Acta Cybernetica **18** (2007) 117–134.

On monotone languages and their characterization by regular expressions

György Gyurica*

To the memory of Balázs Imreh

Abstract

In one of their papers, F. Gécseg and B. Imreh gave a characterization for monotone string languages by regular expressions. It has turned out that the monotone string languages are exactly those languages that can be represented by finite unions of seminormal chain languages. In this paper a similar characterization is given for monotone DR-languages.

1 Introduction

Monotone string and tree languages were introduced by Gécseg and Imreh in [4] where these languages were characterized by means of syntactic monoids. They also used chain languages to represent monotone string languages by regular expressions, and showed that any monotone string language can be represented as the union of finitely many seminormal chain languages and that, conversely, any seminormal chain language can be recognized by a monotone recognizer.

In this paper we continue the investigation of monotone string and DR-languages. Our primary goal was to characterize the monotone DR-languages by regular ΣX-expressions, but we have also introduced the concept of iterational height for regular expressions which was useful to state conditions under which iteration preserves monotonicity. The same result was adapted to DR-languages, too.

Thereafter, a simple characterization of monotone DR-languages was given. The number of the auxiliary variables used in this representation and some decomposition problems were also investigated. Later, we stated some conditions that are required to preserve monotonicity when using the operations of x-product and x-iteration. Finally, we introduced the concept of generalized R-chain languages, for which it will turn out that they represent exactly the monotone DR-languages. For notions and notation not defined in this paper we refer the reader to [4] and [7].

*Department of Informatics, University of Szeged, Árpád tér 2, H-6720 Szeged, Hungary. E-mail: gyurica@inf.u-szeged.hu

2 Monotone string languages

Let X be an alphabet. The set of all words over X is denoted by X^*. Let us denote the *length* of a word $u \in X^*$ by $|u|$ which is the number of occurrences of letters from X in u. The empty word is denoted by e. The set of words with length greater than 0 is denoted by $X^+(= X^* \setminus \{e\})$.

An X*-recognizer* is a system $\mathbf{A} = (A, X, \delta, a_0, A')$, where A is a finite set of *states*, X is the *input alphabet*, $\delta : A \times X \to A$ is the *next-state function*, $a_0 \in A$ is the *initial state*, and $A' \subseteq A$ is the set of *final states*. The next-state function can be extended to a function $\delta^* : A \times X^* \to A$, where $\delta^*(a, e) = a$ and $\delta^*(a, xu) = \delta^*(\delta(a, x), u)$ $(a \in A,\ x \in X,\ u \in X^*)$. If there is no danger of confusion, instead of $\delta^*(a, u)$ we can use the notation $\delta(a, u)$ or simply au.

The *language* $L(\mathbf{A})$ *recognized by* $\mathbf{A}$ is given by

$$L(\mathbf{A}) = \{u \in X^* \mid a_0u \in A'\}.$$

A language $L \subseteq X^*$ is called *regular* or *recognizable* if it can be recognized by an X-recognizer.

An X-recognizer $\mathbf{A} = (A, X, \delta, a_0, A')$ is *monotone* if there is a partial ordering $\leq$ on A such that for all $a \in A$ and $x \in X$, $a \leq \delta(a, x)$ holds. It is obvious that for all $a \in A$ and $u \in X^*$, $a \leq au$ holds, too. A language $L \subseteq X^*$ is *monotone* if $L = L(\mathbf{A})$ for a monotone X-recognizer $\mathbf{A}$. Later we will use the fact that every partial ordering on a finite set can be extended to a linear ordering. For more details we refer the reader to [4].

A language $L \subseteq X^*$ is *fundamental*, if $L = Y^*$ for a $Y \subseteq X$. A language $L \subseteq X^*$ is a *chain language* if L can be given in the form $L = L_0x_1L_1x_2 \dots x_{k-1}L_{k-1}x_kL_k$, where $x_1, \dots, x_k \in X$ and every L_i $(0 \leq i \leq k)$ is a product of fundamental languages. A chain language $L = L_0x_1L_1x_2 \dots \dots x_{k-1}L_{k-1}x_kL_k$ is called *seminormal* if $x_i \notin L_{i-1}$ for every $1 \leq i \leq k$. L is *normal* if $x_i \notin L_{i-1}$ and $x_i \notin L_i$ $(1 \leq i \leq k)$. A seminormal chain language $L = L_0x_1L_1x_2 \dots x_{k-1}L_{k-1}x_kL_k$ is called *simple* if each L_i $(0 \leq i \leq k)$ is fundamental.

Now we recall the main result from the corresponding section in [4].

Theorem 1. *A language is monotone iff it can be given as a union of finitely many seminormal chain languages.* □

Let X be an alphabet. The set RE of all regular expressions and the language $L(\eta)$ represented by $\eta \in RE$ are defined in parallel as follows:

- $\emptyset \in RE$, $\quad L(\emptyset) = \emptyset$,
- $\forall x \in X : x \in RE$, $\quad L(x) = \{x\}$,
- if $\eta_1, \eta_2 \in RE$, then $(\eta_1) + (\eta_2) \in RE$, $\quad L((\eta_1) + (\eta_2)) = L(\eta_1) \cup L(\eta_2)$,
- if $\eta_1, \eta_2 \in RE$, then $(\eta_1)(\eta_2) \in RE$, $\quad L((\eta_1)(\eta_2)) = L(\eta_1)L(\eta_2)$,
- if $\eta \in RE$, then $(\eta)^* \in RE$, $\quad L((\eta)^*) = L(\eta)^*$.

Some parentheses can be omitted from regular expressions, if a precedence relation is assumed between the operations of iteration, concatenation, and union in the given order.

A regular expression ζ is called a *subexpression* of η if ζ occurs in the inductive definition of η. The set of all subexpressions of η will be denoted by *Sub(η)*. The operation *omission* on regular expressions is defined as follows: Let us consider $\eta_1, \eta_2 \in RE$ and the regular expressions $(\eta_1)+(\eta_2)$, $(\eta_1)(\eta_2)$ and $(\eta_1)^*$. By omitting η_1 from them we get η_2 from the first two ones, and the expression η_1 from the third one. We also allow the omission of η_1 from $(\eta_1)^*$ to result in $(\emptyset)^*$. If we omit η_2 from $(\eta_1) + (\eta_2)$ and $(\eta_1)(\eta_2)$ we get η_1 and η_1 respectively. This way, omission is not well-defined, nor does it have to be. Let ζ be a subexpression of a regular expression η. We call ζ *redundant in* η if ζ can be omitted from η so that $L(\eta)$ remains the same after the omission. A regular expression is *reduced* if it has no redundant subexpressions.

The reduction of a regular expression is not necessarily unique as the following example shows.

Example 2. Let us consider the regular expression $\eta = x(yx)^* + z + (xy)^*x$. Obviously the first and the third member of the union represent the same language, that is, both of them are redundant in η. If we omit one of them separately, we get two different reduced regular expressions: $x(yx)^* + z$ and $z + (xy)^*x$ which represent the same language.

Now we define the concept of *iterational height* which is used to identify the length of the longest word that will be used in the iteration of a particular language. Let η be a reduced regular expression in form $(\zeta)^*$. The nonnegative integer $max\{|u| : u \in L(\zeta)\}$ will be called the *iterational height* of η (or $ih(\eta)$ for short), if $L(\zeta)$ is finite. If $L(\zeta)$ is infinite, then let $ih(\eta)$ be the infinity ∞ that we will treat as the largest integer. Let now η be a reduced regular expression in any form. We define $ih(\eta)$ as $max\{ih((\zeta)^*) \mid (\zeta)^* \in Sub(\eta)\}$, if $Sub(\eta)$ contains an expression in form $(\zeta)^*$, and 0 otherwise. The iterational height of a regular language L (or $ih(L)$ for short) is defined as $min\{ih(\eta) \mid L = L(\eta), \eta \in RE\}$.

Example 3. Let us take the regular expression $\zeta = xx + xxx$. By the definition of $ih((\zeta)^*)$ we have $ih((\zeta)^*) = 3$. Let us now consider the regular expression $\eta = x + (\zeta)^*$. It is easy to see that $ih(\eta) = 3$, because η has a subexpression in form $(\zeta)^*$, for which $ih((\zeta)^*) = 3$. Let us now take the language $L(\eta)$, for which we get that $ih(L(\eta)) = 1$, because $L(\eta)$ can also be represented by the regular expression $(x)^*$, for which $ih((x)^*) = 1$.

Lemma 4. *Let η be a reduced regular expression of the form $(\zeta)^*$. If $L(\eta)$ is monotone, then $ih(L(\eta)) \leq 1$.*

Proof. Let η be a reduced regular expression of the form $(\zeta)^*$, and let the monotone X-recognizer $\mathbf{A}$ recognize $L(\eta)$ with the partial ordering $\leq$ on A. We can suppose without the loss of generality that $\mathbf{A}$ is reduced and connected from its initial state, hence there is exactly one final state $a \in A'$ such that $au = a$ for every $u \in L(\zeta)$.

Using the monotonicity of $\mathbf{A}$ we get that $ax = a$ holds for any letter x from the words of $L(\zeta)$. We see that there is no such state $a' \in A \setminus \{a\}$ for which $a' \leq a$ and $a'x = a'$ hold for any $x \in X$, and we also see that there is no final state $a'' \neq a$ such that $a \leq a''$. Hence η can be written in form $\zeta'\zeta''$, where ζ' does not contain the operation *, and represents the set of all words taking $\mathbf{A}$ from a_0 to a, and where ζ'' is in form $(y_1 + \ldots + y_r)^*$, where $y_1, \ldots, y_r$ are the letters from the words of $L(\zeta)$. Since $L(\eta) = L(\zeta'\zeta'')$ and $ih(L(\zeta'\zeta'')) = 1$, we get that $ih(L(\eta)) \leq 1$. □

3 Monotone DR-languages

A *ranked alphabet* is a finite nonempty set of operational symbols, which will be denoted by Σ. The subset of all m-ary operational symbols of Σ will be denoted by Σ_m. We shall suppose in the rest of this paper that $\Sigma_0 = \emptyset$. Let $\mathfrak{p}(S)$ stand for the power set of the set S.

Let X be a set of variables. The set $T_\Sigma(X)$ of ΣX*-trees* is defined as follows:

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

(ii) $\sigma(p_1, \ldots, p_m) \in T_\Sigma(X)$, if $m \geq 0$, $\sigma \in \Sigma_m$ and $p_1, \ldots, p_m \in T_\Sigma(X)$,

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

In the rest of this paper X will stand for the countable set $\{x_1, x_2, \ldots\}$, and for every $n \geq 0$, X_n will denote the subset $\{x_1, \ldots, x_n\}$ of X.

A pair $\mathcal{A} = (A, \Sigma)$ will represent a *deterministic root-to-frontier* Σ*-algebra* (or a DR Σ*-algebra* for short), where A is a nonempty set, and Σ is a ranked alphabet. Every $\sigma \in \Sigma_m$ is represented as a mapping $\sigma^{\mathcal{A}} : A \to A^m$. $\mathcal{A}$ is called *finite*, if Σ is a ranked alphabet and A is finite.

A *deterministic root-to-frontier* ΣX_n*-recognizer* (or a DR ΣX_n*-recognizer* for short) is a system $\mathfrak{A} = (\mathcal{A}, a_0, \mathbf{a})$, where $\mathcal{A} = (A, \Sigma)$ is a finite DR Σ-algebra, $a_0 \in A$ is the *initial state*, and $\mathbf{a} = (A^{(1)}, \ldots, A^{(n)}) \in \mathfrak{p}(A)^n$ is the *final state vector*. If Σ or X_n is not specified, we speak of *DR-recognizers*.

Let $\mathfrak{A} = (\mathcal{A}, a_0, \mathbf{a})$ be a DR ΣX_n-recognizer, and let the mapping $\alpha : T_\Sigma(X_n) \to \mathfrak{p}(A)$ be defined as follows. For every $p \in T_\Sigma(X_n)$

(i) if $p = x_i \in X_n$, then $\alpha(p) = A^{(i)}$,

(ii) if $p = \sigma(p_1, \ldots, p_m)$, then $\alpha(p) = \{a \in A \mid \sigma^{\mathcal{A}}(a) \in \alpha(p_1) \times \ldots \times \alpha(p_m)\}$.

The *tree language* $T(\mathfrak{A})$ *recognized* by $\mathfrak{A}$ can be given by

$$T(\mathfrak{A}) = \{p \in T_\Sigma(X_n) \mid a_0 \in \alpha(p)\}.$$

Tree languages that can be recognized by DR-recognizers are also called *DR-languages*.

Let $\mathfrak{A}$ be a DR ΣX_n-recognizer and $a \in A$ one of its states. We define the tree language $T(\mathfrak{A}, a)$ as the set $\{\ p \in T_\Sigma(X_n) \mid a \in \alpha(p)\}$. A state a is called

0-*state* if $T(\mathfrak{A}, a) = \emptyset$. $\mathfrak{A}$ is called *normalized* if for all $\sigma \in \Sigma_m$ and $a \in A$ it holds that each component of $\sigma^{\mathcal{A}}(a)$ is a 0-state or no component of $\sigma^{\mathcal{A}}(a)$ is a 0-state. Moreover, $\mathfrak{A}$ is called *reduced* if for any states $a, b \in A$ it holds that $a \neq b$ implies $T(\mathfrak{A}, a) \neq T(\mathfrak{A}, b)$. It is a well-known fact that every DR-language can be recognized by a normalized and reduced DR-recognizer (cf. [5], [6] and [7]).

Let π_i be the i-th projection. A DR Σ-algebra $\mathcal{A} = (A, \Sigma)$ is called *monotone* if there is a partial ordering $\leq$ on A such that $a \leq \pi_i(\sigma^{\mathcal{A}}(a))$ holds for all $a \in A$, $\sigma \in \Sigma_m$ and $1 \leq i \leq m$. We say that $\mathfrak{A}$ is a *monotone DR ΣX_n-recognizer* if the underlying DR Σ-algebra $\mathcal{A}$ is monotone. Moreover, $T \subseteq T_\Sigma(X_n)$ is a *monotone DR-language*, if $T = T(\mathfrak{A})$ for a monotone DR ΣX_n-recognizer $\mathfrak{A}$ (see, [2] and [4]).

As every partial ordering on a finite set can be extended to a linear ordering, the following lemma hold.

Lemma 5. *For any monotone DR-recognizer $\mathfrak{A}$ we may assume that the partial ordering on A is total.*

The following lemma is also obvious.

Lemma 6. *Every finite DR-language is monotone.*

4 Basic observations

Before we continue the investigation of monotone DR-languages, we need to introduce some concepts and notions (mainly taken from [4], [6] and [7]).

Let Σ be a ranked alphabet, and let $\hat{\Sigma}$ be an ordinary alphabet defined as follows. For all $\sigma, \tau \in \Sigma$ let

(i) $\hat{\Sigma}_\sigma = \{\sigma_1, \ldots, \sigma_m\}$, if $\sigma \in \Sigma_m$ $(m \geq 1)$, and

(ii) $\hat{\Sigma}_\sigma \cap \hat{\Sigma}_\tau = \emptyset$, if $\sigma \neq \tau$.

We define $\hat{\Sigma}$ as $\hat{\Sigma} = \bigcup(\hat{\Sigma}_\sigma \mid \sigma \in \Sigma)$. We say that the alphabet $\hat{\Sigma}$ corresponds to the ranked alphabet Σ.

Let $n \geq 1$ be fixed arbitrarily. The set $g_{x_i}(t)$ of x_i-*paths* of a tree $t \in T_\Sigma(X_n)$ is defined for each $i \in \{1, \ldots, n\}$ in the following way:

(i) $g_{x_i}(x_i) = \{e\}$, and $g_{x_i}(x_j) = \emptyset$, if $i \neq j$, $i, j \in \{1, \ldots, n\}$,

(ii) If $t = \sigma(t_1, \ldots, t_m)$ $(\sigma \in \Sigma_m)$, then $g_{x_i}(t) = \sigma_1 g_{x_i}(t_1) \cup \ldots \cup \sigma_m g_{x_i}(t_m)$.

For a tree language $T \subseteq T_\Sigma(X_n)$, let $g_{x_i}(T) = \bigcup_{t \in T} g_{x_i}(t)$, which is also denoted by T_{x_i} $(1 \leq i \leq n)$.

Let Σ be a ranked alphabet, and let $\hat{\Sigma}$ be the alphabet corresponding to it. Let $\mathcal{A} = (A, \Sigma)$ be a DR Σ-algebra. For every $u \in \hat{\Sigma}^*$ the mapping $u^{\mathcal{A}} : A \to A$ is defined as follows:

(i) If $u = e$, then $au^{\mathcal{A}} = a$, and

(ii) if $u = \sigma_j v$, then $au^{\mathcal{A}} = \pi_j(\sigma(a))v^{\mathcal{A}}$, $(a \in A,\ \sigma \in \Sigma_m,\ 1 \leq j \leq m,\ v \in \hat{\Sigma}^*)$.

The mapping defined above can be extended to subsets of $\hat{\Sigma}^*$ in a natural way. In the rest of this paper we will omit the superscript $\mathcal{A}$ in $u^{\mathcal{A}}$ if the DR Σ-algebra $\mathcal{A}$ inducing $u^{\mathcal{A}}$ is obvious.

A tree language $T \subseteq T_\Sigma(X_n)$ is *closed* if a tree $t \in T_\Sigma(X_n)$ is in T if and only if $g_x(t) \subseteq T_x$ for all $x \in X_n$. It is a well known result, that a regular tree language is DR-recognizable if and only if it is closed (cf. [1] and [9]).

Now we need to specify some details regarding particular operations on tree languages. The *σ-product* of ΣX_n-tree languages $T_1, \ldots, T_m$ is the tree language $\sigma(T_1, \ldots, T_m) = \{\sigma(t_1, \ldots, t_m) \mid t_i \in T_i,\ 1 \leq i \leq m\}$, where $m \geq 1$ and $\sigma \in \Sigma_m$. We assume that the reader is already familiar with the operations of union, x-product and x-iteration. In the rest of this paper, we will use the operation of x-product in right-to-left manner, that is, for any tree languages $S, T \subseteq T_\Sigma(X_n)$ the x-product $T \cdot_x S$ is interpreted as a tree language in which the trees are obtained by taking a tree s from S and replacing every leaf symbol x in s by a tree from T. Different occurrences of x may be replaced by different trees from T. We will also assume that $T \cdot_y R \cdot_x S$ always means $T \cdot_y (R \cdot_x S)$ for any tree languages $S, R, T \subseteq T_\Sigma(X_n)$ and variables $x, y \in X_n$.

Let Σ be a ranked alphabet, and let X_n be a set of variables. The set $RE(\Sigma X_n)$ of all *regular ΣX_n-expressions* and the tree language $T(\eta)$ represented by $\eta \in RE(\Sigma X_n)$ are defined in parallel as follows:

- $\emptyset \in RE(\Sigma X_n)$, $\quad T(\emptyset) = \emptyset$,
- $\forall x \in X_n : x \in RE(\Sigma X_n)$, $\quad T(x) = \{x\}$,

If $\sigma \in \Sigma_m$, $\eta_1, \eta_2, \ldots, \eta_m \in RE(\Sigma X_n)$, $x \in X_n$, then

- $(\eta_1) + (\eta_2) \in RE(\Sigma X_n)$, $\quad T((\eta_1) + (\eta_2)) = T(\eta_1) \cup T(\eta_2)$,
- $(\eta_2) \cdot_x (\eta_1) \in RE(\Sigma X_n)$, $\quad T((\eta_2) \cdot_x (\eta_1)) = T(\eta_2) \cdot_x T(\eta_1)$,
- $(\eta_1)^{*,x} \in RE(\Sigma X_n)$, $\quad T((\eta_1)^{*,x}) = T(\eta_1)^{*,x}$,
- $\sigma(\eta_1, \ldots, \eta_m) \in RE(\Sigma X_n)$, $\quad T(\sigma(\eta_1, \ldots, \eta_m)) = \sigma(T(\eta_1), \ldots, T(\eta_m))$.

Some parentheses can be omitted from regular ΣX_n-expressions, if a precedence relation is assumed between the operations of σ-product, x-iteration, x-product, and union in the given order.

A regular ΣX_n-expression ζ is *subexpression* of η if ζ occurs in the inductive definition of η. The set of all subexpressions of η will be denoted by *Sub(η)*. The operation omission on regular ΣX_n-expressions is defined as follows: Let us consider $\sigma \in \Sigma_m$, $x \in X$, $\eta_1, \eta_2, \ldots, \eta_m \in RE(\Sigma X_n)$ and the regular ΣX_n-expressions $(\eta_1) + (\eta_2)$, $(\eta_2) \cdot_x (\eta_1)$, $(\eta_1)^{*,x}$ and $\sigma(\eta_1, \ldots, \eta_m)$. By omitting η_1 from them we get η_2, η_2, η_1 and $\sigma(\zeta, \eta_2, \ldots, \eta_m)$ respectively, where ζ is a variable occurring in $T(\eta_1)$, if such exists, otherwise $\zeta = \emptyset$. We allow the omission of η_1 from $(\eta_1)^{*,x}$ to result in x as well. If we omit η_2 from $(\eta_1) + (\eta_2)$ and $(\eta_2) \cdot_x (\eta_1)$ we get η_1

and η_1 respectively. Omission on regular ΣX_n-expressions is not well-defined, but we do not need it to be so. Let η be a regular ΣX_n-expression, and let ζ be a subexpression of η. We call ζ *redundant in* η, if ζ can be omitted from η so that $T(\eta)$ remains unchanged after omission. A regular ΣX_n-expression is *reduced* if it has no redundant subexpressions. As in the string case, a regular ΣX_n-expression may have several different reduced forms.

Now we adapt the concept of *iterational height* for tree languages which will be used to identify the length of the longest x-path that will be used in an x-iteration of a particular tree language. Let $x \in X$ be a variable, and let η be a regular ΣX_n-expression in form $(\zeta)^{*,x}$. The *iterational height* of x in η ($ih_x(\eta)$ for short) is defined as $max\{|u| : u \in g_x(T(\zeta))\}$, if $g_x(T(\zeta))$ is finite. If $g_x(T(\zeta))$ is infinite, then let $ih_x(\eta)$ be the infinity ∞ that we will treat as the largest integer. Let now η be a reduced regular ΣX_n-expression in any form. We define $ih_x(\eta)$ as $max\{ih_x((\zeta)^{*,x}) \mid (\zeta)^{*,x} \in Sub(\eta)\}$, if $Sub(\eta)$ contains an expression in form $(\zeta)^{*,x}$, and 0 otherwise. The iterational height of x in a regular tree language T ($ih_x(T)$ for short) is defined as $min\{ih_x(\eta) \ : \ T = T(\eta)\}$.

Example 7. Let $\Sigma = \Sigma_2 = \{\sigma\}$ and $X = \{x, y\}$ hold and let us consider the regular ΣX-expression $\zeta = \sigma(y, \sigma(y, x)) + \sigma(y, \sigma(y, \sigma(y, x)))$. It is easy to see that $ih_x((\zeta)^{*,x}) = 3$. Taking $\eta = \sigma(y, x) + (\zeta)^{*,x}$ we have $ih_x(\eta) = 3$ because η has a subexpression in form $(\zeta)^{*,x}$ for which $ih_x((\zeta)^{*,x}) = 3$. Considering the tree language $T(\eta)$ we get $ih_x(T(\eta)) = 1$, because $T(\eta)$ can be represented also by $(\sigma(y, x))^{*,x}$, for which $ih_x((\sigma(y, x))^{*,x}) = 1$.

Lemma 8. *Let η be a reduced regular ΣX_n-expression of the form $(\zeta)^{*,x}$. If $T(\eta)$ is a monotone DR-language, then $ih_x(T(\eta)) \leq 1$.*

Proof. Let η be a reduced regular ΣX_n-expression of the form $(\zeta)^{*,x}$, and let $\mathfrak{A}$ be a monotone DR-recognizer which recognizes $T(\eta)$ with the partial ordering $\leq$. Without the loss of generality we can suppose that $\mathfrak{A}$ is reduced and normalized, thus there is exactly one state $a \in A$ for which $a \in \alpha(x)$ and $au = a$ hold for every word $u \in g_x(T(\zeta))$. Since $\mathfrak{A}$ is monotone, we see that $a\sigma = a$ for any letter σ that is present in any of the words of $g_x(T(\zeta))$. Moreover, there is no state $a' \in A \setminus \{a\}$ for which $a \leq a'$ and $a' \in \alpha(x)$, and there is no state $a'' \in \alpha(x) \setminus \{a\}$ for which $a'' \leq a$ and $a''\sigma = a''$ hold for every letter σ that is present in any of the words of $g_x(T(\zeta))$. Hence η can be written in form $(\zeta'')^{*,x} \cdot_x \zeta'$, where ζ' represents the tree language that $\mathfrak{A}$ recognizes by taking $A^{(i)} = \{a\}$, and leaving $A^{(j)}$ unchanged if $j \neq i$, and where ζ'' is the representation of the trees that we can get by decomposition of every tree $t \in T(\zeta)$ at every point of the paths in $g_x(t)$. It is easy to see that $T(\eta) = T((\zeta'')^{*,x} \cdot_x \zeta')$ and $ih_x(T((\zeta'')^{*,x} \cdot_x \zeta')) = 1$, that is $ih_x(T(\eta)) \leq 1$. □

5 A simple characterization

Let $\mathfrak{A} = (\mathcal{A}, a_0, \mathbf{a})$ be a monotone DR ΣX_n-recognizer, where $\mathcal{A} = (A, \Sigma^{\mathcal{A}})$, $A = \{a_0, \ldots, a_k\}$ and $\mathbf{a} = (A^{(1)}, \ldots, A^{(n)})$. Without the loss of generality we can suppose that $a_0 \leq a_1 \leq \ldots \leq a_k$ holds. Let $\Xi_k = \{\xi_0, \ldots, \xi_k\}$ be a set of auxiliary variables for which $X_n \cap \Xi_k = \emptyset$ holds. Furthermore, let $\phi : A \to \Xi_k$ be a bijective mapping defined by $\phi(a_i) = \xi_i$ $(0 \leq i \leq k)$. Now we construct the regular $\Sigma(X_n \cup \Xi_k)$-expression η as follows:

$$\eta = \eta_k \cdot_{\xi_k} \eta_{k-1} \cdot_{\xi_{k-1}} \cdots \cdot_{\xi_1} \eta_0,$$

where for each $i = 0, \ldots, k$

$$\eta_i = (p_1^i + \cdots + p_{l_i}^i + y_1^i + \cdots + y_{r_i}^i) \cdot_{\xi_i} (t_1^i + \cdots + t_{j_i}^i)^{*,\xi_i},$$

and where

1) $y_1^i, \ldots, y_{r_i}^i$ are all the elements of the set $\{x_z \in X_n |\ a_i \in A^{(z)}\}$,
2) $p_s^i = \sigma(\xi_{i_1}, \ldots, \xi_{i_m})$ for such $\sigma \in \Sigma_m$ and $\xi_{i_v} \in \Xi_k$ $(1 \leq v \leq m)$ that $\sigma(a_i) = (\phi^{-1}(\xi_{i_1}), \ldots, \phi^{-1}(\xi_{i_m}))$ and $a_i \notin \bigcup_{1 \leq v \leq m} \{\pi_v(\sigma(a_i))\}$ hold $(1 \leq s \leq l_i)$,
3) $t_s^i = \sigma(\xi_{i_1}, \ldots, \xi_{i_m})$ for such $\sigma \in \Sigma_m$ and $\xi_{i_v} \in \Xi_k$ $(1 \leq v \leq m)$ that $\sigma(a_i) = (\phi^{-1}(\xi_{i_1}), \ldots, \phi^{-1}(\xi_{i_m}))$ and $a_i \in \bigcup_{1 \leq v \leq m} \{\pi_v(\sigma(a_i))\}$ hold $(1 \leq s \leq j_i)$,
4) $\mid \{p_1^i, \ldots, p_{l_i}^i\} \mid + \mid \{t_1^i, \ldots, t_{j_i}^i\} \mid = \mid \Sigma \mid$.

The regular $\Sigma(X_n \cup \Xi_k)$-expression η constructed above is called the *trivial regular expression belonging to* $\mathfrak{A}$, and is denoted by $\eta_{\mathfrak{A}}$. We use the word *trivial* because $\eta_{\mathfrak{A}}$ describes $T(\mathfrak{A})$ by its computation in $\mathfrak{A}$, where for every $0 \leq i \leq k$, η_i is responsible for the computation starting in state a_i. That part of η_i which is iterated by the operation ${}^{*,\xi_i}$ is called the *iterating part* of η_i, and the part of η_i which is inserted by $\cdot_{\xi_i}$ product into the variables ξ_i of the iterating part is called the *terminating part* of η_i. We will call the expressions of the form $\eta_k \cdot_{\xi_k} \ldots \eta_1 \cdot_{\xi_1} \eta_0$ by *chains*.

Let the $a_0 \leq a_1 \leq \ldots \leq a_k$ linear ordering hold on the state set of the monotone DR ΣX_n-recognizer $\mathfrak{A}$. Let us define the DR ΣX_n-recognizer $\mathfrak{A}_i$ as follows: $\mathfrak{A}_i = (\mathcal{A}_i, a_i, \mathbf{a_i})$, where $\mathcal{A}_i = (A \cap \{a_i, \ldots, a_k\}, \Sigma^{\mathcal{A}})$, and
$\mathbf{a_i} = (A^{(1)} \cap \{a_i, \ldots, a_k\}, \ldots, A^{(n)} \cap \{a_i, \ldots, a_k\})$. It is obvious that $\mathfrak{A}_i$ recognizes $T(\mathfrak{A}, a_i)$.

Lemma 9. *For a monotone DR ΣX_n-recognizer $\mathfrak{A}$ the equality $T(\mathfrak{A}) = T(\eta_{\mathfrak{A}})$ holds.*

Proof. Let $\mathfrak{A}$ be a monotone DR ΣX_n-recognizer, and let $\eta_{\mathfrak{A}}$ be the trivial regular expression belonging to $\mathfrak{A}$. Let us also suppose that $\mathfrak{A} = (\mathcal{A}, a_0, \mathbf{a})$, $\mathcal{A} = (A, \Sigma)$, $A = \{a_0, \ldots, a_k\}$, and the linear ordering $a_0 \leq \ldots \leq a_k$ holds on A. The proof is continued by induction on the number of states in $\mathfrak{A}$.

If $k = 0$, then $T(\mathfrak{A}) = T_\Sigma(X_n \cap \{x_i |\ a_0 \in A^{(i)}\})$ holds because A is singleton. Obviously $\eta_{\mathfrak{A}} = \eta_0$ holds, too. By the definition of $\eta_{\mathfrak{A}}$, every $\sigma \in \Sigma$ is present in the iterating part of η_0, and every $x \in \{x_i |\ a_0 \in A^{(i)}\} \subseteq X_n$ is present in the terminating part of η_0. Hence, $T(\eta_{\mathfrak{A}}) = T_\Sigma(X_n \cap \{x_i |\ a_0 \in A^{(i)}\})$, that is, $T(\mathfrak{A}) = T(\eta_{\mathfrak{A}})$.

Let us now suppose as our induction hypothesis that $T(\mathfrak{A}_i) = \eta_k \cdot_{\xi_k} \dots \cdot_{\xi_{i+1}} \eta_i$ holds for every $1 \leq i \leq k$. Now we construct the $\Sigma(X_n \cup \Xi_k)$-recognizer $\mathfrak{A}'$ as follows: $\mathfrak{A}' = (\mathcal{A}, a_0, \mathbf{a}')$, where $\mathbf{a}' = (A^{(1)} \cap \{a_0\}, \dots, A^{(n)} \cap \{a_0\}, \{a_0\}, \dots, \{a_k\}) \in \mathfrak{p}(A)^{n+k+1}$. To interpret the meaning of $T(\mathfrak{A}')$ let us treat $X_n \cup \Xi_k$ as the set X_{n+k+1}, where $x_{n+i+1} = \xi_i$, and let the mapping α be defined as $\alpha(\xi_i) = \alpha(x_{n+i+1}) = A^{(n+i+1)}$ $(i = 0, \dots, k)$.

It can be easily seen that $T(\mathfrak{A}) = T(\mathfrak{A}_k) \cdot_{\xi_k} \dots \cdot_{\xi_2} T(\mathfrak{A}_1) \cdot_{\xi_1} T(\mathfrak{A}')$, and $T(\mathfrak{A}') = T(\eta_0)$. Hence

$$\begin{aligned}
T(\mathfrak{A}) &= T(\mathfrak{A}_k) \cdot_{\xi_k} T(\mathfrak{A}_{k-1}) \cdot_{\xi_{k-1}} \dots \cdot_{\xi_2} T(\mathfrak{A}_1) \cdot_{\xi_1} T(\mathfrak{A}') &&= \\
&= T(\eta_k) \cdot_{\xi_k} T(\eta_k \cdot_{\xi_k} \eta_{k-1}) \cdot_{\xi_{k-1}} \dots \cdot_{\xi_2} T(\eta_k \cdot_{\xi_k} \dots \cdot_{\xi_2} \eta_1) \cdot_{\xi_1} T(\eta_0) &&= \\
&= T(\eta_k) \cdot_{\xi_k} T(\eta_{k-1}) \cdot_{\xi_{k-1}} \dots \cdot_{\xi_2} T(\eta_1) \cdot_{\xi_1} T(\eta_0) &&= \\
&= T(\eta_k \cdot_{\xi_k} \eta_{k-1} \cdot_{\xi_{k-1}} \dots \cdot_{\xi_2} \eta_1 \cdot_{\xi_1} \eta_0) &&= \\
&= T(\eta).
\end{aligned}$$

□

6 Remarks on the decomposition of η

In this section we give some remarks on the decomposition of the regular $\Sigma(X_n \cup \Xi_k)$-expression $\eta = \eta_k \cdot_{\xi_k} \dots \eta_1 \cdot_{\xi_1} \eta_0$. If there is at most one symbol in the terminating part of η_i, then the decomposition in the η_i part makes no sense, hence we assume in this section that there are at least two symbols in the terminating part of η_i.

We say that $\eta = \eta_k \cdot_{\xi_k} \dots \cdot_{\xi_{i+1}} \eta_i \cdot_{\xi_i} \dots \cdot_{\xi_1} \eta_0$ can be decomposed in the η_i part if it can be given in the form

$$\begin{gathered}
\eta = \eta_k \cdot_{\xi_k} \dots \cdot_{\xi_{i+1}} \eta_i \cdot_{\xi_i} \dots \cdot_{\xi_1} \eta_0 = \\
\eta_k \cdot_{\xi_k} \dots \cdot_{\xi_{i+1}} (p_1^i + \dots + p_{l_i}^i + y_1^i + \dots + y_{r_i}^i) \cdot_{\xi_i} (t_1^i + \dots + t_{j_i}^i)^{*,\xi_i} \cdot_{\xi_i} \dots \cdot_{\xi_1} \eta_0 = \\
\eta_k \cdot_{\xi_k} \dots \cdot_{\xi_{i+1}} (y_1^i) \cdot_{\xi_i} (t_1^i + \dots + t_{j_i}^i)^{*,\xi_i} \cdot_{\xi_i} \dots \cdot_{\xi_1} \eta_0 + \\
\vdots \\
+ \eta_k \cdot_{\xi_k} \dots \cdot_{\xi_{i+1}} (y_{r_i}^i) \cdot_{\xi_i} (t_1^i + \dots + t_{j_i}^i)^{*,\xi_i} \cdot_{\xi_i} \dots \cdot_{\xi_1} \eta_0 + \\
+ \eta_k \cdot_{\xi_k} \dots \cdot_{\xi_{i+1}} (p_1^i) \cdot_{\xi_i} (t_1^i + \dots + t_{j_i}^i)^{*,\xi_i} \cdot_{\xi_i} \dots \cdot_{\xi_1} \eta_0 + \\
\vdots \\
+ \eta_k \cdot_{\xi_k} \dots \cdot_{\xi_{i+1}} (p_{l_i}^i) \cdot_{\xi_i} (t_1^i + \dots + t_{j_i}^i)^{*,\xi_i} \cdot_{\xi_i} \dots \cdot_{\xi_1} \eta_0,
\end{gathered}$$

where

(i) $y_s^i \in X_n$ $(1 \leq s \leq r_i,\ 0 \leq r_i \leq n)$,

(ii) $p_s^i = \sigma(\xi_{i_1}, \dots, \xi_{i_m})$, for some $\sigma \in \Sigma_m,\ \xi_{i_v} \in \Xi_k,\ 1 \leq v \leq m,\ 1 \leq s \leq l_i$,

(iii) $t_s^i = \sigma(\xi_{i_1}, \dots, \xi_{i_m})$, for some $\sigma \in \Sigma_m,\ \xi_{i_v} \in \Xi_k,\ 1 \leq v \leq m,\ 1 \leq s \leq j_i$.

Now we state a necessary condition for the existence of such decompositions.

Lemma 10. *The expression $\eta = \eta_k \cdot_{\xi_k} \dots \eta_1 \cdot_{\xi_1} \eta_0$ can be decomposed in the η_i part, if every operational symbol in the iterating part of η_i contains the auxiliary variable ξ_i at most once among its leaves.*

Proof. Let us suppose that the condition of the lemma holds. Let us denote in this proof the regular $\Sigma(X_n \cup \Xi_k)$-expressions $\eta_k \cdot_{\xi_k} \dots \cdot_{\xi_{i+2}} \eta_{i+1}$ and $(t_1^i + \dots + t_{j_i}^i)^{*,\xi_i} \cdot_{\xi_i} \dots \cdot_{\xi_1} \eta_0$ by ζ'' and ζ', respectively. It is easy to see that for every tree $t \in T(\zeta')$ the set $g_{\xi_i}(t)$ is a singleton or the empty set. By the definition of the x-product of tree languages, using the condition of the lemma, we get

$$T(\eta) = T(\zeta'' \cdot_{\xi_{i+1}} (p_1^i + \dots + p_{l_i}^i + y_1^i + \dots + y_{r_i}^i) \cdot_{\xi_i} \zeta') =$$

$$= T(\zeta'') \cdot_{\xi_{i+1}} T(p_1^i + \dots + p_{l_i}^i + y_1^i + \dots + y_{r_i}^i) \cdot_{\xi_i} T(\zeta') =$$

$$= T(\zeta'') \cdot_{\xi_{i+1}} (\ T(p_1^i) \cdot_{\xi_i} T(\zeta') \cup \dots \cup T(p_{l_i}^i) \cdot_{\xi_i} T(\zeta') \cup T(y_1^i) \cdot_{\xi_i} T(\zeta') \cup \dots$$

$$\dots \cup T(y_{r_i}^i) \cdot_{\xi_i} T(\zeta')\ ) =$$

$$= T(\zeta'' \cdot_{\xi_{i+1}} p_1^i \cdot_{\xi_i} \zeta' + \dots + \zeta'' \cdot_{\xi_{i+1}} p_{l_i}^i \cdot_{\xi_i} \zeta' + \zeta'' \cdot_{\xi_{i+1}} y_1^i \cdot_{\xi_i} \zeta' + \dots + \zeta'' \cdot_{\xi_{i+1}} y_{r_i}^i \cdot_{\xi_i} \zeta').$$

Hence the decomposition in η_i led to an equivalent regular $\Sigma(X_n \cup \Xi_k)$-expression. □

It is clear that if the auxiliary variable ξ_i does not occur in the subexpression $\eta_{i-1} \cdot_{\xi_{i-1}} \dots \eta_1 \cdot_{\xi_1} \eta_0$, then the factor η_i can be omitted from the expression of η. Let us note that the decomposed parts will also be called *chains*, that is, the above mentioned chain η is decomposed into finite union of chains.

The variables $y_1^i, \dots, y_{r_i}^i$ can be left in any of the decomposed chains, because by inserting these variables into the iterating part during the ξ_i-product we terminate that path, that is, no auxiliary variable can be reached after from these variables.

Now we state the converse of the Lemma 10.

Lemma 11. *If the expression $\eta = \eta_k \cdot_{\xi_k} \dots \eta_1 \cdot_{\xi_1} \eta_0$ can be decomposed in the η_i part, then every operational symbol in the iterating part of η_i contains the auxiliary variable ξ_i at most once among its leaves.*

Proof. Let us suppose that there is an operational symbol $\sigma \in \Sigma_m$ in the iterating part of the decomposed η_i, where ξ_i occurs at least twice among the leaves of σ. Let ζ'' and ζ' stand for the regular $\Sigma(X_n \cup \Xi_k)$-expressions $\eta_k \cdot_{\xi_k} \dots \cdot_{\xi_{i+2}} \eta_{i+1}$ and $\eta_{i-1} \cdot_{\xi_{i-1}} \dots \cdot_{\xi_1} \eta_0$, respectively. For the sake of simplicity we will write $\tilde{\sigma}(\xi_i, \xi_i)$ instead of $\sigma(\xi'_1, \dots, \xi'_{v_1}, \xi_i, \xi''_1, \dots, \xi''_{v_2}, \xi_i, \xi'''_1, \dots, \xi'''_{v_3})$, where $v_1, v_2, v_3 \in \{0, 1, \dots, m-2\}$, $v_1 + v_2 + v_3 = m - 2$, and $\xi'_{z'}, \xi''_{z''}, \xi'''_{z'''} \in \Xi_k$, $(z' \in \{1, \dots, v_1\},\ z'' \in \{1, \dots, v_2\},\ z''' \in \{1, \dots, v_3\})$. It is obvious that $T(\zeta'' \cdot_{\xi_{i+1}} (p_1^i + \dots + p_{l_i}^i + y_1^i + \dots + y_{r_i}^i) \cdot_{\xi_i} \tilde{\sigma}(\xi_i, \xi_i) \cdot_{\xi_i} \zeta') \subset T(\eta)$. Moreover, $T(\zeta'' \cdot_{\xi_{i+1}} \tilde{\sigma}(s_1, s_2) \cdot_{\xi_i} \zeta') \subset T(\eta)$ holds too for every different pair of symbols $s_1, s_2 \in \{p_1^i, \dots, p_{l_i}^i, y_1^i, \dots, y_{r_i}^i\}$. On the other hand $T(\zeta'' \cdot_{\xi_{i+1}} \tilde{\sigma}(s_1, s_2) \cdot_{\xi_i} \zeta') \not\subseteq \bigcup_{1 \le v \le l_i} T(\zeta'' \cdot_{\xi_{i+1}} \tilde{\sigma}(p_v^i, p_v^i) \cdot_{\xi_i} \zeta') \cup \bigcup_{1 \le v \le r_i} T(\zeta'' \cdot_{\xi_{i+1}} \tilde{\sigma}(y_v^i, y_v^i) \cdot_{\xi_i} \zeta')$, which is a contradiction because there are such trees in $T(\eta)$ which are not present in the decomposed chains of η. □

The above results can be summarized in

Theorem 12. *The expression* $\eta = \eta_k \cdot_{\xi_k} \ldots \eta_1 \cdot_{\xi_1} \eta_0$ *can be decomposed in the* η_i *part if and only if every operational symbol in the iterating part of* η_i *contains the auxiliary variable* ξ_i *at most once among its leaves.*

7 Remarks on the number of the auxiliary variables in $\eta_{\mathfrak{A}}$

In this section we deal with the number of the auxiliary variables in $\eta_{\mathfrak{A}}$. We will also give some methods by which this number can be possibly reduced. It is obvious that if the number of states is k, then the representation can be done with $k+1$ auxiliary variables.

It is said that we *terminate a variable* $x \in X_n$ *in a tree* $t \in T_\Sigma(X_n)$ by a tree $p \in T_\Sigma(X_n)$, if the variable x is not present among the leaves of the trees $p \cdot_x t$. Let ζ be a regular ΣX_n-expression. It is said that *a variable* $x \in X_n$ *is terminated in* ζ, if there is no variable x among the leaves of the trees of $T(\zeta)$.

Obviously, the number of the necessary auxiliary variables can be possibly decreased if we decompose η at every possible place (as seen in the previous section), and we renumber the auxiliary variables from 0 in each decomposed chain of η separately.

It is clear that a variable ξ_i is terminated in the η_i part, that is the variable ξ_i will not occur at any leaf from this point during the right-to-left evaluation of η. Hence we can reuse some auxiliary variables within a chain. Let us suppose that there is an auxiliary variable ξ_j in the chain which has its first occurrence in the terminating part of η_i (during the right-to-left evaluation of the chain). In this case every occurrence of ξ_j in η can be replaced with ξ_i, by which we have done an equivalent transformation. In fact, we can also use the elements of X_n to decrease the number of the auxiliary variables. The idea is the same, that is, an existing auxiliary variable ξ_i can be replaced with a variable x if ξ_i gets terminated before the first occurrence of x.

On the basis of the remarks above the following steps can possibly reduce the number of the auxiliary variables:

(i) decompose $\eta_{\mathfrak{A}}$ into union of as many chains as possible

(ii) decrease the number of the auxiliary variables in these decomposed chains separately

(iii) renumber the auxiliary variables starting with 0 in each chain

Example 13. Let $\mathfrak{A} = (\mathcal{A}, a_0, \mathbf{a})$ be a DR ΣX_3-recognizer, where $\mathcal{A} = (A, \Sigma)$, $A = \{a_0, a_1, a_2, a_3\}$, $\Sigma = \{\sigma_1, \sigma_2, \sigma_3\}$, $\sigma_i \in \Sigma_i$ $(1 \leq i \leq 3)$, and $\mathbf{a} = (\{a_0\}, \{a_0, a_2\}, \{a_1, a_2, a_3\})$. Σ is realized in $\mathcal{A}$ as follows:

$$\begin{array}{lll}\sigma_1(a_0)=(a_1), & \sigma_2(a_0)=(a_0,a_1), & \sigma_3(a_0)=(a_0,a_0,a_1),\\ \sigma_1(a_1)=(a_3), & \sigma_2(a_1)=(a_2,a_2), & \sigma_3(a_1)=(a_1,a_3,a_3),\\ \sigma_1(a_2)=(a_3), & \sigma_2(a_2)=(a_2,a_3), & \sigma_3(a_2)=(a_2,a_3,a_3),\\ \sigma_1(a_3)=(a_3), & \sigma_2(a_3)=(a_3,a_3), & \sigma_3(a_3)=(a_3,a_3,a_3).\end{array}$$

The resulting regular expression is the following:

$$\eta_{\mathfrak{A}} = \eta_3 \cdot_{\xi_3} \eta_2 \cdot_{\xi_2} \eta_1 \cdot_{\xi_1} \eta_0 =$$

$$= (x_3) \cdot_{\xi_3} (\sigma_1(\xi_3)+\sigma_2(\xi_3,\xi_3)+\sigma_3(\xi_3,\xi_3,\xi_3))^{*,\xi_3} \cdot_{\xi_3}$$

$$\cdot_{\xi_3}(\sigma_1(\xi_3)+x_2+x_3) \cdot_{\xi_2} (\sigma_2(\xi_2,\xi_3)+\sigma_3(\xi_2,\xi_3,\xi_3))^{*,\xi_2} \cdot_{\xi_2}$$

$$\cdot_{\xi_2}(\sigma_1(\xi_3)+\sigma_2(\xi_2,\xi_2)+x_3) \cdot_{\xi_1} (\sigma_3(\xi_1,\xi_3,\xi_3))^{*,\xi_1} \cdot_{\xi_1}$$

$$\cdot_{\xi_1}(\sigma_1(\xi_1)+x_1+x_2) \cdot_{\xi_0} (\sigma_2(\xi_0,\xi_1)+\sigma_3(\xi_0,\xi_0,\xi_1))^{*,\xi_0}$$

We can decompose the above chain in the η_1 factor by which we get

$$(x_3) \cdot_{\xi_3} (\sigma_1(\xi_3)+\sigma_2(\xi_3,\xi_3)+\sigma_3(\xi_3,\xi_3,\xi_3))^{*,\xi_3} \cdot_{\xi_3}$$

$$\cdot_{\xi_3}(\sigma_1(\xi_3)+x_2+x_3) \cdot_{\xi_2} (\sigma_2(\xi_2,\xi_3)+\sigma_3(\xi_2,\xi_3,\xi_3))^{*,\xi_2} \cdot_{\xi_2}$$

$$\cdot_{\xi_2}(\sigma_1(\xi_3)+x_3) \cdot_{\xi_1} (\sigma_3(\xi_1,\xi_3,\xi_3))^{*,\xi_1} \cdot_{\xi_1}$$

$$\cdot_{\xi_1}(\sigma_1(\xi_1)+x_1+x_2) \cdot_{\xi_0} (\sigma_2(\xi_0,\xi_1)+\sigma_3(\xi_0,\xi_0,\xi_1))^{*,\xi_0}$$

$$+$$

$$(x_3) \cdot_{\xi_3} (\sigma_1(\xi_3)+\sigma_2(\xi_3,\xi_3)+\sigma_3(\xi_3,\xi_3,\xi_3))^{*,\xi_3} \cdot_{\xi_3}$$

$$\cdot_{\xi_3}(\sigma_1(\xi_3)+x_2+x_3) \cdot_{\xi_2} (\sigma_2(\xi_2,\xi_3)+\sigma_3(\xi_2,\xi_3,\xi_3))^{*,\xi_2} \cdot_{\xi_2}$$

$$\cdot_{\xi_2}(\sigma_2(\xi_2,\xi_2)) \cdot_{\xi_1} (\sigma_3(\xi_1,\xi_3,\xi_3))^{*,\xi_1} \cdot_{\xi_1}$$

$$\cdot_{\xi_1}(\sigma_1(\xi_1)+x_1+x_2) \cdot_{\xi_0} (\sigma_2(\xi_0,\xi_1)+\sigma_3(\xi_0,\xi_0,\xi_1))^{*,\xi_0}$$

Simplifying the above expression we can write

$$(x_3) \cdot_{\xi_3} (\sigma_1(\xi_3)+\sigma_2(\xi_3,\xi_3)+\sigma_3(\xi_3,\xi_3,\xi_3))^{*,\xi_3} \cdot_{\xi_3}$$

$$\cdot_{\xi_3}(\sigma_1(\xi_3)+x_3) \cdot_{\xi_1} (\sigma_3(\xi_1,\xi_3,\xi_3))^{*,\xi_1} \cdot_{\xi_1}$$

$$\cdot_{\xi_1}(\sigma_1(\xi_1)+x_1+x_2) \cdot_{\xi_0} (\sigma_2(\xi_0,\xi_1)+\sigma_3(\xi_0,\xi_0,\xi_1))^{*,\xi_0}$$

$$+$$

$$(x_3) \cdot_{\xi_3} (\sigma_1(\xi_3)+\sigma_2(\xi_3,\xi_3)+\sigma_3(\xi_3,\xi_3,\xi_3))^{*,\xi_3} \cdot_{\xi_3}$$

$$\cdot_{\xi_3}(\sigma_1(\xi_3)+x_2+x_3) \cdot_{\xi_2} (\sigma_2(\xi_2,\xi_3)+\sigma_3(\xi_2,\xi_3,\xi_3))^{*,\xi_2} \cdot_{\xi_2}$$

$$\cdot_{\xi_2}(\sigma_2(\xi_2,\xi_2)) \cdot_{\xi_1} (\sigma_3(\xi_1,\xi_3,\xi_3))^{*,\xi_1} \cdot_{\xi_1}$$

$$\cdot_{\xi_1}(\sigma_1(\xi_1)+x_1+x_2) \cdot_{\xi_0} (\sigma_2(\xi_0,\xi_1)+\sigma_3(\xi_0,\xi_0,\xi_1))^{*,\xi_0}$$

Reusing the variables $(\xi_0 \to \xi_3)$ and $(x_3 \to \xi_1)$ in the above chains we get

$$(x_3) \cdot_{\xi_0} (\sigma_1(\xi_0) + \sigma_2(\xi_0, \xi_0) + \sigma_3(\xi_0, \xi_0, \xi_0))^{*,\xi_0} \cdot_{\xi_0}$$
$$\cdot_{\xi_0} (\sigma_1(\xi_0) + x_3) \cdot_{x_3} (\sigma_3(x_3, \xi_0, \xi_0))^{*,x_3} \cdot_{x_3}$$
$$\cdot_{x_3} (\sigma_1(x_3) + x_1 + x_2) \cdot_{\xi_0} (\sigma_2(\xi_0, x_3) + \sigma_3(\xi_0, \xi_0, x_3))^{*,\xi_0}$$
$$+$$
$$(x_3) \cdot_{\xi_0} (\sigma_1(\xi_0) + \sigma_2(\xi_0, \xi_0) + \sigma_3(\xi_0, \xi_0, \xi_0))^{*,\xi_0} \cdot_{\xi_0}$$
$$\cdot_{\xi_0} (\sigma_1(\xi_0) + x_2 + x_3) \cdot_{\xi_2} (\sigma_2(\xi_2, \xi_0) + \sigma_3(\xi_2, \xi_0, \xi_0))^{*,\xi_2} \cdot_{\xi_2}$$
$$\cdot_{\xi_2} (\sigma_2(\xi_2, \xi_2)) \cdot_{x_3} (\sigma_3(x_3, \xi_0, \xi_0))^{*,x_3} \cdot_{x_3}$$
$$\cdot_{x_3} (\sigma_1(x_3) + x_1 + x_2) \cdot_{\xi_0} (\sigma_2(\xi_0, x_3) + \sigma_3(\xi_0, \xi_0, x_3))^{*,\xi_0}$$

We can see that the initial number of the auxiliary variables is reduced from 4 to 2.

We finish the discussion of the section with

Lemma 14. *If $\Sigma = \Sigma_1$, then for any monotone DR ΣX_n-recognizer $\mathfrak{A}$ one auxiliary variable is enough to represent $\eta_{\mathfrak{A}}$.*

Proof. Let $\Sigma = \Sigma_1$, and let $\eta_{\mathfrak{A}}$ be the $\Sigma(X_n \cup \Xi_k)$-regular expression belonging to $\mathfrak{A}$. As we have only unary operational symbols, ξ_i occurs at most once among the leaves of an operational symbol from the iterating part of each η_i. So η can be decomposed into finite union of chains, moreover, the decomposition can be done at each η_i factor. The condition $\Sigma = \Sigma_1$ implies also that during the evaluation at every step there is exactly one auxiliary variable which is not terminated. Since the variable ξ_0 gets terminated in the terminating part of η_0, we can reuse ξ_0 instead of introducing a new auxiliary variable. Continuing the idea we can rewrite all decomposed chains so that they will use only ξ_0 as an auxiliary variable. □

8 Characterization of monotone DR-languages

It is a well-known fact that the class of DR-languages is closed under σ-products, but not under union, x-product, and x-iteration. It means that the x-product, x-iteration and union of monotone DR-languages are not always deterministic (cf. [3] and [8]). Conversely, using the three operations mentioned above on not closed languages can result in a closed (or even monotone) DR-languages, as it can be seen from the examples below.

Example 15. Let us consider the regular tree languages $S = \{\sigma(x,x),\ \sigma(y,y)\}$ and $T = \{\sigma(x,y),\ \sigma(y,x)\}$. It is clear that they are not closed, but the tree language $S \cup T = \{\sigma(x,x),\ \sigma(y,y),\ \sigma(x,y),\ \sigma(y,x)\}$ is closed, that is, DR-recognizable. Moreover, $S \cup T$ is monotone.

Example 16. Let us now consider the regular tree languages $S=\{z,\ \sigma(x,x),\ \sigma(y,y)\}$ and $T=\{\sigma(x,y),\ \sigma(y,x)\}$. They are not closed, but the tree language $T\cdot_z S=\{\sigma(x,x),\ \sigma(y,y),\ \sigma(x,y),\ \sigma(y,x)\}$ is DR-recognizable, and what is more, $T\cdot_z S$ is monotone.

Example 17. Let S be the following regular tree language: $S=\{\sigma(x,\sigma(x,y)),\ \sigma(x,\sigma(y,x)),\ \sigma(x,x),\ \sigma(y,y),\ \sigma(x,y),\ \sigma(y,x)\}$. S is not closed, but the tree language $(S)^{*,x}$ is closed, moreover, $(S)^{*,x}$ is monotone.

Let $S\subseteq T_\Sigma(X_n)$ be a tree language and let $p\in T_\Sigma(X_n)$ be a tree. The *root* $\mathrm{root}(p)$, *leaves* $\mathrm{leaves}(p)$ and the set of *subtrees* $\mathrm{Sub}(p)$ of the tree p are defined as follows:

(i) If $p\in X_n$, then $root(p)=p,\ leaves(p)=\{p\}$ and $Sub(p)=\{p\}$.

(ii) If $p=\sigma(t_1,\ldots,t_m),\ \sigma\in\Sigma_m,\ t_i\in T_\Sigma(X_n),\ 1\le i\le m$, then $root(p)=\sigma$, $leaves(p)=\bigcup_{1\le i\le m} leaves(t_i)$, and $Sub(p)=\{p\}\cup\bigcup_{1\le i\le m}(Sub(t_i))$.

The above functions are extended from trees to tree languages as follows: $root(S)=\{root(p)\mid p\in S\}$, $leaves(S)=\bigcup_{p\in S} leaves(p)$, and $Sub(S)=\bigcup_{p\in S} Sub(p)$.

Let Σ_S denote the set of *operational symbols appearing in* S, and is defined as $\Sigma_S=root(Sub(S))\setminus X_n$. Let $\Sigma_{S,x}$ denote the set $\{\ \sigma\in\Sigma\mid\exists u\in g_x(S),\ \exists v\in\hat{\Sigma}^*,\ \exists z\in X_n\ :\ uv\in g_z(S),\ v=(\sigma,i)\ldots(\omega,j),\ \omega\in\Sigma,\ i,j\in N\}$.

Now we give a condition by which the x-product of two monotone DR-languages is also monotone.

Theorem 18. *Let $S,T\subseteq T_\Sigma(X_n)$ be monotone DR-languages, $x_i\in X_n$. If $\Sigma_{S,x_i}\cap root(T)=\emptyset$, then $T\cdot_{x_i}S$ is monotone.*

Proof. Assume that the conditions of the theorem hold. Let $\mathfrak{A}=(\mathcal{A},a_0,\mathbf{a})$ and $\mathfrak{B}=(\mathcal{B},b_0,\mathbf{b})$ be monotone DR ΣX_n-recognizers, where $\mathcal{A}=(A,\Sigma^{\mathcal{A}})$, $A=\{a_0,\ldots,a_k\}$, $\mathbf{a}=(A^{(1)},\ldots,A^{(n)})$, $\mathcal{B}=(B,\Sigma^{\mathcal{B}})$, $B=\{b_0,\ldots,b_l\}$, $\mathbf{b}=(B^{(1)},\ldots,B^{(n)})$ and $A\cap B=\emptyset$ such that $T(\mathfrak{A})=S$ and $T(\mathfrak{B})=T$. Let us also suppose that $a_0\le\ldots\le a_k$ and $b_0\le\ldots\le b_l$ hold on the state sets A and B, respectively.

We construct a monotone $\mathfrak{C}=(\mathcal{C},c_0,\mathbf{c})$ that recognizes $T\cdot_{x_i}S$ as follows. Let $\mathcal{C}=(C,\Sigma^{\mathcal{C}})$, $C=A\cup B$, $c_0=a_0$ and $\mathbf{c}=(C^{(1)},\ldots,C^{(n)})$ hold, where $\mathbf{c}$ is defined as follows:

$$C^{(j)}=\begin{cases} A^{(j)}\cup B^{(j)}\cup A^{(i)}, & \text{if } x_j\in T,\ j\neq i\\ A^{(j)}\cup B^{(j)}, & \text{if } x_j\notin T,\ j\neq i\\ B^{(j)}\cup A^{(i)}, & \text{if } x_j\in T,\ j=i\\ B^{(j)}, & \text{if } x_j\notin T,\ j=i\end{cases}$$

It remains to represent the elements of Σ in $\mathcal{C}$. For $\sigma\in\Sigma$ and $c\in C$ let

$$\sigma^{\mathcal{C}}(c)=\begin{cases}\sigma^{\mathcal{B}}(c), & \text{if } c\in B\\ \sigma^{\mathcal{B}}(b_0), & \text{if } c\in A^{(i)},\ \sigma\in root(T)\\ \sigma^{\mathcal{A}}(c), & \text{else}\end{cases}$$

The construction of $\mathfrak{C}$ relies on the condition $\Sigma_{S,x_i} \cap root(T) = \emptyset$. It allows us to determine at every step during the processing of a tree in $\mathfrak{C}$ whether the next input symbol is evaluated in $\mathfrak{A}$ or in $\mathfrak{B}$. Once we reach a state $a \in A^{(i)}$, the symbols from $root(T)$ will lead us to a state $b \in B$, from which we can continue the processing in $\mathfrak{B}$. If the input symbol applied in the state a is from $\Sigma \setminus root(T)$, then we process it according to $\mathfrak{A}$. Therefore, it can be shown by a straightforward computation that $\mathfrak{C}$ recognizes $T \cdot_{x_i} S$, and $\mathfrak{C}$ is monotone under the linear ordering $a_0 \leq \ldots \leq a_k \leq b_0 \leq \ldots \leq b_l$, which means that $T \cdot_{x_i} S$ is monotone. □

Corollary 19. *Let $S, T \subseteq T_\Sigma(X_n)$ be monotone DR-languages, $x_i \in X_n$. If $\Sigma_S \cap root(T) = \emptyset$, then $T \cdot_{x_i} S$ is monotone.*

Proof. The conditions of Theorem 18 hold because $\Sigma_{S,x_i} \subseteq \Sigma_S$. □

The conversion of Theorem 18 does not hold as the counter example below shows.

Example 20. Let T and S stand for the DR-languages $\{\sigma(z,z)\}$ and $\{\sigma(x,z), \sigma(\sigma(z,z),z)\}$, respectively. It is obvious that S and T are monotone and $T \cdot_x S = \{\sigma(\sigma(z,z),\ z)\}$ is also monotone. However, $\Sigma_{S,x} \cap root(T) = \{\sigma\} \neq \emptyset$.

Let $x \in X_n$. A tree language T is called *x-homogeneous* if there exists no $t \in T$ for which there are $u, v \in g_x(t),\ w \in \hat{\Sigma}^*$ and $z \in X_n$ such that $uw \in g_z(T)$ and $vw \notin g_z(T)$.

The condition under which the class of monotone DR-languages is closed under x-iteration can be restricted by the following lemmas.

Lemma 21. *Let $T \subseteq T_\Sigma(X_n)$ be a DR-language, $x \in X_n$, and let $T^{*,x}$ be deterministic. If T is not x-homogeneous, then $T^{*,x}$ is not monotone.*

Proof. Let us suppose that the conditions of the lemma hold. It means that there is a tree $t \in T$ for which there are $u, v \in g_x(t)$ with $u \neq v$, and there are $w \in \hat{\Sigma}^*,\ z \in X_n$ such that $uw \in g_z(T)$ and $vw \notin g_z(T)$. Moreover, let us assume that $\mathfrak{A}$ is a reduced monotone DR ΣX_n-recognizer which recognizes $T^{*,x}$. Let $a_i = a_0 u$ and $a_j = a_0 v$. Since $uw \in g_z(T)$ and $vw \notin g_z(T)$, we get that $a_i \neq a_j$. It is obvious that $a_i, a_j \in \alpha(x)$, hence $T(\mathfrak{A}, a_i) = T^{*,x}$ and $T(\mathfrak{A}, a_j) = T^{*,x}$. Using the fact that $\mathfrak{A}$ is reduced, $T(\mathfrak{A}, a_i) = T(\mathfrak{A}, a_j)$ implies that $a_i = a_j$, which is a contradiction. Therefore, $T^{*,x}$ is not monotone. □

Lemma 22. *Let $T \subseteq T_\Sigma(X_n)$ be a DR-language, $x \in X_n$, and let $T^{*,x}$ be deterministic. If $ih_x(T^{*,x}) > 1$, then $T^{*,x}$ is not monotone.*

Proof. Let us suppose that T is a DR-language for which $T^{*,x}$ is deterministic and $ih_x(T^{*,x}) > 1$. Let the regular ΣX_n-expression ζ represent T. By the definition of ih_x, there is a reduced regular ΣX_n-expression η for which $T(\eta) = T^{*,x},\ ih_x(\eta) > 1$ and η is in form $(\zeta)^{*,x}$. Using Lemma 8 we get that $T(\eta)$ is not monotone, therefore $T^{*,x}$ is not monotone, too. □

Now we give a condition by which the x-iteration of a monotone DR-language is also monotone.

Theorem 23. *Let $T \subseteq T_\Sigma(X_n)$ be a monotone DR-language, $x_i \in X_n$, and let T^{*,x_i} be deterministic. If T is x_i-homogeneous, $ih_{x_i}(T^{*,x_i}) \leq 1$ and $\Sigma_{T,x_i} \cap root(T) = \emptyset$, then T^{*,x_i} is monotone.*

Proof. Let us suppose that the conditions of the theorem hold. Let $\mathfrak{A}$ be a reduced DR ΣX_n-recognizer for which $T(\mathfrak{A}) = T$, and where $\mathfrak{A} = (\mathcal{A}, a_0, \mathbf{a})$, $\mathcal{A} = (A, \Sigma^{\mathcal{A}})$, $A = \{a_0, \ldots, a_k\}$, $\mathbf{a} = (A^{(1)}, \ldots, A^{(n)})$. Let us also assume that $\mathfrak{A}$ is monotone under the linear ordering $a_0 \leq \ldots \leq a_k$.

We construct the monotone DR ΣX_n-recognizer $\mathfrak{B} = (\mathcal{B}, b_0, \mathbf{b})$ with $\mathcal{B} = (B, \Sigma^{\mathcal{B}})$ which recognizes T^{*,x_i}. Let us define the state set B as $A \cup \{b_0\}$, where b_0 is a new state. The final state vector $\mathbf{b}$ is

$$(B^{(1)}, \ldots, B^{(i-1)},\ \{a_0, b_0\},\ B^{(i+1)}, \ldots, B^{(n)}),$$

where the components are defined by two steps in the following order:

(1) For all $j \in \{1, \ldots, n\} \setminus \{i\}$, $\quad B^{(j)} := \begin{cases} A^{(j)} \cup \{b_0\}, & \text{if } a_0 \in A^{(j)} \\ A^{(j)}, & \text{else,} \end{cases}$

(2) For all $a \in A^{(i)}$ and $j \in \{1, \ldots, i-1, i+1, \ldots, n\}$ if $a \in A^{(j)}$, then $B^{(j)} := B^{(j)} \cup \{a_0\}$.

The definition of $\Sigma^{\mathcal{B}}$ is given by four steps in the following order:

(3) For all $\sigma \in root(T)$ and $a' \in A^{(i)}$

$$\sigma^{\mathcal{B}}(a_0) := \begin{cases} (\ldots, a_0, \ldots), & \text{if } \sigma^{\mathcal{A}}(a_0) = (\ldots, a', \ldots) \\ \sigma^{\mathcal{A}}(a_0), & \text{else,} \end{cases}$$

(4) For all $\sigma \in \Sigma \setminus root(T)$

$$\sigma^{\mathcal{B}}(a_0) := \begin{cases} \sigma^{\mathcal{A}}(a'), & \text{if } A^{(i)} \neq \emptyset,\ (a' \in A^{(i)} \text{ is arbitrarily chosen}) \\ \sigma^{\mathcal{A}}(a_0), & \text{if } A^{(i)} = \emptyset, \end{cases}$$

(5) For all $\sigma \in root(T)$ $\sigma^{\mathcal{B}}(b_0) := \sigma^{\mathcal{B}}(a_0)$,

(6) For all $\sigma \in \Sigma$ and $a \in A \setminus \{a_0\}$ $\sigma^{\mathcal{B}}(a) := \sigma^{\mathcal{A}}(a)$.

The construction of $\mathfrak{B}$ relies on the condition $\Sigma_{T,x_i} \cap root(T) = \emptyset$. It guarantees us that in every state $a \in \alpha(x_i)$ for any input symbol σ we can determine whether to continue an already started processing of a tree, or to start a process from the root of a tree from T. In all the other cases $\mathfrak{B}$ is acting as $\mathfrak{A}$ did. The x_i-homogeneous property of T and the inequality $ih_{x_i}(T^{*,x_i}) \leq 1$ ensure us that one state is enough to iterate the x_i-paths of T, which is the basic idea of any iteration related automata construction. Therefore, it can be shown by a straightforward computation that $T(\mathfrak{B}) = T^{*,x_i}$, and $\mathfrak{B}$ is monotone under the linear ordering $b_0 \leq a_0 \leq \ldots \leq a_k$, which means that T^{*,x_i} is monotone. $\square$

The following lemma is obvious.

Lemma 24. *For any fixed variable $x \in X_n$ the x-product of tree languages is associative, that is, for any tree languages S, R and T the equality $T \cdot_x (R \cdot_x S) = (T \cdot_x R) \cdot_x S$ holds.*

A tree language $\eta = \eta_k \cdot_{\xi_k} \cdots \cdot_{\xi_1} \eta_0$ is called *R-chain language*, if every η_i is in form $(T_i) \cdot_{\xi_i} (S_i)^{*,\xi_i}$ $(i = 0, \ldots, k)$, where S_i and T_i are finite DR-languages, for which S_i is ξ_i-homogeneous, $ih_{\xi_i}(S_i) \leq 1$, $root(S_i) \cap \Sigma_{S_i,\xi_i} = \emptyset$ and $root(T_i) \cap (root(S_i) \cup \Sigma_{S_i,\xi_i}) = \emptyset$. Moreover, let us denote the language $\eta_{i-1} \cdot_{\xi_{i-1}} \cdots \cdot_{\xi_1} \eta_0$ by ζ_i. The $\eta = \eta_k \cdot_{\xi_k} \cdots \cdot_{\xi_1} \eta_0$ R-chain language is called *generalized*, if $root(T(\eta_i)) \cap \Sigma_{T(\zeta_i),\xi_i} = \emptyset$ holds for every $i = 1, \ldots, k$.

Theorem 25. *Let T be a DR-language. T is monotone iff it can be given as a generalized R-chain language.*

Proof. Let us suppose that T is a monotone DR-language. Let $\mathfrak{A}$ be the monotone DR-recognizer for which $T(\mathfrak{A}) = T$. Constructing the regular expression $\eta_{\mathfrak{A}}$ belonging to $\mathfrak{A}$ we get a generalized R-chain language for which $T = T(\eta_{\mathfrak{A}})$.

Conversely, let us take a generalized R-chain language $\eta = \eta_k \cdot_{\xi_k} \cdots \cdot_{\xi_1} \eta_0$ which represents T. From Lemma 6, Theorem 18, and Theorem 23 we directly obtain that every $T(\eta_i)$ is monotone $(i = 0, \ldots, k)$. Using Lemma 24 and Theorem 18 we directly get that $T(\eta)$ is monotone. □

9 Conclusion

As we showed above, the monotone DR-languages can be characterized by means of generalized R-chain languages. We gave several conditions by which some particular operations preserve monotonicity, but we did not state conditions by which the class of DR-languages is closed under the operations of x-product, x-iteration and union. However, it seems possible to give appropriate conditions for each operation mentioned above.

10 Acknowledgement

The author is extremely grateful to Professor Ferenc Gécseg for his helpful comments and valuable suggestions. The author is also thankful to the anonymous reviewer whose remarks considerably improved the quality and style of this paper.

References

[1] Courcelle, B.: A representation of trees by languages I, *Theoretical Computer Science*, **6** (1978), 255–279.

[2] Gécseg, F.: On some classes of tree automata and tree languages, *Annales Academiæ Scientiarum Fennicæ, Mathematica*, **25** (2000), 325–336.

[3] Gécseg, F. and Gyurica, Gy.: On the closedness of nilpotent DR tree languages under Boolean operations, *Acta Cybernetica,* **17** (2006), 449–457.

[4] Gécseg, F. and Imreh, B.: On monotone automata and monotone languages, *Journal of Automata, Languages, and Combinatorics,* **7** (2002), 71–82.

[5] Gécseg, F. and Steinby, M.: Minimal ascending tree automata, *Acta Cybernetica,* **4** (1978), 37–44.

[6] Gécseg, F. and Steinby, M.: Minimal Recognizers and Syntactic Monoids of DR Tree Languages, In *Words, Semigroups, & Transductions*, World Scientifics (2001), 155–167.

[7] Gécseg, F. and Steinby, M.: *Tree Automata*, Akadémiai Kiadó, Budapest 1984.

[8] Jurvanen, E.: The Boolean closure of DR-recognizable tree languages, *Acta Cybernetica,* **10** (1992), 255–272.

[9] Virágh, J.: Deterministic ascending tree automata I, *Acta Cybernetica*, **5** (1980), 33-42.

Received December, 2005