Minimal Union-free Decompositions of Regular Languages

Sergey Afonin and Denis Golomazov

Lomonosov Moscow State University, Institute of Mechanics
Moscow, Russia
serg@msu.ru

Abstract. A regular language is called *union-free* if it can be represented by a regular expression that does not contain the union operation. Every regular language can be decomposed into a union of a finite number of union-free languages (the so-called *union-free decomposition*). We call the number of components in the minimal union-free decomposition of a regular language *the union width* of the regular language. In this paper we prove that union width of a regular language is computable and we present an algorithm for constructing a corresponding decomposition.

1 Introduction

Regular expressions are a natural formalism for the representation of regular languages. It is well known that there exist regular languages that can be represented by infinitely many equivalent regular expressions, and a number of "canonical" forms of regular expressions representing a given regular language have been proposed in the literature, such as concatenative decomposition [1,2], or union-free decomposition (see e.g. [3]). In this paper we consider union-free decompositions of a regular language.

This paper is devoted to the task of finding a minimal union-free decomposition of a regular language. A language is called *union-free* if it can be represented by a regular expression without the usage of the union operation. For example, the language represented by the expression $(a + b^*)^*$ is union-free because there exists an equivalent expression $(a^*b^*)^*$. Union-free languages have been introduced under the name "star-dot regular" languages by J. Brzozowski in [4]. It follows from the definition that every union-free language L can be written as

$$L = S_{01}^* S_{02}^* \cdots S_{0k_0}^* u_1 S_{11}^* \cdots S_{1k_1}^* u_2 \cdots S_{l-1,1}^* \cdots S_{l-1,k_{l-1}}^* u_l S_{l,1}^* \cdots S_{l,k_l}^*, \qquad (1)$$

where S_{ij} are regular languages, $u_1, \ldots, u_l$ are non-empty words, and $l \geqslant 0$. We call (1) *a general form* of a union-free language and denote it $GF(L)$.

Every regular expression r can be transformed into a regular expression r' in which union operations appear only on the "top level" of the expression, i.e., it takes the following form: $r' = r_1 + \ldots + r_m$, and the regular expressions $r_1, \ldots, r_m$ do not contain the "+" operator (see [3]). This means that every regular language can be represented as a finite union of union-free languages. But this decomposition is not necessarily unique: for example, $(a+b)^* = (a^*b^*)^* = \{\varepsilon\} + a^*ba^* + b^*ab^*$, and these are two different union-free decompositions of the language $(a+b)^*$. We call the minimal number of components in such representation *the union width* of the regular language and a corresponding decomposition (that is not necessarily unique) is called *minimal*. Our goal is an algorithm that computes the union width and constructs a minimal union-free decomposition of a regular language.

Union width of a regular language and corresponding decompositions may be considered as canonical representations of a regular expression, as well as a complexity measure of a regular language [5], similar to the restricted star height. Union-free decompositions play an important role in the algorithm for checking membership of a regular language in a rational subset of a finitely generated semigroup of regular languages with respect to concatenation as a semigroup product. In order to check such membership one should verify that at least one of the distance automata corresponding to the components of an arbitrary union-free decomposition of the regular language is limited. We do not go into the details here (see [6]). We will just mention that checking the limitedness property is PSPACE-complete [7], thus, the number of components in a union-free decomposition is an important parameter influencing membership-checking complexity.

The main result of the current paper is that union width of a regular language can be effectively computed, and we present an algorithm that constructs the corresponding decomposition. This result is achieved by using the combinatorial technique we have adopted from [2]. We prove that for any regular language L there exists an effectively computable finite set $C(L)$ of union-free languages such that there exists a minimal union-free decomposition of L that consists of languages from $C(L)$. We also present an algorithm for checking that a given regular language is union-free. This decidability result is already known (see Theorem 2.1 below) but it is based on reduction to the computationally expensive problem of checking limitedness of distance automata, which is PSPACE-complete.

The paper has the following structure: Section 2 provides some basic definitions, Section 3 presents results concerning general properties of union-free languages, Section 4 is devoted to the algorithm for finding a minimal union-free decomposition of a regular language, and Section 5 contains conclusions and ideas for the further work.

2 Preliminaries

Let $\Sigma = \{a_1, \ldots, a_n\}$ be a finite alphabet, $L \subseteq \Sigma^*$ be a regular language and $V = \langle \Sigma, Q, q_0, F, \varphi \rangle$ be the corresponding minimal deterministic finite automaton, where $Q = \{q_1, \ldots, q_m\}$ is the set of all states of V, q_0 is the initial state, $F \subseteq Q$ is the set of final states and $\varphi : Q \times \Sigma \to Q$ is the transition function of the automaton. Let $M \subseteq \Sigma^*$, $q_1 \in Q$. The definition of the function φ is extended as follows: $\varphi(q_1, M) = \{q \in Q \mid \exists\, \alpha \in M : \varphi(q_1, \alpha) = q\}$.

An ordered list of states $\{q_1,\dots,q_m\}(q_i \in Q)$ is called *a path marked with a word* $w \in \Sigma^*$ iff $w = a_1 \cdots a_{m-1}$ and $\varphi(q_i, a_i) = q_{i+1}$, $i = 1,\dots,m-1$. A path in an automaton is called *cycle-free* iff it starts at the initial state q_0, ends at a final state $q_f \in F$ and does not contain any cycles, i.e., there is no state occurring in the list more than once. It should be noted that when we mention "a cycle-free path in a language" we actually mean a word in the language that is represented by a cycle-free path in the minimal automaton associated with the language.

A language $W \subseteq \Sigma^*$ is called *a star language* iff $W = V^*$ for some $V \subseteq \Sigma^*$.

Let L be a union-free language. We denote $\mathrm{tsw}(L)$ the shortest word in L. Proposition 3.1 shows that the definition is correct, i.e., there cannot exist two different words of minimum length.

Let L be a regular language. Then a representation $L = L_1 \cup L_2 \cup \cdots \cup L_k$ is called *a union-free decomposition* of L iff L_i is a union-free language for all $i = 1,\dots,k$. The decomposition is called *minimal* iff there is no other union-free decomposition of L with fewer elements.

Theorem 2.1 (K. Hashiguchi [8]). *Let L be a regular language, $T \subseteq \{\cdot,\cup,{}^*\}$ be a subset of the rational language operations (concatenation, union, and star), and $M = \{M_1,\dots,M_n\}$ be a finite set of regular languages. Then it is decidable whether L can be constructed from elements of M using a finite number of operations from T.*

As an immediate corollary we obtain that it is decidable whether a regular language L is union-free, by taking singleton languages as M and $T = \{\cdot,{}^*\}$.

Let $B \subseteq Q$. The set of words $\{x \in \Sigma^* \mid \forall q \in B, \varphi(q,x) \in B\}$ is denoted $\mathrm{str}(B)$.

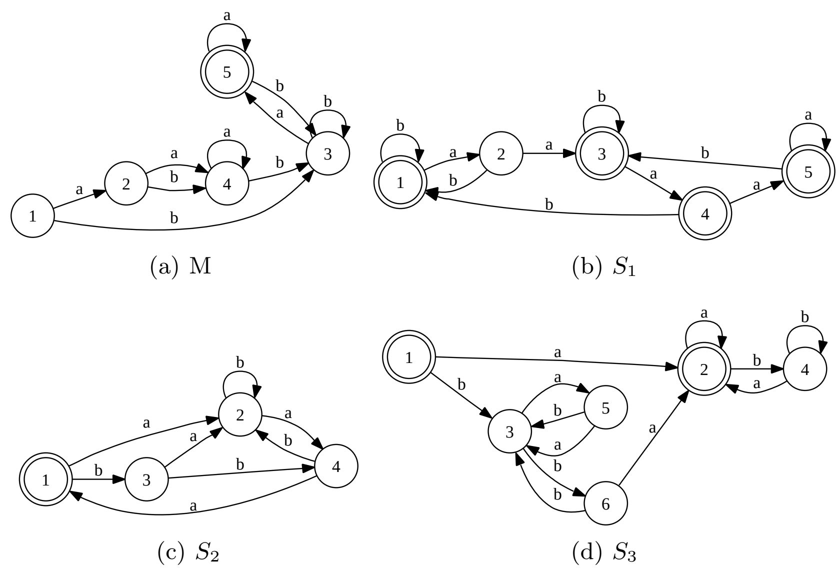
Lemma 2.1 (J.A. Brzozowski, R. Cohen [2]). *Let $B \subseteq Q$. Then* $\mathrm{str}(B)$ *is a regular star language.*

3 Union-free Languages

In this section some common properties of union-free languages are studied. In particular, we present an algorithm for checking whether a regular language is union-free.

First, we adduce an example of a union-free language. Its associated finite automaton is shown in Fig. 1(a). We believe that it is not a simple task to recognize a union-free language by looking at the automaton. For example, the well-known Kleene algorithm constructs a regular expression that contains three union operations on the top level. The language can be represented as $M = S_1^* b S_2^* a S_3^*$ where S_1, S_2, and S_3 are shown in Fig. 1(b),1(c), and 1(d), respectively (the initial states of these automata are marked by 1).

In this section we assume that $\Sigma = \{a_1,\dots,a_n\}$ is a finite alphabet, $L \subseteq \Sigma^*$ is a regular language and $V = \langle \Sigma, Q, q_0, F, \varphi \rangle$ is its associated deterministic finite automaton.



(a) M (b) S_1

(c) S_2 (d) S_3

Fig. 1. Example of the union-free language $M = S_1^* b S_2^* a S_3^*$

Proposition 3.1. *Let L be a union-free language. Then* $\mathrm{tsw}(L)$ *is meaningfully defined, i.e., if u and v are shortest words in L then $u = v$.*

Proof. Suppose $u = u_1 \cdots u_l$, $v = v_1 \cdots v_l$. Consider $GF(L)$. It should have the following form:

$$L = S_{01}^* S_{02}^* \cdots S_{0k_0}^* u_1 S_{11}^* \cdots S_{1k_1}^* u_2 \cdots S_{l-1,1}^* \cdots S_{l-1,k_{l-1}}^* u_l S_{l,1}^* \cdots S_{l,k_l}^*$$

Since $v \in L$ and length of v is equal to that of u, $v_i = u_i$ for $i = 1, \ldots, l$, hence $u = v$. $\square$

Remark 3.1. Obviously, the word $u_1 \cdots u_l$ in the general form of a union-free language L is equal to $\mathrm{tsw}(L)$.

Definition 3.1. *Let $2^Q = \{B_1, \ldots, B_k\}$. We denote*

$$B(L) = \{\mathrm{str}(B_1), \ldots, \mathrm{str}(B_k)\}$$

By Lemma 2.1, $B(L)$ is the fixed set of regular star languages that can be constructed for every regular language. We now show that for every representation of a subset of L as a product of a prefix language, a star language and a suffix language the star language can be replaced with a language from $B(L)$. Thus we can extend each subset of L by replacing all star languages within it with star languages from the fixed set $B(L)$.

Lemma 3.1. *Let $M \subseteq L$. Then for every representation $M = PR^*T$ there exists a language $D \in B(L)$ so that $M \subseteq PDT \subseteq L$.*

Proof. First it should be noted that we do not consider the automaton associated with M and work only within the automaton for L.

Given a representation $M = PR^*T$ we define

$$C = \{q \in Q \mid \exists\, w \in P,\, \varphi(q_0, w) = q\},$$

Then we denote $\widehat{C} = \{q \in Q \mid \forall w \in T\ \ \varphi(q, w) \in F\}$. Obviously, $C \subseteq \widehat{C} \subseteq Q$. We define $D = \mathrm{str}(\widehat{C})$. Taking any words $p \in P$ and $r \in R^*$, we obtain that $\varphi(q_0, p) \in C \subseteq \widehat{C}$ and $\varphi(q_0, pr) \in \widehat{C}$, because $\varphi(prt) \in F$ for all $t \in T$. This means that $\varphi(q, r) \in \widehat{C}$ for all $q \in \widehat{C}$, $r \in R^*$. Hence, $R^* \subseteq D \in B(L)$ and $M \subseteq PDT$. $PDT \subseteq L$, because we extend the language R^* to the language D working within the same unmodified automaton for L and therefore cannot obtain a language that contains more words than L does. □

Corollary 3.1. *For every representation $L = PR^*T$ there exists a language $D \in B(L)$ so that $L = PDT$.*

Proof. We consider $M = L$ and apply Lemma 3.1. Then $L \subseteq PDT \subseteq L$ hence $L = PDT$. □

Definition 3.2. *We denote $C(L)$ a set of all maximal finite concatenations of languages from $B(L)$ and letters such that every concatenation is a subset of L. Maximal means that if C_1, C_2 are such finite concatenations and $C_2 \subseteq C_1$ then we include only the language C_1 in $C(L)$.*

Lemma 3.2. *Let $M \subseteq L$, $B_1 \subseteq Q$, $B_2 \subseteq Q$ and $M = P\,\mathrm{str}(B_1)\,\mathrm{str}(B_2)T$, where P and T are regular languages. Then $\varphi(q_0, P\,\mathrm{str}(B_1)) \subset \varphi(q_0, P\,\mathrm{str}(B_1)\,\mathrm{str}(B_2))$ or there exists $B_3 \subseteq Q$ such that $M \subseteq P\,\mathrm{str}(B_3)T \subseteq L$.*

Proof. We denote three sets of states: $D_1 = \varphi(q_0, P)$, $D_2 = \varphi(q_0, P\,\mathrm{str}(B_1))$, and $D_3 = \varphi(q_0, P\,\mathrm{str}(B_1)\,\mathrm{str}(B_2))$. Since $\mathrm{str}(B_1)$ and $\mathrm{str}(B_2)$ contain the empty word, $D_1 \subseteq D_2 \subseteq D_3$. We also obtain that $\mathrm{str}(B_1) \subseteq \mathrm{str}(D_2)$, because the set $\varphi(q_0, P\,\mathrm{str}(B_1)\,\mathrm{str}(B_1)) = \varphi(q_0, P\,\mathrm{str}(B_1)) = D_2$. Suppose $D_2 = D_3$. This means that $\mathrm{str}(B_2) \subseteq \mathrm{str}(D_2)$. Therefore, $\mathrm{str}(B_1)\,\mathrm{str}(B_2) \subseteq \mathrm{str}(D_2)$ and $M = P\,\mathrm{str}(B_1)\,\mathrm{str}(B_2)T \subseteq P\,\mathrm{str}(D_2)T$. Finally, we take $B_3 = D_2$. $P\,\mathrm{str}(B_3)T \subseteq L$ because we have not modified the automaton for L and still work within it. □

Lemma 3.3. *$C(L)$ is a finite set.*

Proof. Every element in $C(L)$ is a concatenation of star languages and letters. As already mentioned, all the letters concatenated form the shortest word in the language represented by the concatenation. First we limit the number of letters in each concatenation by $|Q| - 1$. We do that as follows: if a concatenation $L_i \in C(L)$ contains more than $|Q| - 1$ letters, we show that there is a language $M_i \in C(L)$ such that $L_i \subseteq M_i$, and we come to a contradiction with the definition of $C(L)$. We show how to effectively construct the language M_i given the language L_i. Suppose $L_i \in C(L)$ and its general form contains more than $|Q|-1$ letters. This means that $\operatorname{tsw}(L_i)$ contains cycles in the automaton for L. Then $\operatorname{tsw}(L_i) = u_1 v_1 \cdots u_h v_h$, where $u_j \in \Sigma^*$, $v_j \in \Sigma^+$ and every $v_j = v_{j_1} \cdots v_{j_{l_j}}$ $(j = 1, \ldots, h)$ represents a cycle in the path $u_1 v_1 \cdots u_h v_h$ in the minimal automaton for L (and every u_j does not contain any cycles). This means that $L_i = L_{u_1} L_{v_1} \cdots L_{u_h} L_{v_h}$ where languages L_{u_j} and L_{v_j} are parts of the general form of L_i corresponding to the words u_j, v_j, respectively. For example, $L_{v_1} = v_{11} S_{p1} \cdots S_{pk_p} v_{12} S_{p+1,1} \cdots S_{p+1,k_{p+1}} v_{13} \cdots v_{1l_1}$. Then we define the language M_i as $M_i = L_{u_1}(L_{v_1})^* \cdots L_{u_h}(L_{v_h})^*$. First, $L_i \subset M_i$. Second, $\operatorname{tsw}(M_i) = u_1 \cdots u_h$ and $u_1 \cdots u_h$ represents a cycle-free path in L. Third, $M_i \subseteq L$, because it has been constructed within the automaton for L. This means that $L_i \subset M_i \subseteq L$ and we come to a contradiction, since $C(L)$ contains only maximal languages.

Now we prove that there is only a limited number of star languages between every pair of adjacent letters in every concatenation $M \in C(L)$. For every representation $M = P \operatorname{str}(B_1) \cdots \operatorname{str}(B_k) T$ we apply Lemma 3.2 and obtain that either $\varphi(q_0, P \operatorname{str}(B_1)) \subset \varphi(q_0, P \operatorname{str}(B_2)) \subset \ldots \subset \varphi(q_0, P \operatorname{str}(B_k))$ or we can replace the language M with the language M' such that $M \subseteq M'$ and $M' = P \operatorname{str}(D_1) \cdots \operatorname{str}(D_l) T$ and

$$\varphi(q_0, P \operatorname{str}(D_1)) \subset \varphi(q_0, P \operatorname{str}(D_2)) \subset \ldots \subset \varphi(q_0, P \operatorname{str}(D_l)).$$

In this case $M \notin C(L)$. We conclude that every element in $C(L)$ can be written as a concatenation with not more than $|Q|-1$ star languages between every pair of adjacent letters. These two limitations complete the proof. □

Corollary 3.2. *Let $M \in C(L)$. Then* $\operatorname{tsw}(M)$ *represents a cycle-free path in the automaton associated with L.*

Proof. Since $M \subseteq L$, $\operatorname{tsw}(M) \in L$. Suppose $\operatorname{tsw}(M)$ contains cycles in the automaton associated with L. Then applying the procedure described in the proof of Lemma 3.3 (which constructs the language M_i using the language L_i), we obtain a language $M' \in C(L)$ such that $M \subset M'$. This means that $M \notin C(L)$ and we get a contradiction. □

Corollary 3.3. $|C(L)| \leqslant c|Q| \left(2^{|Q|}\right)^{|Q|-1}$*, where c is the number of cycle-free paths in the automaton associated with L.*

Proof. First, we fix a cycle-free path in the automaton associated with L (c possibilities). Then we fix a position of star languages: since there are not more than $|Q|-1$ letters, we have $|Q|$ possibilities (because star languages can appear before the first letter and after the last one). Then we choose not more than $|Q|-1$ languages from $B(L)$ (each language can appear more than once), having $\left(2^{|Q|}\right)^{|Q|-1}$ possibilities. Finally, we multiply all three expressions and come to the statement of the corollary. □

Remark 3.2. To construct $C(L)$ given a language L, we simply take all possible concatenations that contain letters that being concatenated form cycle-free paths in the automaton for L and that contain not more than $|Q|-1$ languages from $B(L)$ between each pair of letters. Finally, we exclude the languages that are not subsets of L and the languages that are subsets of other languages from the set.

Lemma 3.4. *Let L be a regular language and $M \subseteq L$ be a union-free language. Then there exists a language $C_M \in C(L)$ such that $M \subseteq C_M$.*

Proof. We take each star language $S^*_{i,j}$ from the general form of M. Thus $M = PS^*_{i,j}T$ where

$$P = S^*_{01} \cdots S^*_{0k_0} a_1 \cdots S^*_{i1} \cdots S^*_{i,j-1}$$

and $T = S^*_{i,j+1} \cdots S_{i,k_i} a_{i+1} \cdots a_l S^*_{l,1} S^*_{l,2} \cdots S^*_{l,k_l}$. Then we apply Lemma 3.1 and derive that $M \subseteq P\,\mathrm{str}(B_k)T$ where $B_k \in B(L)$. Thus we extended the "unknown" language $S^*_{i,j}$ to the known language $\mathrm{str}(B_k)$ from the fixed set $B(L)$. After applying the procedure of extension to each star language in the general form for M, we get a language C_M that is a finite concatenation of languages from $B(L)$ and letters and also an extension of M. Hence $M \subseteq C_M$ and $C_M \in C(L)$. $\square$

Theorem 3.1. *Let L be a regular language. Then L is a union-free language iff $L \in C(L)$.*

Proof. Necessity. We consider $M = L$ and apply Lemma 3.4. Then there exists a language $C_L \in C(L)$ such that $L \subseteq C_L$. But since all languages from $C(L)$ are subsets of L, $L = C_L$ and hence $L \in C(L)$.
Sufficiency. Suppose $L \in C(L)$ and L is a non-union-free language. Then it cannot be represented as a finite concatenation from $C(L)$ because every concatenation from $C(L)$ only consists of union-free languages (languages from $B(L)$ and letters), and we come to a contradiction. $\square$

4 Union-free Decomposition

Theorem 4.1. *Let L be a regular language. Then there exists an algorithm that results in a minimal union-free decomposition of L: $L = L_1 \cup L_2 \cup \cdots \cup L_k$ (the algorithm is described within the proof).*

Proof. To construct a minimal union-free decomposition, we examine all the subsets of $C(L)$ and choose the subset containing a minimum number of languages (among all the subsets) which being added up are equal to L. It should be noted that there is at least one subset containing languages which being added up are equal to L, because there exists at least one union-free decomposition of L and to every component of the decomposition we can apply Lemma 3.4, thus obtaining a decomposition of L into languages from $C(L)$. The final step is to prove that the decomposition obtained is minimal, i.e., there exists no decomposition containing fewer elements than the one we got. Suppose we have such a decomposition $L = N_1 \cup N_2 \cup \cdots \cup N_p$, $p < k$. We take each language N_i $(i = 1, \ldots, p)$

and apply Lemma 3.4 to it, getting a union-free language $C_{N_i} \in C(L)$ such that $N_i \subseteq C_{N_i}$. Thus we get the new decomposition $L = C_{N_1} \cup C_{N_2} \cup \cdots \cup C_{N_p}$, $p < k$ and every language C_{N_i} belongs to the set $C(L)$. But since we have already examined all the subsets of $C(L)$, we have examined the subset $\{C_{N_1}, \ldots, C_{N_p}\}$ too, and we must have chosen this subset for the minimal decomposition. This contradiction completes the proof. □

The algorithm for constructing a minimal union-free decomposition of a given regular language L is computationally hard since it requires checking all the subsets of the set $C(L)$, which can contain up to $c|Q| \left(2^{|Q|}\right)^{|Q|-1}$ elements, where c is the number of cycle-free paths in the automaton associated with L (see Corollary 3.3). We believe that there exist more effective algorithms that results in a minimal union-free decomposition of a given regular language. Some ideas on creating such an algorithm are given below.

A promising way of constructing minimal union-free decompositions can be developed using the technique of cutting maximum star languages introduced in [2]. In short, the technique is as follows. Let L be a regular language. The equation $L = X^* L$ is proved to have the unique maximal solution X_0 (w.r.t. inclusion). Moreover, the equation $L = X_0^* Y$ is proved to have the unique minimal solution Y_0. To construct a minimal union-free decomposition of L we solve these two equations and obtain the language Y_0. Then we apply the same procedure to the language Y_0 and get the minimal language Y_1 such that $L = X_0^* X_1^* Y_1$. If the process ends (and it is an open problem, see [9]) we either obtain the language $Y_m = \{\varepsilon\}$ or get a language Y_m such that the equation $Y_m = X^* Y_m$ has no solutions. Then we check whether all the words in the language Y_m start with the same letter. If it is the case and, for example, all the words in Y_m start with a, we write $Y_m = aY'_m$ and apply the procedure described above to the language Y'_m (solve the equation $Y'_m = X^* Y'_m$ etc.). If it is not, and there are words in Y_m that start with different letters, e.g. $a_1, \ldots, a_n$, we can write $Y_m = Y_{m_1} \cup \cdots \cup Y_{m_n}$ so that every language $Y_{m_1}, \ldots, Y_{m_n}$ contains only words starting with the same letter a_i, $1 \leqslant i \leqslant n$. Then we write $L = X_0^* X_1^* \cdots X_m^* Y_{m_1} \cup X_0^* X_1^* \cdots X_m^* Y_{m_2} \cup \cdots \cup X_0^* X_1^* \cdots X_m^* Y_{m_n}$ and apply the procedure described above to every language $Y_{m_1}, \ldots, Y_{m_n}$. If the process ends, thus we obtain the union-free decomposition of the language L, which is likely to be minimal, but this is yet to be proved. As already mentioned, another open problem connected with this technique is that the described process of "cutting stars" has not yet been proved to always be finite (see [9]).

5 Conclusions and Further Work

In this paper we have presented an algorithm for constructing a minimal union-free decomposition of a regular language. The algorithm includes an exhaustive search but we tend to think that there exist more effective algorithms that solve the problem.

We have also studied some common properties of union-free languages. In particular, we have presented the new algorithm for checking whether a given language is union-free which can be more effective than the one existing in the field (see [8]).

There are some other interesting questions connected with the problems considered. For instance, whether a minimal union-free decomposition consists of disjoint languages (as sets of words) and if it is possible to construct a minimal decomposition that contains disjoint languages only. If it is not always the case, one can consider minimal disjoint union-free decompositions (that consist of disjoint languages) and ways of constructing them.

Another open problem is connected with star height. Given a star height of a regular language is it possible to construct a minimal union-free decomposition that consists of languages of the same star height?

References

1. Paz, A., Peleg, B.: On concatenative decompositions of regular events. IEEE Transactions on Computers **17**(3) (1968) 229–237
2. Brzozowski, J., Cohen, R.: On decompositions of regular events. Journal of the ACM **16**(1) (1969) 132–144
3. Nagy, B.: A normal form for regular expressions. In: Eighth International Conference on Developments in Language Theory, CDMTCS Technical Report 252, CDMTCS, Auckland (2004) 10
4. Brzozowski, J.: Regular expression techniques for sequential circuits. PhD thesis, Princeton University, Princeton, New Jersey (1962)
5. Ehrenfeucht, A., Zeiger, P.: Complexity measures for regular expressions. In: STOC '74: Proceedings of the Sixth Annual ACM Symposium on Theory of Computing, New York, NY, USA, ACM (1974) 75–79
6. Afonin, S., Khazova, E.: Membership and finiteness problems for rational sets of regular languages. International Journal of Foundations of Computer Science **17**(3) (2006) 493–506
7. Leung, H., Podolskiy, V.: The limitedness problem on distance automata: Hashiguchi's method revisited. Theoretical Computer Science **310**(1–3) (January 2004) 147–158
8. Hashiguchi, K.: Representation theorems on regular languages. Journal of Computer and System Sciences **27** (1983) 101–115
9. Brzozowski, J.: Open problems about regular languages. In Book, R.V., ed.: Formal Language Theory, Santa Barbara, CA, Univ. of CA at Santa Barbara, New York: Academic Press (1980) 23–47
10. Nagy, B.: Union-free languages and 1-cycle-free-path-automata. Publicationes Mathematicae Debrecen **68** (2006) 183–197