UNIVERSITY of SZEGED
Faculty of Science and Informatics
Doctoral School in Mathematics and Computer Science
Department of Computer Algorithms and Artificial Intelligence

Characterization of special classes of tree automata

– abstract of the Ph.D. Thesis –

by György Gyuricza

Supervisor: Dr. Ferenc Gécseg

Szeged, 2010

Introduction

Deterministic root-to-frontier tree languages (DR-languages in short) were given special attention in the past mainly because they form a proper subclass of all regular tree languages. This means that some properties of regular tree languages do not necessarily remain valid on DR-languages. Moreover, we do not know too much about DR-languages in general. We wanted therefore to aim our focus on a well defined specific area. Some subclasses of DR-languages (like monotone, nilpotent, definite, etc.) have been already studied and they were characterized in various ways, for example by means of syntactic monoids. For monotone string languages (where by string languages we mean the languages recognized by classical finite recognizers) Gécseg F. and Imreh B. gave a characterization by means of regular expressions (cf. [5]). This gave the idea of examining monotone DR-languages and nilpotent (DR-)languages to give similar characterization by means of regular expressions. The result of our research on this topic represents the basis of the dissertation.

In the first chapter of the thesis we defined the basic concepts that are essential in order to follow and understand the results. We have also done some preparation as well, for example, here we defined the concept of reduced regular expressions. Beyond the basic concepts we needed some algebraic concepts that we assumed the Reader is familiar with.

Gécseg and Imreh stated in [5] that a string language is monotone if and only if it can be given as a finite union of seminormal chain languages. We wanted to follow the same approach when characterizing monotone DR-languages, that is, to identify the sequences of states of a monotone recognizer while it is recognizing a tree. Ultimately we wanted to reuse these sequences to characterize the recognized language.

Nilpotent languages were also studied by Gécseg and Imreh, in [4] for example they characterized nilpotent DR-languages by means of syntactic monoids, however characterization by regular expressions did not take place. Therefore our goal here was to give a characterization for both nilpotent string languages and nilpotent DR-languages by means of regular expressions.

During our research we had to use some closure properties – or conditions that guarantee closedness – in order to characterize monotone- and nilpotent DR-languages. In the last chapter therefore we summarized these closure properties of DR-languages (with respect to monotone- and nilpotent subclasses) under Boolean- (union, intersection, complementation) and regular operations (union, x-product, x-iteration, σ-product).

In some cases we identified various conditions that were needed in order to preserve closedness under a particular operation. The majority of our results in this chapter are taken from [3], [10], [11] and [12].

Results

As we have referred to it in the introduction, the focus of our study was directed towards the monotone- and nilpotent languages. This research resulted in characterization of the above classes by means of regular expressions both for string languages and DR-languages. We have also studied some closure properties of these classes under Boolean- and regular operations. The goal of the thesis is to present these results where the primary focus was on the characterization by means of regular expressions so the study of closure properties was secondary. The thesis however does not aim to examine other properties of monotone- and nilpotent languages so we do not refer to them either.

The results are presented in the following chapters:

1. **Preliminaries** – some of the concepts in this chapter were introduced in [10].
2. **Monotone languages** – the characterization of monotone languages was published in [10].
3. **Nilpotent languages** – the characterization of nilpotent languages comes from [11].
4. **Closure properties** – majority of the results on closure properties were published in [3] although two of them are coming from [10] and [11].

1 Preliminaries

Before we detail the results we need to introduce some basic concepts like alphabet, languages, recognizers and regular expressions. These are substantial concepts that we need to clarify in order to understand the results. For regular expressions we introduce the concept of redundant subexpressions which are those parts of regular expressions that can be omitted without changing the languages they represent. This will be used then to define reduced regular expressions as those expressions that do not contain redundant subexpressions.

Similarly, we introduce the concept of deterministic root-to-frontier algebras, tree automata and tree languages, and then we define the usual and reduced regular ΣX_n-expressions. We recall the concept of x-paths (the set of which is denoted by g_x) that is a very useful tool in studying DR-languages. Furthermore, we highlight some functions on trees like root, height, leaves and Sub, these are used quite commonly. Note that we exclude nullary operational symbols from our ranked alphabets due to practical reasons.

2 Monotone languages

2.1 Monotone string languages

It was proved by Gécseg and Imreh that a string language is monotone if and only if it can be given as a finite union of seminormal chain languages. This representation indeed describes the sequence of states a monotone recognizer would use to process a word, and each of these potential state sequences corresponds to a seminormal chain language in the representation. Hence came the idea that monotone DR-languages could be characterized using a similar approach.

Then we established the concept of iterational height for both regular expressions and string languages. Iterational height identifies the length of the longest word that will be used in the iteration of a particular string language. The importance of iterational height will come up when studying monotone DR-languages although the corresponding results are valid for the string case, too.

Lemma 2.1.7 *If a reduced regular expression is in form* $(\zeta)^*$ *and it represents a monotone string language, then its iterational height is not greater than* 1.

2.2 Monotone DR-languages

In case of DR-languages we define (x-based) iterational height as the length of the longest x-path that will be used in an x-iteration of a particular tree language. Similarly to monotone string languages we show the relationship between monotonicity and iterational height of regular ΣX_n-expressions. This will be important later when we examine the closure properties of monotone DR-languages under x-iteration.

Lemma 2.2.5 *If a reduced regular ΣX_n-expression is in form $(\zeta)^{*,x}$ and it represents a monotone tree language, then its (x-based) iterational height is not greater than* 1.

Let us now take a monotone DR-recognizer $\mathfrak{A}$. As we have seen it in the string case, $\mathfrak{A}$ has a sequence of states that are used for processing trees. To describe this, we have established the trivial regular expression belonging to $\mathfrak{A}$ as

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

where every η_i is in form

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

This form, reading it from right to left, indeed describes the processing in $\mathfrak{A}$. Every single η_i simulates the functionality of a state a_i, where t^is represent trees in form $\sigma(\xi, \ldots, \xi)$ for which the state a_i appears at least once among the elements of $\sigma(a_i)$ and p^is represent trees in form $\omega(\xi, \ldots, \xi)$ for which the state a_i does not appear among the elements of $\omega(a_i)$. Each of the ξs is a member of the auxiliary variable set $\{\xi_0, \ldots, \xi_k\}$, they are corresponding exactly to the states of $\mathfrak{A}$. The y^is are the variables that can be derived from the state a_i. The trees of t^is are encapsulated into a ξ_i-iteration because the application of the operational symbols of the t^is any number of times will still keep a_i in the resulting state vector. For easier reference, we will call the entire expression of $\eta_{\mathfrak{A}}$ as chain, moreover, the part containing the t^is will be called the iterating part of η_i, and the part containing p^is and y^is will be called the terminating part of η_i. We have showed that $\eta_{\mathfrak{A}}$ represents the tree language $T(\mathfrak{A})$.

Lemma 2.3.1 *For any monotone DR-recognizer $\mathfrak{A}$ the equality $T(\mathfrak{A}) = T(\eta_{\mathfrak{A}})$ holds.*

Most of the time the recently introduced trivial representation can be simplified because there are many different DR-recognizers recognizing the same DR-language. It justifies the examination of equivalent transformations on $\eta_{\mathfrak{A}}$ since it may lead to a more general form. One of the obvious transformations would be a decomposition in η_i, if it is possible at all. The research on this led to the following theorem.

Theorem 2.4.3 *The expression $\eta = \eta_k \cdot_{\xi_k} \ldots \cdot_{\xi_1} \eta_0$ can be decomposed in the η_i part if and only if every tree in the iterating part of η_i contains the auxiliary variable ξ_i at most once among its leaves.*

Another potential simplification would be the reduction of the number of the auxiliary variables. This can be achieved by reusing some (auxiliary) variables in the trivial form above. For example, if there is an auxiliary variable ξ_j appearing the first time in the terminating part of η_i (when reading the chain from right to left), then every occurrence of ξ_j in the chain can be replaced with ξ_i. This replacement can be done because ξ_i is not used in the terminating part of η_i or in any further parts of the chain. There is an interesting statement regarding this in the case $\Sigma = \Sigma_1$ as follows.

Lemma 2.5.2 *For every monotone DR-recognizer $\mathfrak{A}$ one auxiliary variable is enough to represent $\eta_{\mathfrak{A}}$.*

It is a well known fact that the class of (monotone) DR-languages is closed under σ-product but it is not closed under other regular operations. This is why it was necessary to find conditions beside which the class of monotone DR-languages is closed under x-product and x-iteration. In order to find them, let $\Sigma_{S,x}$ be the set of those σ operational symbols for which there exists an x-path in S that we can extend by a letter from $\hat{\Sigma}_\sigma$ and then by any applicable word from $\hat{\Sigma}^*$ so that we end up in a path in S. Using this concept we stated the following important theorem.

Theorem 2.6.5 *If S and T are monotone DR-languages with $\Sigma_{S,x} \cap \mathrm{root}(T) = \emptyset$, then $T \cdot_x S$ is monotone.*

Another important concept is the x-homogeneous property. We say that a tree language T is x-homogeneous if there is no such tree $p \in T$ for which there exist $u, v \in g_x(p)$, $w \in \hat{\Sigma}^*$ and $z \in X_n$, where $uw \in g_z(T)$ but $vw \notin g_z(T)$. This will guarantee us that if there are two different states in a DR-recognizer from which we can derive the variable x, then from these two states we can derive exactly the same set of trees. The following statements are also valid.

Corollary 2.6.11 *For any DR-language T if $T^{*,x}$ is a monotone DR-language, then T is x-homogeneous and the iterational height of $T^{*,x}$ is not greater than 1.*

Theorem 2.6.12 *If a monotone DR-language T is x-homogeneous, the iterational height of $T^{*,x}$ is not greater than 1 and $\Sigma_{T,x} \cap \mathrm{root}(T) = \emptyset$, then $T^{*,x}$ is a monotone DR-language.*

Now we have all the necessary concepts to characterize monotone DR-languages. A tree language $\eta = \eta_k \cdot_{\xi_k} \ldots \cdot_{\xi_1} \eta_0$ is called R-chain language if every η_i is given in the form $(T_i) \cdot_{\xi_i} (S_i)^{*,\xi_i}$, where S_i and T_i are finite DR-languages, S_i is ξ_i-homogeneous, the iterational height of $(S_i)^{*,\xi_i}$ is not greater than 1, $\mathrm{root}(S_i) \cap \Sigma_{S_i,\xi_i} = \emptyset$ and $\mathrm{root}(T_i) \cap (\mathrm{root}(S_i) \cup \Sigma_{S_i,\xi_i}) = \emptyset$ $(0 \leq i \leq k)$. The above R-chain language is said to be generalized if $\mathrm{root}(T(\eta_i)) \cap \Sigma_{T(\eta_{i-1} \cdot_{\xi_{i-1}} \cdots \cdot_{\xi_1} \eta_0),\xi_i} = \emptyset$ for every $i \in \{1, \ldots, k\}$. This helps us to state the main result of this section.

Theorem 2.6.15 *A DR-language is monotone if and only if it can be given as a generalized R-chain language.*

3 Nilpotent languages

3.1 Nilpotent string languages

It is common across all nilpotent recognizers that they have a nilpotent element (also known as trap state) and the processing of every word not shorter than the degree of nilpotency of the given recognizer will lead to this element. What is more, no other states are reachable from this state and this is the only state from which there is transition into itself. This means that if we would describe the potential sequences of states that a nilpotent recognizer would go through when reading a word, then we would get a monotone sequence, this is why every nilpotent language is monotone. We would use this normality when characterizing nilpotent languages.

We say that an $L_0 x_1 L_1 x_2 \ldots x_k L_k \subseteq X^*$ chain language is plain if for every index i $(< k)$ $L_i = \{e\}$ holds and if $L_k = Y^*$, where $Y = \emptyset$ or $Y = X$. Plain chain languages are therefore in form $\zeta = x_1 x_2 \ldots x_k L_k$. We say that ζ is finite if $L_k = \{e\}$ or ζ is infinite when $L_k = X^*$, moreover, the length of ζ is k. A plain chain language $\zeta' = x_1 x_2 \ldots x_j$ is called to be a prefix of ζ if either $1 \leq j \leq k$ or $j > k$ with $x_{k+1} \ldots x_j \in L_k$. This also means that every word in X^* is a plain chain language. Furthermore, every finite string language can be given as a finite union of plain chain languages.

Plain chain languages and their prefixes can be used effectively to characterize nilpotent string languages. We proved that if an infinite string language is given as a finite union of plain chain languages and every word in X^* is a prefix of a plain chain language from this union, then the string language in question is nilpotent. Moreover, we proved the converse, that is, every nilpotent string language can be given as a finite union of plain chain languages so that in case the language in question is infinite, then every word in X^* is a prefix of a plain chain language from the above representation. Thus we got the representation of nilpotent string languages by means of regular expressions.

Theorem 3.1.18 *A regular language is nilpotent if and only if it can be given as a finite union of plain chain languages so that in case the language in question is infinite, then every word in X^* is a prefix of a plain chain language from the above representation.*

3.2 Nilpotent DR-languages

Since nilpotent DR-languages are monotone it seems obvious to examine the trivial regular expression belonging to a nilpotent DR-recognizer $\mathfrak{A}$. Let us take the resulting chain $\zeta_{\mathfrak{A}} = \eta_k \cdot_{\xi_k} \eta_{k-1} \cdot_{\xi_{k-1}} \dots \cdot_{\xi_1} \eta_0$. It is clear that apart from η_k the iteration part in every η_i is empty because there are no such operational symbol σ and state a different from the nilpotent element for which $\sigma(a)$ contains a. This means that we can omit these iterational parts from $\zeta_{\mathfrak{A}}$ hence we simplified the trivial regular expression belonging to $\mathfrak{A}$. We will call the result the plain regular expression belonging to $\mathfrak{A}$.

Now we have to study those conditions that will make nilpotent DR-languages closed under x-product. We will need the following concepts. A tree language S is called path complete if in any prefix of any path in S we replace the last letter with any other letter, we get a prefix of a path in S. We have proved the following.

Lemma 3.3.4 *The x-product of two path complete tree languages is path complete.*

Another important concept is the following. A tree language S is said to be x-terminating if every path u in S is an x-path in S provided u is not a proper prefix of any other paths in S. These concepts help us to state conditions by which the x-product of two nilpotent DR-languages is nilpotent.

Theorem 3.3.6 *Let S and T be nilpotent DR-languages where S is finite, path complete and x-terminating, and let $\Sigma_{S,x} \cap \operatorname{root}(T) = \emptyset$. Then $T \cdot_x S$ is nilpotent.*

The only thing remained is to characterize nilpotent DR-languages by regular expressions. A tree language represented by $\eta = \eta_k \cdot_{\xi_k} \dots \cdot_{\xi_1} \eta_0$ is called plain R-chain language if for every $i \in \{0, \dots, k-1\}$ $T(\eta_i)$ is finite and path complete, the set of leaves of the trees from $T(\eta_i) \setminus X_n$ is a nonempty subset of $\{\xi_{i+1}, \dots, \xi_k\}$, $\mathrm{root}(T(\eta_{i+1})) \cap \Sigma_{T(\eta_i \cdot_{\xi_i} \cdots \cdot_{\xi_1} \eta_0), \xi_{i+1}} = \emptyset$ and $T(\eta_k) = Z \cdot_{\xi_k} T_\Sigma(Y \cup \{\xi_k\})$, where Y and Z are subsets of the set of variables. Using this we stated the main result regarding characterization of nilpotent DR-languages.

Theorem 3.3.9 *A DR-language is nilpotent if and only if it is a plain R-chain language.*

4 Closure properties

Since we had to use some closure properties of DR-languages under Boolean- and regular operations (with respect to the monotone- and nilpotent subclasses), it was reasonable to summarize (or even to examine) these properties. In some cases when a class was not closed under an operation, we gathered conditions that guarantee closedness of that particular class under that particular operation. Let us clarify beforehand that the direct product of DR Σ-algebras $\mathcal{A} = (A, \Sigma)$ and $\mathcal{B} = (B, \Sigma)$ is the DR Σ-algebra $\mathcal{A} \times \mathcal{B} = (A \times B, \Sigma)$ where for every $\sigma \in \Sigma_m$ and $(a, b) \in A \times B$ it holds that $\sigma^{\mathcal{A}\times\mathcal{B}}((a,b)) = ((\pi_1(\sigma^{\mathcal{A}}(a)), \pi_1(\sigma^{\mathcal{B}}(b))), \dots, (\pi_m(\sigma^{\mathcal{A}}(a)), \pi_m(\sigma^{\mathcal{B}}(b))))$ and where π_i is the i-th projection.

4.1 Union

We know that DR-languages are not closed under union and so neither monotone- nor nilpotent DR-languages are closed under union. There can be identified however such a condition beside which closedness can be ensured. To achieve this, we need to introduce the following concept. The union direct product of DR ΣX_n-recognizers $\mathfrak{A} = (\mathcal{A}, a_0, \mathbf{a})$ and $\mathfrak{B} = (\mathcal{B}, b_0, \mathbf{b})$ is the DR ΣX_n-recognizer $\mathfrak{A} \times^{\cup} \mathfrak{B} = (\mathcal{A} \times \mathcal{B}, (a_0, b_0), \mathbf{a} \times^{\cup} \mathbf{b})$ such that $\mathbf{a} \times^{\cup} \mathbf{b} \in \mathfrak{p}(A \times B)^n$ and $(\mathbf{a} \times^{\cup} \mathbf{b})^{(i)} = (A^{(i)} \times B) \cup (A \times B^{(i)})$ hold $(1 \le i \le n)$.

Theorem 4.1.4 *Let $\mathfrak{A}$ and $\mathfrak{B}$ be two normalized DR ΣX_n-recognizers. Then $T(\mathfrak{A}) \cup T(\mathfrak{B})$ is deterministic if and only if $T(\mathfrak{A}) \cup T(\mathfrak{B}) = T(\mathfrak{A} \times^{\cup} \mathfrak{B})$.*

Using the above theorem we can conclude that for any two nilpotent (monotone) DR-languages their union is nilpotent (monotone) if and only if it is deterministic.

Later we give conditions beside which the union of two DR-languages is not deterministic.

Theorem 4.1.7 *Let S and T be DR-languages. Then $S \cup T$ is not deterministic if and only if there are a tree p, two variables x, y, and two different paths $u \in g_x(p)$ and $v \in g_y(p)$ such that $u \in g_x(S) \setminus g_x(T)$ and $v \in g_y(T) \setminus g_y(S)$.*

Since the trees satisfying the conditions of the above theorem are not unary, we can conclude the following. If for any DR-languages S and T one of them differs from the other one only in unary trees then $S \cup T$ is deterministic. The same statement is valid for monotone- and nilpotent DR-languages. The previous theorem also implies that if the sets of root symbols of two (monotone, nilpotent) DR-languages are disjoint then the union of these DR-languages is a (monotone, nilpotent) DR-language.

Then we gave conditions beside which the union of two nilpotent DR-languages is not nilpotent. For example, if S and T are nilpotent DR-languages for which $S \setminus T$ contains at least one non-unary operational symbol and there is a variable x such that $g_x(T) \setminus g_x(S)$ is infinite, then $S \cup T$ is not nilpotent. Or if S and T are finite- and infinite nilpotent DR-languages respectively and $S \setminus T$ contains at least one operational symbol with arity greater than 1, then $S \cup T$ is not nilpotent.

4.2 Intersection

Similarly to union direct product we can define intersection direct product. The intersection direct product of two DR ΣX_n-recognizers $\mathfrak{A} = (\mathcal{A}, a_0, \mathbf{a})$ and $\mathfrak{B} = (\mathcal{B}, b_0, \mathbf{b})$ is the DR ΣX_n-recognizer $\mathfrak{A} \times^{\cap} \mathfrak{B} = (\mathcal{A} \times \mathcal{B}, (a_0, b_0), \mathbf{a} \times^{\cap} \mathbf{b})$ for which $(\mathbf{a} \times^{\cap} \mathbf{b}) \in \mathfrak{p}(A \times B)^n$ and $(\mathbf{a} \times^{\cap} \mathbf{b})^{(i)} = A^{(i)} \times B^{(i)}$ hold $(1 \leq i \leq n)$. Then we have the following theorem.

Theorem 4.2.2 *Let $\mathfrak{A}$ and $\mathfrak{B}$ be DR ΣX_n-recognizers. Then $T(\mathfrak{A}) \cap T(\mathfrak{B}) = T(\mathfrak{A} \times^{\cap} \mathfrak{B})$ holds.*

This means that the class of DR-languages is closed under intersection. The structure of $T(\mathfrak{A} \times^{\cap} \mathfrak{B})$ also implies that both monotone- and nilpotent DR-languages are closed under intersection.

4.3 Complementation

It is a known fact that the class of DR-languages is not closed under complementation and so neither monotone DR-languages nor nilpotent DR-languages are closed under complementation. In case of nilpotent DR-languages however we have identified some conditions that ensure closedness. For any given tree language T let $T(x)$ be consisted of all (unary) trees from T whose leaves are x. Let us also denote the complement of T by $\mathrm{c}(T)$. In case the ranked alphabet is consisted of unary operational symbols, we proved that T is nilpotent if and only if $T(x)$ or $\mathrm{c}(T)(x)$ is finite. Moreover, still in the unary case, T is nilpotent if and only if $\mathrm{c}(T)$ is nilpotent.

Let us now assume that there is at least one operational symbol in Σ with arity greater than 1. Then a tree language consisted of unary operational symbols is nilpotent if and only if it is finite. Moreover, if it is nilpotent, then its complement is nilpotent, too. We have also stated that if T is an infinite nilpotent tree language and the complement of T contains at least one non-unary tree, then $\mathrm{c}(T)$ is not nilpotent. The following theorem also holds.

Theorem 4.3.7 *Let us suppose that there is at least one operational symbol in Σ with arity greater than 1. Then a tree language T and its complement $\mathrm{c}(T)$ are simultaneously nilpotent if and only if T or $\mathrm{c}(T)$ is consisted of finitely many unary trees.*

4.4 x-product

We know that the class of DR-languages is not closed under the operation of x-product and so neither monotone DR-languages nor nilpotent DR-languages are closed under the same. In both cases however we gave conditions that guaranteed closedness under x-product, theorems 2.6.5. and 3.3.6. are dealing with these, respectively.

4.5 x-iteration

x-iteration is yet another operation that the class of DR-languages is not closed under. It is clear that none of monotone- or nilpotent DR-languages are closed under x-iteration either. In case of monotone DR-languages we had to identify such a condition that helped preserving monotonicity under x-iteration, theorem 2.6.12. refers to the result on the same.

For nilpotent DR-languages we did not need such a condition so we have not dealt with it either.

4.6 σ-product

The study of closure properties regarding σ-product brought interesting results. It turned out that unlike the previously mentioned operations here we observed difference between the examined classes of DR-languages. Namely, it is known that the class of DR-languages is closed under σ-product and so is the class of monotone DR-languages. However the class of nilpotent DR-languages is not closed as we have showed it using a counter example.

4.7 Summary of closure properties

The below table summarizes the closure properties of DR-languages under Boolean- and regular operations with respect to the monotone- and nilpotent DR-languages.

Closed	$\cup$	$\cap$	$\setminus$	x-product	x-iteration	σ-product
DR	false	true	false	false	false	true
nilpotent	false	true	false	false	false	false
monotone	false	true	false	false	false	true

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 definite and nilpotent DR tree languages, *Journal of Automata, Languages, and Combinatorics.* **9:1** (2004), 55-60.

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

[6] Gécseg, F. and Peák, I.: *Algebraic Theory of Automata*, Akadémiai Kiadó, Budapest 1972.

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

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

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

[10] Gyurica, Gy.: On monotone languages and their characterization by regular expressions, *Acta Cybernetica*, **18** (2007), 117-134.

[11] Gyurica, Gy.: On nilpotent languages and their characterization by regular expressions, *Acta Cybernetica*, **19** (2009), 231-244.

[12] Jurvanen, E.: *On Tree Languages Defined by Deterministic Root-to-frontier Recognizers*, Ph.D. Thesis, University of Turku, Turku, 1995, ISBN 952-90-7096-9.

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

[14] Ševrin, L. N.: On some classes of abstract automata. *Uspehi matem. nauk*, **17:6 (108)** (1962), 219.

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