North-Holland Publishing Company

NOTE

SYSTOLIC TREES AND SYSTOLIC LANGUAGE RECOGNITION BY TREE AUTOMATA

Magnus STEINBY
Department of Mathematics, University of Turku, SF-20500 Turku 50, Finland

Communicated by A. Salomaa
Received March 1982
Revised May 1982

Abstract. Recently K. Culik II, J. Gruska, A. Salomaa and D. Wood have studied the language recognition capabilities of certain types of systolically operating networks of processors. Here their model for systolic VLSI trees is formalized in terms of standard tree automaton theory, and we show how some known facts about recognizable forests and tree transductions can be applied in the VLSI tree theory.

1. Introduction

In a set of recent reports K. Culik II, J. Gruska, A. Salomaa and D. Wood [2, 3, 4] have considered some regularly organized systolically operating networks of processors in an attempt to estimate the language recognition power of VLSI chips. Here we shall concentrate on one of their models, the systolic tree automaton. The reader is referred to the above-mentioned writings for a more general treatment of the subject.

In [4] it was noted that their VLSI tree automata operate like deterministic frontier-to-root tree automata, but this fact is somewhat obscured by the formulation of the theory, and no use of traditional tree automaton theory is made. I shall try to make the connection more explicit by reformulating the VLSI tree theory so that it becomes fully compatible with tree automaton theory. The new formulation represents a slightly altered point of view, but it allows us to present formal proofs in the style of tree automaton theory and to apply this theory, while the families of languages to be studied remain unchanged.

In Section 2 the basic definitions relating to trees are presented. The underlying structure of a VLSI tree is conveniently represented by its domain.

Section 3 reviews some concepts and results from the theory of tree automata. The authors favorite variant of the tree recognizer concept is quite suitable here as it involves natural counterparts to the constituents of VLSI tree automata.

0304-3975/83/0000-0000/$03.00

In Section 4 a mode of operation is defined for tree recognizers which makes them act as systolic VLSI language recognizers. The families of languages to be studied are introduced. Also, we comment on the relationships to the VLSI tree theory as formulated by Culik et al.

In Section 5 we demonstrate the applicability of tree automaton theory by giving concise proofs for some basic results concerning the language families associated with VLSI trees. In Section 6 it is shown that languages recognized by balanced VLSI tree automata are target languages of root-to-frontier tree transducers. Hence the theory of tree transformations appears also to be relevant to the VLSI tree theory.

2. Tree domains, trees and VLSI trees

Let N^* be the free monoid generated by the set of positive integers which we denote by N. The operation is denoted by $\cdot$ and the identity element by 0. The *depth* $\mathrm{d}(p)$ of an element $p \in N^*$ is defined inductively:

(1) $\mathrm{d}(0) = 0$,

(2) $\mathrm{d}(p) = \mathrm{d}(q) + 1$ for $p = q \cdot i$ with $i \in N$.

A partial ordering $\leqslant$ on N^* is defined as follows: for any $p, q \in N^*$,

$$p \leqslant q \quad \text{iff} \quad p \cdot p' = q \text{ for some } p' \in N^*.$$

If $p = q \cdot i$ for some $i \in N$, then p is the ith *son* of q and q is the *father* of p.

Definition. A *tree domain* is a nonempty subset D of N^* satisfying the following three conditions:

D1. $p \leqslant q$ and $q \in D$ imply $p \in D$.

D2. $p \cdot j \in D$ and $i < j$ imply $p \cdot i \in D$ $(i, j \in N)$.

D3. There is a number $n \in N$ such that no element of D has more than n sons in D.

A tree domain may be finite or infinite. The elements of a tree domain are called its *nodes*. Nodes maximal with respect to $\leqslant$ are called *leaves*. The *height* $\mathrm{hg}(D)$ of a finite tree domain is $\max(\mathrm{d}(p): p \in D)$. For each p in D, we write

$$D_p = \{q \in N^*: p \cdot q \in D\}.$$

The *rank* $\mathrm{r}_D(p)$ of a node $p \in D$ is the number of its sons in D. The *width* $\mathrm{wd}_D(k)$ of a tree domain D *at level* $k \geqslant 0$ is defined as the number of nodes of D of depth k.

Example. The finite tree domain $D = \{0, 1, 2, 1.1, 1.2, 1.2.1\}$ is to be viewed as a representation of the tree structure shown in Fig. 1. The height of D is 3, its leaves are 1.1, 1.2.1 and 2, $\mathrm{r}_D(0) = \mathrm{r}_D(1) = 2$, $\mathrm{r}_D(1.2) = 1$, $\mathrm{r}_D(1.1) = \mathrm{r}_D(1.2.1) = \mathrm{r}_D(2) = 0$, $\mathrm{wd}_D(0) = 1$, $\mathrm{wd}_D(1) = \mathrm{wd}_D(2) = 2$, $\mathrm{wd}_D(3) = 1$, and $\mathrm{wd}_D(k) = 0$ for $k = 4, 5, \ldots$.

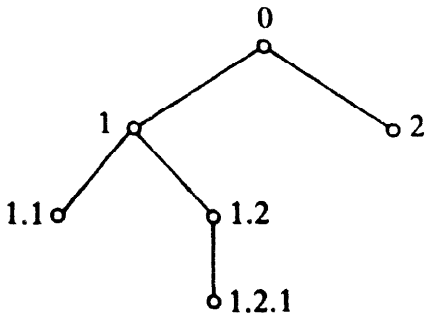

Fig. 1.

Definition. A tree domain D is called a *VLSI tree domain* if it satisfies the following two additional conditions:

D4. Every p in D has at least one son in D.

D5. There is a constant $c > 1$ such that $\mathrm{wd}_D(k) > c^k$ for all $k > 0$.

Condition D4 is implied by the systolic mode of operation of the VLSI tree automata. Condition D5 is the exponential growth condition. A VLSI tree domain is always infinite and it has no leaves.

A *ranked alphabet* is a finite nonempty set Σ of symbols together with a mapping $\mathrm{r}: \Sigma \to N_0$ which assigns a non-negative *rank* $\mathrm{r}(\sigma)$ to each symbol $\sigma \in \Sigma$. Usually we do not mention r, but express Σ as a union $\Sigma_0 \cup \Sigma_1 \cup \cdots$, where for each $m \geq 0$, $\Sigma_m = \{\sigma \in \Sigma : \mathrm{r}(\sigma) = m\}$. Note that $\Sigma_m \neq \emptyset$ for a finite number of m's only. In what follows, Σ will always denote a ranked alphabet.

Definition. A *Σ-tree* over a tree domain D is a mapping $f: D \to \Sigma$ such that $\mathrm{r}(f(p)) = \mathrm{r}_D(p)$ for every p in D. For each p in D, $f(p)$ is called the *label* of p. D is called the *domain* of f, and it is denoted by $\mathrm{dom}\, f$. The *subtree* of f at node $p \in D$ is defined as the Σ-tree $f_p: D_p \to \Sigma$, where $f_p(q) = f(p \cdot q)$ for every $q \in D_p$. Finally, a *VLSI Σ-tree* is a Σ-tree over a VLSI tree domain with a finite number of different subtrees.

Note that a Σ-tree over a given tree domain D exists only in case $\Sigma_m \neq \emptyset$ for every $m \geq 0$ such that D contains a node with exactly m sons. On the other hand, condition D3 implies that for every tree domain there exist ranked alphabets which can be used to define trees over this domain. For each tree domain D we distinguish a minimum ranked alphabet Γ^D such that for each $m \geq 0$,

$$\Gamma_m^D = \begin{cases} \{\gamma_m\} & \text{if } m = \mathrm{r}_D(p) \text{ for some } p \in D, \\ \emptyset & \text{otherwise.} \end{cases}$$

The concepts and notations defined above for tree domains are applied to trees in the natural way. For example, the height $\mathrm{hg}(t)$ of a finite tree t (which means, of course, that its domain is finite) is the height of its domain.

3. Tree recognizers, recognizable forests, and languages

We shall now define tree recognizers, and review some basic facts about recognizable forests and their relation to languages. For proofs and more information the reader may consult, for example, [5, 7, 12, 13].

Let X be an alphabet disjoint from the ranked alphabet Σ. Let Σ^X be the ranked alphabet we get by adding the letters of X to Σ as nullary symbols. Then a *ΣX-tree* is a finite Σ^X-tree as defined in the previous section. The set of ΣX-trees is denoted by $T_\Sigma(X)$. Subsets of $T_\Sigma(X)$ are called *ΣX-forests*.

A ΣX-tree differs from a finite Σ-tree only in that its leaves may also be labelled by letters from X. It is often more convenient to define ΣX-trees as Σ-terms in variables X. Then $T_\Sigma(X)$ is the smallest set such that

(1) $X \cup \Sigma_0 \subseteq T_\Sigma(X)$, and

(2) $\sigma(t_1, \ldots, t_m) \in T_\Sigma(X)$ whenever $m > 0$, $\sigma \in \Sigma_m$ and $t_1, \ldots, t_m \in T_\Sigma(X)$.

There is a simple inductive translation back to the original definition of $T_\Sigma(X)$, so we may use the two definitions in parallel.

Example. Suppose $\gamma \in \Sigma_0$, $\tau \in \Sigma_1$, $\sigma \in \Sigma_2$ and $x, y \in X$. The ΣX-tree shown in Fig. 2 may also be written as the term $\sigma(\sigma(x, \tau(\gamma)), y)$.

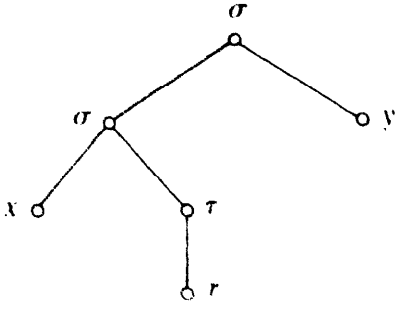

Fig. 2.

A *Σ-algebra* $\mathcal{A}$ consists of a nonempty set A of elements in which an operation $\sigma^{\mathcal{A}}$ is defined for each $\sigma \in \Sigma$; if $\sigma \in \Sigma_m$, then $\sigma^{\mathcal{A}}$ is an m-ary operation $A^m \to A$. We write $\mathcal{A} = (A, \Sigma)$, and call $\mathcal{A}$ *finite* if A is finite.

Let $\mathcal{A} = (A, \Sigma)$ be a Σ-algebra. Any mapping $\alpha : X \to A$ can be extended to a mapping

$$\hat{\alpha} : T_\Sigma(X) \to A$$

as follows:

(1) for $x \in X$, let $x\hat{\alpha} = x\alpha$; for $\sigma \in \Sigma_0$, let $\sigma\hat{\alpha} = \sigma^{\mathcal{A}}$,

(2) if $t = \sigma(t_1, \ldots, t_m)$, then $t\hat{\alpha} = \sigma^{\mathcal{A}}(t_1\hat{\alpha}, \ldots, t_m\hat{\alpha})$.

Definition. A *ΣX-recognizer* $\mathbf{A}$ is a system $(\mathcal{A}, \alpha, A_F)$, where

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

(2) $\alpha: X \to A$ is the *initial assignment*, and

(3) $A_F \subseteq A$ is the set of *final states*.

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

$$T(\mathbf{A}) = \{t \in T_\Sigma(X): t\hat{\alpha} \in A_F\}.$$

A ΣX-forest T is said to be *recognizable* if $T = T(\mathbf{A})$ for some ΣX-recognizer **A**. The set of all recognizable ΣX-forests is denoted by $\mathrm{Rec}(\Sigma, X)$.

The value $t\hat{\alpha}$ of the term t when evaluated in the algebra $\mathcal{A}$ for the assignment α of values to the variables is interpreted as the state of the recognizer **A** when it arrives at the root of the tree t. The tree t is accepted iff this state is a final state.

A *nondeterministic ΣX-recognizer* may have several choices for starting states and next states in any given situation. It accepts a tree if there is a set of mutually compatible choices such that the recognizer arrives at the root in a final state. We omit the formal definition, but note the following fact.

Theorem 3.1. *For each nondeterministic ΣX-recognizer one may construct an equivalent ΣX-recognizer.*

The proof is by the usual subset construction. Also the following theorem can be established by standard constructions.

Theorem 3.2. $\mathrm{Rec}(\Sigma, X)$ *is closed under all Boolean operations.*

Let Ω be a ranked alphabet and let Y be an alphabet. Suppose we are given a mapping $h_X: X \to Y$, and for each $m \geq 0$ such that $\Sigma_m \neq \emptyset$, a mapping $h_m: \Sigma_m \to \Omega_m$. (Hence $\Omega_m \neq \emptyset$ whenever $\Sigma_m \neq \emptyset$.) The *alphabetic tree homomorphism* determined by these mappings, is the mapping

$$h: T_\Sigma(X) \to T_\Omega(Y)$$

defined as follows. For each ΣX-tree t, $h(t)$ is the ΩY-tree over the same tree domain as t defined as follows:

(1) if $t = x (\in X)$, then $h(t) = h_X(x)$; if $t = \sigma (\in \Sigma_0)$, then $h(t) = h_0(\sigma)$,

(2) if $t = \sigma(t_1, \ldots, t_m)$ $(m > 0)$, then $h(t) = h_m(\sigma)(h(t_1), \ldots, h(t_m))$.

Hence an alphabetic tree homomorphism just relabels the nodes of a tree without changing its 'shape'. The two statements of the following theorem hold even for more general tree homomorphisms.

Theorem 3.3. *If $h: T_\Sigma(X) \to T_\Omega(Y)$ is an alphabetic tree homomorphism, then*

(a) $T \in \mathrm{Rec}(\Sigma, X)$ *implies* $h(T) \in \mathrm{Rec}(\Omega, Y)$, *and*

(b) $T \in \mathrm{Rec}(\Omega, Y)$ *implies* $h^{-1}(T) \in \mathrm{Rec}(\Sigma, X)$.

The set of all words over an alphabet X is denoted by X^*. The empty word is denoted by e. A language over X will also be called an X-language.

The *yield* $\mathrm{yd}(t)$ of a ΣX-tree t is defined inductively thus:

(1) for $x \in X$, $\mathrm{yd}(x) = x$; if $\sigma \in \Sigma_0$, then $\mathrm{yd}(\sigma) = e$,

(2) if $t = \sigma(t_1, \ldots, t_m)$, then $\mathrm{yd}(t) = \mathrm{yd}(t_1) \cdots \mathrm{yd}(t_m)$.

Hence we have a mapping $\mathrm{yd}: T_\Sigma(X) \to X^*$ which assigns to each ΣX-tree the word spelled out by the labels of its leaves when read in order from left to right ignoring leaves labelled by nullary symbols. The yd-mapping is extended to forests in the natural way: the *yield* of a ΣX-forest T is the X-language $\mathrm{yd}(T) = \{\mathrm{yd}(t): t \in T\}$.

Definition. The *language recognized* by a ΣX-recognizer $\mathbf{A}$ is the X-language $L(\mathbf{A}) = \mathrm{yd}(T(\mathbf{A}))$.

Hence a word over X is accepted by $\mathbf{A}$ iff it is the yield of at least one tree accepted by $\mathbf{A}$. The following theorem lists some of the basic facts about tree recognizers and languages.

Theorem 3.4. (a) *Every language recognized by a tree recognizer is context-free, i.e.* $T \in \mathrm{Rec}(\Sigma, X)$ *implies that* $\mathrm{yd}(T)$ *is a context-free* X*-language.*

(b) *If* $\Sigma_0 \neq \emptyset$ *and* $\Sigma_m \neq \emptyset$ *for at least one* $m > 1$, *then every context-free* X*-language is recognized by a* ΣX*-recognizer.*

(c) *If* L *is a regular* X*-language, then* $\mathrm{yd}^{-1}(L) \in \mathrm{Rec}(\Sigma, X)$.

4. Systolic language recognition

From here on, the ranked alphabets are assumed to include exactly one nullary symbol which we take to be §. A modified yield operation

$$\mathrm{yd}_\S: T_\Sigma(X) \to (X \cup \{\S\})^*$$

is introduced. It differs from the original one only in that $\mathrm{yd}_\S(\S) = \S$.

Definition. Let $f: D \to \Sigma$ be a VLSI Σ-tree and let X be an alphabet. The *systolic* ΣX*-forest* $S_f(X)$ of f over X consists of the one-node trees § and x ($x \in X$) and all ΣX-trees t of height $k > 0$ which satisfy the following conditions:

S1. $\mathrm{dom}\, t \subseteq D$.

S2. $d(p) = k$ for every maximal node p of $\mathrm{dom}\, t$.

S3. $\mathrm{yd}_\S(t) = w\S^i$, where $w \in X^*$ and $\mathrm{wd}_f(k-1) < |w| \leq \mathrm{wd}_f(k)$.

S4. For all nonmaximal nodes p of $\mathrm{dom}\, t$, $t(p) = f(p)$.

It is obvious that for each word w in X^* there is at most one t in $S_f(X)$ such that $\mathrm{yd}(t) = w$. On the other hand, the exponential growth condition D5 satisfied by f assures that such a tree exists for every word. Therefore we may state

Lemma 4.1. *Let* f *be a VLSI* Σ*-tree and* X *an alphabet. Then* $\mathrm{yd}(S_f(X)) = X^*$ *and for every* $w \in X^*$ *there is a unique* f^w *in* $S_f(X)$ *such that* $\mathrm{yd}(f^w) = w$.

Definition. Let f be a VLSI Σ-tree, X an alphabet and $\mathbf{A}=(\mathcal{A}, \alpha, A_F)$ a ΣX-recognizer. The *language f-recognized* by $\mathbf{A}$ is the X-language

$$L_f(\mathbf{A})=\{w \in X^*: f^w \hat{a} \in A_F\}.$$

An X-language is said to be *f-recognizable* if it is f-recognized by some ΣX-recognizer, and it is said to be *systolically recognizable* if it is f-recognizable for some VLSI tree f. If D is a VLSI tree domain, then a language is said to be *D-recognizable* if it is *f-recognizable* for some VLSI tree f such that $\operatorname{dom} f = D$.

Hence a word w is in $L_f(\mathbf{A})$ iff $\mathbf{A}$ accepts the systolic tree f^w. Let $\mathscr{L}_f$ denote the family of all f-recognizable languages. Similarly, $\mathscr{L}_D$ denotes the family of D-recognizable languages for a given VLSI tree domain D. For any alphabet X, let $\mathscr{L}_f(X)$ and $\mathscr{L}_D(X)$ denote the set of all X-languages belonging to $\mathscr{L}_f$ and $\mathscr{L}_D$, respectively.

It should be fairly obvious that the definitions stated above provide us with a formalism and terminology for discussing the families of languages introduced by Culik et al. Our ranked alphabet Σ is their 'processor alphabet', our X is their 'terminal alphabet', the state set A corresponds to their 'operating alphabet', the VLSI tree domain dom f accounts for their 'underlying structure' of a VLSI acceptor etc. However, some details deserve special comments.

In our formulation languages are recognized by ordinary tree automata in the usual manner except that the input trees are restricted to the systolic forest. In [4] it is thought that the input words are fed into tree shaped networks of processors. The latter point of view may, however, be maintained here too simply by assuming that the tree recognizers are realized as iterative tree networks (cf. [10]).

We have also altered slightly the manner in which input symbols are encoded into internal states. In [4] the processors themselves act as encoders, which requires that they are equipped with input terminals for receiving external inputs. It also means that the input alphabets compatible with a given VLSI chip are fixed in advance. By using the initial assignment α, which stands for separate input-to-state encoders, we make external and internal alphabets independent of each other. Moreover, this approach fits better into our definition of tree recognizers. The condition that a VLSI tree has just a finite number of different subtrees assures that this difference in the input process will not change the families of languages to be studied.

5. Some basic properties of the families $\mathscr{L}_f$ and $\mathscr{L}_D$

Throughout this section Σ is a ranked alphabet with the single nullary symbol s, f is VLSI Σ-tree, and X is an alphabet. Having fixed f and X, we denote $S_f(X)$ simply by S.

The following lemma is immediately clear from the definition of f-recognition.

Lemma 5.1. *For any X-language L, $L \in \mathscr{L}_f(X)$ iff there exists a recognizable ΣX-forest R such that $L = \mathrm{yd}(R \cap S)$.*

One immediate consequence of this lemma and Theorem 3.1 is that the families $\mathscr{L}_f$ would not change if we allowed nondeterministic tree recognizers. For every ΣX-forest R and any word $w \in X^*$, $w \in \mathrm{yd}(R \cap S)$ iff $f^w \in R$. Parts (b) and (c) of the next lemma follow from this observation; part (a) is of course valid for all forests.

Lemma 5.2. *For any ΣX-forests R_1 and R_2,*

(a) $\mathrm{yd}((R_1 \cup R_2) \cap S) = \mathrm{yd}(R_1 \cap S) \cup \mathrm{yd}(R_2 \cap S)$,

(b) $\mathrm{yd}((R_1 \cap R_2) \cap S) = \mathrm{yd}(R_1 \cap S) \cap \mathrm{yd}(R_2 \cap S)$, *and*

(c) $\mathrm{yd}((R_1 - R_2) \cap S) = \mathrm{yd}(R_1 \cap S) - \mathrm{yd}(R_2 \cap S)$.

By combining these two lemmas with the obvious fact that $X^* \in \mathscr{L}_f(X)$, we get the following result. It differs from Theorem 3 of [4] in that here the VLSI tree, and not just the domain, is fixed.

Theorem 5.3. *$\mathscr{L}_f(X)$ is closed under all Boolean operations.*

The next theorem is the counterpart to Theorems 1 and 2 of [4].

Theorem 5.4. *$\mathscr{L}_f$ includes the family of all regular languages.*

Proof. For any regular language $L \subseteq X^*$, $\mathrm{yd}^{-1}(L) \in \mathrm{Rec}(\Sigma, X)$. By Lemma 5.1 this implies $L \in \mathscr{L}_f(X)$ as

$$L = \mathrm{yd}(\mathrm{yd}^{-1}(L) \cap S)$$

by Lemma 4.1. □

Let $\mathrm{dom}\, f = D$. We denote the minimum alphabet $\Gamma^{(D)}$ by Γ and the unique VLSI Γ-tree over D by f. The following theorem includes the 'normal form theorem' (i.e., Theorem 5) of [4].

Theorem 5.5. $\mathscr{T}_f = \mathscr{L}_f$.

Proof. We may assume that every symbol from Σ appears in t. Then there is a unique alphabetic tree homomorphism

$$h : T_\Sigma(X) \to T_\Gamma(X)$$

such that h_X is the identity mapping of X. Obviously, $\mathrm{yd}(h(t)) = \mathrm{yd}(t)$ for every $t \in T_\Sigma(X)$. (Two yield-mappings are involved, but our notation does not distinguish between them.) Moreover, $h(f^w) = f^w$ for every $w \in X^*$. These facts imply that for

every ΣX-forest R,

$$\mathrm{yd}(R \cap S) = \mathrm{yd}(h(R) \cap S_f(X)), \tag{1}$$

and for every ΓX-forest T,

$$\mathrm{yd}(h^{-1}(T) \cap S) = \mathrm{yd}(T \cap S_f(X)). \tag{2}$$

Together with Theorem 3.3 and Lemma 4.1, (1) and (2) imply that for any X-language L, $L \in \mathscr{L}_f(X)$ iff $L \in \mathscr{L}_f(X)$. □

Corollary 5.6. *If f and g are* VLSI *trees such that* dom f = dom g, *then* $\mathscr{L}_f = \mathscr{L}_g$.

Corollary 5.7. *For any VLSI tree f, $\mathscr{L}_f = \mathscr{L}_{\mathrm{dom}\, f}$.*

By using Theorem 5.3 we also get the following corollary which appears as Theorem 3 in [4].

Corollary 5.8. *For any VLSI tree domain D and any alphabet X, $\mathscr{L}_D(X)$ is closed under all Boolean operations.*

6. Balanced VLSI trees and tree transducers

To focus the attention on balanced VLSI trees not only simplifies matters, but may also be justified from a practical point of view. One of the simplifying factors is that for a balanced VLSI tree it is easy to control the condition S3 concerning the length of the yield of a systolic tree.

For each $m \geq 2$ there is a unique VLSI tree domain mD in which every node has rank m. Assuming that a fixed m is given, let Ω be a ranked alphabet consisting of a single m-ary symbol ω and the 0-ary symbol §. The unique VLSI Ω-tree

$$mf : mD \to \Omega$$

is called the *m-ary VLSI tree*. Jointly the m-ary VLSI trees ($m \geq 2$) are called *balanced VLSI trees*.

A *tree transformation* is a relation $\tau \subseteq T_\Sigma(X) \times T_\Omega(Y)$, where Σ and Ω are ranked alphabets (not restricted by the assumption made above), X and Y are alphabets. Tree transformations may be defined by *tree transducers*. One of the main types of these is the *root-to-frontier tree transducer*, or the *R-transducer* for short, which transforms input trees into output trees working from the root toward the leaves. The formal definition is given for a special case, which is the only type to be considered here.

Definition. A *1-unary R-transducer* is a system $\mathfrak{B} = (\Sigma, X, B, \Omega, Y, B_0, P)$ where Σ is a ranked alphabet consisting of a single unary symbol σ, X and Y are alphabets,

B is a finite set of *states* which are viewed as unary symbols, Ω is a ranked alphabet, $B_0(\subseteq B)$ is the set of *initial states*, and P is a finite set of *productions* which are of type (1) or of type (2):

(1) $bx \to r$, where $b \in B$, $x \in X$ and $r \in T_\Omega(Y)$:

(2) $b\sigma \to r$, where $b \in B$ and $r \in T_\Omega(Y \cup B(\xi))$. Here ξ is a new symbol and $B(\xi) = \{b(\xi): b \in B\}$.

Furthermore, it is assumed that B is disjoint from the other alphabets involved.

Note that now

$$T_\Sigma(X) = \{\sigma^n(x): n \geq 0, x \in X\},$$

where

$$\sigma^0(x) = x \quad \text{and} \quad \sigma^{i+1}(x) = \sigma(\sigma^i(x)) \quad \text{for} \quad i \geq 0.$$

In order to describe the operation of such a $\mathfrak{B}$ we shall first define a set U of trees which represent possible initial, intermediate and final stages in a transformation process. Let $B(T_\Sigma(X))$ be the set of all trees $b(s)$ where $b \in B$ and $s \in T_\Sigma(X)$. The presence of such a tree $b(s)$ means that the transducers remaining tasks include the transforming of s, which is a subtree of the original input tree, starting in state b. Now U is defined as the smallest set such that

(1) $Y \cup \Omega_0 \cup B(T_\Sigma(X)) \subseteq U$, and

(2) $\omega(t_1, \ldots, t_m) \in U$ whenever $m \geq 1$, $\omega \in \Omega_m$ and $t_1, \ldots, t_m \in U$.

Note that $B(T_\Sigma(X)) \subset U$ and $T_\Omega(Y) \subset U$; each successful transformation begins with a tree $b(s) \in B(T_\Sigma(X))$, where $b \in B_0$, and ends in a tree t belonging to $T_\Omega(Y)$.

The stepwise transformation process is modelled by the relation $\Rightarrow (\subseteq U \times U)$ defined as follows. For any $s_1, s_2 \in U$, $s_1 \Rightarrow s_2$ iff there are strings s' and s'' such that either

(i) $s_1 = s'b(x)s''$ and $s_2 = s'rs''$ for some production $bx \to r$ of type (1) belonging to P, or

(ii) $s_1 = s'b(\sigma^n(x))s''$ and $s_2 = s'r(\xi \leftarrow \sigma^{n-1}(x))s''$, where $n \geq 1$, $\sigma^n(x) \in T_\Sigma(X)$, $b\sigma \to r$ is a production of type (2) in P, and $r(\xi \leftarrow \sigma^{n-1}(x))$ is the tree in U which is obtained when every occurrence of ξ in r is replaced by $\sigma^{n-1}(x)$.

Let $\Rightarrow^*$ be the reflexive, transitive closure of $\Rightarrow$. Then the *1-unary R-transformation defined by* $\mathfrak{B}$ is

$$\tau_{\mathfrak{B}} = \{(t, t'): t \in T_\Sigma(X), t' \in T_\Omega(Y), b(t) \Rightarrow^* t' \text{ for some } b \in B_0\}.$$

If τ is such a 1-unary tree transformation and T is a recognizable ΣX-forest, then

$$T\tau = \{t': (t, t') \in \tau \text{ for some } t \in T\}$$

is called a *1-unary R-surface forest*, and the language $\mathrm{yd}(T\tau)$ is a *1-unary R-target language*.

Theorem 6.1. *For every $m \geq 0$, all mf-recognizable languages are 1-unary R-target languages.*

Proof. Let $L \in \mathscr{L}_{mf}(X)$ be a language *mf*-recognized by an ΩX-recognizer $\mathbf{A} = (\mathscr{A}, \alpha, A_F)$. ($\Omega$ consists again of the single m-ary symbol ω and the 0-ary symbol §.) Let $\Sigma = \Sigma_1 = \{\sigma\}$ and let $Z = \{z\}$. We define a 1-unary R-transducer $\mathfrak{B} = (\Sigma, Z, B, \Omega, X, B_0, P)$ as follows:

(a) $B = A \times \{0, 1, 2, 3\}$ and $B_0 = A_F \times \{0\}$,

(b) P consists of the following four lists of productions:

(0) All productions $(x\alpha, 0)z \to x$ such that $x \in X$ and $x\alpha \in A_F$, the production $(\S^{\mathscr{A}}, 0)z \to \S$ in case $\S^{\mathscr{A}} \in A_F$, and all productions

$$(a, 0)\sigma \to \omega((a_1, 1)(\xi), \ldots, (a_i, 1)(\xi), (a_{i+1}, 2)(\xi), (a_{i+2}, 3)(\xi), \ldots, (a_m, 3)(\xi))$$

such that $\sigma^{\mathscr{A}}(a_1, \ldots, a_m) = a$, $a \in A_F$, and $1 \leq i \leq m$.

(1) All productions $(x\alpha, 1)z \to x$ with $x \in X$, and all productions

$$(a, 1)\sigma \to \omega((a_1, 1)(\xi), \ldots, (a_m, 1)(\xi))$$

such that $\sigma^{\mathscr{A}}(a_1, \ldots, a_m) = a$.

(2) All productions $(x\alpha, 2)z \to x$ with $x \in X$, and all productions

$$(a, 2)\sigma \to \omega((a_1, 1)(\xi), \ldots, (a_i, 1)(\xi), (a_{i+1}, 2)(\xi), (a_{i+2}, 3)(\xi), \ldots, (a_m, 3)(\xi))$$

such that $\sigma^{\mathscr{A}}(a_1, \ldots, a_m) = a$ and $0 \leq i < m$.

(3) The production $(\S^{\mathscr{A}}, 3)z \to \S$ and all productions

$$(a, 3)\sigma \to \omega((a_1, 3)(\xi), \ldots, (a_m, 3)(\xi))$$

such that $\sigma^{\mathscr{A}}(a_1, \ldots, a_m) = a$.

for any $(a, i) \in B$ and $n \geq 0$, let

$$\tau_{(a,i)}(n) = \{t \in T_\Omega(X) : (a, i)(\sigma^n(z)) \Rightarrow^* t\}.$$

Moreover, for every $n \geq 0$, let E_n denote the set of all ΩX-trees t such that $d(p) = n$ for every leaf p of dom t.

By induction on n, one may easily verify that for all $n \geq 0$ and $a \in A$, the following claims (C1)–(C3) are valid:

(C1) $\tau_{(a,1)}(n) = \{t \in E_n : t\hat{\alpha} = a, \mathrm{yd}_\S(t) \in X^*\}$.

(C2) $\tau_{(a,2)}(n) = \{t \in E_n : t\hat{\alpha} = a, \mathrm{yd}_\S(t) = w\S^i \text{ with } w \in X^* \text{ and } i = m^n - |w|\}$.

(C3) $\tau_{(a,3)}(n) = \{t \in E_n : t\hat{\alpha} = a, \mathrm{yd}_\S(t) = \S^{m^n}\}$.

Any derivation of an output tree t from an input tree $\sigma^n(z)$ begins with the application of one of the productions in list (0). If $n = 0$, then the possible t's are the letters x such that $x\alpha \in A_F$, i.e. $x \in T(\mathbf{A})$, and § in case $§ \in T(\mathbf{A})$. If $n > 0$, then the first production used creates the root of the output tree and m copies of $\sigma^{n-1}(z)$. From (C1)–(C3) we see that the choice of this first production determines in which of the m main subtrees of t the rightmost letter from X will appear; it appears in the subtree which is derived from the copy of $\sigma^{n-1}(z)$ starting in state $(a_{i+1}, 2)$. The location of the division line within the subtree is determined by the subsequent choices of productions from list (2). Because the leftmost $\sigma^{n-1}(z)$ is transformed starting in a state $(a_1, 1)$ and the next one in state $(a_2, 1)$ or $(a_2, 2)$, the output tree will satisfy condition S3. In fact, it should now be obvious that if we ignore the restrictions expressed in terms of the first components of the states of $\mathfrak{B}$, all systolic ΩX-trees of height n can be derived from $\sigma^n(z)$, and that these conditions restrict the output trees to those accepted by $\mathbf{A}$. Hence

$$\tau_{\mathfrak{B}} = \{(\sigma^n(z), t);\ n \geq 0,\ t \in T(\mathbf{A}) \cap S_{mf}(X),\ hg(t) = n\}.$$

This implies that

$$T(\mathbf{A}) \cap S_{mf} = T_{\Sigma}(Z)\tau_{\mathfrak{B}},$$

which means that L is a 1-unary target language. □

We shall now note a few facts which can be derived by Theorem 6.1 from the theory of tree transformations. For the first three corollaries the special nature of our R-transformations is not needed. The first one follows from a result due to Baker [1].

Corollary 6.2. *For any $m \geq 2$, $\mathcal{S}_{mf}$ is a proper subfamily of the family of deterministic context-sensitive languages.*

The next two corollaries follow by results due to Rounds [8]. Corollary 6.3 was proved (for the binary case) in [3] in a direct way.

Corollary 6.3. *The emptiness problem is solvable for mf-recognizable languages ($m \geq 2$).*

Corollary 6.4. *The finiteness problem is solvable for mf-recognizable languages ($m \geq 2$).*

Engelfriet [6] has shown that the 1-unary target languages are exactly the EOL languages (cf. [9], for example). Hence the following result of [3] (proved there for $m = 2$) is also a corollary of Theorem 6.1. The properness of the inclusion follows, for example, from the fact that the EOL language $\{x^n y^n;\ n \geq 0\}$ is not mf-recognizable for any m.

Corollary 6.5. *For any $m \geq 2$, $\mathscr{L}_{mt}$ is a proper subfamily of the family of EOL languages.*

7. Concluding remarks

We have seen that the theory of VLSI systolic tree acceptors can be formalized in terms of conventional tree automaton theory, and that a number of basic results follows immediately from this formalization. Although the systolic mode of operation adds a distinctive flavour to the VLSI tree theory, one might still view it as a new branch of the theory of tree automata, where it raises some new questions. In particular, it seems to motivate a closer study of some special tree transformations.

Our formulation of the theory is intended to reflect the intentions of the originators of the theory (with the exception of the way inputs are handled). However, the model itself seems still to be open to some discussion. For example, some rather unnatural cases could be excluded by requiring that all subtrees of a VLSI tree are also VLSI trees. But such changes would not call for any changes in our formalism, and the balanced VLSI trees would probably remain the most natural objects of study under any reasonable definition. One might also argue that there is no reason to associate the theory of systolically operating networks too exclusively with the forthcoming VLSI technology, but that it should rather be seen as a new outgrowth of the general theory of iterative networks.

Acknowledgment

I would like to thank Professor Arto Salomaa for his remarks and for encouraging me to write this note.

References

[1] B. Baker, Generalized syntax directed translation, tree transducers, and linear space, *SIAM J. Comput.* **7** (1978) 376–391.

[2] K. Culik II, J. Gruska and A. Salomaa, On a family of L languages resulting from systolic tree automata, Research Report CS-81-36, Department of Computer Science, University of Waterloo, Waterloo, Ontario (1981).

[3] K. Culik II, J. Gruska and A. Salomaa, Systolic automata for VLSI on balanced trees, Research Report CS-82-01, Department of Computer Science, University of Waterloo, Waterloo, Ontario (1982).

[4] K. Culik II, A. Salomaa and D. Wood, VLSI systolic trees as acceptors, Research Report CS-81-32, Department of Computer Science, University of Waterloo, Waterloo, Ontario (1981).

[5] J. Engelfriet, Tree automata and tree grammars, DAIMI Report FN-10, University of Aarhus, Aarhus, Denmark (1975).

[6] J. Engelfriet, Surface tree languages and parallel derivation trees, *Theoret. Comput. Sci.* **2** (1976) 9–27.

[7] F. Gécseg and M. Steinby, A faautomatak algebrai elmélete I-II, *Matem. Lapok* **26** (1978) 169-207 and **27** (1979) 283-336.
[8] W.C. Rounds, Mappings and grammars on trees, *Math. Systems Theory* **4** (1970) 257-287.
[9] G. Rozenberg and A. Salomaa, *The Mathematical Theory of L Systems* (Academic Press, New York, 1980).
[10] M. Steinby, On the structure and realizations of tree automata, *Proc. 2éme Colloquium sur les Arbres en Algèbre et en Programmation*, Université de Lille 1, Lille (1977) 235-248.
[11] J.W. Thatcher, Generalized^2 sequential machine maps, *J. Comput. System Sci.* **4** (1970) 339-367.
[12] J.W. Thatcher, Tree automata: an informal survey, in: A.V. Aho, Ed., *Currents in the Theory of Computation* (Prentice-Hall, Englewood Cliffs, NJ, 1973) 143-172.
[13] J.W. Thatcher and J.B. Wright, Generalized finite automata theory with an application to a decision problem of second order logic, *Math. Systems Theory* **2** (1968) 57-81.