KYBERNETIKA ČÍSLO 6, ROČNÍK 5/1969

The Theory of Regular Events II

IVAN M. HAVEL

Part II of the survey on regular events* is concerned with some applications of regular events and regular expressions mainly to the synthesis problem of finite automata. The classification of regular events is also mentioned.

6. INTRODUCTION

6.1

The principal task of applied automata theory is the design of finite automata on the basis of a given behaviour, possibly with additional requirements mainly of economical character. We shall rectrict ourselves to the synthesis of automata, the behavioural description of which is given by means of regular expressions.

The outlines of problems related to the abstract synthesis are presented below:

(1) The point of departure is a regular expression characterizing the behaviour of an automaton. The expression may be written using arbitrary operators.

(2) The original expression may be transformed in some or other way. Then the problems of equivalence of expressions, their canonical representation and reduction of complexity arise.

(3) When we choose the modular approach, the decomposition of expressions, corresponding to the decomposition of automata, is desirable.

(4) The proper synthesis procedure applies to a regular expression and yields the transition table (or equivalent description) of a finite automaton. This may also involve the reduction of an automaton.

(5) The design in its entirety may be proposed to be carried out with the help of a computer. The procedures are then desirable to be of exact algorithmic character without any intuitive step.

* Cf. I. M. Havel: The theory of regular events I. Kybernetika 5 (1969), 5, 400–419.

Before the study of the application of regular expressions it is worth to mention the relationship of regular events and automata. We have seen that the event A recognized by an automaton $\mathscr{A}$ consists just of words leading from the initial state s_0 to some of the final states of $\mathscr{A}$. Similarly we may attribute to any other state s of $\mathscr{A}$ an event A_s of words leading from s to some of the final states. Now, let w be a word leading from s_0 to s (i.e. $s = \delta(s_0, w)$). You may see that A_s is just the left quotient of A by w:

$$A_s = w \setminus A .$$

Thus computing the left quotients of the event A we may come to an insight to a structure of the automaton recognizing A. This very consideration is applied to the synthesis which will be described in 9.1.

The main problem is that we have to accomplish it in terms of regular expressions instead of regular events.

The synthesis of automata using this method was firstly described by Brzozowski [8] and independently by Spivak [114]–[116] (cf. also [104]). An almost exhaustive presentation of the application of regular expressions to this purpose is comprised in the monography prepared by Brzozowski [11]. The manuscript of this monography was the main source for brief explanations in Sec. 7–9.

In Sec. 10 the decomposition problem is mentioned and in the last Sec. 11 the classification of events by their special properties and by their complexity (star height) is discussed.

6.3

In the following we shall not use the concept of abstract Kleenean algebra (see Part I of this paper), but only its special case: the algebra of regular events, and the appropriate language of regular expressions. The alphabet will be $X := \{x_1, \ldots, x_k\}$ (arbitrary but fixed). The language of regular expressions is supposed to be extended by the complement and intersection and also by any other useful regularity preserving operation. Two expressions α and β will be called *equivalent* (written $\alpha = \beta$) iff they denote the same event, i.e. iff $|\alpha| = |\beta|$. Similarly we shall use the notation $\alpha \leqq \beta$ in the case when $|\alpha| \subseteq |\beta|$.

7. DERIVATIVES OF REGULAR EXPRESSIONS

7.1

The derivatives of expressions provide a very powerful tool for the study of regular expressions: they correspond to the left quotient of events. As an operation over regular expressions they were introduced by Brzozowski [8].

First we need a special operator recognizing whether λ belongs to some event A. Let us define

$$\varrho(A) := \begin{cases} \Lambda & \text{if } \lambda \in A\,, \\ \emptyset & \text{otherwise}\,. \end{cases}$$

We shall define a similar operator for regnlar expressions resursively as follows:

1. $$\varrho(\emptyset) = \varrho(x_i) = \emptyset \quad (i = 1, \ldots, k)\,,$$
$$\varrho(\Lambda) = \Lambda\,,$$

2. $$\varrho(\alpha \cup \beta) = \varrho(\alpha) \cup \varrho(\beta)\,,$$
$$\varrho(\alpha\beta) = \varrho(\alpha \cap \beta) = \varrho(\alpha) \cap \varrho(\beta)\,,$$
$$\varrho(\alpha^*) = \Lambda\,,$$
$$\varrho(\bar{\alpha}) = \Lambda \cap \overline{\varrho(\alpha)} = \Lambda - \varrho(\alpha)\,.$$

Easily we may verify that

$$|\varrho(\alpha)| = \varrho(|\alpha|)\,.$$

The definition of a *derivative with respect to the symbol* $x_i \in X$ is also recursive:

1. $$\partial_{x_i} x_i = \Lambda\,,$$
$$\partial_{x_i}\emptyset = \partial_{x_i}\Lambda = \partial_{x_i} x_j = \emptyset \quad (i \neq j)\,,$$

2. $$\partial_{x_i}(\alpha \cup \beta) = \partial_{x_i}\alpha \cup \partial_{x_i}\beta\,,$$
$$\partial_{x_i}(\alpha \cap \beta) = \partial_{x_i}\alpha \cap \partial_{x_i}\beta\,,$$
$$\partial_{x_i}(\alpha\beta) = (\partial_{x_i}\alpha)\,\beta \cup \varrho(\alpha)\,\partial_{x_i}\beta\,,$$
$$\partial_{x_i}\alpha^* = (\partial_{x_i}\alpha)\,\alpha^*\,,$$
$$\partial_{x_i}\bar{\alpha} = \overline{\partial_{x_i}\alpha}\,.$$

We extend the definition of the derivative to cover words of arbitrary length:

3. $$\partial_\Lambda \alpha = \alpha\,.$$

4. Let $w \in X^*$, $w = ux_i$. Then $\partial_w\alpha = \partial_{x_i}(\partial_u\alpha)$ (it is nothing else than the repeated first derivative).

Example (over $\{0, 1\}$). Let $\alpha = 1(00^* \cup 1)^* 1$. Then

$$\begin{aligned}
\partial_0\alpha &= \emptyset\,,\\
\partial_1\alpha &= (00^* \cup 1)^*\,1\,,\\
\partial_{10}\alpha &= (\partial_0(00^* \cup 1))\,(00^* \cup 1)^*\,1 \cup \emptyset = 0^*(00^* \cup 1)^*\,1\,,\\
\partial_{11}\alpha &= (00^* \cup 1)^*\,1 \cup \Lambda\,.
\end{aligned}$$

An important fact is that derivatives denote the left quotients of events (2.2d)):

$$|\partial_w\alpha| = w \setminus |\alpha|$$

(as a matter of fact they were introduced just for this purpose).

7.2

Knowing the relationship between the left quotients of an event and the states of a finite automaton we may expect the total number of quotients of a given event being finite. In other words any regular expression may have only a finite number of non-equivalent derivatives.

We may formulate a somewhat stronger result. Let us call two expressions α and β *similar* iff they are equivalent and their equivalence can be proved without using more than the associative, commutative and idempotent laws for union, and the following properties of $\emptyset$ and Λ: $\alpha \cup \emptyset = \alpha$, $\alpha\emptyset = \emptyset$ and $\alpha\Lambda = \Lambda\alpha = \alpha$ (i.e. the axioms A1 – A3, A7 – A9).

Theorem. *Any regular expression may have only a finite number of dissimilar derivatives.*

The proof can be accomplished by induction over the number of regular operators: (Let us denote the number of dissimilar derivatives of α by d_α.)

1. $$d_\emptyset = 1\,; \quad d_\Lambda = 2\,; \quad d_{x_i} = 3 \quad (i = 1, \ldots, k)\,;$$

2. $$d_{\alpha\cup\beta} \leqq d_\alpha d_\beta\,; \quad d_{\alpha\cap\beta} \leqq d_\alpha d_\beta\,;$$
$$d_{\alpha\beta} \leqq d_\alpha \,.\, 2^{d_\beta}\,; \quad d_{\alpha^*} \leqq 2^{d_\alpha} - 1\,; \quad d_{\bar\alpha} = d_\alpha\,.$$

Let us define the *derivative closure* $\mathscr{D}(\alpha)$ (in Spivaks's papers called a *basis*) of α as the least set of expressions (up to the similarity) satisfying

(1) $$\alpha \in \mathscr{D}(\alpha)\,;$$

(2) $$\beta \in \mathscr{D}(\alpha) \Rightarrow \partial_{x_i}\beta \in \mathscr{D}(\alpha)\,, \quad i = 1, \ldots, k\,.$$

We have seen that $\mathscr{D}(\alpha)$ is finite. Moreover, it always can be constructed by taking derivatives with respect to words of the length not exceeding $d_\alpha - 1$.

As a matter of fact the construction of $\mathscr{D}(\alpha)$ is nothing else than the construction of finite automaton recognizing $|\alpha|$. We shall return to this in 9.1.

7.3

It may be easily verified that any event can be expanded in terms of its first left quotients:

$$A = \bigcup_{i=1}^{k} \{x_i\} \, . \, x_i \setminus A \ \cup \varrho(A) \, .$$

In the case of regular events we may rewrite it in terms of regular expressions:

$$\alpha = \bigcup_{i=1}^{k} x_i \, \partial_i \alpha \ \cup \varrho(\alpha)$$

(we write ∂_i instead of ∂_{x_i}). Let $\mathscr{D}(\alpha_1) = \{\alpha_1, \ldots, \alpha_n\}$ be the derivative closure of α_1. The elements of $\mathscr{D}(\alpha_1)$ then satisfy the following system of equations:

$$\begin{aligned}
\xi_1 &= x_1 \xi_{11} \cup x_2 \xi_{12} \cup \ldots \cup x_k \xi_{1k} \cup \varrho_1 \, , \\
\xi_2 &= x_1 \xi_{21} \cup x_2 \xi_{22} \cup \ldots \cup x_k \xi_{2k} \cup \varrho_2 \, , \\
&\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots \\
\xi_n &= x_1 \xi_{n1} \cup x_2 \xi_{n2} \cup \ldots \cup x_k \xi_{nk} \cup \varrho_n
\end{aligned}$$

where

$$\xi_{ij} \in \{\xi_1, \ldots, \xi_n\} \, , \quad 1 \leqq i \leqq n \, , \quad 1 \leqq j \leqq k \, ;$$

$\{\xi_1, \ldots, \xi_n\}$ is a set of indeterminate variables and ϱ_i is Λ or $\emptyset$ in accordance with the value of $\varrho(\alpha_i)$.

These equations will be called the *derivative equations* of α_1 or also the *equational characterization of* α_1 (equations of basis according to Spivak).

Every regular expression can be equationally characterized. Conversely, as we shall see now, it can be computed (up to the equivalence) from its system of derivative equations.

Generally it holds that any set of equations of the form

$$\xi_i = \bigcup_{j=1}^{n} \beta_{ij} \xi_j \cup \gamma_i \, , \quad i = 1, \ldots, n$$

with $\varrho(\beta_{ij}) = \emptyset$, $(1 \leqq i, j \leqq n)$ can be solved for ξ_i uniquely. This solution proceeds by repeated application of the rule R from 3.1 and substitution.

The special case of such equations is clearly the set of derivative equations of an expression α. After the solution* we obtain an expression α' equivalent to α but containing only the $\cup$, . and $*$ operators. This implies that any regular expression can be transformed to the so called *restricted regular expression.*

An important result also follows from the above explanations:

Theorem. *An event is regular iff it has a finite number of distinct left (or right) quotients.*

Equations over regular expressions are also studied in [78], [79], [87], [88], [117].

8. THE EQUIVALENCE PROBLEM

8.1

The testing of equivalence of regular expressions is a very important problem. It is always solvable trivially by constructing the corresponding automata, reducing them and testing the isomorphism. There exist, however, also other methods. The first one described here uses the derivatives of expressions and is essentially related to the mentioned trivial method (see [11]).

Besides the equivalence we shall also test the inclusion (which otherwise can be handled also as an equivalence: $\alpha \subseteq \beta$ iff $\alpha \cup \beta = \beta$). The method is based on the following four statements, which can be easily verified:

(1) $\alpha = \emptyset$ iff no derivative of α contains Λ ,

(2) $\alpha = X^*$ iff all derivatives of α contain Λ ,

(3) $\alpha \subseteq \beta$ iff $\alpha \cup \bar{\beta} = X^*$,

(4) $\alpha = \beta$ iff $\alpha \triangle \beta = \emptyset$.

Thus the testing of equivalence or inclusion of α and β consists in

(a) finding all derivatives, i.e. constructing the derivative closure of the expression $\alpha \triangle \beta$ or $\alpha \cup \bar{\beta}$ respectively (some redundancy in the number of derivatives is allowable; the finiteness of the procedure is not affected even if we distinguish only dissimilar derivatives) and

(b) finding the values of the function ϱ for all the derivatives (the useful tool for this is the definition of ϱ).

A more convenient method can be used when we have to test the equivalence of two derivatives of some expression and when the derivative equations are known. The method resembles the testing of indistinguishability of two states when reducing the transition table of automaton. With help of this method the so called reduced

* The example of the solution of derivative equations is in 9.2.

derivative equations may be obtained. We have the important result that every regular expression can be characterized uniquely by a set of derivative equations in which all derivatives are distinct and which can be constructed effectively.

8.2

Now we shall mention some further methods of deciding the equivalence, which are not based on finding the derivatives.

(a) The most exact method is the *proving of equations* as identities of abstract Kleenean algebra. Because of the completeness of the axiom system described in 3.1, this method is by all means general. Its disadvantage is that it usually claims for some mathematical skill and can be scarcely automatized.

(b) Another technique of deriving equalities is the proof by *reparsing* [46]. The equivalence is proved in terms of regular events as word sets. Let us demonstrate it on the example:

We have to prove that e.g. $\alpha(\beta\alpha)^* = (\alpha\beta)^* \alpha$.

An arbitrary word $w \in |\alpha(\beta\alpha)^*|$ may be written as $w = u_0(v_1u_1)(v_2u_2), \ldots, (v_nu_n)$ where $u_i \in |\alpha|$ and $v_i \in |\beta|$. Using associative law for concatenation we obtain $w = (u_0v_1)(u_1v_2), \ldots, (u_{n-1}v_n)\, u_n \in |(\alpha\beta)^* \alpha|$. Hence $\alpha(\beta\alpha)^* \leqq (\alpha\beta)^* \alpha$. The converse inclusion can be obtained similarly. However, this technique is not systematic and can be used only for special cases.

(c) The last method we shall mention is described by McNaughton [46]. It is based on the fact that regular expressions are easily characterized by the so called *transition graphs*. These graphs resemble the state graphs of automata with the exception that they are non-deterministic and their branches may be labelled also by the empty word λ. The equivalence of expressions are proved by simple transformations of corresponding graphs, as long as the isomorphism of graphs is obvious. Alternatively the graphs may be processed by easy mechanical procedure [31], [32]. The advantage of this technique consists in the fact that the graph preserves the structure of the corresponding expression and hence the method is very insightful.

8.3

The most difficult problem in the theory of regular expressions is defining some *canonical representation* of regular events. As it is known from Boolean algebra the existence of canonical or normal formulas is very useful. Till now no general canonical form of regular expressions has been found. The only results are concerned with some classes of regular events as e.g. definite events [7] and chain events [85]; for ultimate and symmetric definite events [54], star events [10] and comet events [13] the representation is canonical only to a certain level (the mentioned classes of events are described in 11.1). The same holds for the concatenative canonical expansion of all (nonempty) events studied by Paz and Peleg [55] (cf. also [13]).

9.1

We are coming to our principal problem of construction of finite automata from their regular expressions. We shall, however, see that we have the greatest part of work already done.

First let us briefly recapitulate our results. We have seen (6.2) that we may associate with each state of a given automaton an event consisting of all words leading from this state to some of the final states, and, moreover, this event is the left quotient of the event associate with the initial state. The latter is the event recognized by an automaton. Because we put no restriction on the admittable input words we must consider all possible left quotients of the event (the number of which is finite, indeed). A transition from one state to another corresponds to any new symbol coming to the input. Eventually the events associated with the final states must contain λ because it trivially leads from a final state to a final state, namely to the same state.

But we know (7.1, 7.2) how to construct all derivatives of an expression (the derivative closure). Besides, all information on transitions in an automaton is involved in the system of derivative equations, as well as the information about the fact which expressions contain the empty word.

The synthesis algorithm is then as follows:

(1) Construct the system of derivative equations to the given expression α.

(2) Interpret each derivative as a state of an automaton, interpret α as the initial state.

(3) Define the transition function $\delta(s_i, x_j)$ by the derivative $\partial_{x_j}\alpha_i$, where α_i corresponds to s_i.

(4) The derivatives containing λ are the final states.

The described synthesis algorithm, due to Brzozowski and Spivak, has two important properties: (a) it can be applied to any regular expression without restriction on the operators used, and (b) if we use only mutually non-equivalent derivatives the resulting automaton is reduced.

Let us illustrate the synthesis algorithm by an example.

Example. Given $\alpha = (0 \cup 10)^* \, 110^* =: \alpha_1$. The derivatives of α:

$$\partial_0\alpha_1 = (0 \cup 10)^* \, 110^* = \alpha_1 \,,$$

$$\partial_1\alpha_1 = 0(0 \cup 10)^* \, 110^* \cup 10^* =: \alpha_2 \,,$$

$$\partial_0 x_2 = (0 \cup 10)^* \, 110^* = \alpha_1$$

$$\partial_0\alpha_2 = 0^* =: \alpha_3 \quad (\Lambda \in \alpha_3),$$

$$\partial_0\alpha_3 = 0^* = \alpha_3 ,$$

$$\partial_1\alpha_3 = \emptyset =: \alpha_4 ,$$

$$\partial_0\alpha_4 = \partial_1\alpha_4 = \alpha_4 .$$

The derivative equations:

$$\alpha_1 = 0\alpha_1 \cup 1\alpha_2 ,$$

$$\alpha_2 = 0\alpha_1 \cup 1\alpha_3 ,$$

$$\alpha_3 = 0\alpha_3 \cup 1\alpha_4 \cup \Lambda ,$$

$$\alpha_4 = 0\alpha_4 \cup 1\alpha_4 .$$

The state table is in Table 1. The state graph is on Fig. 1.

Table 1.

	0	1
$\to 1$	1	2
2	1	3
$\leftarrow 3$	3	4
4	4	4

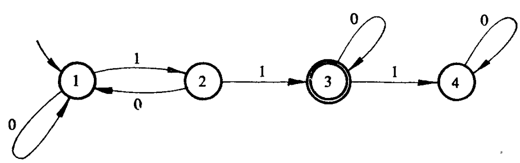


Fig. 1.

9.2

The converse problem is the problem of analysis. This can be accomplished by solving the derivative equations written in accordance with the state table. We shall demonstrate it on the same example as above.

Example. The fourth derivative equation is $\alpha_4 = (0 \cup 1)\,\alpha_4$, using the rule R we have $\alpha_4 = \emptyset$. We substitute it to the third equation (and again use the rule R):

$$\alpha_3 = 0\alpha_3 \cup \emptyset \cup \Lambda \Rightarrow \alpha_3 = 0^*\Lambda = 0^*$$

similarly

$$\alpha_2 = 0\alpha_1 \cup 10^*$$

and consequently

$$\alpha_1 = 0\alpha_1 \cup 1(0\alpha_1 \cup 10^*) = (0 \cup 10)\,\alpha_1 \cup 110^* \Rightarrow$$
$$\Rightarrow \alpha_1 = (0 \cup 10)^*\, 110^* = \alpha\,.$$

In this case we have obtained exactly the same expression from which we have started.

The analysis procedure was firstly described by McNaughton and Yamada [50]. Their algorithm starts from the state graph. Similar algorithm was also presented by Glushkov [91]–[93]. The analysis by solving derivative equations was described in [8] and [117].

The analysis procedure can be also usefully formalized. For example we may start from the transition matrix, the size of which is step by step reduced by means of rules of the type

$$\alpha'_{ij} = \alpha_{ij} \cup \bigcup_{\substack{k \\ k \neq j}} \alpha_{ik}\alpha^*_{kk}\alpha_{kj}\,.$$

The result is the desired regular expression (cf. [101]).

9.3

It should be noted that there exists an alternative approach to finite automata in terms of regular events, which is to some extent dual to the described one. Let us call the set

$$A_s := \{w \mid \delta(s_0, w) = s\}$$

the *destination set* [11]. It is precisely the set of all words leading from the initial state to a given state. Now we associate with each state its destination set instead of the quotient set as before. In the place of derivative equations we obtain now the so called *destination equations* of the following form

$$\alpha_i = \bigcup_{j=1}^{k} \alpha_{ij} x_j \cup \varrho_i\,, \quad i = 1, \ldots, n\,,$$

where $\varrho_1 = \Lambda$, $\varrho_i = \emptyset$ for $i = 2, \ldots, n$.

Let $\mathscr{A}$ be a finite automaton recognizing a regular event A. If its destination equations are rewritten to the form of right quotients and reversed, we obtain derivative equations for the reverse event $\overleftarrow{A}$. For synthesis techniques from reverse regular expression see also [20].

9.4

Beside the synthesis procedure using derivatives there exists also a different technique called the *position algorithm*. It was developed by McNaughton and Yamada

 [50] and in a modified version presented by Glushkov [90], [92], [93]. We shall not describe the position algorithm here, but show only its application by the example (it is the same example as before).

Example. Given $\alpha = (0 \cup 10)^* \, 110^*$. The so called positions are marked as follows:

$$_\lambda(0_1 \cup 1_1 0_2)^* \, 1_2 1_3 0_3^* .$$

The initial positions are: $0_1, 1_1, 1_2$,

the terminal positions are: $1_3, 0_3$.

We construct the set of transitions admitted in the event:

$$\begin{array}{lll} (\lambda, 0_1), & (\lambda, 1_1), & (\lambda, 1_2), \\ (0_1, 0_1), & (0_1, 1_1), & (0_1, 1_2), \\ (0_2, 0_1), & (0_2, 1_1), & (0_2, 1_2), \\ (0_3, 0_3) & & \\ (1_1, 0_2) & & \\ (1_2, 1_3) & & \\ (1_3, 0_3) . & & \end{array}$$

Table 2.

	0	1	
$\to \lambda$	0_1	$1_1, 1_2$	(1)
0_1	0_1	$1_1, 1_2$	(1)
$1_1, 1_2$	0_2	1_3	(2)
0_2	0_1	$1_1, 1_2$	(1)
$\leftarrow 1_3$	0_3	$\emptyset$	(3)
$\leftarrow 0_3$	0_3	$\emptyset$	(3)
$\emptyset$	$\emptyset$	$\emptyset$	(4)

$\to 1$	1	2
2	1	3
$\leftarrow 3$	3	4
4	4	4

The states of the resulting automaton correspond to sets of positions; the initial state to $\{\lambda\}$, the final states to the sets containing some terminal position. The obtained transition table and its reduced form are in Table 2.

The position algorithm does not necessarily yield the reduced transition table. Besides, it can be applied only to restricted regular expressions (using only $\cup$, ., *). On the other hand it is easily adaptable for computer processing ([60]).

The following example shows a similar but more objective procedure using the transition graph (cf. 8.2 (c)):

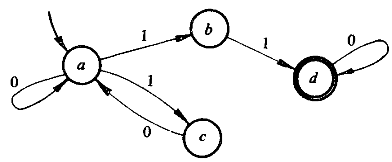


Fig. 2.

Example. The expression $\alpha = (0 \cup 10)^* \, 110^*$ may be characterized by the transition graph on Fig. 2.

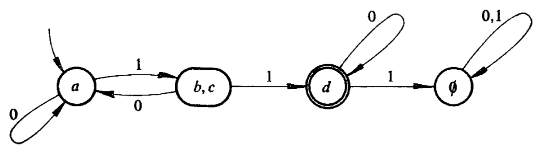


Fig. 3.

The states of the automaton are represented by suitably chosen subsets of states of this graph (Fig. 3).

10. COMPOSITION AND DECOMPOSITION OF AUTOMATA

10.1

The composition and decomposition are usually understood as structural operations over automata and are studied within the framework of the algebraic theory of automata (cf. [32]). In this section we shall briefly mention another approach to the problem, emphasizing the behavioural point of view. When designing large automata it is desirable to split the problem from the very beginning and to accomplish the synthesis separately. The desired automaton is then a system composed of several smaller component automata. The system as a whole as well as its components are studied in terms of their behavioural descriptions rather than their transition functions. Here the theory of regular expressions may be profitable.

We shall show the behavioural treatment of composition on the case of Boolean operations and basic operations of Kleenean algebra described by Copi, Elgot and Wright [19] and Arden [3] (see also [5], [6], [15], [25], [52]).

In the case of union, intersection and complement the compositions are quite easy. The automata* are supposed to have one binary output indicating the occurrence of recognized event by the value 1. We have at our disposal the special small combinational automata OR gate, AND gate and inverter. The compositions are shown on Fig. 4a, b, c. (The first two compositions are sometimes called *parallel compositions.*) Similarly we may obtain all other operations of Boolean algebra.

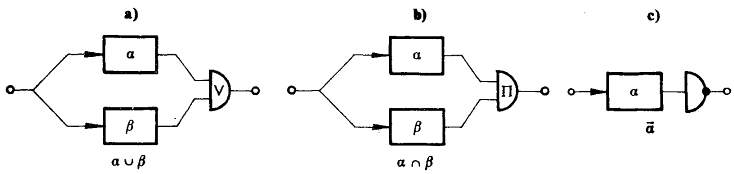


Fig. 4.

The situation is more complicated in the case of concatenation and star. We must assume the following:

(1) automata are provided by a special so called *starting input.* The occurrence of a value 1 on this input brings the automaton to the initial state and makes it sensitive to input signals;

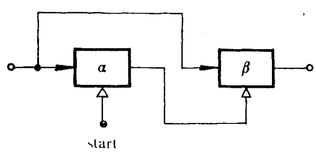


Fig. 5.

(2) automata have a special behaviour that Yamada [81] calls the *disjunctively linear behaviour*: roughly speaking, if the starting input is excited more times, the automaton is capable to recognize independently all events formed by "shifting" the original one in accordance with the starting signals. There are fortunately no difficulties with construction of such special automata in practice.

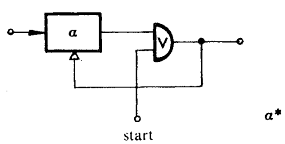


Fig. 6.

* The automaton is here understood as a realization of an abstract automaton (e.g. by means of a logical net) rather than the abstract automaton itself.

Under the assumptions stated above we may describe the following compositions – the *concatenation* (Fig. 5) and the *star composition* (introduction of the feedback – Fig. 6):

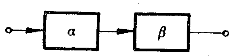
There is an important fact that the union, concatenation and star composition preserve the disjunctive linearity of automata (cf. [15], [81]).

It is possible to describe similar compositions for some other regularity preserving operations but corresponding constructions are rather intricate and a significant change to the structure of automata is usually required.

10.2

On the other hand there is not known any general operation over regular expressions corresponding to the simple serial composition (written $\alpha \circ \beta$) on Fig. 7

Fig. 7.



$\alpha \circ \beta$

(where both input and output alphabets are supposed to be $\{0, 1\}$). This composition may be of great practical interest as it is seen from the following example:

$$\mathsf{C}_n(\alpha) := \alpha \circ [(0^* 1)^n]^* \, 0^*$$

(the number of occurrences of α equals to $0, n, 2n, 3n, \ldots$ – the generalized counter). We also may generalize this composition to the serio-parallel one (written $(\alpha, \beta) \circ \gamma$ – Fig. 8): (the input alphabet of $\mathscr{A}_\gamma$ is supposed to be $\{0, 1, 2, \widehat{12}\}$).

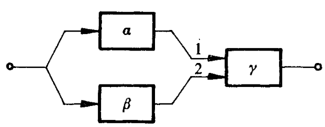


Fig. 8.

$(\alpha, \beta) \circ \gamma$

Example.

$$(\alpha, \beta) \circ (0^* \, 10^* \, 2)^* \, 0^*$$

(the occurence of α and β alternates).

Boolean compositions mentioned above are the special cases:

$$\alpha \cup \beta = (\alpha, \beta) \circ X^*(1 \cup 2) \,,$$
$$\alpha \cap \beta = (\alpha, \beta) \circ X^* \, \widehat{12} \,,$$
$$\bar{\alpha} = \alpha \circ X^* \, 0 \,.$$

10.3

The problem of composition may be attacked also from the opposite side: investigating the decomposition of regular events (or expressions). The known results concern only the Kleenean operations and we mention only three recent related papers: Paz and Peleg [55] study some special concatenative decompositions, Brzozowski [10] inverses the star operation and computes the so called roots of star events (cf. 11.1 (8)) and Brzozowski and Cohen [13] factor out the star component from a given event.

11. CLASSIFICATION OF REGULAR EVENTS

11.1

We shall close our exposition on regular events by several brief remarks on their classification.

Very often it is helpful to divide our field of study to smaller parts where our problems may be solved by easier methods. Moreover, in automata theory such differentiated approach may bring valuable insights to automata. As for regular events, it is very convenient to study several classes of events separately. Usually specific classes of events correspond to specific classes of automata. We shall present some illustrative examples of such classes below. The diagram of hierarchy of these classes is on Fig. 9. (Lines denote the inclusion of classes of events; in cases denoted by the asterisk the inclusion does not apply to the event Λ.)

(1) *Elementary events* or atoms and trivial events $\emptyset$ and Λ were already mentioned.

(2) *Finite events* (also called initial events) are a very simple class of events. Algebraically they form a closure $[X; \cup, ., \emptyset, \Lambda]$ of X without the use of the star. Automata recognizing finite events "finish their work" after a finite time.

(3) *Combinational events* are infinite but of a very degenerate structure. They are of the form X^*A, where $A \subseteq X$. They are realized by combinational circuits and it is usual to describe them by means of Boolean expressions defining A.

(4) *Definite events* and corresponding definite automata have been described by Perles, Rabin and Shamir [56]. Definite events are of the form $A \cup X^*B$, where A, B are finite. (Special cases: for $B = \emptyset$ finite events and for $A = \emptyset$ the so called *non-initial definite events.*) The canonical expressions for definite events have been described by Brzozowski [7]. Definite events can be further classified by the maximal length of words in B. The automata recognizing definite events have finite depth of the memory and can be always realized by the delay (register) and combinational circuit. Some authors ([7], [30]) use also the concept of *reverse definite events* $A \cup BX^*$ (A, B finite) and more general events of the type

$$A \cup \bigcup_i B_i X^* C_i \quad (A_i, B_i, C_i \text{ finite}).$$

We may go further with this generalization and define the

(5) *Generalized definite events* as the closure of definite events under concatenation or as the closure $[X; \cup, ., \emptyset, \Lambda, \Omega]$ where $\Omega = X^*$ is the universal event. This class contains e.g. events corresponding to the occurrence of some given sequences: X^*AX^* (A finite).

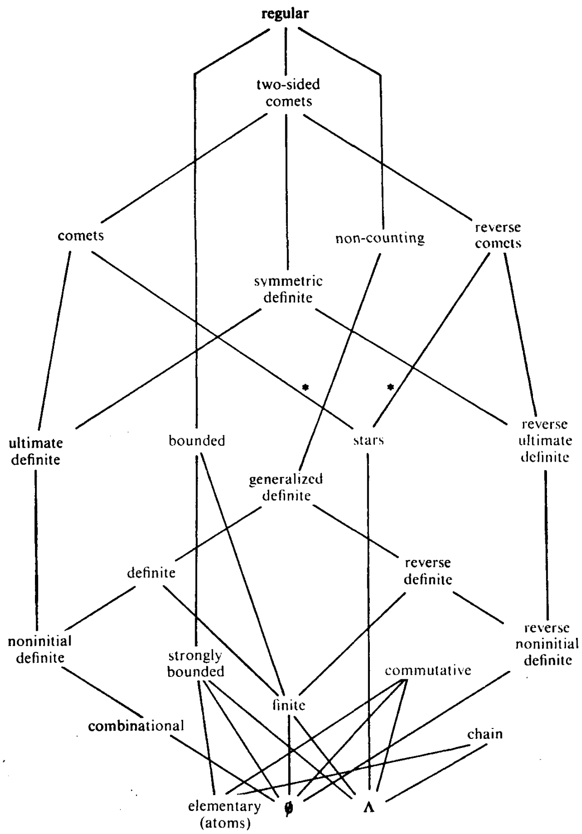


Fig. 9.

(6) *Non-counting events* are the most interesting in the above hierarchy. Algebraically they are the closure $[X; \cup, \cap, ., -, \emptyset]$ and thus can be described by extended regular expressions without star (they are also called the *star-free events*). These events are exhaustively studied by Papert and McNaughton [53] (see also [18], [67], [68]). The non-counting events admit equivalent characterizations from many different aspects:

(a) They satisfy the following condition (for A):

$$\exists n\ \forall u, v, w \in X^*, \quad uv^{n+1}w \in A \Leftrightarrow uv^n w \in A$$

(this means that they cannot count modulo an integer);

(b) they characterize the ability of an automaton to provide local testing of input words and recognize their internal order (and nothing more);

(c) they characterize precisely the class of automata, the behaviour of which can be described within the first-order logic.

(d) they describe the so called group-free or permutation-free automata, the transition function of which permits only trivial permutations inside of all subsets of the state set;

(e) they correspond to the threshold (neural) nets with only small and excitory feedback loops.

(7) *Ultimate-definite* and *symmetric definite events*, studied by Paz and Peleg [54], are another generalization of definite (precisely noninitial definite) events. Ultimate-definite events are of the type X^*A (A arbitrary) and symmetric definite events are AX^*B (A, B arbitrary).

(8) *Star events*, studied by Brzozowski [10], are events of the type A^* for some regular A. The property of being star event may not be directly seen from the corresponding regular expression. But all three following properties of an expression α are equivalent with the property of α being star:

(a) $$\alpha = \alpha^*,$$

(b) $$\alpha = \alpha^2,$$

(c) $$\Lambda \subseteq \alpha \quad \text{and} \quad \forall w \in X^*, \quad \Lambda \leqq \partial_w\alpha \Leftrightarrow \alpha \subseteq \partial_w\alpha .$$

There is an important fact that for every star event A an expression for its minimal *root* V can be found, viz. $V = (A - \Lambda) - (A - \Lambda)^2$ (V is said to be a root of A iff $V^* = A$). In more general case A need not be a star but it may contain subsets which are stars. An algorithm for finding such subsets is presented in [13].

(9) *Comet events* [13] can be decomposed to a concatenation A^*B of a star event A^* (not Λ) and an arbitrary the so called *tail event* B.

(10) *Commutative events* ([43]–[45], [109]): An event A is *commutative* iff with every word $w \in A$ it contains also all possible permutations of the letter tokens of w. In other words

$$uvv'w \in A \Leftrightarrow uv'vw \in A .$$

Very often the term *commutative closure* $c(A)$ of a regular event A is used. Obviously $c(A)$ need not be regular. In [45] a method is presented by which, given any $c(A)$ (by means of a regular expression for A) it can be decided whether or not $c(A)$ is regular and in this case there is an algorithm for finding the automaton.

(11) *Bounded events* [28]. An event A is bounded iff there exists a finite number of words $w_1, \ldots, w_s$ such that $A \subseteq w_1^*, \ldots, w_s^*$. The class of all regular bounded events is the smallest class which contains finite events, all events w^* ($w \in X^*$) and is closed with respect to union and dot (they are the compound closure $[[[X; ., \emptyset]; *]; \cup, .]$).

An event A is *strongly bounded* iff there exists a word $w \in X^*$ such that $A \subseteq w^*$. Strongly bounded events are in [28] called commutative (this should not be confused with the commutative events mentioned above) because of their property:

$$\forall u, v \in A , \quad uv = vu .$$

(12) *Chain events* are of the form

$$A_0 x_{i_1} A_1 x_{i_2}, \ldots, A_{r-1} x_{i_r} A_r , \quad r \geqq 0 ,$$

where A_j are star events the roots of which are subsets of X and $X_{ij} \in X$. Yoeli [85] has found unique canonical forms for two special subclasses of chain events (*normal* where $x_{i_j} \notin A_{j-1} \cup A_j$ and *regular* where $x_{i_1} = x_{i_2} = \ldots = x_{i_r}$).

Note. The definitions of some classes are valid not only for regular events (particularly (7), (9)–(11)). However, in the diagram on Fig. 9 we consider only their regular subclasses.

11.2

Now we shall mention one special parameter a enabling classification of regular events by some measure of complexity.

The *star height* $h(\alpha)$ of a regular expression α is the length of the longest sequence of stars in the expression, such that each star is within the scope of the preceding star in the sequence. More precise definition is the recursive one:

1. $$h(\emptyset) = h(\Lambda) = h(x_i) = 0 , \quad x_i \in X ,$$
2. $$h(\alpha \cup \beta) = h(\alpha\beta) = \max(h(\alpha), h(\beta)) ,$$
$$h(\alpha^*) = h(\alpha) + 1$$

(we are restricted here to regular expressions without complement and intersection).

Then the star height (or the *loop complexity*) $h(A)$ of a regular event A is defined as

$$h(A) = \min \{h(\alpha) \mid |\alpha| = A\} .$$

Roughly speaking, the star height of an event characterizes the complexity of the associated state graph by the number of subordinate loops in the graph.

Whereas the star height of regular expressions can be easily determined, the star height of regular events is difficult to compute. Eggan [24] proved the existence of events of arbitrarily large star height provided one is willing to extend the alphabet arbitrarily. Later Dejean and Schutzenberger [23] proved the same over the fixed alphabet $\{0, 1\}$ (see also [47]). The corresponding state graph is on Fig. 10.

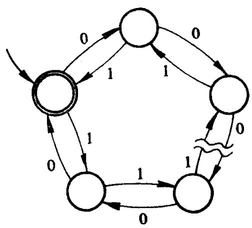


Fig. 10.

Till the present time no algorithm of computing the star height of regular events in general is known. Beside some particular cases solved by McNaughton [47] (see also 3.5, eq. (7)–(10)), the only results are algorithms for two closely related classes of regular events: (1) regular events with finite intersection property (the intersection of every pair of distinct derivatives of an event is finite): Cohen and Brzozowski [17]; (2) pure-group events (each input of the corresponding automaton yields a permutation of states): McNaughton [49].

An open problem [53] is that of properties of the so called *generalized star height* defined on the basis of extended regular expressions (with the complement and intersection). The only known facts are (a) that events of generalized star heights 0 exist: they are precisely the non-counting events, and (b) that also events of generalized star height 1 exist (00)* for example).

CONCLUSION

We have tried to present a general discussion of the theory of regular events and regular expressions. It was not, however, possible to cover all problems and results connected with this theory. Some of the omitted topics are the following: application of the synthesis procedure to Mealy automata and multiple output automata [8], [11], [92], [93], [118] and to incompletely specified automata [106], [107]; computer programs dealing with regular expressions [57], [60], [76]; the metric space of events [89]; regular expressions for linear nets [9] and for asynchronous nets [99]; the input-output translation and generation of output regular expressions [27], [40], [41], [83], [97]; sequential relations, experiments and transductions [35], [118], [119]; various generalizations [2], [12], [33], [34], [37], [43]–[45], [48], [66], [74], [75], [105], [108], events recognized by probabilistic automata [72], [77], [96]; relationship to other behavioural languages for automata [23], [36], [53]; relationship to algebraic theory of automata [32]–[34], [53], [67]–[69], [75], to mathematical theory of languages, to graph theory, etc.

(Received July 31st, 1968.)

REFERENCES

Abbreviations of some titles of journals:

C, EC	Transactions of the IEEE (IRE) on Computers, on Electronic Computers
IC	Information and Control
JACM	Journal of the Association for Computing Machinery
AT	Автоматика и телемеханика
ДАН	Доклады Академии наук СССР
Киб.	Кибернетика (АН Украинской ССР)
УМЖ	Украинский математический журнал

[1] S. Aanderaa: On the algebra of regular expressions. Appl. Math., Harvard University, Jan. 1965, 1–18.

[2] V. Amar, G. Putzolu: Generalizations of regular events. IC *8* (1965), 1, 56–63.

[3] D. N. Arden: Delayed logic and finite state machines. In: Theory of Computing Machine Design, Univ. of Michigan Press, Ann Arbor, 1960, 1–35.

[4] J. A. Brzozowski: Properties of regular expressions and state diagrams. Tech. Rept. No 15, Princeton University, 1962.

[5] J. A. Brzozowski: Regular expression techniques for sequential circuits. Ph. D. dissertation, Princeton University 1962.

[6] J. A. Brzozowski: A survey of regular expressions and their applications. EC *11* (1962), 324–335.

[7] J. A. Brzozowski: Canonical regular expressions and minimal state graphs for definite events. In: Mathematical Theory of Automata, Polytechnic Press, 1963.

[8] J. A. Brzozowski: Derivatives of regular expressions. JACM *11* (1964), 4, 481–494.

[9] J. A. Brzozowski: Regular expressions for linear sequential circuits. EC *14* (1965), 2, 148–156.

[10] J. A. Brzozowski: Roots of star events. JACM *14* (1967), 3, 466–77.

[11] J. A. Brzozowski: Regular theory of sequential machines. (Manuscript) Univ. of Waterloo, 1968.

[12] J. A. Brzozowski: Regular-like expressions for some irregular languages. In: IEEE 8th. Ann. Symp. on Switching and Automata Theory, 1968.

[13] J. A. Brzozowski and R. Cohen: On decompositions of regular events. In: IEEE 7th Ann. Symp. Switching and Automata Theory, 1967, 255–264.

[14] J. A. Brzozowski and E. J. McCluskey: Signal flow graph techniques for sequential circuit state diagrams. EC *12* (1963), 67–76.

[15] J. A. Brzozowski and Poage: On the construction of sequential machine from regular expressions. EC *12* (1963), 4, 402—403.

[16] R. Cohen: Transition graphs and the star heights problem. In: IEEE 8th Ann. Symp. on Switching and Automata Theory, 1968.

[17] R. Cohen and J. A. Brzozowski: On the star height of regular events. In: IEEE 7th Ann. Symp. Switching and Automata theory, 1967, 265—280.

[18] R. Cohen and J. A. Brzozowski: On star-free events. Hawaii Int. Conf. on System Sciences, 1968.

[19] I. Copi, C. C. Elgot and J. B. Wright: Realization of events by logical nets. JACM *5* (1958), 2, 181—196.

[20] H. Allen Curtis: Polylinear sequential circuit realizations of finite automata. C *17* (1968), 3, 251—259.

[21] F. Dejean and M. P. Schutzenberger: On a question of Eggan IC *9* (1966), 23—25.

[22] L. C. Eggan: Transition graphs and the star height of regular events. Michigan Math. Journal *10* (1963), 385—397.

[23] C. C. Elgot: Decision problems of finite automata design and related arithmetic. Trans. Am. Math. Soc. *98* (1961), 1, 21—51.

[24] S. Even: Rational numbers and regular events. EC *13* (1964), 6, 740—741.

[25] T. Frey: Über die Konstruktion endlicher Automaten. Acta Math. Acad. Sci. Hungar, *15* (1964), 383—398.

[26] H. Ghiron: Rules to manipulate regular expressions of finite automata. EC *11* (1962), 574—575.

[27] S. Ginsburg and T. N. Hibbard: Solvability of Machine Mappings of regular sets to regular sets. JACM *11* (1964) *3*, 302—312.

[28] S. Ginsburg and E. H. Spanier: Bounded regular sets. Proc. Am. Math. Soc. *17* (1966), 1043—1049.

[29] S. Ginsburg and E. H. Spanier: Quotients of context-free languages. JACM *10* (1963), 4, 487—492.

[30] A. Ginzburg: About some properties of definite, reverse definite and related automata. EC *15* (1966), 5, 806—810.

[31] A. Ginzburg: A procedure for checking equality of regular expressions. JACM *14* (1967), 2, 355—362.

[32] A. Ginzburg: Algebraic theory of automata. Academic Press, New York, 1958.

[33] Y. Give'on: The theory of algebraic automata I. Univ. of Michigan, Ann Arbor, 1964.

[34] Y. Give'on: Outline for an algebraic study of event automata. Univ. of Michigan, Ann Arbor, 1964.

[35] J. N. Gray and M. A. Harrison: The theory of sequential relations, IC *9* (1966), 5, 435—468.

[36] I. M. Havel: Jazyky zápisu a zadání konečných automatů. Thesis EF ČVUT, Praha 1966.

[37] I. M. Havel: Regular expressions over generalized alphabet and design of logical nets, Kybernetika *4* (1968), 6, 516—537.

[38] D. Ion Ion: Un sistem de axiome pentru algebra evenimen telor. Studii ci cercetări mat. Acad. RPR *17* (1965), 4, 599—606.

[40] T. Kasami, K. Torii and H. Ozaki: Generalized sequential machine mapping of regular set to regular set. Electron. Commun. Japan *48* (1965), 15—24.

[41] T. Kasami, K. Torii and H. Ozaki: Translation of finite state languages by a sequential machine. Journ. Inst. Elec. Commun. Engs. Japan (1966).

[42] S. C. Kleene: Representation of events in nerve nets and finite automata. In: Automata Studies, Princeton University Press, 1956.

[43] R. Laing: Tape machined realization sof commutative-regular events. Techn. Rept. Michigan Univ. 1965.

[44] R. Laing: Realization and complexity of commutative events. Rept., Michigan Univ. 1967.
[45] R. Laing and J. B. Wright: Commutative machines. Techn. Rept. Michigan Univ. 1962.
[46] R. McNaughton: Techniques for manipulating regular expressions. In: Systems and Computer Science. University of Toronto Press, 1967.
[47] R. McNaughton: The loop complexity of regular events. Rept. M.I.T. 1966.
[48] R. McNaughton: Testing and generating infinite sequences by a finite automaton. IC *9* (1966), 5, 521—530.
[49] R. McNaughton: The loop complexity of pure-group events. IC *11* (1967), 167—176.
[50] R. McNaughton and H. Yamada: Regular expressions and state graphs for automata. EC *9* (1960), 1, 39—47.
[51] J. Myhill: Finite automata and the representation of events. WADC Tech. Rept. 1957.
[52] G. Ott and N. Feinstein: Design of sequential machines from their regular expressions. JACM *8* (1961), 4, 585—600.
[53] S. Papert and R. McNaughton: Non-counting automata. Rensselaer Polyt. Inst. (manuscript), 1967.
[54] A. Paz and B. Peleg: Ultimate-definite and symmetric-definite events and automata. JACM *12* (1965), 3, 399—410.
[55] A. Paz and B. Peleg: On concatenative decompositions of regular events. C *17* (1968), 3, 229—237.
[56] M. Perles, M. O. Rabin and E. Shamir: The theory of definite automata. EC *12* (1963), 3, 233—243.
[57] T. F. Piatkowski: Computer programs dealing with finite state machines. Techn. Rept., Michigan University, 1967.
[58] M. O. Rabin and D. Scott: Remarks on finite automata. In: Summer Inst. Symb. Logic, Cornell Univ. 1957, 106—112.
[59] M. O. Rabin and D. Scott: Finite automata and their decision problems. IBM Journal Res. Dev. *3* (1959), 2, 114—125.
[60] F. Saez Vacas: Programmation de l'algorithme de synthèse de Gloushkov pour les tables de fluence des systèmes séquentiels. Thèse à l'Ecole Nationale Supérieure de l'Aéronautique, Paris, 1966.
[61] F. Saez Vacas, E. Daclin: Sur une méthode algorithmique de synthèse d'une machine séquentielle. Automatisme *13* (1968), 10, 510—520.
[62] A. Salomaa: Theorems on the representation of events in Moore-automata. Ann. Univ. Turku, AI. Vol. 69 (1964).
[63] A. Salomaa: Axiom systems for regular expressions of finite automata. Ann. Univ. Turku, AI Vol. 75 (1964), 14—29.
[64] A. Salomaa: The complete axiom systems for the algebra of regular events. JACM *13* (1966), 1, 152—169.
[65] A. Salomaa and V. Tixier: Two complete axiom systems for the extended language of regular expressions. C *17* (1968) 7, 700—701.
[66] M. P. Schützenberger: Finite counting automata. IC *5* (1962), 91—107.
[67] M. P. Schützenberger: On finite monoids having only trivial subgroups. IC *8* (1965), 2, 190—194.
[68] M. P. Schutzenberger: On a family of sets related to McNaughton's L-language. In: Automata theory (ed. Caianiello), Academic Press, 1966, 320—324.
[69] M. P. Schützenberger: Sur certaines variétés de monoides finis. In: Automata theory (ed. Caianiello), Academic Press 1966, 314—319.
[70] E. Shamir: On sequential languages and two classes of regular events. Zeit. Phonetik Sprachwiss. Komm. *18* (1965), 61—69.

[71] P. H. Starke: Über die Darstellbarkeit von Ereignissen in nicht-initialen Automaten. Zeit. Math. Log. und Grund. Math. *9* (1963), 4, 315–319.

[72] P. H. Starke: Stochastische Ereignisse und Wortmengen. Zeit. Math. Log. u. Grund. Math. *12* (1966), 1–2, 61–68.

[73] R. E. Stearns and J. Hartmanis: Regularity preserving modifications of regular expressions. IC *6* (1963), 55–69.

[74] V. Tixier: Recursive functions of regular expressions in language analysis. Thesis, Stanford University, 1967.

[75] J. W. Thatcher, J. B. Wright: Generalized automata theory with an application to a decision problem of second-order logic. Math. Syst. Theory *2* (1968), 57–81.

[76] K. Thompson: Regular expression search algorithm. Comm. ACM *11* (1968), 6, 419–422.

[77] P. Turakainen: On non-regular events representable in probabilistic automata with one input letter. Ann. Univ. Turku, AI., Vol. 90 (1966).

[78] K. Udagawa, Y. Inagaki and H. Tange: State characteristic equations of finite automata and their regular expressions. Electron. Commun. Japan *48* (1965), 25.

[79] K. Udagawa and H. Tange: Transformation of state diagrams by state characteristic equations of finite automata. Electron. Commun. Japan *48* (1965), 48.

[80] V. Vučkovič: On a class of regular sets. Notre Dame Journ. Form. Log. *5* (1964), *2*, 113 to 124.

[81] H. Yamada: Disjunctively linear logic nets. EC *11* (1962), 5, 623–639.

[82] H. Yamada: Franctionalization of regular expressions (unpublished note).

[83] S. S. Yau: Generation on an output regular expression of a sequential machine with a specified input regular expression. Rept. Northwestern Univ., Evanston, 1966.

[84] M. Yoeli: Lattice-ordered semigroups, graphs and automata. Journal of the SIAM *13* (1965), 2, 411–422.

[85] M. Yoeli: Canonical representation of chain events. IC *8* (1965), 2, 180–189.

[86] В. Г. Боднарчук: Автоматы и события. УМЖ *14* (1962), 2, 351—361.

[87] В. Г. Боднарчук: Анализ взвешенных графов методом решения уравнений в алгебре событий. В сборнике: Теория конечных и вероятностных автоматов, Москва 1965, 246—249.

[88] В. Г. Боднарчук: Системы уравнений в алгебре событий. Журн. вычисл. математики и мат. физики *3* (1963) 6, 1077—1088.

[89] В. Г. Боднарчук: Метрические пространство событий I./II. Киб. (1965), 1, 24—27; 2, 22—23.

[90] В. Н. Глушков: Об одном алгоритме синтеза абстрактных автоматов. УМЖ *12* (1960) 2, 147—156.

[91] В. Н. Глушков: Про один метод аналізу абстрактних автоматів. Доповіді Акад. Наук Украинской ССР *12* (1960), 9, 1151—1154.

[92] В. Н. Глушков: Некоторые проблемы синтеза цифровых автоматов. Журн. вычисл. математики и мат. физики *1* (1961) 3, 371—411.

[93] В. Н. Глушков: Синтез цифровых автоматов. Физматгиз, Москва 1962.

[94] В. Н. Глушков: Абстрактная теория автоматов. Успехи мат. наук *16* (1961) 5.

[95] В. И. Казначеев: Построение тестов конечных автоматов с помощью языка регулярных выражений. В сб. Проблемы синтеза цифровых автоматов, Москва 1967, 145.

[96] И. Н. Коваленко: Замечание о сложности представления событий в вероятностных и детерминированных конечных автоматах. Киб. (1965) 2, 35—36.

[97] А. А. Курмит: Обращение автоматов по отношению к регулярному событию. Автоматика и вычислительная техника (1967), 4, 9—16.

[98] О. П. Кузнецов: Об одном классе регулярных событий. В сб.: Структурная теория релейных устройств. Москва 1963.

[99] О. П. Кузнецов: Представление регулярных событий в асинхронных автоматах. AT *26* (1965) 6, 1086—1093.

[100] О. П. Кузнецов, Е. Д. Стоцкая: Исследования по языке регулярных событий. Колл. по языкам конечных автоматов, Томск 1964.

[101] А. Г. Лунц: Метод анализа конечных автоматов. ДАН *160* (1965) 4, 778—780.

[102] Ю. И. Любич: О свойствах периодичности событий, представлимых в конечных автоматах. УМЖ *16* (1964), 3, 396—402.

[103] Ю. Т. Медведев: О классе событий, допускающих представление в конечном автомате. В сб.: Автоматы, Москва 1956.

[104] В. Г. Миркин: Новый алгоритм построения базиса в языке регулярных выражений. Известия АН СССР — Техническая кибернетика (1966) 5, 113—119.

[105] В. Г. Миркин: О языке псевдорегулярных выражений. Киб. *2* (1966) 6, 8—11.

[106] В. Г. Миркин: Минимизация последовательностных машин относительно регулярных полных справа событий. AT (1967) 11, 149—153.

[107] В. Г. Миркин: О распознавании относительной эквивалентности последовательностных машин. AT (1967), 2, 133—136.

[108] Г. С. Плеснович: О событиях, связанных с семейством автоматов 1. Вычислительные системы *9* (1963), 44—64.

[109] В. Н. Редько: Про коммутативные замыкания подий. Доповиди АН УССР (1963), 1156—1158.

[110] В. Н. Редько: Об алгебре коммутативных событий. УМЖ *16* (1964) 2, 185—195.

[111] В. Н. Редько: Об определяющей совокупности соотношений алгебры регулярных событий. УМЖ *16* (1964) 1, 120—126.

[112] В. Н. Редько: О коммутативных автоматах. В сб.: Теория конечных и вероятностных автоматов. Москва 1965, 253—256.

[113] А. Саломаа: Аксиоматизация алгебры событий, реализуемых логическими сетями. Проблемы кибернетики 17, Москва 1966, 237—246.

[114] М. А. Спивак: Новый алгоритм абстрактного синтеза автоматов. Материалы научных семинаров на теорет. и прикл. вопросам кибернетики 1 (1963), 3.

[115] М. А. Спивак: Разложение регулярного выражения по базису и его применения. ДАН *162* (1965), 3, 520—522.

[116] М. А. Спивак: Алгоритм абстрактного синтеза автоматов для расширенного языка регулярных выражений. Извести АН СССР — Техническая кибернетика (1965) 1, 51—57.

[117] М. А. Спивак: К методу анализа абстрактных автоматов с помощью уравнений в алгебре событий, Киб. *1* (1965) 1, 28.

[118] М. А. Спивак: Представление автоматных отображений регулярными выражениями. Киб. *1* (1965) 6, 15—17.

[119] М. А. Спивак: Некоторые свойства множества экспериментов автомата. Киб. *2* (1966) 6, 1—7.

[120] Ю. И. Янов: О тождественных преобразованиях регулярных выражений. ДАН *147* (1962), 2, 327—330.

[121] Ю. И. Янов: Об инвариантных операциях над событиями. Пробл. кибернетики 12, Москва 1964, 253—258.

[122] Ю. И. Янов: Об эквивалентных преобразованнях регулярных выражений. В сб.: Теория конечных и вероятностных автоматов. Москва 1965, 230—231.

[123] Ю. И. Янов: О некоторых подалгебрах событий, не имеющих конечных полных систем тождеств. Пробл. кибернетики 17, Москва 1966, 255—258.

Teorie regulárních událostí II

IVAN M. HAVEL

Druhá část přehledu teorie regulárních událostí a regulárních výrazů se zabývá její aplikací v teorii konečných automatů. Hlavní pozornost je věnována syntéze konečných automatů, a to zejména metodou používající derivací regulárních výrazů. Práce obsahuje též klasifikaci regulárních událostí dle různých hledisek.

Ing. Ivan M. Havel, Ustředí pro rozvoj automatizace a výpočetní techniky, Loretánské nám. 3, Praha 1.
Present address: Department of Computer Science, University of California, Berkeley, Calif. U.S.A.