# Union-Freeness, Deterministic Union-Freeness and Union-Complexity

Benedek Nagy

# Union-Freeness, Deterministic Union-Freeness and Union-Complexity

Benedek Nagy(✉)

Department of Mathematics, Faculty of Arts and Sciences,
Eastern Mediterranean University, Famagusta, North Cyprus, Mersin-10, Turkey
nbenedek.inf@gmail.com

**Abstract.** Union-free expressions are regular expressions without using the union operation. Consequently, union-free languages are described by regular expressions using only concatenation and Kleene star. The language class is also characterised by a special class of finite automata: 1CFPAs have exactly one cycle-free accepting path from each of their states. Obviously such an automaton has exactly one accepting state. The deterministic counterpart of such class of automata defines the deterministic union-free languages. A regular expression is in union (disjunctive) normal form if it is a finite union of union-free expressions. By manipulating regular expressions, each of them has equivalent expression in union normal form. By the minimum number of union-free expressions needed to describe a regular language, its union-complexity is defined. For any natural number $n$ there are languages such that their union complexity is $n$. However, there is not known any simple algorithm to determine the union-complexity of any language. Regarding the deterministic union-free languages, there are regular languages such that they cannot be written as a union of finitely many deterministic union-free languages.

## 1 Introduction

The family of regular languages is one of the most known, most common and most applied class of languages. It is the smallest, the simplest class of the Chomsky hierarchy. The descriptions of the regular languages by regular expressions are widely used. They are generated by regular, by left-linear and by right-linear grammars. They are accepted by finite state automata: both nondeterministic and deterministic variants characterize this class of languages. Recently various classes of subregular languages play also importance [5,9].

In this paper we will consider special subclasses of the regular languages. The main topic is the class of union-free languages, they are defined by regular expressions without the union. They were first mentioned as star-dot regular languages in [2]. Later on, in [4], their description by equations were examined, and it was shown that this class cannot be axiomatized by a finite set of equations. Automata theoretical characterisation was given in [11] allowing to define the deterministic counterpart of the class, the family of deterministic union-free languages [3,7,8].

It is also known [2,10] that every regular language is a finite union of union-free languages. The union-complexity of the regular languages is defined subsequently based on minimal decompositions [1,10,12]. However, there are regular languages that cannot be obtained as a finite union of deterministic union-free languages [8]. On the other hand allowing infinite unions one is able to describe every recursively enumerable language. Therefore infinite unions are usually not allowed when languages are described.

The structure of the paper is as follows. In the next section we define the union-free languages based on regular expressions, we show their characterisation by 1-cycle-free path automata, and we define deterministic union-free regular languages and a new class between the union-free and deterministic union-free class. In Section 3 some properties of the mentioned three language classes, e.g., closure properties are summarised. Section 4 is about the union decomposition of regular languages and the union-complexity.

## 2 The Union-Free Language Classes and Their Corresponding Automata Classes

In this section first we define the union-free languages and then we recall the corresponding class of finite automata. We assume that the reader is familiar with the basic concepts of formal languages and automata, thus for each unexplained concepts she/he is referred to any standard textbook on the topic, e.g., to [6] or the Handbook chapter [13]. Here we show only specific notions closely related to the topic of this paper. The empty word is denoted by $\lambda$, $V$ is a finite alphabet, while $+, \cdot, *$ are the regular operations on languages, i.e., the union, the concatenation and the Kleene star.

**Definition 1 (Union-free expression, union-free language).** *A regular expression is union-free expression if only the operators concatenation and Kleene star are used in its description. A language is union-free if there is a union-free description that defines it.*

A kind of related idea is to define and use star-free expressions, where only union and concatenation are allowed in regular expressions. They define exactly the class of finite languages. Since the class of finite languages has already their well-known name, in the literature, the terms of star-free expressions and languages usually refer to expressions which are defined by operations union, concatenation and complement, and the corresponding language family [13]. Similarly, in the literature sometimes a wider class of languages are called union-free, those which have a description by operations concatenation, Kleene star and complement [9]. However, in this paper, the above stricter concept is used.

The empty language is described as regular expression $\emptyset$. Each other regular expression can be written in a tree form, in which the leaves are representing elements of $V \cup \{\lambda\}$ and the other nodes are representing the regular operations. The language $\emptyset$ is very special, in the rest of the paper we assume that the language we consider is not the empty one.

**Definition 2.** *A 5-tuple* $\mathbf{A} = (Q, S, V, \delta, F)$ *is a non-deterministic finite automaton, with the finite set of states* $Q$*. Further,* $S \in Q$ *is the initial state,* $V$ *is the (input) alphabet and* $F \subset Q$ *is the set of final (or accepting) states. The function* $\delta : Q \times (V \cup \{\lambda\}) \to 2^Q$ *is the transition function. A path is called accepting path of the word* $w$ *if it is written as* $(S = Q_0)a_1Q_1a_2Q_2...a_{n-1}Q_{n-1}a_nQ_n$ *where* $Q_{i+1} \in \delta(Q_i, a_{i+1})$ *for every* $0 \leq i < n$ *with* $Q_n \in F$ *and* $w = a_1a_2...a_n$ *(*$a_i \in V \cup \{\lambda\}$*). A word is accepted by the finite automata if it has an accepting path.*

A path $Q_0a_1Q_1a_2Q_2 \dots a_{n-1}Q_{n-1}a_nQ_n$ is called a cycle if $Q_0 = Q_n$ (where $n > 0$). A path without any repeated state is called cycle-free path. Two cycle-free paths $Q_0a_1 \dots a_nQ_n$ and $P_0b_1 \dots b_mP_m$ are called alternative paths, if they are not identical, but $Q_0 = P_0$ and $Q_n = P_m$.

In this paper we use only automata with the following property: for each state $Q_i$ of the automaton there is an accepting path that contains $Q_i$. Consequently, there is no useless and sink states and the automaton may not be fully determined, i.e., it may happen that for a state $Q_i$ and an input letter $a$ the transition function assigns the empty set.

**Definition 3 (1CFPA, d-1CFPA and n-1CFPA).** *A nondeterministic finite automaton* $\mathbf{A}$ *is a* ***1*** ***c****ycle-****f****ree* ***p****ath* ***a****utomaton, a* 1CFPA*, for short, if there is a unique cycle-free accepting path from each of its states. Moreover, if the automaton* $\mathbf{A}$ *has no* $\lambda$*-transition, then it is an* n-1CFPA*, and if it is a deterministic, then it* d-1CFPA*.*

Figure 1 shows an example. As a consequence of the definition above, a 1CFPA has exactly one final state. From now on $F$ will refer not only for the set of final states, but for its unique element as well. The following result is proven in [11].

**Theorem 4.** *The family of languages which are described by union-free expressions and the family of languages recognized by 1CFPAs are exactly the same.*

**Proposition 5.** *Since from every state* $R$ *there is exactly one transition is going to the direction of* $F$ *(without cycle), the word which transfers the state* $R$ *to* $F$ *in cycle-free path is unique for each state.*

**Definition 6 (backbone).** *The backbone of the automaton is the cycle-free path from the initial state (*$S$*) to the final state (*$F$*). The other parts of the automaton are the loops, sub-loops etc. The word accepted by the backbone is called the backbone word.*

In a directed graph there are two different concepts that are somewhat analogous to cycles in the undirected graphs. Cycles in undirected graph allow to visit a node more than once in a path, while they also allow to connect nodes in alternative paths. The directed cycles allow to return to an already visited node in a path (in a directed graph). Two alternative paths (as two halves of an undirected cycle) give the chance to reach the same target node from a start node in two different cycle-free way. Based on that we can compare the graphs of the classes of automata accepting the union-free and the finite languages.

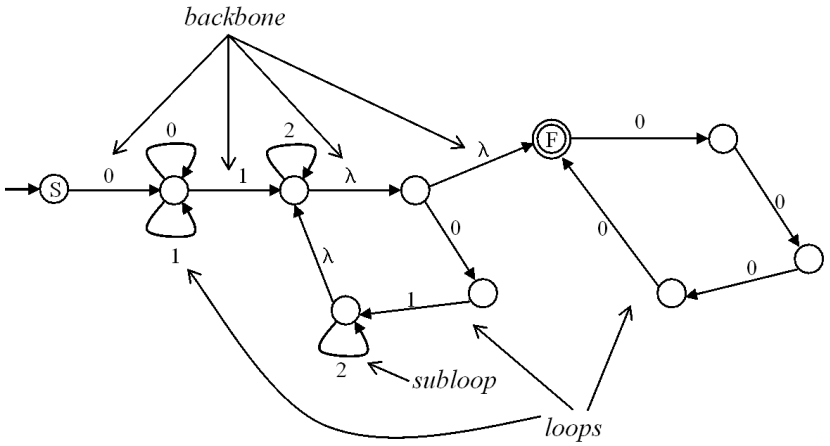


**Fig. 1.** An example for a 1-cycle-free-path-automaton.

**Proposition 7.** *Every union-free language is accepted by automata with graphs having no alternative paths. Every automaton with one final state and without alternative paths is accepting a union-free language. Every finite language is accepted by automata with no cycles. Cycle-free automata accept finite languages.*

As we have already mentioned the class of finite languages is an important subclass of the class of regular languages. Both the classes of nondeterministic and deterministic variants of cycle-free finite state automata correspond to the class of finite languages. The classes of the union-free languages form a hierarchy as we describe here. The result of Theorem 4 allows us to call the language classes accepted by n-1CFPAs and d-1CFPAs as $\lambda$*-free nondeterministic union-free* and *deterministic union-free*, respectively. We also use the abbreviated names *n-union-free* and *d-union-free* for these classes of languages.

## 3 Properties of Union-Free Language Classes

Now we detail some properties of the languages defined above.

**Lemma 8.** *There are infinitely many non-comparable x-union-free languages with* $x \in \{\lambda, n, d\}$.

**Lemma 9.** *A union-free language is infinite if and only if every regular expression contain Kleene star that describes it.*

**Corollary 10.** *A union-free language is either infinite or contains at most one word.*

**Lemma 11.** *Let $L$ be an infinite x-union-free language ($x \in \{\lambda, n, d\}$). There are infinitely many sequences of union-free languages starting with $L$, in which each language is a proper subset of the previous one.*

The next proposition gives a necessary condition for a language to be union-free.

**Proposition 12.** *The shortest word of a union-free language $L$ is unique and it is the backbone word. In a union-free language each word contains the backbone word in scattered way.*

It is well-known that the Parikh images of regular languages coincide with the semi-linear sets. The Parikh images of the union-free languages form a special subset of the semi-linear sets, and at the same time, they form a special superset of the linear sets.

**Definition 13 (Conditional-linear sets).** *A set of vectors $W$ is conditional-linear if the following condition holds. Every vector $\alpha$ is in $W$ if and only if it can be written in the form*

$$\alpha = \alpha_0 + \delta_1 n_1 \alpha_1 + \delta_2 n_2 \alpha_i + \cdots + \delta_m n_m \alpha_m,$$

*where $n_j$ are non-negative integers and $\alpha_j$ are fixed vectors of non-negative integers, and $\delta_i$ are conditional coefficients defined in the following way: $\delta_1 = 1$, and if $i > 1$, then $\delta_i$ is either without any condition and equals to 1, or depends on the coefficient of some $\alpha_j$ with $j < i$, and in such a case it equals to 1 if $\delta_j n_j > 0$ and to 0 if $\delta_j n_j = 0$:*

$$\delta_i = 1, \qquad \text{if there is no condition for } \alpha_i;$$

$$\delta_i = \begin{cases} 1, & \text{if } \delta_j n_j > 0; \\ 0, & \text{if } \delta_j n_j = 0; \end{cases} \qquad \text{if } \alpha_i \text{ depends on the coefficient of } \alpha_j.$$

Having $\delta_i = 1$ for all $i$ without any conditions, the linear sets can be obtained. Thus conditional-linear sets are a kind of generalisations of linear sets. Moreover, all conditional linear sets are semilinear, i.e., they are finite unions of linear sets. However, there are semilinear sets that are not conditional linear.

**Theorem 14.** *Conditional-linear sets coincide with the Parikh images of union-free languages.*

Let $L$ be a union-free language. Note that $\lambda \in L$ if and only if the backbone word is the empty word. This implies that every terminal is under a Kleene star in the tree of the regular expression. Under these circumstances the language can be accepted by a 1CFPA with backbone word $\lambda$. If $L$ is n-union-free and $\lambda \in L$, then $S = F$ in the corresponding n-1CFPA. Since every 1CFPA (and thus d-1CFPA) has exactly one accepting state, languages which cannot be accepted by deterministic finite automata with only one final state are not d-union-free languages.

Now, we are in the position to claim the theorem about the closure properties of union-free languages.

**Theorem 15 (Closure properties of union-free languages).** *The family of union-free languages is closed under the operations concatenation and Kleene star. Further, it is closed under the following operations: reversal, homeomorphism, substitution by union-free expression, Kleene plus, and for any fixed natural number $n$ it is closed under the $n$-th power.*

*The family of union-free languages is not closed under the following operations: union (of course), intersection, intersection with regular languages, complement, difference, symmetric difference, cyclic permutation, permutation, shuffle, inverse morphism and substitution by regular expression.*

**Corollary 16.** *The family of union-free languages is not a cone, not an AFL and not an anti-AFL.*

On the other hand we have only anti-closure properties for the deterministic counterpart:

**Theorem 17 (Closure properties of d-union-free languages).** *The class of deterministic union-free languages is not closed under union, complement, difference, intersection, intersection by regular languages, concatenation, square, Kleene star, reversal, cyclic shift, permutation, homomorphism, and inverse morphism.*

We are turning to hierarchy results. On one hand it is clear by definition that all d-union-free languages are n-union-free languages and all n-union-free languages are union-free. The language $a^*b^*$ is union-free. However, it is not n-union-free. The language $(aa+ab+ba+bb)^*$ is n-union-free, but not d-union-free. Thus, we can state the following:

**Corollary 18.** *There is a proper hierarchy among the union-free classes:*

$$\textit{d-union free} \subsetneq \textit{n-union-free} \subsetneq \textit{union-free.}$$

## 4 Union-Complexity of Regular Languages

We start this section by a decomposition result [2,10].

**Definition 19.** *A regular expression is in union normal form if it is a finite union of union-free expressions.*

**Theorem 20.** *For each regular language there is a regular expression in union normal form that describes it.*

Moreover, based on the following equivalences among regular expressions,

1. $(x+y)^*$ can be written in the form $(x^*y^*)^*$,
2. $(x+y)z$ can be written in the form $(xz+yz)$,
3. $x(z+v)$ can be written in the form $(xz+xv)$,
4. $(x+y)(z+v)$ can be written in the form $(xz+xv)+(yz+yv)$,

where $x, y, z$ and $v$ are arbitrary regular expressions, one can efficiently find an equivalent expression in union normal form for any regular expression.

And now we have some notes about the regular expressions containing the union operation, but describing union-free regular languages.

Let $r$ be a regular expression. For the sake of simplicity assume that its tree is a binary tree, which means that all unions and concatenations have exactly two components, while the Kleene stars have exactly one.

**Theorem 21.** *Let $r$ be a regular expression. If every union operation is under a Kleene star operation in the tree form of $r$, then $r$ defines a union-free regular language.*

The class of union-free languages is an interesting class including several languages since we have:

**Corollary 22.** *For each regular language $L$ the language $L^*$ is union-free regular.*

Now, based on [10,12] we are going to define the union-complexity of languages by specific decompositions.

A decomposition $L = \bigcup_{i=1}^{n} L_i$ is called proper, if there is no language $L_j$ such that $L_j \subseteq \bigcup_{i=1}^{j-1} L_i \cup \bigcup_{i=j+1}^{n} L_i$. (One needs all the languages $L_i$ to describe $L$, i.e., there is no useless member of the union.) A normal form is called proper normal form if it gives a proper decomposition.

**Definition 23 (Union-complexity).** *$L = \bigcup_{i=1}^{n} L_i$ is a minimal decomposition of the language $L$ if each $L_i$ is a union-free language and there is no $m < n$ such that $L = \bigcup_{i=1}^{m} L_i$, where each $L_i$ is union-free. Then, $n$ is called the union-complexity of language $L$.*

Every minimal decomposition is a proper decomposition, but the converse does not hold, there are proper decompositions which are not minimal.

Now we are showing special minimal decompositions: a minimal decomposition of $L$ is given by maximal union-free languages, if there is no $L_i'$ such that $L_i' \supset L_i$ and replacing $L_i$ with $L_i'$ in the union, the resulted language $L$ is the same as before.

**Proposition 24.** *The minimal decomposition by maximal union-free languages of a regular language may not be unique.*

Consider the language over $V = \{a, b\}$ containing all words that do not contain $bb$ as a consecutive substring. Its union-complexity is 2, but there are two minimal decompositions using maximal union-free languages:

$$((ba)^*a^*)^* + ((ba)^*a^*)^*b((ab)^*a^*)^*,$$

and

$$((ab)^*a^*)^* + ((ba)^*a^*)^*b((ab)^*a^*)^*.$$

Let $\mathbf{L}_n$ be the family of languages which can be written as union of $n$ union-free languages.

**Theorem 25.** *The families $\boldsymbol{L}_n$ and $\boldsymbol{L}_m$ are in the following relation:*

$$\boldsymbol{L}_n \supsetneq \boldsymbol{L}_m \text{ iff } n > m.$$

The previous theorem presents an infinite hierarchy of regular languages, in which the union-free ones are the simplest ones. The planet of regular languages is shown in Fig. 2, where the "west pole" is the empty language, the west region contains the union-free languages, the south region contains the finite languages. The intersection of the classes of union-free and finite languages includes the empty language and all the singleton languages. A singleton contains exactly one word. The union-complexity grows to the east direction.

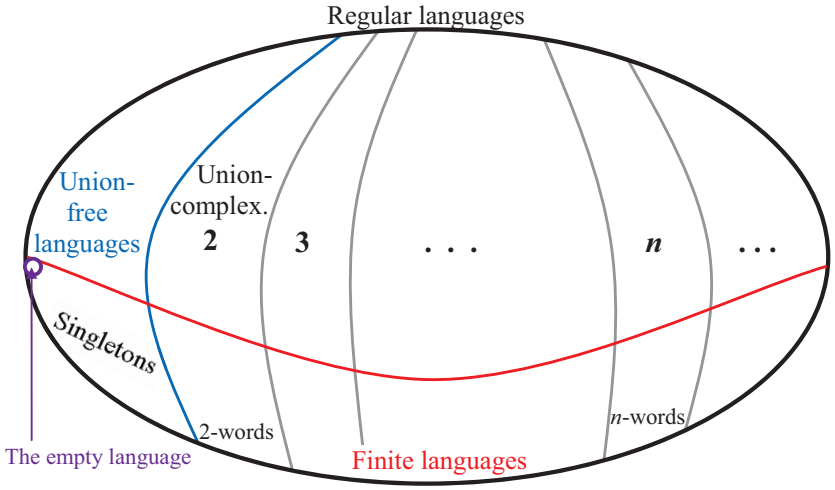


**Fig. 2.** The "planet" of regular languages.

We can summarise the known results about the union-complexity in the following theorem.

**Theorem 26.** *The union-complexity of union-free languages is at most 1: it is 0 for the empty language and 1 for every nonempty union-free language.*
*For every finite language, its union-complexity is exactly the cardinality of the language.*
*A language is regular if and only if its union-complexity is finite.*
*For each regular language $L$ the union-complexity of $L^*$ is 1.*

In [1] it has been proven that the union-complexity of regular languages is computable. However, the method is very complex and cannot be used in practical applications. Some lower and upper bound may be computed much faster, e.g., a union normal form of a regular language defines an upper bound for its union-complexity. On the other hand one can also obtain lower bounds by analysing the short words of the language:

**Proposition 27.** *Let $L$ be a regular language. Fix a natural number $n$ and let $Z_n = \{w \in L \mid |w| \leq n\}$. Consider any subset $Z$ of $Z_n$ with maximal cardinality such that any of the elements of $Z$ does not contain any other elements of $Z$ in a scattered way. Then $|Z|$ is a lower bound for the union-complexity of $L$.*

While every regular language can be expressed as a union of a finite number of union-free languages, this is not true if we replace union-free languages by d-union-free languages.

**Theorem 28.** *The language described by the regular expression $((a+b)(a+b))^*$ cannot be expressed as a union of a finite number of deterministic union-free languages.*

As new subregular classes of languages, we may consider the finite unions of deterministic union-free languages [8].

**Definition 29.** *For every positive integer $m$, we define $\boldsymbol{dL}_m$ as the family of languages that can be expressed as a union of $m$ d-union-free languages. Furthermore, let*

$$\boldsymbol{dL}_* = \bigcup_{i=1}^{\infty} \boldsymbol{dL}_i.$$

The following result shows that the classes $\mathbf{dL}_n$ define a proper infinite hierarchy similarly to the classes shown in Theorem 25.

**Theorem 30.** *Let $n$ and $m$ be positive integers. The families $\boldsymbol{dL}_n$ and $\boldsymbol{dL}_m$ are in the following relation:*

$$\boldsymbol{dL}_n \supsetneq \boldsymbol{dL}_m \text{ iff } n > m.$$

*Moreover,*

$$\boldsymbol{dL}_* \supsetneq \boldsymbol{dL}_m.$$

## 5 Conclusions

We examined in detail the classes of union-free and deterministic union-free languages, moreover we have defined a new class between them, namely the class of n-union-free languages. Various properties of these classes were shown. Since every regular language is a finite union of union-free languages, the union-complexity as a complexity measure of the regular languages was considered. Although it is known that the union-complexity of the regular languages can be computed, there is not known any efficient algorithm to do it. It was also recalled that there is a union-free language which cannot be written as a finite union of d-union-free languages.

## References


1. Afonin, S., Golomazov, D.: Minimal union-free decompositions of regular languages. In: Dediu, A., Ionescu, A., Martín-Vide, C. (eds.) LATA 2009. LNCS, vol. 5457, pp. 83–92. Springer, Heidelberg (2009). https://doi.org/10.1007/978-3-642-00982-2_7
2. Brzozowski, J.A.: Regular expression techniques for sequential circuits. Ph.D. thesis. Department of Electrical Engineering, Princeton University, Princeton, NJ (1962)
3. Brzozowski, J.A., Davies, S.: Most complex deterministic union-free regular languages. In: Konstantinidis, S., Pighizzini, G. (eds.) DCFS 2018. LNCS, vol. 10952, pp. 37–48. Springer, Cham (2018). https://doi.org/10.1007/978-3-319-94631-3_4
4. Crvenković, S., Dolinka, I., Ésik, Z.: On equations for union-free regular languages. Inform. and Comput. **164**(1), 152–172 (2001). https://doi.org/10.1006/inco.2000.2889
5. Holzer, M., Kutrib, M.: Structure and complexity of some subregular language families. In: Konstantinidis, S., Moreira, N., Reis, R., Shallit, J. (eds.) The Role of Theory in Computer Science - Essays Dedicated to Janusz Brzozowski. pp. 59–82. World Scientific (2017). https://doi.org/10.1142/9789813148208_0003
6. Hopcroft, J.E., Ullman, J.D.: Introduction to Automata Theory, Languages and Computation. Addison-Wesley (1979)
7. Jirásková, G., Masopust, T.: Complexity in union-free regular languages. Internat. J. Found. Comput. Sci. **22**(7), 1639–1653 (2011). https://doi.org/10.1142/S0129054111008933
8. Jirásková, G., Nagy, B.: On union-free and deterministic union-free languages. In: Baeten, J.C.M., Ball, T., de Boer, F.S. (eds.) TCS 2012. LNCS, vol. 7604, pp. 179–192. Springer, Heildelberg (2012). https://doi.org/10.1007/978-3-642-33475-7_13
9. Kutrib, M., Wendlandt, M.: Expressive capacity of subregular expressions. RAIRO Theor. Inform. Appl. **52**(2-3-4), 201–218 (2018). https://doi.org/10.1051/ita/2018014
10. Nagy, B.: A normal form for regular expressions. In: Calude, C., Calude, E., Dinneen, M.J. (eds.) Supplemental Papers for DLT 2004, pp. 51–60. CDMTCS Report 252, Auckland (2004)
11. Nagy, B.: Union-free regular languages and 1-cycle-free-path-automata. Publ. Math. Debrecen **68**, 183–197 (2006)
12. Nagy, B.: On union-complexity of regular languages. In: Proc. 11th IEEE International Symposium on Computational Intelligence and Informatics, CINTI 2010. pp. 177–182 (2010)
13. Yu, S.: Regular languages. In: Rozenberg, G., Salomaa, A. (eds.) Handbook of Formal Languages, Vol. 1, pp. 41–110. Springer, Heidelberg (1997). https://doi.org/10.1007/978-3-642-59136-5_2