18. Let $\mathfrak{A} = (\Sigma, X, A, Y, P, A')$ be a GSDT and Ω a ranked alphabet. Let $\{n_1, \ldots, n_r\}$ be the set of lengths of right-hand sides of all rules from P (each element of $A\Xi$ is counted as one symbol). Moreover, let $r(\Omega) = \{m_1, \ldots, m_s\}$. Assume that there exists a mapping $f\colon \{n_1, \ldots, n_r\} \to r(\Omega)$ such that the equality

$$n_k = m_{f(k)} + l_1(m_1 - 1) + \ldots + l_s(m_s - 1)$$

holds for every $k(= 1, \ldots, r)$, where $l_1, \ldots, l_s \geq 0$. Then there is an R-transducer $\mathfrak{B} = (\Sigma, X, B, \Omega, P', B')$ with $\tau_{\mathfrak{A}} = \{(p, \mathrm{yd}(q)) \mid (p, q) \in \tau_{\mathfrak{B}}\}$.

19. Find an R-transducer $\mathfrak{A}$ such that $\tau_{\mathfrak{A}}$ preserves recognizability, but $\mathfrak{A}$ is not equivalent to any LF-transducer.

20. An R-transducer $\mathfrak{A} = (\Sigma, X, A, \Omega, Y, P, a_0)$ is called *k-metalinear* if the following conditions are satisfied:
    (1) a_0 does not appear in the right-hand sides in rules from P,
    (2) for each rule $a_0\sigma \to q$ $(\sigma \in \Sigma_m)$ in P every ξ_i $(1 \leq i \leq m)$ can occur in q at most k times, and
    (3) for each rule $a\sigma \to q$ $(a \neq a_0,\ \sigma \in \Sigma_m)$ in P the number of occurences of each ξ_i $(1 \leq i \leq m)$ in q is 0 or 1.

    Let $\mathfrak{A}$ be a k-metalinear R-transducer. Does $\tau_{\mathfrak{A}}$ preserve recognizability?

21. For a ranked alphabet Σ let $\tilde{\Sigma} = \tilde{\Sigma}_0 \cup \tilde{\Sigma}_1$ be the ranked alphabet with $\tilde{\Sigma}_0 = \Sigma_0$ and $\tilde{\Sigma}_1 = \{\tilde{\sigma} \mid \sigma \in \Sigma_m,\ m > 0\}$. Define the mapping $\mathrm{ph}\colon F_\Sigma(X) \to \mathfrak{p}F_{\tilde{\Sigma}}(X)$ by $\mathrm{ph}(d) = \{d\}$ $(d \in \Sigma_0 \cup X)$ and

$$\mathrm{ph}\big(\sigma(p_1, \ldots, p_m)\big) = \{\tilde{\sigma}(t) \mid t \in \mathrm{ph}(p_1) \cup \ldots \cup \mathrm{ph}(p_m)\}$$

$\big(\sigma \in \Sigma_m,\ m > 0,\ p_1, \ldots, p_m \in F_\Sigma(X)\big)$. Show that if $T \in \mathrm{Surf}(\mathcal{R})$ then $\mathrm{ph}(T) = \bigcup\big(\mathrm{ph}(t) \mid t \in T\big)$ is recognizable.

22. Is Surf($\mathcal{R}$) closed under intersection?

23. Give a recursive definition of the concepts of state-sequence and production-sequence.

24. For every F-transducer there is an equivalent totally defined F-transducer with a single final state.

25. For every DF-transducer (DR-transducer) one can effectively give an equivalent DF-transducer (DR-transducer) with a minimal number of states.

4.11 NOTES AND REFERENCES

The concept of the R-transducer was introduced by ROUNDS [215] and THATCHER [238] thus extending generalized sequential machines from strings to trees and to give a tree automaton formalism for parts of mathematical linguistics (in particular, for the theory of syntax directed compilation). The F-transducer is due to THATCHER [239]. As in the case of tree recognizers, many of the authors dealing with tree transducers allow a symbol from a ranked alphabet to have more than one rank, and most of them use no separate frontier alphabets.

The results of Section 4.2 can be found in ENGELFRIET [75], and most results of Section 4.3 are also from this work. Theorems 4.3.3, 4.3.12, 4.3.13 were obtained by BAKER [26].

Tree transducers with regular look-ahead are defined and investigated in ENGELFRIET [78]. Generalized syntax directed translations were introduced by AHO and ULLMAN [2] in the special case where the domain of the translation is the forest of all *parse trees* of a given context-free grammar. (Parse trees are almost the same as our production trees.) Applying a generalized syntax directed translation in the sense of Aho and Ullman is equivalent to applying a DGSDT of Section 4.5 which, by Theorem 4.5.4, is equivalent to applying a DR-transducer and then taking the yield of the resulting tree. The more general concept of a GSDT was introduced in BAKER [28]. In the same work she proved that for each n, ydSurf($\mathcal{R}^n$) and ydSurf($\mathcal{F}^n$) are properly contained in the family of deterministic context-sensitive languages.

The results of Section 4.6 are from ENGELFRIET [75], GÉCSEG [102] and ROUNDS [215].

The first result about the Surf($\mathcal{R}^n$)-hierarchy can be found in OGDEN and ROUNDS [190], where they proved that Surf($\mathcal{R}$) is a proper subclass of Surf($\mathcal{R}^2$) and conjectured the properness of the hierarchy. It was ENGELFRIET [80, 83] who succeeded in proving that the $\mathcal{R}^n$-, Surf($\mathcal{R}^n$)-, and ydSurf($\mathcal{R}^n$)-hierarchies (and their F-transducer counterparts) are proper. Section 4.7 and 4.8 are based on his work.

The decidability results of Section 4.9 are from ÉSIK [90]. Using a different technique ZACHAR [254] also proved the decidability of the equivalence problem of DF-transducers.

As a conclusion we mention some other topics relevant to the subject matter of Chapter 4.

A *sequential program machine* (sp-*machine*) introduced by BUDA [46] is such a generalization of a gsm whose inputs are strings and whose outputs are n-tuples of n-ary trees. Buda showed that the equivalence problem of sp-machines is solvable and that this implies that the equivalence of certain program schemes is also decidable.

Engelfriet and Filè introduced a new type of tree transducer called *macro tree transducer* which is a combination of the R-transducer and the context-free tree grammar (see ENGELFRIET [82]). They propose to use macro tree transducers to model *attribute grammars* of D. E. Knuth (Math. Systems Theory 2 (1968), 127–145: Correction: ibid 5 (1971), 95–96). For tree transformations in terms of magmoids we refer the reader to ARNOLD and DAUCHET [13, 16], DAUCHET [61, 62], and LILIN [159, 160].

Finally, we note that much of the category theoretic work mentioned in the Notes and References of Chapter 2 deal with tree transductions.

Bibliography

We hope that most of the literature dealing with tree automata, tree grammars, forests, tree transductions, or their applications (published by the end of 1982) is listed in this bibliography. It also includes some more general works which devote at least a part to our subject, as well as a few items on closely related topics. As to the latter category the decision on inclusion or exclusion has sometimes been difficult. Of a paper published more than once in almost identical form, just the more complete, or the more widely available, version is mentioned. Preliminary reports and unpublished theses are not included except for a few cases. Items published by the same author(s) in the same year are distinguished for reference by a letter after the year. For some of the most often recurring journals and proceedings we use the following abbreviations:

n. Ann. ACM STC = Proceedings of the n^{th} Annual ACM Symposium on Theory of Computing
n. Coll. Lille = Les Arbres en Algébre et en Programmation, n^{th} Colloque du Lille, Université de Lille I
IC = Information and Control
n. IEEE Symp. ($n \leq 15$) = n^{th} Annual Symposium on Switching and Automata Theory
n. IEEE Symp ($n > 15$) = n^{th} Annual Symposium on Foundations of Computer Science
J. ACM = J. Assoc. Comput. Mach.
J. CSS = J. Comput. System Sci.
LN in CS = Lecture Notes in Computer Science (Springer-Verlag)
MST = Mathematical Systems Theory
S-C-C = Systems-Computers-Controls

[1] Adámek, J. and Trnková, V. (1981): Varietors and machines in a categry. – Algebra Universalis *13* (1981), 89-132.

[2] Aho, A. V. and Ullman, J. D. (1971): Translations on a context-free grammar. – IC *19* (1971), 439-475.

[3] Alagić, S. (1975a): Categorical theory of tree processing. – Category Theory Applied to Computation and Control (Proc. Symp., San Francisco, 1974), LN in CS *25* (1975), 65-72.

[4] Alagić, S. (1975b): Natural state transformations. – J. CSS *10* (1975), 266-307.

[5] Arbib, M. A. and Give'on, Y. (1968): Algebra automata I: Parallel programming as a prolegomena to the categorical approach. – IC *12* (1968), 331-345.

[6] Arbib, M. A. and Manes, E. G. (1974): Machines in a category: An expository introduction. – SIAM Review *16* (1974), 163-192.

[7] Arbib, M. A. and Manes, E. G. (1978): Tree transformations and the semantics of loop-free programs. – Acta Cybernet. *4* (1978), 11-17.

[8] Arbib, M. A. and Manes, E. G. (1979): Interwined recursion, tree transformations, and linear systems. – IC *40* (1979), 144-180.

[9] Arnold, A. (1977a): Rational sets of trees. – 2. Coll. Lille (1977), 20-28.

[10] Arnold, A. (1977b): Systèmes d'equations dans le magmoide. Ensembles rationnels et algébriques d'arbres. – Thèse de doctorat, Université de Lille I (1977).

[11] Arnold, A. (1980): Le théorème de transversale rationnelle dans les langages d'arbres. – MST 13 (1980), 275-282.

[12] Arnold, A. and Dauchet, M. (1976a): Theorie des magmoides. – 1. Coll. Lille (1976), 15-30.

[13] Arnold, A. and Dauchet, M. (1976b): Bimorphismes de magmoides. – 1. Coll. Lille (1976), 31-43.

[14] Arnold, A. and Dauchet, M. (1976c): Transductions de forêts reconnaissables monadiques. Forêts corégulières – RAIRO Informat. Théor. *10* (1976), No. 3, 5-28.

[15] Arnold, A. and Dauchet, M.(1976d): Un théorème de duplication pour les forêts algébriques. – J. CSS *13* (1976), 223-244.

[16] Arnold, A. and Dauchet, M. (1976e): Bi-transductions de forêts, – Automata, Languages and Programming (Conf. Rec., Edinburgh, 1976), University Press, Edinburgh (1976), 74-86.

[17] Arnold, A. and Dauchet, M. (1977): Un théorème de Chomsky-Schützenberger pour les forêts algébriques. – Calcolo *14* (1977), 161-184.

[18] Arnold, A. and Dauchet, M. (1978a): Forêts algébriques et homomorphismes inverses. – IC *37* (1978), 182-196.

[19] Arnold, A. and Dauchet, M. (1978b): Sur l'inversion des morphismes d'arbres. – Automata, Languages and Programming (Fifth Coll., Udine 1978), LN in CS *62* (1978), 26-35.

[20] Arnold, A. and Dauchet, M. (1978c): Une relation d'equivalence decidable sur la classe des forêts reconnaissables. – MST *12* (1978), 103-128.

[21] Arnold, A. and Dauchet, M. (1978d, 1979): Theorie des magmoides

(I) – RAIRO Inform. Théor. *12* (1978), 235-257.

(II) – RAIRO Inform. Théor. *13* (1979), 135-154.

[22] Arnold, A. and Dauchet, M. (1982): Morphismes et bimorphismes d'arbres. – Theor. Comput. Sci. *20* (1982), 33-93.

[23] Arnold, A. and Leguy, B. (1979a): Une propriété des forêts algébriques "de Greibach". – 4. Coll. Lille (1979), 1-17.

[24] Arnold, A. and Leguy, B. (1979b): Forêts de Greibach et homomorphismes inverses. – Fundam. Comput. Theory '79 (Proc. Conf., Berlin/Wendisch-Rietz 1979), Akademie-Verlag Berlin (1979), 31-37.

[25] Asveld, P. R. J. and Engelfriet, J. (1979): Extended linear macro grammars, iteration grammars, and register programs. – Acta Inform. *11* (1979), 259-285.

[26] Baker, B. S. (1973): Tree transductions and families of tree languages. – 5. Ann. ACM STC (1973), 200-206.

[27] Baker, B. S. (1978a): Tree transducers and tree languages. – IC *37* (1978), 241-266.

[28] Baker, B. S. (1978b): Generalized syntax directed translation, tree transducers, and linear space. – SIAM J. Comput. *7* (1978), 876-891.

[29] Baker, B. S. (1979): Composition of top-down and bottom-up tree transductions. – IC *41* (1979), 186-213.

[30] Barrero, A. and Gonzalez, R. C. (1976): Minimization of deterministic tree grammars and automata. – Proc. IEEE Conf. Decision and Control and the 15$^{\text{th}}$ Symp. Adaptive Processes (Clearwater, Fla., 1976), Inst. Electr. Electron. Engrs., New York (1976), 404-407.

[31] Barrero, A., Gonzalez, R. C. and Thomason, M. G. (1981): Equivalence and reduction of expansive tree grammars. – IEEE Trans. Pattern Anal. & Mach. Intell. PAMI – 3 (1981), 204-206.

[32] Benson, D. B. (1975): Semantic preserving translations. – MST *8* (1975), 105-126.

[33] Berger, J. and Pair, C. (1978): Inference for regular bilanguages. – J. CSS *16* (1978), 100-122.

[34] Berstel, J. and Reutenauer, C. (1982): Recognizable power series on trees. – Theor. Comput. Sci. *18* (1982), 115-148.

[35] Bertsch, E. (1973): Some considerations about classes of mappings between context-free derivation systems. – GI. 1. Fachtagung Automatentheorie Formale Sprachen (Bonn, 1973), LN in CS *2* (1973), 278-283.

[36] Bilstein, J. and Damm, W. (1981): Top-down tree-transducers for infinite trees I. – CAAP'81 (Trees in algebra and programming, 6$^{\text{th}}$ Coll., Genoa, March 1981), LN in CS *112* (1981), 117-134.

[37] Bloom, S. L. and Elgot, C. C. (1976): The existence and construction of free iterative theories. – J. CSS *12* (1976), 305-318.

[38] Bobrow, L. S. and Arbib , M. A. (1974): Discrete Mathematics, Applied Algebra for Computer and Information Science. – W. S. Saunders Co., Philadelphia (1974).

[39] Brainerd, W. S. (1968): The minimalization of tree automata. – IC *13* (1968), 484-491.

[40] Brainerd, W. S. (1969a): Tree generating regular systems. – IC *14* (1969), 217-231.

[41] Brainerd, W. S. (1969b): Semi-Thue systems and representations of trees. – 10. IEEE Symp. (1969), 240-244.

[42] Brayer, J. M. and Fu, K.-S. (1977): A note on the k-tail method of tree grammar inference. – IEEE Trans. Systems Man Cybernetics SMC – *7* (1977), 293-300.

[43] Buda, A. (1978a): The equivalence problem for sequential program machines. – 3. Coll. Lille (1978), 19-26.

[44] Буда, А.О. (1978b): Абстактные машины программ. – Акад. Наук СССР Сиб. отд., Вычисл. центр, Препринт 108, Новосибирск (1978).

[45] Buda, A. (1978c): Languages of program machines (Russian). – C. R. Acad. Bulgare Sci. *31* (1978), 1543-1544.

[46] Buda, A. (1979): Generalized$^{1.5}$ sequential machines. – Inform. Process. Lett. *8* (1979), No. 1, 38-40.

[47] Buttelmann, H. W. (1971): On generalized finite automata and unrestricted generative grammars. – 3. Ann. ACM STC (1971), 63-77.

[48] Buttelmann, H. W. (1975a): On the syntactic structures of unrestricted grammars I: Generative grammars and phrase structure grammars. – IC *29* (1975), 29-80.

[49] Buttelmann, H. W. (1975b): On the syntactic structures of unrestricted grammars II: Automata. – IC *29* (1975), 81-101.

[50] Casteran, P. (1978): Représentation rationelle d'arbres infinis. – 3. Coll. Lille (1978), 27-39.

[51] Catalano, A., Gnesi, S. and Montanari, U. (1978): Shortest path problems and tree grammars: An algebraic framework. – Graph-grammars and their application to computer science and biology (International workshop, Bad Honnef, 1978), LN in CS *73* (1978), 167-179.

[52] Costich, O. L. (1972): A Medvedev characterization of sets recognized by generalized finite automata. – MST *6* (1972), 263-267.

[53] COURCELLE, B. (1976): Arbres algébriques et langages déterministes. – 1. Coll. Lille (1976), 60-64.

[54] COURCELLE, B. (1978): Frontiers of infinite trees, – 3. Coll. Lille (1978), 76-102.

[55] CRESPI REGHIZZI, S. and DELLA VIGNA, P. (1973): Approximation of phrase markers by regular sets. – Automata, Languages and Programming (Proc. Coll., Rocquencourt, 1972), North Holland, Amsterdam (1973), 367-376.

[56] ČULIK, K. II (1974): Structured OL-systems. – L Systems, LN in CS *15* (1974), 216-229.

[57] ČULIK, K. II and MAIBAUM, T. S. E. (1974): Parallel rewriting systems on terms. – Automata, Languages and Programming (Proc. Symp., Saarbrücken, 1974), LN in CS *14* (1974), 495-511.

[58] DAMM, W. (1977): Languages defined by higher program schemes. – Automata, Languages and Programming (Proc. Coll., Turku, 1977), LN in CS *52* (1977), 164-179.

[59] DAMM, W. (1979): An algebraic extension of the Chomsky-hierarchy. – 4. Coll. Lille (1979), 66-78.

[60] DAMM, W. (1982): The IO- and OI-hierarchies. – Theor. Comput. Sci. *20* (1982), 95-207.

[61] DAUCHET, M. (1977a): Grammaires transformationelles et bimorphismes de magmoides. – 2. Coll. Lille (1977), 249-273.

[62] DAUCHET, M. (1977b): Transductions de forêts, bimorphismes de magmoides. – Thèse de doctorat Universite de Lille I (1977).

[63] DAUCHET, M. and MONGY, J. (1979a): Image de noyaux reconnaissables par diverses classes de transformations. – 4. Coll. Lille (1979), 79-101.

[64] DAUCHET, M. and MONGY, J. (1979b): Transformations de noyaux reconnaissables capacité générative des bimorphismes de forêts, – Fundam. Computation Theory '79 (Proc. Conf. Berlin/Wendisch-Rietz 1979), Akademie-Verlag, Berlin (1979), 92-97.

[65] DONER, J. E. (1965): Decidability of the weak second-order theory of two successors. – Notices Amer. Math. Soc. *12* (1965), Abstract 65T-468, 819.

[66] DONER, J. E. (1970): Tree acceptors and some of their applications. – J. CSS *4* (1970), 406-451.

[67] DUBINSKY, A. (1975): Computation on arbitrary algebras. – Symp. on λ-calculus and Computer Science Theory (Rome, 1975), LN in CS *37* (1975), 319-341.

[68] Duske, J. (1970): Funktionenautomaten. – Automaten und Formale Sprachen (Tagung Math. Forschungsinst., Oberwolfach, 1969), Bibliographisches Institut, Mannheim (1970), 23-26.

[69] Eilenberg, S. and Wright, J. B. (1967): Automata in general algebras. – IC *11* (1967), 452-470.

[70] Elgot, C. C. (1975): Monadic computation and iterative algebraic theories. – Logic Colloquium '73, Studies in Logic, Vol. 80 (Eds. M. E. Rose and J. C. Sheperdson), North-Holland, Amsterdam (1975), 175-230.

[71] Elgot, C. C., Bloom, S. L. and Tindell, R. (1978): On the algebraic structure of rooted trees. – J. CSS *16* (1978), 362-399.

[72] Ellis, C. A. (1971): Probabilistic tree automata. – IC *19* (1971), 401-416.

[73] Engelfriet, J. (1972): A note on infinite trees. – Information Processing Lett. 1 (1972), 229-232.

[74] Engelfriet, J. (1975a): Tree automata and tree grammars, – Lecture notes, DAIMI FN-10, Inst. Math., Aarhus Univ., Aarhus (1975).

[75] Engelfriet, J. (1975b): Bottom-up and top-down tree transformations. A comparison. – MST *9* (1975), 198-231.

[76] Engelfriet, J. (1976a): Surface tree languages and parallel derivation trees. – Theor. Comput, Sci. *2* (1976), 9-27.

[77] Engelfriet, J. (1976b): Some remarks on classes of macro languages. – 1. Coll. Lille (1976), 71-79.

[78] Engelfriet, J. (1976/77): Top-down tree transducers with regular look-ahead. – MST *10* (1976/77), 289-303.

[79] Engelfriet, J. (1977): Macro grammars, Lindenmayer systems and other copying devices. – Automata, Languages and Programming (Proc. Coll., Turku, 1977), LN in CS *52* (1977), 221-229.

[80] Engelfriet, J. (1978a): A hierarchy of tree transducers. – 3. Coll. Lille (1978), 103-106.

[81] Engelfriet, J. (1978b): On tree transducers for partial functions. – Inform. Process. Lett. *7* (1978), 170-172.

[82] Engelfriet, J. (1980): Some open questions and recent results on tree transducers and tree languages. – Formal language theory. Perspectives and open problems (ed. R. V. Book), Academic Press, New York (1980), 241-286.

[83] Engelfriet, J. (1982): Three hierarchies of transducers. – MST *15* (1982), 95-125.

[84] Engelfriet, J., Rozenberg, G. and Slutzki, G. (1980): Tree transducers, L systems, and two way machines. – J. CSS *20* (1980), 150-202.

[85] Engelfriet, J. and Schmidt, E. M. (1977, 1978): IO and OI.

I – J. CSS *15* (1977), 328-353.

II – J. CSS *16* (1978), 67-99.

[86] Engelfriet, J. and Skyum, S. (1976): Copying theorems. – Information Processing Lett. *4* (1976), 157-161.

[87] Engelfriet, J. and Skyum, S. (1982): The copying power of one-state tree transducers. – J. CSS *25* (1982), 418-435.

[88] Ésik, Z. (1978): On decidability of injectivity of tree transducers. – 3. Coll. Lille (1978), 107-133.

[89] Ésik, Z. (1979): On functional tree transducers. – Fundam. Computation Theory '79 (Proc. Conf., Berlin/Wendisch-Rietz 1979), Akademie-Verlag, Berlin (1979), 121-127.

[90] Ésik, Z. (1980): Decidability results concerning tree transducers I. – Acta Cybernet. *5* (1980). 1-20.

[91] Ésik, Z. (1981): An axiomatization of regular forests in the language of algebraic theories with iteration. – Fundamentals of computation theory (Proc, Conf., Szeged 1981), LN in CS *117* (1981), 130-136.

[92] Estenfeld, K. (1982): A new characterization theorem of treetransductions – Elektron. Informationsverarbeit. Kybernet. *18* (1982), 187-204.

[93] Ferenci, F. (1976) A new representation of context-free languages by tree automata – Found. Control. Engrg. *1* (1976) 217-222.

[94] Ferenci, F. (1980): Groupoids of pseudoautomata. – Acta Cybernet. *4* (1980), 389-399.

[95] Fischer, M. J. (1968): Grammars with macro-like productions – 9. IEEE Symp. (1968), 131-142.

[96] Fu, K.-S. (1980): Picture syntax. – Pictorial Information Systems (Eds, S. K. Chang and K.-S. Fu), LN in CS *80* (1980), 104-127.

[97] Fu, K.-S. (1982): Syntactic pattern recognition and applications. – Prentice-Hall, Englewood Cliffs, N. J. (1982).

[98] Fu, K.-S. and Bharkava, B. K. (1973): Tree systems for syntactic pattern recognition. – IEEE Trans. Computers C-*22* (1973), 1087-1099.

[99] Fu, K.-S. and Fan, T.-I. (1982): Tree translation and its application to a time-varying image analysis problem. – IEEE Trans. Systems, Man and Cybernetics, SMC – *12* (1982), 856-867.

[100] Fülöp, Z. (1981): On attributed tree transducers. – Acta Cybernet. *5* (1981), 261-279.

[101] Gécseg, F. (1977): Universal algebras and tree automata. – Fundamentals of Computation Theory (Proc, Symp., Poznań-Kórnik, 1977), LN in CS *56* (1977), 98-112.

[102] Gécseg, F. (1981): Tree transformations preserving recognizability. – Finite Algebra and Multiple-valued Logic (Record Coll. Universal Algebra, Szeged 1980), North-Holland, Amsterdam (1981), 251-273.

[103] Gécseg, F. and Horváth, Gy. (1976): On representation of trees and context-free languages by tree automata. – Found. Control Engrg, *1* (1976), 161-168.

[104] Gécseg, F. and Steinby, M. (1978a): Minimal ascending tree automata. – Acta Cybernet. *4* (1978), 37-44.

[105] Gécseg, F. and Steinby, M. (1978b, 1979): A faautomatak algebrai elmélete.

I – Mat. Lapok *26* (1978), 169-207.

II – Mat. Lapok *27* (1979), 283-336.

[106] Gécseg, F. and E.-Tóth, P. (1977): Algebra and logic in theoretical computer science. – Mathematical Foundations of Computer Science, 1977 (Tatranska Lomnica), LN in CS *53* (1977), 78-92.

[107] Georgeff, M. P. (1981): Interdependent translation schemes. – J. CSS *22* (1981), 198-219.

[108] Ginali, S. (1979): Regular trees and the free iterative theory. – J. CSS *18* (1979), 228-242.

[109] Ginsburg, G. and Mayer, O. (1982): Tree acceptors and grammar forms. – Computing *29* (1982), 1-9.

[110] Give'on, Y. (1971): Algebraic theory of m-ary systems. – Theory of machines and computations (Eds. Z. Kohavi and A. Paz), Academic Press, New York (1971), 275-286.

[111] Give'on, Y. and Arbib, M. A. (1968): Algebra automata II: the categorical framework for dynamic analysis. – IC *12* (1968), 346-370.

[112] Gnesi, S., Montanari, U. and Martelli, A. (1981): Dynamic programming as graph searching: an algebraic approach. – J. ACM *28* (1981), 737-751.

[113] Goguen, J. A. (1975): Semantics of computation. – Category Theory Applied to Computation and Control (Proc. Symp., San Francisco, 1974), LN in CS *25* (1975), 151-163.

[114] Goguen, J. A. and Thatcher, J. W. (1974): Initial algebra semantics. – 15. IEEE Symp. (1974), 63-77.

[115] Goguen, J. A., Thatcher, J. W., Wagner, E. G. and Wright, J. B. (1977): Initial algebra semantics and continuous algebras. – J. ACM *24* (1977), 68-95.

[116] Gonzalez, R. C., Edwards, J. J. and Thomason, M. G. (1976): An algorithm for the inference of tree grammars. – Intern. J. Comput, Information Sci. *5* (1976), 145-164.

[117] Gonzalez, R. C. and Thomason, M. G. (1978): Syntactic pattern recognition. – Addison Wesley, New York (1978).

[118] Hart, J. M. (1974): Acceptors for the derivation languages of phrase-structure grammars. IC *25* (1974), 75-92.

[119] Hart, J. M. (1976): The derivation language of a phrase structure grammar. – J. CSS *12* (1976), 64-79.

[120] Helton, F. J. (1976): The semigroup of an algebra automaton. – J. CSS *12* (1976), 13-24.

[121] Höpner, M. (1971): Eine Charakterisierung der Szilardsprachen. – GI-4. Jahrestagung (Berlin, 1974), LN in CS *26* (1975), 113-121.

[122] Horváth, Gy. (1979): On machine maps in categories. – Fundamentals of Computation Theory '79 (Proc. conf., Berlin/Wendisch-Rietz 1979), Akademie-Verlag, Berlin (1979), 182-186.

[123] Horváth, Gy. (1981): Functor state machines. – Acta Cybernet, *6* (1981), 147-172.

[124] Hübler, A. (1975): Zur Dechiffrierung von Baum-Akzeptoren mittels Mehrfachexperimenten. – Elektron. Informationsverarb. Kybernet. *11* (1975), 590-593.

[125] Hupbach, U. L. (1978): Rekursive Funktionen in mehrsortigen Peano-Algebren. – Elektron. Informationsverarb. Kybernet. *14* (1978), 491-506.

[126] Inoue, K. and Nakamura, A. (1976): Some topological properties of Σ-structure automata. – S-C-C *7* (1976), No. 5, 19-27.

[127] Ito, T. and Ando, S. (1974): A complete axiom system of super-regular expressions. – Proc. IFIP Congress 74 (Stockholm, 1974), 661-665.

[128] Ito, H. and Fukumura, T. (1974): Dendrolanguage generating systems on sets of control strings. – S-C-C *5* (1974), No. 4, 9-17.

[129] Ito, H., Inagaki, Y. and Fukumura, T. (1973a): Characterization of derivation trees of context sensitive tree generating systems. – S-C-C *4* (1973), No.2, 24-32.

[130] Ito, H., Inagaki, Y. and Fukumura, T. (1973b): Scattered tree automata and scattered context-sensitive tree-generating systems. – S-C-C *4* (1973), No.4, 22-28.

[131] Ito, H., Inagaki, Y. and Fukumura, T. (1973c): Hierarchy of the families of dendrolanguages. – S-C-C *4* (1973), No. 5, 48-56.

[132] Ito, H., Inagaki, Y. and Fukumura, T. (1974): Dendrolanguage generating systems on control state sets. A hierarchy between context-free and context-sensitive dendrolanguages, – S-C-C *5* (1974), No. 5, 1-8.

[133] Jacob, G. (1979): Elements de la théorie algébriques des arbres. – Fundamentals of Computation Theory '79 (Proc, Conf., Berlin/Wendisch-Rietz 1979), Akademie-Verlag, Berlin (1979), 193-206.

[134] Joshi, A. K. and Levy, L. S. (1977): Constraints on structural descriptions: Local transformations. – SIAM J. Comput. *6* (1977), 272-284.

[135] Joshi, A. K., Levy, L. S. and Takahashi, M. (1973): A tree generating system. – Automata, Languages and Programming (Proc. Symp., Rocquencourt, 1972), North-Holland, Amsterdam (1973), 453-465.

[136] Joshi, A. K., Levy, L. S. and Takahashi, M. (1975): Tree adjunct grammars. – J. CSS *10* (1975), 136-163.

[137] Joshi, A. K., Levy, L. S. and Yueh, K. (1980): Local constraints in programming languages. Part I: Syntax. – Theoret. Comput. Sci. *12* (1980), 265-280.

[138] Kamimura, T. and Slutzki, G. (1979): DAGs and Chomsky hierarchy (extended abstract). – Automata, languages and programming, (6th Colloq., Graz 1979), LN in CS *71* (1979), 331-337.

[139] Kamimura, T. and Slutzki, G. (1982): Transductions of dags and trees. – MST *15* (1982), 225-249.

[140] Karpiński, M. (1973a, b, c, 1974a): Free structure tree automata.

I – Equivalence. – Bull. Acad. Polon. Sci. Sér, Sci. Math. Astron. Phys. *21* (1973), 441-446.

II – Nondeterministic and deterministic regularity. – ibid *21* (1973), 447-450.

III – Normalized climbing automata. – ibid. *21* (1973), 567-572.

IV – Sequential representation. – ibid. *22* (1974), 87-91.

[141] Karpiński, M. (1974b): Probabilistic climbing and sinking languages. – Bull. Acad. Sci. Sér, Sci. Math. Astron. Phys. *22* (1974), 1057-1061.

[142] Karpiński, M.(1975): Stretching by probabilistic tree automata and Santos grammars. – Mathematical Foundations of Computer Science (Proc. Symp., Jadwisin 1974), LN in CS *28* (1975), 249-255.

[143] Karpiński, M. (1977): The equivalence problems for binary EOL-systems are decidable. Fundamentals of Computation Theory (Proc, Symp., Poznań-Kórnik, 1977), LN in CS *56* (1977), 423-434.

[144] Kawahara, Y. (1980): Relational tree automata and context-free sets. – Bull. Kyushu Inst. Technol., Math. Nat. Sci. *27* (1980), 17-25.

[145] Kawahara, Y. and Yamaguchi, M. (1980): Minimal realization theory for free process machines in monoidal categories. – Mem. Fac. Sci. Kyushu Univ. Ser. A. *34* (1980), No. 1, 71-78.

[146] Kojima, M. and Honda, N. (1972): Properties of context-sensitive tree automata and characterizations of derivation trees of context-sensitive grammars. – S-C-C *3* (1972), No. 5, 23-30.

[147] Kojima, M. and Honda, N. (1973): A characterization of sets of trees acceptable by tree automata. – S-C-C 4 (1973), No. 1, 40-47.

[148] Kozen, D. (1977): Complexity of finitely presented algebras. – 9. Ann. ACM STC (Boulder, Co1. 1977), 164-177.

[149] Lawvere, F. W. (1963): Functorial semantics of algebraic theories. – Proc. Nat. Acad. Sci. USA *50* (1963), 869-872.

[150] Lescanne, P. (1976): Equivalence entre la famille des ensembles réguliers et la famille des ensembles algébriques. – RAIRO Inform. Théor. Sér. Rouge *10* (1976), No. 8, 57-81.

[151] Lescanne, P. (1977): Quelques applications des classes équationelles conformes. – 2. Coll. Lille (1977), 199-212.

[152] Levine, B. (1981): Derivatives of tree sets with applications to grammatical inference. – IEEE Trans. Pattern Anal. & Mach. Intell., PAMI-*3* (1981), 285-293.

[153] Levine, B. (1982): The use of tree derivatives and a sample support parameter for inferring tree systems. – IEEE Trans. Pattern Anal. Mach. Intell., PAMI-*4* (1982), 25-34.

[154] Levy, L. S. (1971): Tree adjunct, parenthesis, and distributed adjunct grammars. – Theory of machines and computations (Eds. Z. Kohavi and A. Paz), Academic Press, New York (1971), 127-142.

[155] Levy, L. S. (1973): Structural aspects of local adjunct grammars. – IC *23* (1973), 260-287.

[156] Levy, L. S. (1980): Discrete structures of computer science. – John Wiley & Sons, New York (1980).

[157] Levy, L. S. and Joshi, A. K. (1973): Some results in tree automata. – MST *6* (1973), 334-342.

[158] Levy, L. S. and Joshi, A. K. (1978): Skeletal structural descriptions. – IC *39* (1978), 192-211.

[159] Lilin,E. (1978a): S-transducteurs de forêts. – 3. Coll. Lille (1978), 189-206.

[160] Lilin, E. (1978b): Une generalization des transducteurs d'etats finis d'arbres: les S-transducteurs. – Thése de doctorat, Université de Lille I (1978).

[161] Lilin, E. (1981): Transducteurs finis d'arbres et tests d'egalite. – RAIRO Inform. Theor. *15* (1981), 213-232.

[162] Lippe, W.-M. (1982): Context-sensitive top-down creative dendrogrammars. – Bull. EATCS, No. 9 (Oct. 1979), 41-45.

[163] Lu, S. Y. (1979a): Stochastic tree grammar inference for texture synthesis and discrimination. – Comput. Graphics and Image Process. *9* (1979), 234-245.

[164] Lu, S. Y. (1979b): A tree-to-tree distance and its application to cluster analysis. – IEEE Trans. Pattern. Anal. & Mach. Intell., PAMI-*1* (1979), 219-224.

[165] Lu, S.Y.and Fu, K.-S. (1978): Error-correcting tree automata for syntactic pattern recognition. IEEE Trans. Comput, C-*27* (1978), 1040-1053.

[166] Magidor, M. and Moran, G. (1969): Finite automata over finite trees. – Technical Report 30, Hebrew University, Jerusalem (1969).

[167] Magidor, M. and Moran, G. (1970): Probabilistic tree automata. – Israel J. Math. *8* (1970), 340-348.

[168] Mahn, F. K. (1969): Primitiv-rekursive Funktionen auf Termmengen, – Arch. Math. Logik Grundlagenforsch. *12* (1969), 54-65.

[169] Maibaum, T. S. E. (1972): The characterization of the derivation trees of context-free sets of terms as regular sets. – 13. IEEE Symp. (1972), 224–230.

[170] Maibaum, T. S. E. (1974): A generalized approach to formal languages. – J. CSS *8* (1974), 409-439.

[171] Maibaum, T. S. E. (1978): Pumping lemmas for term languages. – J. CSS *17* (1978), 319-330.

[172] Marchand, P. (1976): Bigrammes et systemes transformationnels. – I. Coll. Lille (1976), 175-195.

[173] MARCHAND, P. (1979): Construction des algèbres minimales des sous-ensembles des algèbres libres. Applications aux parties reconnaissables. – 4. Coll. Lille (1979), 134-158.

[174] MARCHAND, P. (1980): Grammaires paranthésés et bilangages réguliers, – RAIRO Inform. Theor. *14* (1980), 3-38.

[175] MARCHAND, P. (1981): Langages d'arbres. Langages dans les algèbres libres. – Thesis, CRIN 81-T-030, Universitè de Nancy, Nancy (1981).

[176] MARÓTI, G. (1977): Rational representation of forests by tree automata. – Acta Cybernet. *3* (1977), 309-320.

[177] MARTIN, D. E. and VERE, S. A. (1970): On syntax-directed transduction and tree transducers. – 2. Ann. ACM STC (1970), 129-135.

[178] MAYER, O. (1975): On the analysis and synthesis problems for context-free expressions. – Mathematical Foundations of Computer Science (Proc. Symp., Mariánské Lázně 1975), LN in CS *32* (1975), 308-314.

[179] MEISSNER, H.-G. (1976): Über die Fortsetzbarkeit von sequentiellen Baumoperatoren mit endlichem Gewicht. – Elektron. Informationsverarbeit. Kybernet. *11* (1976), 578-579.

[180] MEISSNER, H.-G. (1977): Zu einigen Begriffen und Resultaten aus der Theorie der Baumautomaten. – Rostock. Math. Kolloq. *3* (1977), 85-102.

[181] MERZENICH, W. (1979): A binary operation on trees and an initial algebra characterization for finite tree types. – Acta Inform. *11* (1979), 149-168.

[182] MEZEI, J. and WRIGHT, J. B. (1967): Algebraic automata and context-free sets. – IC *11* (1967), 3-29.

[183] Модина, Л. С. (1975a): Деревные грамматики и языки. – Кибернетика (Киев) (1975). No. 5, 86-93.

[184] MODINA, L. S. (1975b): On some formal grammars generating dependency trees. – Mathematical Foundations of Computer Science 1975 (Proc. Symp. Mariánské Lázně), LN in CS *32* (1975), 326-329.

[185] MOSTOWSKI, A. W. (1979): A note concerning the complexity of a decision problem for positive formulas in SkS. – 4. Coll. Lille (1979), 173-180.

[186] MOSTOWSKI. A. W. (1982): Determinancy of sinking automata on infinite trees and inequalities between Rabin's pair indices. – Information Processing Lett. *15* (1982), 159-163.

[187] NG, P. and YEH, R. T. (1973): Tree transformations via finite recursive transition machines. – Mathematical Foundations of Computer Science (Proc. Symp., High Tatras 1973), 273-278.

[188] NG, P. A. and YEH, R. T. (1976): Sequential tree-walking automata. – Nanta Math. IX (1976), 159-167.

[189] NIVAT, M. (1973): Langages algébriques sur le magma libre et sémantique des schémas de programme. – Automata, Languages and Programming (Proc. Symp., Rocquencourt 1972), North-Holland, Amsterdam (1973), 367-376.

[190] OGDEN, W. F. and ROUNDS, W. C. (1972): Compositions of n tree transducers. – 4. Ann. ACM STC (1972), 198-206.

[191] OPP, M. (1975a): Eine Beschreibung contextfreier Sprachen durch endliche Mengensysteme. Automata Theory and Formal Languages (2^{nd} GI Conf., Kaiserslautern 1975), LN in CS *33* (1975), 190-197.

[192] OPP, M. (1975b): Allgemeine Σ-Grammatiken. – GI-5. Jahrestagung (Dortmund 1975), LN in CS *34* (1975), 420-428.

[193] OPP, M. (1976): Characterizations of recognizable subsets in generic algebras. – 1. Coll. Lille (1976), 164-174.

[194] PAIR, C. (1976a): Inference for regular bilanguages. – Formal Languages and Programming (Proc. Semin., Madrid 1975), North-Holland, Amsterdam (1976), 15-30.

[195] PAIR, C. (1976b): Les arbres en theorie des langages. – 1. Coll. Lille (1976),196-216.

[196] PAIR, C. and QUERE, A. (1968): Definition et étude des bilangages réguliers, – IC *13* (1968), 565-593.

[197] PERRAULT, C. R. (1976a): Intercalation lemmas for tree transducer languages. – J. CSS *13* (1976), 246-277.

[198] PERRAULT, C. R. (1976b): Augmented transition networks and their relation to tree transducers. – Information Sci. *11* (1976), 93-120.

[199] PETROV, S. V. (1978): Graph grammars and automata (survey). – Autom. Remote Control *39* (1978), 1034-1050.

[200] PETTOROSSI, A. (1976): Combinators as tree transducers. – 2. Coll. Lille (1976), 213-223.

[201] PYSTER, A. (1978): Context-dependent tree automata. – IC *38* (1978), 81-102.

[202] PYSTER, A. and BUTTELMANN, H. W. (1978): Semantic-syntax-directed translation. – IC *36* (1978), 320-361.

[203] RABIN, M. O. (1967): Mathematical theory of automata. – Mathematical Aspects of Computer Science (Proc. Symp. Appl. Math. XIX), Amer. Math. Soc., Providence (1967), 153-175.

[204] RABIN, M. O. (1969): Decidability of second-order theories and automata on infinite trees. Trans. Amer. Math. Soc. *141* (1969), 1-35.

[205] RABIN, M. O. (1970): Weakly definable relations and special automata. – Mathematical Logic and Foundations of Set Theory (Proc. Coll., Jerusalem 1968), North-Holland, Amsterdam (1970), 1-23.

[206] RAOULT J.-C. (1981): Finiteness results on rewritting systems. – RAIRO Inform. Théor, *15* (1981), 373-391.

[207] REISIG, W. (1979): A note on the representation of finite automata. – Inform. Process. Lett. *8* (1979), 239-240.

[208] RÉVÉSZ, Gy. (1977): Algebraic properties of derivation words. – 2. Coll. Lille (1977), 224-234.

[209] RICCI, G. (1973): Cascades of tree-automata and computations in universal algebras. – MST *7* (1973), 201-218.

[210] RIHA, A. (1981): A certain type of dependency tree transformations. – Mathematical logic in computer science (Proc. Coll., Salgótarján, Hungary, Sept. 10-15, 1978), Elsevier North-Holland Publ. Co., New York (1981), 699-709.

[211] ROSEN, B. K. (1973): Tree-manipulating systems and Church-Rosser theorems. – J. ACM *20* (1973), 160-187.

[212] ROSEN, B. K. (1974): Syntactic complexity. – IC *24* (1974), 305-335.

[213] ROUNDS, W. C. (1969): Context-free grammars on trees. – 1. Ann. ACM STC (1969), 143-148.

[214] ROUNDS, W. C. (1970a): Tree-oriented proofs of some theorems on context-free and indexed languages. – 2. Ann. ACM STC (1970), 109-116.

[215] ROUNDS, W. C. (1970b): Mappings and grammars on trees. – MST *4* (1970), 257-287.

[216] ROUNDS, W. C. (1973): Complexity of recognition in intermediate-level languages. – 14. IEEE Symp. (1973), 145-158.

[217] SCHREIBER, P. P. (1976): Tree transducers and syntax-connected transductions. – 1. Coll. Lille (1976), 217-238.

[218] SCHÜTT, D. (1970): Baumautomaten. – Bericht 36, Gesellschaft für Math. u. Datenverarbeitung, Bonn (1971).

[219] SCHÜTT, D. (1973): Zustandsfolgenabbildungen von verallgemeinerten endlichen Automaten. – 1. Fachtagung über Automatentheorie und Formale Sprachen (Bonn 1973), LN in CS *2* (1973), 88-97.

[220] Shepard, C. D. (1969): Languages in general algebras. – 1. Ann. ACM STC (1969), 155-163.

[221] Shi, Q.-Y. and Fu, K.-S. (1982): Efficient error-correcting parsing for (attributed and stochastic) tree grammars. – Information Sciences *26* (1982), 159-188.

[222] Siefkes, D. (1978): An axiom system for the weak monadic second-order theory of two successors. – Israel J. Math. *30* (1978), 264-284.

[223] Sommerhalder, R. (1974): Monoids associated with algebras and automata. – Unpublished Report, Delft (1974).

[224] Steinby, M. (1977a): On algebras as tree automata. – Contributions to Universal Algebra (Record Coll. Universal Algebra, Szeged 1975), North-Holland, Amsterdam (1977), 441-455.

[225] Steinby, M.(1977b): On the structure and realizations of tree automata. – 2. Coll. Lille (1977), 235-248.

[226] Steinby, M. (1979): Syntactic algebras and varieties of recognizable sets. – 4. Coll. Lille (1979), 226-240.

[227] Steinby, M. (1981): Some algebraic aspects of recognizability and rationality. – Fundamentals of computation theory (Proc Conf., Szeged 1981), LN in CS *117* (1981), 360-372.

[228] Steyart, J.-M. (1977a): Sur les index rationelles des feuillages de forêts lineaires. – C. R. Acad. Sci. Paris, Sér, A, t. 285 (1977), 473-476.

[229] Steyart, J.-M. (1977b): Evaluation des index rationnels de quelques familles de langages. – Technical Report No. 261, IRIA, Rocquencourt, France (1977).

[230] Steyart, J.-M. (1978): Index rationnel des ETOL-Jangages. – 3. Coll. Lille (1978), 246-249.

[231] Szilard, A. L. (1974): Ω-OL systems. – L-systems, LN in CS *15* (1974), 258-291.

[232] Tai, K.-CH. (1979): The tree-to-tree correction problem. – J. ACM *26* (1979), 422-433.

[233] Takahashi, M. (1973): Primitive transformations of regular sets and recognizable sets. – Automata, Languages and Programming (Proc. Coll., Roquencourt 1972), North-Holland, Amsterdam (1973), 475-480.

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

[235] Takahashi, M. (1975b): A mathematical approach to the structure of language. On the fundamental concept of a tree (Japanese). – Sugaku *27* (1975), 241-252.

[236] Takahashi, M. (1977): Rational relations on binary trees. – Automata, Languages and Programming (Proc, Coll. Turku 1977), LN in CS *52* (1977), 524-538.

[237] Thatcher, J. W. (1967): Characterizing derivation trees of context-free grammars through a generalization of finite automata theory. – J. CSS *1* (1967), 317-322.

[238] Thatcher, J. W. (1970): Generalized^2 sequential machines. – J. CSS *4* (1970), 339-367.

[239] Thatcher, J. W. (1973): Tree automata: an informal survey. – Currents in the Theory of Computing (ed. A. V. Aho), Prentice-Hall, Englewood Cliffs, N. J. (1973), 143-172.

[240] Thatcher, J. W. and Wright, J. B. (1965): Generalized finite automata. – Notices Amer. Math. Soc. *12* (1965), Abstract No. 65T- 649, 820.

[241] Thatcher, J. W. and Wright, J. B. (1968): Generalized finite automata theory with an application to a decision problem of second order logic. – MST *2* (1968), 57-81.

[242] Tiuryn, J. (1977a, b): Fixed-points and algebras with infinitely long expressions.

I – Mathematical Foundations of Computer Science 1977 (Proc. Symp., Tatranska Lomnica), LN in CS *53* (1977), 513-522.

II – Fundamentals of Computation Theory (Proc. Symp., Poznań-Kórnik 1977), LN in CS *56* (1977), 332-339.

[243] Tokura, N. and Kasami, T. (1974): Automata with labelled tree inputs. – S-C-C *5* (1974), No. 3, 88-95.

[244] Trnková, V. and Adámek, J. (1979): Tree-group automata. – Fundamentals of Computation Theory '79 (Proc. Conf., Berlin/Wendisch-Rietz 1979), Akademie-Verlag (1979), 462-468.

[245] Turner, R. (1973): An infinite hierarchy of term languages - an approach to mathematical complexity. – Automata, Languages and Programming (Proc. Symp., Rocquencourt 1972), North-Holland Amsterdam (1973), 593-608.

[246] Turner, R. (1975): An algebraic theory of formal languages. – Mathematical Foundations of Computer Science (Proc. Symp. Mariánské Lázně 1975), LN in CS *32* (1975), 426-431.

[247] Upton, R. A. (1981): An extension of tree adjunct grammars. – IC *51* (1981), 248-274.

[248] Virágh, J. (1980): Deterministic ascending tree automata I. – Acta Cybernet. *5* (1980), 33-42.

[249] Wagner, E. G. (1971): An algebraic theory of recursive definitions and recursive languages. – 3. Ann. ACM STC (1971), 12-23.

[250] Wagner, E. G., Wright, J. R, Goguen, J. A. and Thatcher, J. W. (1976): Some fundamentals of order-algebraic semantics. – Mathematical Foundations of Computer Science (Proc. Symp. Gdańsk 1976), LN in CS *45* (1976), 153-168.

[251] Williams, K. L. (1975): A multidimensional approach to syntactic pattern recognition. – Pattern Recognition *7* (1975), 125-137.

[252] Wright, J. B., Thatcher, J. W., Wagner, E. G. and Goguen, J. A. (1976): Rational algebraic theories and fixed-point solutions. – 17. IEEE Symp. (1976), 147-158.

[253] Yeh, R. T. (1971): Some structural properties of generalized automata and algebras. – MST *5* (1971), 306-318.

[254] Zachar, Z. (1979): The solvability of the equivalence problem for deterministic frontier-to-root tree transducers. – Acta Cybernet. *4* (1979), 167-177.

Index

5 APPENDIX

SOME FURTHER TOPICS AND REFERENCES

by Magnus Steinby

The purpose of this Appendix is to supplement the original book with notes on some further topics and a selection of more recent references. The choice of topics and references is partly influenced by personal preferences, but I trust that the areas included deserve to be mentioned, and that the general expositions, surveys and research papers appearing in the bibliography are useful. Hence, I hope that these notes may serve as an initial guide to the subjects discussed, and that they give an idea of the continuing vitality of the theory and of its applications.

Before considering any specific areas, let me note some works of a general nature published after *Tree Automata* was written. J. R. Büchi's posthumous book *Finite Automata, Their Algebras and Grammars* [11] appeared in 1989 (edited by D. Siefkes). The main part of it treats unary algebras, finite acceptors, regular languages and production systems, but in a manner that suggests tree automata and tree languages as natural generalizations. The last two chapters deal with terms, trees, algebras as tree automata, tree grammars, and connections between context-free languages and pushdown automata. Especially this latter part of the book appears quite unfinished, but the author's grand design, a theory that would encompass algebras, automata, formal languages and rewriting systems, is clearly discernible. The terminology and notation is often nonstandard, sometimes even confusing, but a patient reader is rewarded by original insights and interesting historical remarks.

The book *Tree Automata and Languages* [66] edited M. Nivat and A. Podelski, which appeared in 1992, is a collection of papers that discuss a variety of topics involving trees. The survey paper [46] by F. Gécseg and M. Steinby may be viewed as a condensed and somewhat modernized version of *Tree Automata*, but it also takes up some further topics and its bibliography includes many additional items.

In their book *Syntax-Directed Semantics. Formal Models Based on Tree Transducers* [44], Z. Fülöp and H. Vogler consider formal models of syntax-directed semantics based on tree transducers. They also develop a fair amount of the general theory of total deterministic top-down, macro, attributed, and macro attributed tree transducers. In particular, they compare with each other the classes of tree transformations defined by the different types of tree transducers, and they present several composition and decomposition results for these tree transformations.

The internet book *Tree Automata Techniques and Applications* [13], to be referred to as TATA, is a joint enterprise of several authors. First launched in 1997, it has already been revised and extended a few times. The presentation is often rather informal, but the ideas are richly illustrated by examples and many interesting facts are also given as exercises. The first two chapters review some basic material about finite tree recognizers, regular tree languages, and regular tree grammars, but also mention context-free tree languages. Chapter 6 contains a brief account of tree transducers (without proofs). The remaining five chapters deal with topics not covered by our book. The tutorial [55] by C. Löding focuses on applications of tree automata and emphasizes algorithmic aspects.

Automata on infinite trees and the connections between *tree automata and logic* were the most important topics excluded from *Tree Automata*. The two are strongly linked with each other and have been studied intensively ever since tree automata were introduced, and by now they form an extensive theory with important applications to logic and computer science. Although mainly concerned with the word case, the survey papers [81] and [82] by W. Thomas offer very readable introductions to this area, and they also include extensive bibliographies. Chapter 3 of TATA [13] is a further useful general reference, and some of the papers in [66] deal with this topic. The book *Automata, Logics, and Infinite Games* [50] edited by E. Grädel, Thomas and T. Wilke contains twenty tutorial papers that form an excellent overview of the study of automata, logics and games. About half of them concern trees and tree automata. Besides MSO logics, they elucidate the uses and properties of various modal logics, fixed-point logics and guarded logics, and demonstrate the usefulness of alternating tree automata.

The continual development of the theory of *tree transformations* is also largely driven by applications, and tree transducers will be mentioned also in connection with some the other themes to be discussed below. Here I shall note separately a few important topics. The study of compositions of tree transformation classes initiated by B. S. Baker (1973, 1979)[1] and J. Engelfriet (1975) has been pursued further especially by Fülöp and S. Vágvölgyi [40, 42, 43, 35]. In particular, they have considered semigroups of the compositions of some given tree transformation classes, and presented rewriting systems by which the equality of two composition classes can be decided. They have also considered some variants of Engelfriet's (1977) important idea of regular look-ahead for top-down tree transducers ([41], for example). Recently, Engelfriet, S. Maneth and H. Seidl [25] have shown that in certain cases it can be decided whether a deterministic top-down tree transducer with regular look-ahead is equivalent to a deterministic top-down tree transducer, and that such a transducer without look-ahead can be constructed if the answer is positive. Macro tree transducers were first defined by Engelfriet (1980) but, as noted in [44] for example, the primitive recursive program schemes independently introduced by B. Courcelle and P. Franchi-Zannettacci [14] amount to many-sorted versions of them. Macro and other higher-level tree transducers have been studied in depth by Engelfriet and Vogler [26, 27, 28, 29] (cf. also [21, 22]). For further information about these matters, I recommend the bibliographic notes in [44]. The work [8] on equational tree transformations by S. Bozapalidis, Fülöp and G. Rahonis is a natural extension of a classical theme.

The decidability of the question whether the image of a given regular tree language

[1]The references can be found in the original bibliography of *Tree Automata*

under a given tree homomorphism is regular, has been a relatively long-standing open problem, but recently an affirmative solution was presented by G. Godoy and O. Giménez [48]. Their approach uses tree automata with equality or disequality tests, and their work contains also some results of independent interest concerning such automata. Moreover, it has some applications to term rewriting and XML theory. Fülöp and P. Gyenizse [37] have shown that injectivity is undecidable for tree homomorphisms while it is decidable for linear deterministic top-down tree transformations. Furthermore, in [36] Fülöp proves that several questions concerning the ranges of deterministic top-down tree transformations are undecidable. The decidability of the equivalence of deterministic top-down tree transducers was proved by Ésik already in 1980. More recently, Engelfriet, Maneth and Seidl [24] showed that the equivalence of total deterministic top-down tree transducers can be decided in polynomial time by reducing the transducers to a certain canonical form, and their method can be applied also to deterministic top-down tree transducers with regular look-ahead. In [34], S. Friese, Seidl and Maneth present a corresponding equivalence checking algorithm based on normal forms for bottom-up tree transducers. In [23], Engelfriet and Maneth prove that the equivalence of deterministic MSO tree transducers is decidable. These results, as well as many other decidability questions for tree transducers are discussed in the recent survey paper [58] by Maneth. Finally, two quite recent contributions should be mentioned. Firstly, Seidl, Maneth and G. Kemper [78] prove the decidability of the equivalence of deterministic top-down tree-to-string transducers. In [33], E. Filiot, Maneth, P.-A. Reynier and J.-M. Talbot introduce tree transducers for which every output tree is augmented with information about the origin of each of its nodes, and prove several decidability results concerning the equivalence or injectivity of such transducers.

Since terms can be seen as syntactic representations of trees over ranked alphabets, it is to be expected that there are some connections between *tree automata and term rewriting systems* (TRSs). Indeed, various tree automata and tree grammars are often defined as special term rewriting systems. On the other hand, tree automata can be used for solving problems concerning TRSs and such applications have, in turn, inspired new developments in the theory tree automata. In the mid-1980s it was noted that the set $Red(\mathcal{R})$ of terms reducible by a finite left-linear TRS $\mathcal{R}$, as well as its complement, the set $Irr(\mathcal{R})$ of irreducible terms, are regular tree languages. Since this means that many questions concerning reducibility and normal forms are decidable for such TRSs, the observation was quickly followed by several studies of related matters. Thus it was shown that a finite TRS $\mathcal{R}$ for which $Red(\mathcal{R})$ is regular can be "linearized" and that the regularity of $Red(\mathcal{R})$ is decidable, the regular sets $Red(\mathcal{R})$ were characterized in terms of a new class of finite tree automata, and questions of ground reducibility were considered. So-called monadic and semi-monadic TRSs were studied using tree pushdown automata. For extending such applications to TRSs that are not left-linear, new classes of tree automata are needed. The problem here is that automata that are able to recognize also non-regular sets $Red(\mathcal{R})$ or the sets of all ground instances of a given non-linear term, tend to be too powerful to be manageable themselves. An example of increased power combined with good decidability properties is provided by the automata with comparisons between brothers introduced in the 1990s. The ground tree transducer is another important tree automaton sprung from the theory of term rewriting. Much material concerning these matters can be found in TATA [13], and introductions to this subject and many references are provided also by the surveys [47], [67] and [79]. For some recent work on this theme, cf. [83], for example.

Weighted tree automata, tree series and weighted tree transformations have been studied quite extensively in recent years. Most aspects of this work (up to around 2009) are reviewed in the handbook chapter [45] by Fülöp and Vogler, and a broad introduction is provided also by the survey paper [31] by Z. Ésik and W. Kuich. *Weighted logics* for weighted tree automata have been studied by M. Droste, Vogler and others, cf. [18, 39], for example. Equational weighted tree transformations are considered by Bozapalidis, Fülöp and Rahonis [9]. In [71] Rahonis introduces weighted Muller-automata on infinite trees and a corresponding weighted MSO-logic. The dissertation [61] of C. Mathissen contains, among other matters, also much interesting material belonging to this area as well as a useful bibliography.

In an *unranked tree* a node labeled with a given symbol may have any number of children. Languages of such trees were considered already in the 1960s in two notable papers. J. W. Thatcher (1967) introduced finite unranked tree recognizers and showed that the yields of the recognizable unranked tree languages are precisely the context-free languages. C. Pair and A. Quere (1968) created an algebraic framework for the study of regular unranked tree languages that also incorporated *hedges*, i.e. finite sequences of unranked trees, and they proved many of the usual properties of regular sets for recognizable unranked tree languages. Nevertheless, the topic received little attention before it was discovered that it is natural to represent XML documents by unranked trees and that unranked tree automata may be useful for handling questions concerning them. The revival of the theory of unranked tree and hedge languages by M. Murata et al. [63, 64, 10] initiated a lively activity in the area. TATA [13] devotes a chapter to unranked tree languages and their applications. As a sample from the extensive literature, let us mention just the papers [15, 59, 60, 65] and the survey [77] by T. Schwentick. Since this work is mostly quite application-oriented, algorithmic and complexity issues are much to the fore. X. Piao and K. Salomaa [69, 70] have considered state complexity questions connected with conversions between different types of unranked automata as well as lower bounds for the size of unranked tree automata. An overview of logics for unranked trees is given by L. Libkin [54]. Weighted unranked tree automata are studied in [19] and [17] by Droste, Vogler, and D. Heusel.

Natural language description and processing has become an important area of application of the theory of tree automata and tree languages. Of course, parse trees of natural languages have always been prime examples of 'trees' and some of the early works on tree automata explicitly refer to linguistic motivations, but the current activity took really off much later. In his book [62] F. Morawietz discusses *formalizations of natural language syntax* that are based on monadic second-order (MSO) logic on trees and tree language theory. A key fact here is the effective correspondence between weak MSO logic and finite tree automata established already by Thatcher and J. B. Wright (1965, 1968) and J. Doner (1965, 1970), but actually a whole array of tree language-related notions are utilized or noted as potentially useful. These include tree walking automata [4, 6, 7] macro tree transducers [26, 21], and tree-adjoining grammars (cf. [51], for a survey). Recently, the theory of tree automata has attracted the attention of linguists especially because of the promise shown by tree-based approaches to *machine translation*. Besides classical notions and results appearing already in our book, work in this area draws also upon some newer developments. In particular, it has both utilized and inspired work on unranked and weighted tree languages as well as weighted tree transducers. Furthermore, it has revived the interest in the generalized top-down tree transducers studied much earlier by E. Lilin (1978). Also compositions and decompositions of various tree transformations are used in machine translation systems. The papers [52, 53] expose some of the relevant questions from a linguist's point of view, while the papers [20, 49, 56, 57] form a sample of theoretical work in the area.

Almost all papers on *varieties of tree languages*, and *classes of special regular tree languages* in general, have appeared after 1984. Most of the work in this area published before 2005 is at least mentioned in the survey [80], and all the references pointed to (by author and year) below can be found there. Eilenberg-like variety theories for tree languages were presented by Steinby (1979, 1992, 1998) and J. Almeida (1990, 1995). Ésik (1999) has set forth a variety theory in which finitary algebraic theories take the place of finite algebras, and later he together with P. Weil [32] formulated a similar theory in terms of preclones. Syntactic monoids of tree languages were introduced by Thomas (1982, 1984) and studied further by Salomaa (1983). A similar notion for binary trees has been used by Nivat and Podelski (1989, 1992). The families of regular tree languages considered in the literature include those of the finite and co-finite tree languages (Gécseg and B. Imreh 1988), definite, reverse definite and generalized definite tree languages (U. Heuter 1989, 1992), k-testable tree languages (Heuter 1989, T. Knuutila 1992), and aperiodic tree languages (Thomas 1984). All of them are varieties of tree languages (cf. Steinby 1992, 1998), and in some cases the corresponding varieties of finite algebras are also known.

Although Thomas (1984) could characterize the aperiodic tree languages by their syntactic monoids, it was obvious that such a characterization is not possible for all varieties of tree languages. This was confirmed when S. Salehi [72] described the (generalized) varieties definable by syntactic monoids or semigroups. His result shows, for example, that the definite tree languages cannot be characterized by syntactic semigroups (as claimed in an earlier paper). However, in [12] A. Cano Gomez and Steinby introduce generalized syntactic semigroups (and monoids) in terms of which the definite tree languages can be characterized. Wilke (1996) gave an effective characterization of the reverse definite binary tree languages in terms of so-called tree algebras. Salehi and Steinby [74] studied the tree algebra formalism in some detail and presented a variety theorem for it. Noticing that the well-known equivalence of aperiodicity, star-freeness, and first-order definability of string languages fails for trees, Thomas (1984) introduced logics in which set quantifications are limited to chains or to antichains of nodes. He proved then, for example, that a regular tree language is star-free iff it is antichain-definable. This line of research has been pursued further by Heuter (1989, 1991) and A. Potthoff (1994, 1995), for example.

Some families of tree languages have been introduced by first defining a class of finite algebras. For example, the monotone tree languages studied by Gécseg and Imreh (2002) were defined as the languages recognized by monotone algebras. Similarly, Ésik and Sz. Iván [30] introduce a hierarchy of aperiodicity notions for finite algebras and consider then, besides the properties of the obtained varieties of finite algebras, the corresponding families of tree languages. There are a few different extensions of the variety theory of tree languages: positive varieties of tree languages by T. Petković and Salehi [68], varieties of many sorted sets (with tree languages as a special case) by Salehi and Steinby [73], and varieties of recognizable tree series by Fülöp and Steinby [38].

A section of *Tree Automata* is devoted to deterministic root-to-frontier (DR) recognizers and DR tree languages, but the topic has been studied quite extensively also later. In her thesis E. Jurvanen (1995) considers closure properties and the variety generated by DR tree languages as well as ways of strengthening DR recognizers. The latter include, in particular, the regular frontier check mechanism introduced by Jurvanen, Potthoff and Thomas (1994). The thesis is also a good general introduction and a reference for work done before 1995. In the synchronized deterministic top-down automata of Salomaa [75, 76] a limited communication between the computations in different branches is allowed. Gécseg and Steinby (2001) introduced syntactic monoids for DR tree languages, and these were used by Gécseg and Imreh (2002, 2004) for characterizing monotone, nilpotent and definite DR tree languages. In [59] W. Martens, F. Neven and Schwentick discuss several aspects of DR-recognition. In particular, motivated by applications to schema languages for XML, they study DR recognizers of unranked tree languages.

The book *Grammatical Picture Generation. A Tree-Based Approach* [16] by F. Drewes is a comprehensive treatment of *tree-based picture generation*. The picture generating systems considered consist, roughly speaking, of a device for producing a tree language and a picture algebra that interprets trees as pictures. The devices used for producing the tree languages include regular tree grammars, ET0L tree grammars, branching tree grammars, and tree transducers. The needed tree language theory is given in several inserts in the main text and in a separate appendix. Thus this fascinating book offers also a general introduction to tree languages.

A great number of concepts and results from several branches of mathematics are used in the theory of tree automata. However, as a conclusion of this appendix, I shall mention some introductions to just two subjects most intimately connected with tree automata: universal algebra and term rewriting. Besides the texts listed at the end of Chapter I of *Tree Automata*, there are several other good books on universal algebra. As general introductions, I recommend the classic [5] by S. Burris and H. P. Sankappanavar and the more recent textbook by C. Bergman [3]. The book [84] by W. Wechler, written expressly for computer scientists, is also very useful. The books [1] by J. Avenhaus and [2] by F. Baader and T. Nipkow offer two good introductions to term rewriting systems.

Bibliography of the Appendix

[1] Avenhaus, J. (1995): Reduktionssysteme. Springer-Verlag, Berlin 1995.

[2] Baader, F. and Nipkow, T. (1998): Term *Re*writing and *All That*. Cambridge University Press, Cambridge, UK 1998.

[3] Bergman, C. (2012): Universal Algebra. Fundamentals and Selected Topics. CRC Press, A Chapman & Hall Book, Boka Raton, Fl 2012.

[4] Bloem, J. and Engelfriet, J. (1997): Monadic second order logic and node relations on graphs and trees. – Structures in Logic and Computer Science (Eds. J. Mycielski, G. Rozenberg and A. Salomaa), Lecture Notes in Computer Science 1261, Springer-Verlag, Berlin 1997, 144-161.

[5] Burris, B. and Sankappanavar, H.P. (1981): A Course in Universal Algebra. Springer-Verlag, New York 1981.

[6] Bojanczyk, M. and Colcombet, T. (2006): Tree-walking automata cannot be determinized. Theoretical Computer Science 350 (2006), 164-173.

[7] Bojanczyk, M. and Colcombet, T. (2008): Tree-walking automata do not recognize all regular languages. SIAM Journal of Computing 38 (2008), 658-701.

[8] Bozapalidis, S., Fülöp, Z. and Rahonis, G. (2011): Equational tree transformations. Theoretical Computer Science 412 (2011), 3676-3692.

[9] Bozapalidis, S., Fülöp, Z. and Rahonis, G. (2012): Equational weighted tree transformations. Acta Informatica 49 (2012), 29-52.

[10] Brüggemann-klein, A., Murata, M. and Wood, D. (2001): Regular tree and regular hedge languages over unranked alphabets: Version 1, April 3, 2001. Technical Report HKUST-TCSC-2001-05, The Hongkong University of Technology 2001.

[11] Büchi, J. R (1989): Finite Automata, Their Algebras and Grammars. Towards a Theory of Formal Expressions (Ed. D. Siefkes), Springer-Verlag, New York 1989.

[12] Cano Gomez, A. and Steinby, M. (2011): Generalized contexts and n-ary syntactic semigroups of tree languages. Asian-European Journal of Mathematics 4 (2011), 49-79.

[13] Comon, H., Dauchet, M., Gilleron, R., Jacquemard, F., Lugiez, D., Löding, C., Tison, S. and Tommasi, M. (2008): Tree Automata Techniques and Applications. Available at `http://tata.gforge.inria.fr`.

[14] Courcelle, B. and Franchi-zannettacci, P. (1982): Attribute grammars and recursive program schemes I and II. Theoretical Computer Science 17 (1982), 163-191 and 235-257.

[15] Cristau, J., Löding, C. and Thomas, W. (2005): Deterministic automata on unranked trees. – Foundations of Computation Theory, FCT 2005 (Eds. M. Liśkiewicz and R. Reinschuk), Lecture Notes in Computer Science 3623, Springer-Verlag, Berlin 2005, 68-79.

[16] Drewes, F. (2006): Grammatical Picture Generation. A Tree-Based Approach, Springer-Verlag, Berlin 2006.

[17] Droste, M. and Heusel, D. (2015): The supports of weighted unranked tree automata. Fundamenta Informaticae 136 (2015), 37-58.

[18] Droste, M. and Vogler, H. (2006): Weighted tree automata and weighted logics. Theoretical Computer Science 366 (2006), 228-247.

[19] Droste, M. and Vogler, H. (2011): Weighted logics for unranked tree automata. Theory of Computing Systems 48 (2011), 23-47.

[20] Engelfriet, J., Lilin, E. and Maletti, A. (2009): Extended multi bottom-up tree transducers – Composition and decomposition. Acta Informatica 46 (2009), 561-590.

[21] Engelfriet, J. and Maneth, S. (1999): Macro tree transducers, attribute grammars, MSO definable tree translations. Information and Computation 154 (1999), 34-91.

[22] Engelfriet, J. and Maneth, S. (2003): Macro tree translations of linear size are MSO definable. SIAM Journal of Computing 32 (2003), 950-1006.

[23] Engelfriet, J. and Maneth, S. (2006): The equivalence problem for deterministic MSO tree transducers is decidable. Information Processing Letters 100 (2006), 206-212.

[24] Engelfriet, J., Maneth, S. and Seidl, H. (2009): Deciding equivalence of top-down XML transformations in polynomial time. Journal of Computer and Systems Science 75 (2009), 271-286.

[25] Engelfriet, J., Maneth, S. and Seidl, H. (2014): How to remove the look-ahead of top-down tree transducers. – Developments in Language Theory, DLT 2014 (Eds. A.M Shur and M.V. Volkov), Lecture Notes in Computer Science 8633, Springer International Publishing Switzerland 2014, 103-115.

[26] Engelfriet, J. and Vogler, H. (1985): Macro tree transducers. Journal of Computer and Systems Science 31 (1985), 71-146.

[27] Engelfriet, J. and Vogler, H. (1986): Pushdown machines for the macro tree transducer. Theoretical Computer Science 42 (1986), 251-368.

[28] Engelfriet, J. and Vogler, H. (1988): High level tree transducers and iterated pushdown tree transducers. Acta Informaticae 26 (1988), 131-192.

[29] Engelfriet, J. and Vogler, H. (1991): Modular tree transducers. Theoretical Computer Science 78 (1991), 267-304.

[30] Ésik, Z. and Iván, Sz. (2007): Aperiodicity in tree automata. – Algebraic Informatics CAI 2007 (Eds. S. Bozapalidis and G. Rahonis), Lecture Notes in Computer Science 4782, Springer-Verlag, Berlin 2007, 189-207.

[31] Ésik, Z. and Kuich, W. (2003): Formal tree series. Journal of Automata, Languages and Combinatorics 8(2) (2003), 219-285.

[32] Ésik, Z. and Weil, P. (2005): Algebraic recognizability of tree languages. Theoretical Computer Science 340 (2005), 291-321.

[33] Filiot, E., Maneth, S., Reynier, P.-A. and Talbot, J.-M. (2015): Decision problems of tree transducers. – Automata, Languages, and Programming (Proc. 42nd Intern. Coll. ICALP 2015, Kyoto, Japan, July 2015), Lecture Notes in Computer Science 9135, Springer-Verlag, Berlin 2015, 209-221.

[34] Friese, S., Seidl, H. and Maneth, S. (2011): Earliest normal form and minimization for bottom-up tree transducers. International Journal of Foundations of Computer Science 22 (2011), 1607-1623.

[35] Fülöp, Z. (1991): A complete description for a monoid of deterministic bottom-up tree transformation classes. Theoretical Computer Science 88 (1991), 253-268.

[36] Fülöp, Z. (1994): Undecidable properties of top-down tree transducers. Theoretical Computer Science 134 (1994), 311-328.

[37] Fülöp, Z. and Gyenizse, P. (1993): On injectivity of deterministic top-down tree transducers. Information Processing Letters 48 (1993), 183-188.

[38] Fülöp, Z. and Steinby, M. (2011): Varieties of recognizable tree series over fields. Theoretical Computer Science 412 (2011), 736-752.

[39] Fülöp, Z., Stüber, T. and Vogler, H (2012): A Büchi-like theorem for weighted tree automata over multioperator monoids. Theory of Computation Systems 50 (2012), 241-278.

[40] Fülöp, Z. and Vágvölgyi, S. (1987): Results on compositions of deterministic root-to-frontier tree transformations. Acta Cybernetica 8 (1987), 49-61.

[41] Fülöp, Z. and Vágvölgyi, S. (1989): Variants of top-down tree transducers with look-ahead. Mathematical Systems Theory 21 (1989), 125-145.

[42] Fülöp, Z. and Vágvölgyi, S. (1990): A complete rewriting system for a monoid of tree transformation classes. Information and Computation 86 (1990), 195-212.

[43] Fülöp, Z. and Vágvölgyi, S. (1991): A complete classification of deterministic root-to-frontier tree transformation classes. Theoretical Computer Science 81 (1991), 1-15.

[44] Fülöp, Z. and Vogler, H. (1998): Syntax-Directed Semantics. Formal Models Based on Tree Transducers, Springer-Verlag, Berlin 1998.

[45] Fülöp, Z. and Vogler, H. (2009): Weighted tree automata and tree transducers. – Handbook of Weighted Automata (Eds. M. Droste, W. Kuich and H. Vogler), Springer-Verlag, Berlin 2009, 313-403.

[46] Gécseg, F. and Steinby, M. (1997): Tree languages. – Handbook of Formal Languages, Vol. 3 (Eds. G. Rozenberg and A. Salomaa), Springer-Verlag, Berlin 1997, 1-68.

[47] Gilleron, R. and Tison, S. (1995): Regular tree languages and rewrite systems. Fundamenta Informaticae 24 (1995), 157-175.

[48] Godoy, G. and Giménez, O. (2013): The HOM problem is decidable. Journal of the ACM 60(4) (2013), Article 23.

[49] Graehl, J., Knight, K. and May, J. (2008): Training tree transducers. Computational Linguistics 34 (2008), 391-427.

[50] Grädel, E, Thomas, W. and Wilke, T. (Eds.) (2002): Automata, Logics, and Infinite Games, Springer-Verlag, Berlin 2002.

[51] Joshi, A. K. and Schabes, Y. (1997): Tree-adjoining grammars. – Handbook of Formal Languages, Vol. 3 (Eds. G. Rozenberg and A. Salomaa), Springer-Verlag, Berlin 1997, 69-123.

[52] Knight, K. (2007): Capturing practical natural language transformations, Machine Translation 21 (2007), 212-133.

[53] Knight, K. and Graehl, J. (2005): An overview of probabilistic tree transducers for natural language processing. – Computational Linguistics and Intelligent Text Processing (Proc. 6th International Conference, CICLing 2005, Mexico City, Mexico, February 2005), Lecture Notes in Computer Science 3406, Springer-Verlag, Berlin 2005, 1-24.

[54] Libkin, L. (2006): Logics for unranked trees: an overview. Logical Methods in Computer Science 2 (2006), 1-31.

[55] Löding, C. (2012): Basics on tree automata. – Modern Applications of Automata Theory (Eds. D. D'Souza and P. Shankar), World Scientific, Singapore 2012, 79-109.

[56] Maletti, A. (2011a): *Survey.* Weighted top-down tree transducers. Part I – Basics and expressive power. Acta Cybernetica 20 (2011), 223-250.

[57] Maletti, A. (2011b): Applications in machine translation of *Survey*: Weighted top-down tree transducers. Fundamenta Informaticae 112 (2011), 239-261.

[58] Maneth, S. (2014): Equivalence problems for tree transducers: a brief survey. – Automata and Formal Languages 2014, AFL 2014 (Eds. Z. Ésik and Z. Fülöp), EPTCS 151, 2014, 74-93.

[59] Martens, W., Neven, F. and Schwentick, T. (2008): Deterministic top-down automata: past, present and future. – Logic and Automata. Texts in Logic and Games, Vol. 2 (Eds. J. Flum, E. Grädel and T. Wilke), Amsterdam University Press, Amsterdam 2008, 515-541.

[60] Martens, W. and Niehren, J. (2007): On the minimization of XML Schemas and tree automata for unranked trees. Journal of Computer and System Sciences 73 (2007), 550-583.

[61] Mathissen, C. (2009): Weighted Automata and Weighted Logics over Tree-like Structures. Dissertation, Faculty of Mathematics and Informatics, University of Leipzig, Leipzig 2009.

[62] Morawietz, F. (2003): Two-Step Approach to Natural Language Formalisms. Studies in Generative Grammar 64, Mouton de Gruyter, Berlin 2003.

[63] Murata, M. (1995): Forest-regular and tree-regular languages. Technical Report, Fuji-Xerox, Japan 1995.

[64] Murata, M. (2000): Hedge automata: A formal model for XML schemata. Fuji-Xerox Information Systems, Japan 2000.

[65] Neven, F. (2002): Automata, logic, and XML. – Computer Science Logic (Proc. 16th Internat. Workshop, CSL 2002, Edinburgh, UK, 2002). Lecture Notes in Computer Science 2471, Springer-Verlag, Berlin 2002, 2-26.

[66] Nivat, M. and Podelski, A. (Eds.) (1992): Tree Automata and Languages, Studies in Computer Science and Artificial Intelligence 10, North-Holland, Amsterdam 1992.

[67] Otto, T. (1999): On the connections between rewriting and formal languages. – Rewriting Techniques and Applications, RTA-99 (Proc. Conf., Trento, Italy, 1999), Lecture Notes in Computer Science 1631, Springer-Verlag, Berlin 1999, 332-355.

[68] Petković, T. and Salehi, S. (2005): Positive varieties of tree languages. Theoretical Computer Science 347 (2005), 1-35.

[69] Piao, X. and Salomaa, K. (2011): Transformations between different models of unranked bottom-up tree automata. Fundamenta Informaticae 109 (2011), 405-424.

[70] Piao, X. and Salomaa, K. (2012): Lower bounds for the size of deterministic unranked tree automata. Theoretical Computer Science 454 (2012), 231-239.

[71] Rahonis, G. (2007): Weighted Muller tree automata and weighted logics. Journal of Automata, Languages and Combinatorics 12 (2007), 455-483.

[72] Salehi, S. (2005): Varieties of tree languages definable by syntactic monoids. Acta Cybernetica 17 (2005), 21-41.

[73] Salehi, S. and Steinby, M. (2007a): Varieties of many-sorted recognizable sets. PU.M.A. 18 (2007), 319-343. Also as: TUCS Technical Report No 626, Turku 2004.

[74] Salehi, S. and Steinby, M. (2007b): Tree algebras and varieties of tree languages. Theoretical Computer Science 377 (2007), 1-24.

[75] Salomaa, K. (1994): Synchronized tree automata. Theoretical Computer Science 127 (1994), 25-51.

[76] Salomaa, K. (1996): Decidability of equivalence for deterministic synchronized tree automata. Theoretical Computer Science 167 (1996), 171-192.

[77] Schwentick, T. (2007): Automata for XML – A survey. Journal of Computer and System Sciences 73 (2007), 289-315.

[78] Seidl, H., Maneth, S. and Kemper, G. (2015): Equivalence of deterministic top-down tree-to-string transducers is decidale. arXiv: 1503.09163v [cs.FL] 31Mar2015.

[79] Steinby, M. (2003): Tree automata in the theory of term rewriting. – Words, Languages and Combinatorics III (Proc. Intern. Conf., Kyoto, Japan, 2000) (Eds. M. Ito and T. Imaoka), World Scientific, New Jersey 2003, 434-449.

[80] Steinby, M. (2005): Algebraic classifications of regular tree languages. – Structural Theory of Automata, Semigroups and Universal Algebra (Eds. V.B. Kudryavtsev and I.G. Rosenberg), NATO Science Series, Mathematics, Physics and Chemistry, vol. 207 (2005), 381-432.

[81] Thomas, W. (1990): Automata on infinite objects. – Handbook of Theoretical Computer Science, Vol. B (Ed. J. van Leeuwen), Elsevier, Amsterdam 1990, 133-191.

[82] Thomas, W. (1997): Languages, automata, and logic. – Handbook of Formal Languages, Vol. 3 (Eds. G. Rozenberg and A. Salomaa), Springer-Verlag, Berlin 1997, 389-455.

[83] Vágvölgyi, S. (2013): Rewriting preserving recognizability of finite tree languages. The Journal of Logic and Algebraic Programming 82 (2013), 71-94.

[84] Wechler, W. (1992): Universal Algebra for Computer Scientists, Springer-Verlag, Berlin 1992.