ALGORYTMY

VOL. VIII • NO. 14 • 1971

INSTYTUT MASZYN MATEMATYCZNYCH

A L G O R Y T M Y
Vol. VIII N° 14 1971

Adres Redakcji: Warszawa, ul. Krzywickiego 34, tel. 28-37-29

Pow. w IMM pap. offset. kl. III g. 70 zam. 32/72.n.450 GP-II-1416/68

TREŚĆ

CONTENTS

ALGORYTMY, VIII, N° 14, 1971, 5-25

AN ALPHABETIC CODE FOR THE INFLEXIONAL ANALYSIS OF POLISH TEXTS

by Janusz Stanisław BIEŃ

Received April 28th, 1970

The Polish language is inflected. This means that for every lexical unit there is a set of its forms, i.e. words of the same meaning but of different syntactical functions. In automatic text processing it is necessary to connect given forms with descriptions of proper lexical units. This is called inflexional analysis. The main difficulty here arises with the stem alternations. The "paraphonetic code", worked out by the author, makes it possible to identify, when necessary, alternating elements; thereby the algorithm of the analysis becomes simpler and the index of lexical units can be smaller. The paper contains a description of the paraphonetic code "PF" and the ALGOL-60 procedures for its conversion from and to the code of the GIER computer.

1. INTRODUCTION

In automatic text processing the recognition of an input word[1)] is an indispensable stage. By recognition we mean ob-

1) Describing the inflexion we use the terminology which will be presented in [1]. It differs from the Polish linguistic terminology as well as from the English one, so we have to explain the notions which will be mentioned thereafter. By a word (in Polish "słowo") we understand a string of characters between two delimiters, usually punctuation marks. Adjectives and nouns are examples of lexical units called formems (in Polish "wyrazy"), i.e. sets of words which have varied syntactical functions but basically the same meaning. Possible semantic differences do not go beyond the grammatical categories of number, case, gender and person. For the sake of economy we found it convenient to join suitable formems into items called groups (in Polish "gniazda"). A Polish verb in the traditional sense is a good example of such an item: the group CZYTAĆ includes, among others, the formems czyta, czytał, czytający, czytany which in turn include their inflexional forms, e.g. the words "czytam", "czytasz", "czytający", "czytającego". For the sake of simplicity by group we mean also a group or a formem description kept in a computer store.

taining from the computer store the description referring to the word. Usually the description refers not to a single word, but to a complex of them, constituting the paradigm (the set of inflexional forms) of a so-called formeme or a group. The task of inflexional analysis can therefore be expressed as the transition from a given string of letters to the address of that group whose paradigm includes the string. It is necessary to utilitize appropriate rules of grammar as well as to search through a special index, containing the stems of selected forms of the accepted groups. This poses no problem in the case of languages with rudimentary inflexion. For example, in English an adjective possesses only one form (e.g. "blue"), a noun possesses two forms (e.g. "dog", "dogs"), a verb at most five (e.g. "learn", "learns", "learned"), but only for some of them these forms differ essentially (e.g. "see", "saw", "seen").

In Polish, the situation is far more complicated. Here an adjective has eleven forms (e.g. "dobry", "dobra", "dobre", "dobrego" etc.), a noun has thirteen forms (e.g. "szkoła", "szkoły", "szkole" etc.), while the number of simple forms of a verbal group equals thirty odd[2]. Besides, the similarity between inflexional forms of the same group (or of the same formeme) is often affected by stem alternations, for instance: "rów":"rowu", "ciasto":"cieście", "wisi":"wiszą" etc. In the past such exchanges were justified, e.g. the alternations "o":"ó" and "ą":"ę" originated in the fifteen and sixteen centuries as a result of distinguishing short vowels from long ones; now almost all the alternations are individual properties of particular groups. Another difficulty is a triple way of marking the palatalization of some consonants. For instance in the verbal group CZYNIĆ, the same sound [n'] is denoted by "ni" ("czynię" - the letter "i" is not pronounced), by "n" ("czyni" - the letter "i" is here pronounced) and by "ń" ("czyń").

[2] We mean the total number of the personal forms, the different inflexional forms of the declinable participles and some other forms specific for a Polish verb.

At first sight, there are two ways of solving the problem. One is to keep in the index a number of items which exhausts the variety of graphic stem variants for a given set of groups. Thus in order to recognize the Past Participle (e.g. "pisał", "pisali"; they are used mainly as a component part of the Future Tense, the Past Tense and the Conditional) of the groups: CZYTAĆ, MOKNAĆ, GIAĆ, LAĆ, WIEŚĆ, LEŹĆ we ought to store the following stems: "czyta" ("czytała", "czytali", "czytał",...), "mók" ("mókł"), "mok" ("mokła", "mokli",...), "gią" ("giął"), "gię" ("gięła", "gięli",...), ("lał", "lała", ,,,), "le" ("leli"), "wiód" ("wiódł"), "wiod" ("wiodła", "wiodły",...), "wied" ("wiedli"), "laz" ("lazł", "lazła",...), "leź" "leźli" . For the recognition of other forms of the groups, further stems are necessary; in addition to all this some alternations occur rather often and the index would therefore become too extensive.

The other method is based on the rule: one stem for one group this does not concern the exceptions like IŚĆ . An input word should be modified until it is found to be an item in the index. For instance, if we want to analyse the word "lazła", we need first of all to find its stem "laz". The next step is to separate the last vowel and the following consonants, and to check which alternations should be taken into account (in our case only "a":"e" and "z":"zi"). Then we should generate the set of stem modifications (e.g. "laź", "leź", "lezi") and search the index for every member of it. In our case "lezi" should fit the proper item, provided that for verbal groups we keep in the index the stem of 3rd person singular of the Present Tense (e.g. "lezie"). The algorithm allows for a concise index but it is complicated, and therefore slow. Thus none of these solutions is satisfactory. It may be that taking a middle course could yield better results but luckily there is a third way out the situation - the paraphonetic code.

The idea of a code which makes possible, when necessary, the identification of alternating elements is due to Prof.

J. Tokarski. He is the author of a series of papers [4] published at irregular intervals in the "Poradnik Językowy" from 1961 to 1964 and devoted to the description of the Polish inflexion from the point of view of machine translation. The conception has been realized by the present writer as the paraphonetic code "PF" and some ALGOL-60 procedures for the GIER computer.

2. CODE DESCRIPTION

The code under discussion is called paraphonetic because of the fact that some sounds of Polish have their counterparts in the code regardless of the way they are denoted in writing. To every symbol of the code an eight-bit binary number is assigned. Appendix 1 contains the table of code symbols, their binary equivalents and decimal values.

The code possesses forty two symbols, but the number can be increased if necessary. It includes letters and also sequences of letters which denote single sounds and which take part in stem exchanges, e.g. "ch" because of "głuchy":"głusi"[3]. The palatality of sounds is marked in two ways: by means of the letter "i" for sounds which have no graphic counterparts in the Polish alphabet and do not alternate[4], and thus have no symbols of their own in the code. They are the labials [p'], [b'], [m'] and the labio-dentals [f'], [w']. For the rest of them (e.g. for "ń", "dź") we use special symbols. Thus the word "dzień" is coded as dź-e-ń, and the word "nikt" as ń-i-k-t, but "wiosna" as w-i-o-s-n-a and "miasto" as m-i-a-s-t-o. While decoding from the paraphonetic code it is necessary to reinstate the normal, context depending,

[3] The list of stem alternation was fixed on the basis of works [3], [4], [6]. It makes us think that the list is complete.

[4] We do not take into account the alternation consisting in making a consonant hard (e.g. "łapie":"łap"), because it is more convenient to treat them in a strictly graphic way.

way of marking palatality. To do this we have to check if the palatal consonant is followed by the letter "i" or another vowel - thus presents no difficulty because the vowels in the code have values not greater than forty.

The main idea is very simple: the alternating symbols should differ only on some settled bits, which can be masked during matching. If all the sets of exchanging symbols (e.g. "n"-"ń", "l"-"ł") were disjoint, it would be enough to design some bits for numbering the sets (let us call them the main bits) and others (let us call them the serial bits) for numbering their elements. Unfortunately, this is not the case, the sets usually intersect, e.g. "d"-"dź", "dź"-"g", "g"-"ż", "ż"-"ź", "ź"-"z". Therefore we have to introduce some auxiliary bits and assign values to the elements of such sets under the following rule: the main bits describe a class of intersecting sets, the auxiliary bits differenciate the sets and the serial bits number their member. For the symbols included in a disjoint set the auxiliary bits have the same function as the main bits. To symbols which take no part in the alternations we can assign just successive numbers.

Finally, the way in which the exchanging symbols (e.g. "z":"ź", "ź":"ż") are united into a set of "congruent symbols" (e.g. "z"-"ź"-"ż") is a compromise between the number of auxiliary bits and the probability of undesirable identifications.

The estimation of the latter is made intuitively, because of the lack of statistical information about the occurrence of the alternations in texts.

We consider three as the optimal number of auxiliary bits. It allows us to separate the alternations "dz":"dź" (ch-o-dz-ę : ch-o-dź-i), "g":"dz" (n-o-g-a : n-o-dz-e) and "g":"ż" (m-o-g-ę : m-o-ż-e); these bits are between the main bits (on the left-hand side) and the serial ones (two bits on the opposite side) - the values of congruent elements, owing to such a division, differ relatively little, e.g. "r"-100

and "rz"-101, "ch"-144 and "ś"-147. For masking a symbol we can use different bit patterns. The most important are: 111 100 00, 111 010 00, 111 001 00, which are called respectively the left mask, the central mask and the right mask. It may sometimes be convenient to call the pattern 111 111 11 a full mask. Every symbol of an item in the index has a corresponding mask, which is used while matching an input string to cover irrelevant bits. For instance, because the item "w-l-e-cz" should have the full-full-central-central sequence of masks, then while comparing "cz"-011 010 11 with "k"-011 110 01 we compare in fact, owing to the central mask, only the main bits and the central auxiliary bit. We should receive a positive answer, which happens to be correct, because "wlecze" and "wloką" are forms of the same formeme. Occasionally it may lead to undesirable results, e.g. to the identification of "cz" in "piecze" with "c" in "piece". These must be treated as cases of normal homonymy (e.g. "mam" is the form of mieć or of mama or of mamić); incidentally, they can be avoided by a proper segmentation of the index and an appropriate order of searching.

The elements "c", "ć", "k", "t" can be identified with the help of the left mask as well as the central one. It is a consequence of the fact that these symbols together with "cz" compose the set of symbols which alternate in the conjugation and which can be identified with the left mask[5]. Because there are five of them, we need to use also one auxiliary bit to number them. In this way we connect one mask more with the set. We employ it - by appropriate assignment values to the above mentioned symbols - to ignore the alternations "c":"k" and "ć":"t", which occur in the declension of nouns and adjectives.

Below we present the list of the exchanges which were taken into account and the possibilities of ignoring them by the use of the three main masks.

---

5) The possibility of using the same mask for paradigmatic words of the same type proves sometimes useful (cf the last chapter).

The left mask can be used to identify the symbols:

a. "d"-"dz"-"dź"-"dż",
e.g. j-a-d-ę : j-e-dź-e, j-e-ż-dż-ę : j-e-ź-dź-i, w-i-dz-ę : w-i-dź-i, r-u-d-y : r-u-dź-i, s-a-d : s-a-dź-e;

b. "o"-"ć"-"k"-"t",
e.g. sz-y-b-k-i : sz-y-b-c-y, s-y-t-y : s-y-ć-i, ch-a-t-a : ch-a-ć-e;

c. "ch"-"s"-"sz"-"ś",
e.g. g-ł-u-ch-y : g-ł-u-ś-i, l-e-p-sz-y : l-e-p-ś-i, ł-y-s-y : ł-y-ś-i, r-o-s-a : r-o-ś-e, ń-o-s-ę : ń-e-ś-e, g-o-sz-cz-ę : g-o-ś-ć-i.

The central mask identifies the following symbols:

a. "dz"-"g"-"ż",
e.g. n-a-g-i : n-a-dz-y, m-o-g-ę : m-o-ż-e;

b. "o"-"cz"-"ć"-"k"-"t",
e.g. p-l-o-t-ę : p-l-e-ć-e, m-ł-ó-c-ę : m-ł-ó-ć-i, w-l-o-k-ę : w-l-e-cz-e;

c. "n"-"ń",
e.g. ł-a-d-n-y : ł-a-d-ń-i, cz-ó-ł-n-o : cz-ó-ł-ń-e, p-ł-y-n-ę : p-ł-y-ń-e;

d. "a"-"e"-"o"-"ó",
e.g. w-e-s-o-ł-y : w-e-s-e-l-i, r-ó-w : r-o-w-u, m-i-a-s-t-o : m-i-e-ś-ć-e, ń-o-s-ł-e-m : ń-ó-s-ł, w-l-o-k-ę : w-l-e-cz-e, j-a-d-ę : j-e-dź-e.

And the right one:

a. "z"-"ź"-"ż",
e.g. d-u-ż-y : d-u-ź-i, w-i-ą-z : w-i-ą-ź-e;

b. "l"-"ł",
e.g. c-a-ł-y : c-a-l-i, w-a-ł : w-a-l-e;

c. "r"-"rz",
e.g. s-t-a-r-y : s-t-a-rz-y, ż-a-r : ż-a-rz-e, b-i-o-r-ę : b-i-e-rz-e;

d. "ą"-"ę",
e.g. r-ę-k-a : r-ą-k, w-z-i-ą-ł : w-z-i-ę-ł-a.

3. CONVERSION PROCEDURES

The procedures were prepared for the GIER computer used in the Computation Centre of the Warsaw University (Zakład Obliczeń Numerycznych UW). They are written in Gier Algol 4. This hardware representation of ALGOL 60 has some convenient extensions of the reference language [2]. One of them is that Boolean variables can be treated as patterns of forty bits and be manipulated with high speed. A special case of their application is storing strings in Boolean arrays. To read a string into an array we use the standard procedure "readstring". There are also generalizations of conditional expressions and statements, called the case expressions or statements. Here is an example of a case expression: "case k of (11, 4, 5)", it has the value 11 for k = 1, the value 4 for k = 2 and the value 5 for k = 3. Because of the lack of some Polish letters in the input-output devices, it was necessary to use the following transcription: ą - a, ę - e, ć - c, ł - l, ń - n, ó - o, ś - s, ź - z, ż - r.

The procedures "fromGIERtoPF" (App. 3) and "fromPFtoGIER" (App. 4) convert a string from the GIER code (App. 2) to the paraphonetic code PF and vice versa. Their formal parameters are the same: the Boolean arrays "sfile", "ffile" and the integers "sp", "fp". The source string is stored in the array "sfile" starting in the element "sp"+1, and the final string in the array "ffile" starting in the element "fp+1". The procedures "read6bits" and "read8bits" yield consecutive symbols of a string and the procedures "write6bits", "write8bits" store them. The procedures "read6bits" and "write6bits" work according to the Gier Algol 4 string format, i.e. six bits per symbol, the value 10 means the end of a string. They use the integers "r6start", "r6char", "w6start", "w6char". The procedures "read8bits" and "write8bits" work according to the

PF format: eight bits per symbol, the value 0 means the end of a record. They use the integers "r8start", "r8char", "w8start", "w8char". The procedure "trace" has been added for debugging purposes. It prints, when wanted, the trace of the program; it uses the variable "spa".

The body of the procedure "fromGIERtoPF" contains some declarations: the integer array "buf" (cf line 82 of App. 3) for buffering the input, the integer variable "i" for counting symbols which are already coded but still remain in the buffer and the variable "j" to keep the total number of symbols in the buffer. Every input symbol passes through the buffer (cf l.120). It is the simplest case if a given letter cannot start a sequence denoting a single PF symbol (including digraphs with the letter "i" meaning palatalisation), i.e. if the letter is one of the following letters: "a", "b", "e", "f", "g", "h", "i", "j", "k", "l", "m", "o", "p", "q", "t", "u", "v", "w", "x", "y". We look for the position of the character in the list of GIER characters and then we take the corresponding position from the list of PF symbols (l.165-171). The task of the procedure "return" (l.83-87) is to store the given symbol, change the content of the counter "i" and go over to the analysis of a new character. We treat the digraphs starting with the underlining ("a", "c", "e", "l", "n", "o", "s", "z", "r") in a similar way. After the recognition of the underlining (l.122) we read the next character in, which decides what symbol will be put out by means of "return" (l.128).

In the case of the letter "r" (l.136-140) we have also to check the next character, because the letter may occur as a part of the digraph "rz". When the given sequence of characters can denote a palatalized sound as well as a hard one, the situation is far more complicated. To examine such cases we introduce the procedure "palat" (l.88-112). It is called with the value of the palatal sound in the PF code as the actual parameter. If the next letter (l.92) is not "i", the

procedure involves nothing more. If it is "i", the actual parameter is stored as the PF counterpart of the input sequence of characters and the next character is read in. Now we have one of the following cases: either it is a vowel which means that "i" was used only as the palatality mark and ought to be skipped while the vowel is coded (l.102,108), or it is a consonant which means that "i" denoted a sound and therefore it is coded (l.103,109); the procedure ends by going over to the analysis of the next character. With the help of the procedure "palat" all the remaining sound denotations are coded, namely: "z(i)" - lines 130-131, "s(i)", "sz" - lines 132-133, "n(i)" - lines 134-135, "c(i)", "cz", "ch" - lines 141-145, "d", "dz̲", "dr̲", "dz(i)" - lines 146-164. The exit from the procedure "fromGIERtoPF" is caused by receiving the character with the value 10.

The procedure "fromPFtoGIER" is much simpler. The only difficulty is to reinstate the normal way of marking palatality. It is enough to check the next symbol, so instead of buffering we use a recursive procedure called "decode" (cf lines 172-224 of App. 4). Its parameter is a symbol of the PF code. If it refers to a palatal consonant (cf l.177), the next symbol is read in. Depending on whether it is a vowel or not, we decode the sound using a diacritic sign (e.g. "dz̲") or we decode its hard counterpart, after which we add the letter "i" (e.g. "dzi"), except the case where "i" is the vowel just read in. The exit from the procedure "decode" takes place after decoding the remaining symbol (l.194). The digraphs with the underlining are decoded in lines 197-201, the rest of them in lines 202-208 and the sequences of three characters ("dz̲", "dr̲") in lines 209-213. The symbols corresponding to single characters are decoded in lines 215-222. If the symbol of the value 0 is received, it changes the Boolean variable "e" which is unlocal for the procedure "decode"

(1.175). Thus the proper action of the procedure "fromPFtoGIER" is to read a symbol in, to call "decode" and so on, as long as the variable "e" has the value true.

4. REMARKS ON APPLICATIONS

The way of assigning values to symbols of the paraphonetic code depends on the assumption about the contents of groups and the segmentation of the index. The values chosen here were found convenient for no index segmentation and for the following contents of the groups:

a. an uninflected group - a single uninflected form,

b. a noun group - a noun only,

c. an adjectival group - the positive, comparative and superlative degrees of an adjective and its adjectival adverb, the adjectival noun and the forms of the type "zdrów", "wesół" and "polsku", "rusku",

d. a verbal group - the Present (or Simple Future) Tense and simple forms of the Imperative Mood, the Past Participle, the Passive (e.g. "pisany") and the adjectival Simultaneous (e.g. "piszący") Participles, The Infinitive, the Simultaneous (e.g. "pisząc") and the Anticipatory (e.g. "napisawszy") Participles and the adverbial form of the Passive Participle (e.g. "pisano").

If we change the above assumption it may happen that a different set of values will be more useful.

In almost all the cases we need not store as many masks as there are symbols in a stem. The alternations affect the endings of stems, so it is enough to keep only a few masks for an item. Another aspect of the question is that we can store a set of masks for every item or only for some of them. In the latter case if we search the index sequentially we may put

the complexes of masks between proper items and use the same masks all the time until we meet a new complex of them. This method has been applied in the program written for testing the code.

To conclude, I suppose that the idea of the paraphonetic code may prove quite useful for other inflexional languages in which the stem exchanges also occur.

References

[1] BIEŃ J.St.: O pewnych problemach przetwarzania języków fleksyjnych na maszynach cyfrowych. Prace Filologiczne t. XXIII, [to appear].

[2] NAUR P.: A Manual of Gier Algol 4. Copenhagen 1967.

[3] SZOBER St.: Gramatyka języka polskiego. Warszawa 1969.

[4] TOKARSKI J.: Czasowniki polskie. Warszawa 1951.

[5] TOKARSKI J.: Fleksja polska, jej opis w świetle możliwości mechanizacji w urządzeniu przekładowym. Poradnik Językowy 1961-64.

[6] TOKARSKI J.: Elements of Polish Grammar. In J. Stanisławski: The Great Polish-English Dictionary. Warsaw 1969, [also in Polish].

Appendix 1

PF Code

a	8	000 010 00
ą	4	000 001 00
b	160	101 000 00
c	123	011 110 11
ch	144	100 100 00
cz	106	011 010 10
ć	120	011 110 00
d	81	010 100 01
dz	88	010 110 00
dź	82	010 100 10
dż	83	010 100 11
e	9	000 010 01
ę	5	000 001 01
f	161	101 000 01
g	73	010 010 01
h	162	101 000 10
i	40	001 010 00
j	163	101 000 11
k	121	011 110 01
l	132	100 001 00
ł	133	100 001 01
m	164	101 001 00
n	136	100 010 00
ń	137	100 010 01
o	10	000 010 10
ó	11	000 010 11
p	165	101 001 01
q	166	101 001 10
r	100	011 001 00
rz	101	011 001 01
s	145	100 100 01
sz	146	100 100 01
ś	147	100 100 11
t	122	011 110 10
u	32	001 000 00
v	167	101 001 11
w	168	101 010 00
x	169	101 010 01
y	33	001 001 01
z	68	010 001 00
ź	69	010 001 01
ż	90	010 011 10

Appendix 2

GIER Code

a	49
b	50
c	51
d	52
e	53
f	54
g	55
h	56
i	57
j	33
k	34
l	35
m	36
n	37
o	38
p	39
q	40
r	41
s	18
t	19
u	20
v	21
w	22
x	23
y	24
z	25
-	14

APPENDIX 3
The procedure "fromPFtoGIER"

```
79    procedure fromGIERtoPF(sfile,sp,ffile,fp);
80    boolean array sfile,ffile; integer sp,fp;
81    begin comment read6bits,write8bits;
82      integer array buf[1:10]; integer i,j,k,c,c1;
83      procedure return(char,n); integer char,n;
84      begin
85        spa:=spa+2; write8bits(ffile,char); i:=i+n;
86        trace(‡<return‡,-2,i,j,0,char); goto TEST
87      end return;
88      procedure palat(char); integer char;
89      begin
90        trace(‡<palat‡,2,i,j,0,char);
91        j:=j+1; c:=read6bits(sfile,buf[j]);
92        if c=57 then
93        begin
94          write8bits(ffile,char); j:=j+1;
95          c:=read6bits(sfile,buf[j]);
96          if c=14 then
97          begin
98            j:=j+1; c:=read6bits(sfile,buf[j]);
99            k:=1;
100           for c1:=38,49,53 do
101           if c≠c1 then k:=k+1 else
102           begin c1:=case k of(11,4,5); return(c1,4) end;
103           return(40,2);
```

```
104           end if 14;
105           k:=1;
106           for c1:=20,24,38,49,53,57 do
107           if c#c1 then k:=k+1 else
108           begin c1:=case k of(32,33,10,8,9,40); return(c1,3) end;
109           return(40,2)
110         end if 57;
111         spa:=spa-2;
112       end palat;
113       trace(‡<<fromGIERtoPF‡,2,sp,fp,0,0);
114       r6start:=sp+1; w8start:=fp+1; r6char:=w8char:=0;
115       ffile[w8start]:=false;
116       j:=0; i:=1;
117   TEST:
118       if i>j then
119       begin j:=1; i:=1; read6bits(sfile,buf[j]); goto TEST end;
120       c:=buf[i];
121       if c=10 then goto E;
122       if c=14 then
123       begin
124         j:=j+1; c:=read6bits(sfile,buf[j]);
125         k:=1;
126         for c1:=18,25,35,37,38,41,49,51,53 do
127         if c#c1 then k:=k+1 else
128         begin c1:=case k of(147,69,133,137,11,90,4,120,5); return(c1,2) end;
129       end if 14;
130       if c=25 then
131       begin palat(69); return(68,1) end;
```

```
132     if c=18 then
133     begin palat(147); if c=25 then return(146,2); return(145,1) end;
134     if c=37 then
135     begin palat(137); return(136,1) end;
136     if c=41 then
137     begin
138       j:=j+1; c:=read6bits(sfile,buf[j]);
139       if c=25 then return(101,2); return(100,1)
140     end if 41;
141     if c=51 then
142     begin
143       palat(120); if c=25 then return(106,2);
144       if c=56 then return(144,2); return(123,1)
145     end if 51;
146     if c=52 then
147     begin
148       j:=j+1;c:=read6bits(sfile,buf[j]);
149       if c=14 then
150       begin
151         j:=j+1; c:=read6bits(sfile,buf[j]);
152         if c=41 then return(83,3); if c=25 then return(82,3);
153         k:=1;
154         for c1:=49,53,35,38,37,18 do
155         if c<>c1 then k:=k+1 else
156         begin
157           c1:= case k of (4,5,133,11,137,147);
158           write8bits(ffile,81);return(c1,3)
```

```
159           end;
160           return(81,1)
161         end;
162         if c=25 then begin i:=i+1; palat(82);return(88,1) end;
163         return(81,1)
164       end if 52;
165       k:=1;
166       for c1:=19,20,21,22,23,24,33,34,35,36,38,39,40,49,50,53,
                54,55,56,57 do
167       if c≠c1 then k:=k+1 else
168       begin
169         c1:=case k of(122,32,167,168,169,33,163,121,132,164,10,165,
                          166,8,160,9,161,73,162,40);
170         return(c1,1)
171       end;
172   E:  sp:=r6start;;
173       fp:=w8start;
174       trace(‡<GtoP‡,-2,sp,fp,0,0)
175     end fromGIERtoPF;
```

APPENDIX 4

The procedure "fromPFtoGIER"

```
169   procedure fromPFtoGIER(sfile,sp,ffile,fp);
170   boolean array sfile,ffile; integer sp,fp;
171   begin integer c;boolean e;
172     procedure decode(c); integer c;
173     begin integer k,c1,c2;
174       trace(‡<decode‡,2,0,0,0,c);
175       if c=0 then begin e:=false;goto E end;
176       k:=1;
177       for c1:=120,137,147,69,82 do
178       if c≠c1 then k:=k+1 else
179       begin
180         read6bits(sfile,c);if c=0 then e:=false;
181         if c≤40∧c≠0 then
182         begin
183           c1:=case k of(123,136,145,68,88); decode(c1);
                  write6bits(ffile,57);
184         end
185         else
186         if c1=82 then
187         begin
188           write6bits(ffile,52); write6bits(ffile,14);
              write6bits(ffile,255);
189         end
190         else
191         begin
192           c1:=case k of(51,37,18,25); write6bits(ffile,14);
                  write6bits(ffile,c1);
193         end;
194         if c≠40 then decode(c); goto E;
```

```
195       end;
196       k:=1;
197       for c1:=4,5,133,11,90 do
198       if c<>c1 then k:=k+1 else
199       begin
200         c1:=case k of(49,53,35,38,41); write6bits(ffile,14);
              write6bits(ffile,c1); goto E
201        end;
202       k:=1;
203       for c1:=144,106,88,101,146 do
204       if c<>c1 then k:=k+1 else
205       begin
206         c1:=case k of(51,51,52,41,18); c2:=if k=1 then 56 else 25;
207         write6bits(ffile,c1); write6bits(ffile,c2); goto E
208       end;
209       if c=82\/c=83 then
210       begin
211         write6bits(ffile,52); write6bits(ffile,14);
212         write6bits(ffile,if c=82 then 25 else 41); goto E
213       end;
214       k:=1;
215       for c1:=8,160,123,81,9,161,73,162,40 do
216       if c<>c1 then k:=k+1 else
          begin write6bits(ffile,48+k); goto E end;
217       k:=1;
218       for c1:=163,121,132,164,136,10,165,166,100 do
219       if c<>c1 then k:=k+1 else
          begin write6bits(ffile,32+k); goto E end;
220       k:=1;
221       for c1:=145,122,32,167,168,169,33,68 do
```

```
          if c≠c1 then k:=k+1 else
          begin write6bits(ffile,17+k); goto E end;
   E:     spa:=spa-2;
        end decode;
        trace(‹<fromPFtoGIER›,2,sp,fp,0,0);
        r8start:=sp+1; w6start:=fp+1; r8char:=w6char:=00;e:=true;
        ffile[w6start]:=4106106106106106106106 10;
   NEXT:
        read8bits(sfile,c);
        decode(c);
        if e then goto NEXT;
        sp:=r8start; fp:=w6start;
        trace(‹<FtoG›,-2,sp,fp,0,0)
      end from PFtoGIER;
```

KOD MASZYNOWY DO ANALIZY FLEKSYJNEJ TEKSTÓW POLSKICH

Streszczenie

Język polski jest językiem fleksyjnym, co oznacza, że z jednym wyrazem związany jest zespół jego form, nie różniących się znaczeniem, ale pełniących odmienne role składniowe. Przy maszynowym przetwarzaniu tekstów niezbędne jest kojarzenie konkretnej formy z odpowiadającym jej wyrazem, a ściślej - z przechowywanym w pamięci opisem tego wyrazu. Czynność tę nazywamy analizą fleksyjną; jest ona niełatwa m.in. ze względu na tzw. wymiany tematowe. Opracowany przez autora kod parafonetyczny pozwala utożsamiać elementy wymieniające się, a tym samym zarówno upraszcza sam algorytm analizy, jak i znacznie zmniejsza objętość indeksu będącego podstawą rozpoznawania form. Artykuł ten zawiera opis kodu parafonetycznego PF oraz algolowych procedur przejścia pomiędzy kodem PF a kodem EMC GIER.

МАШИННЫЙ КОД ДЛЯ ФЛЕКСИОННОГО АНАЛИЗА ПОЛЬСКИХ ТЕКСТОВ

Резюме

Польский язык является флексионным; это обозначает, что с одним словом связан состав его форм не различающихся по значению, но исполняющих разную синтаксическую роль. При машинной обработке текстов необходимо согласовать конкретную форму с отвечающим ей словом, более точно - с описанием этого слова сохраненным в запоминающем устройстве. Такое действие называем флексионным анализом; это не простой анализ, между прочим в виду так называемого обмена тем. Парафометический код, разработанный автором, позволяет идентифицировать элементы обмена, следовательно упрощает сам алгоритм анализа, а также значительно уменьшает объем индекса являющейся основой опознания форм. Настоящая статья заключает описание парафонетического кода PF и процедур Алгола - перехода между кодом PF и кодом EMC GIER.

Warszawa, ul. Krzywickiego 34
A.M.

ALGORYTMY, VIII, No 14 1971, 27-67

O MINIMALIZACJI AUTOMATÓW

Józef WELO

Pracę złożono 1969,4.10

W artykule podano algorytmy i metody minimalizacji dowolnych automatów skończonych częściowo określonych albo niezupełnych w oparciu o pojęcie zbiorów domkniętych w grafie zorientowanym skończonym oraz algorytm znalezienia automatu minimalnego dla danego automatu - **patrz punkt 5.9.**

Podkreślamy, że zawarte tu pojęcia są formułowane pod kątem minimalizacji automatów w praktyce inżynierskiej.

Dowodów wielu własności i twierdzeń nie podano zakładając, że są one oczywiste bądź łatwe do znalezienia.

1. WSTĘP

Głównym celem niniejszego artykułu jest podanie algorytmów i metod minimalizacji dowolnych automatów skończonych, częściowo określonych albo niezupełnych, w oparciu o pojęcie zbioru domkniętego w grafie zorientowanym skończonym, a także podanie algorytmu znalezienia automatu minimalnego danego automatu, tj. takiego automatu równoważnego danemu, który składa się z najmniejszej liczby stanów wewnętrznych spośród wszystkich automatów równoważnych.

Ponadto udowodniono tu, że zagadnienie minimalizacji automatów należy do teorii grafów zorientowanych skończonych. Ściślej - zagadnienie znalezienia automatu równoważnego danemu i złożonego z mniejszej liczby wszystkich stanów wewnętrznych od liczby wszystkich stanów wewnętrznych danego automatu jest równoważne znalezieniu odpowiedniego zbioru domkniętego w pew-

nym grafie zorientowanym zależnym od tego automatu, albo zależnym od funkcji opisujących ten automat. Temu zagadnieniu poświęcono rozdziały 4–5. W rozdziale 3 podano algorytm I, który ma duże znaczenie dla minimalizacji automatów za pomocą metod tu zaprezentowanych.

Dowodów wielu własności i twierdzeń nie podajemy z uwagi na ich oczywistość.

Podkreślamy także, że podane tu pojęcia są wyłącznie dostosowane do naszych potrzeb, a więc pod kątem minimalizacji automatów w praktyce.

Zwracamy również uwagę, że interpretacja zbioru X jest wieloraka: w jednym przypadku nazywamy go po prostu zbiorem, w innym - zbiorem wierzchołków grafu, a jeszcze w innym, zbiorem stanów wewnętrznych automatu itp; analogiczna uwaga dotyczy elementów tego zbioru. W każdym przypadku spełnia on odpowiednią rolę.

2. NIEKTÓRE OZNACZENIA

X oznacza dowolny, ustalony, niepusty skończony zbiór; x, x', x'', x_1, x_2,... oznaczają elementy zbioru X.

Duże litery alfabetu łacińskiego ze wskaźnikami u dołu lub bez tych wskaźników, pisane cienką kursywą, oznaczają dowolne niepuste podzbiory zbioru X- np. A, B, C,..., A_1, A_2,..., B_1, B_2,..., z tym, że symbole K, K_1, K_2,..., rezerwujemy dla oznaczania podzbiorów dwuelementowych zbioru X.

Duże litery alfabetu łacińskiego ze wskaźnikami u dołu lub bez tych wskaźników, pisane grubą kursywą, oznaczają dowolne niepuste rodziny podzbiorów zbioru X, np. $\mathbf{A}$, $\mathbf{B}$, $\mathbf{C}$,..., $\mathbf{A}_1$, $\mathbf{A}_2$,...

$(A)^2$ jest to rodzina wszystkich podzbiorów dwuelementowych zbioru A.

Duże litery alfabetu łacińskiego ze wskaźnikiem 2 u góry oraz ze wskaźnikami u dołu lub bez wskaźników u dołu, pisane cienką kursywą, oznaczają dowolne niepuste podzbiory zbioru $(X)^2$, tj. rodziny podzbiorów dwuelementowych zbioru X, np. A^2, B^2, C^2,..., A_1^2, A_2^2,... Należy więc odróżnić symbol $(A)^2$ od symbolu A^2.

2^A jest to rodzina wszystkich podzbiorów zbioru A. Symbol $\emptyset$ oznacza zbiór pusty.

Ponadto stosujemy symbole powszechnie przyjęte w teorii mnogości, jak znak sumy $\cup$, różnicy $\setminus$, iloczynu $\cap$, należenia $\in$, negacja należenia $\notin$, zawierania $\subset$, negacja zawierania $\not\subset$, równości $=$, negacja równości $\neq$, tożsamości $\equiv$. Czasami symbol $\equiv$ oznacza to samo, co "wtedy i tylko wtedy". Natomiast znak $\subsetneq$ oznacza ostre zawieranie, np. $A \subsetneq B$ oznacza, że $A \subset B$ i $A \neq B$.

Należy więc bezwzględnie odróżnić symbol $A \cup B$ od symbolu $\mathbf{A} \cup \mathbf{B}$, z których pierwszy jest podzbiorem zbioru X, a drugi - podzbiorem zbioru 2^X. Ściślej, element $x \in X$ należy do zbioru $A \cup B$ wtedy i tylko wtedy, gdy $x \in A$ lub $x \in B$. Natomiast zbiór $D \subset X$ należy do rodziny $\mathbf{A} \cup \mathbf{B}$ wtedy i tylko wtedy, gdy $D \in \mathbf{A}$ lub $D \in \mathbf{B}$. Odpowiednia uwaga dotyczy również symboli $A \setminus B$ i $\mathbf{A} \setminus \mathbf{B}$ oraz symboli $A \cap B$ i $\mathbf{A} \cap \mathbf{B}$. W szczególności, np. symbol $\mathbf{A}$ może być symbolem C^2 lub $(R)^2$.

Spotkamy dalej także symbol

$$B = \bigcup_{j=0}^{p} A_j = A_0 \cup A_1 \cup A_2 \cup \ldots \cup A_p.$$

Mamy tutaj zbiór $B \subset X$, do którego element x należy wtedy i tylko wtedy, gdy istnieje $j \in \{0,1,2,\ldots,p\}$ i $x \in A_j$. Jeśli więc symbol A_j, $j = 0,1,2,\ldots,p$, zastąpimy np. symbolem $\mathbf{A}_j$, to wtedy symbol B jest odpowiednim symbolem $\mathbf{B} \subset 2^X$, a więc $B \neq \mathbf{B}$, na co zwracamy uwagę.

$\mathbf{A} \times \mathbf{B}$ oznacza iloczyn kartezjański zbiorów / rodzin / $\mathbf{A}$ i $\mathbf{B}$, tj. zbiór wszystkich par uporządkowanych (A,B), gdzie $A \in \mathbf{A}$

i $B \in \mathbf{B}$, co symbolicznie zapisujemy $(A,B) \in \mathbf{A} \times \mathbf{B}$. W szczególności np. symbol $\mathbf{A}$ może być symbolem R, a symbol $\mathbf{B}$ symbolem D. Wówczas otrzymamy symbol R×D, tj. zbiór wszystkich par uporządkowanych (x_1, x_2), gdzie $x_1 \in R$ i $x_2 \in D$, czyli $(x_1, x_2) \in R \times D$. Zbiory R i D mogą z kolei być dowolnymi niepustymi skończonymi zbiorami (niekoniecznie podzbiorami zbioru X).

$[A]^2$ jest to rodzina, do której zbiór K należy wtedy i tylko wtedy, gdy istnieje $A \in \mathbf{A}$ i $K \in (A)^2$. Zatem $[A]^2$ jest podzbiorem zbioru $(X)^2$.

Chociaż w teorii mnogości A^2 oznacza to samo, co A×A, to jednak rezygnujemy z tej zasady dla odróżnienia symbolu $(A)^2$ od symboli A^2 i $[A]^2$.

Przez $\{x\}$ / $\{A\}$, $\{\mathbf{A}\}$ / oznaczamy zbiór złożony tylko z jednego elementu x /A, $\mathbf{A}$/.

U(A) jest to rodzina, do której zbiór B należy wtedy i tylko wtedy, gdy istnieje $x \in A$ i $B = \{x\}$.

3. RODZINY $m(A, B^2)$ i $\bar{m}(A, D^2)$

Głównym celem rozdziału 3 jest podanie algorytmu znalezienia rodzin $m(A, B^2)$ i $\bar{m}(A, D^2)$, co ma duże znaczenie dla minimalizacji automatów. Ponadto określona relacja $\rightarrow$ pomiędzy podzbiorami zbioru 2^X pozostaje także w pewnym związku z zagadnieniem minimalizacji automatów.

3.1. Zbiór stabilny symbolu: $\langle A, B^2 \rangle$

Dana jest uporządkowana dwójka: $\langle A, B^2 \rangle$, w której A jest dowolnym niepustym skończonym zbiorem oraz $B^2 \subset (A)^2$, przy czym nie wykluczamy $B^2 = \emptyset$.

Zbiór $D \subset A$, $D \neq \emptyset$, nazywamy zbiorem stabilnym symbolu: $\langle A, B^2 \rangle$, jeżeli $(D)^2 \subset B^2$.

Elementem maksymalnym rodziny $\mathbf{A}$ nazywamy taki zbiór $A \in \mathbf{A}$, że jeśli $A \subset M \in \mathbf{A}$, to $A = M$.

Przez $\eta(A)$ oznaczamy zbiór (rodzinę) wszystkich elementów maksymalnych rodziny A. Przez $m(A, B^2)$ oznaczamy zbiór (rodzinę) wszystkich elementów maksymalnych rodziny wszystkich zbiorów stabilnych symbolu: $\langle A, B^2\rangle$, czyli

$$m(A, B^2) \overset{\text{def}}{=\!=} \eta(\{D \subset A : (D)^2 \subset B^2\}) .$$

Oznaczamy

$$\bar{m}(A, D^2) \overset{\text{def}}{=\!=} \eta\left(\left\{R : \bigvee_{A \in A}\left[R \in m(A, (A)^2 \setminus D^2)\right]\right\}\right), \qquad (1)$$

gdzie $A \neq \emptyset$ i nie wykluczamy $D^2 = \emptyset$.

3.2. Relacja $\rightarrow$

Dla dowolnych A, B określamy relację $\rightarrow$, jak następuje:

1. $\emptyset \rightarrow D$ dla każdego D;
2. $A \rightarrow B$, jeżeli dla każdego $A \in A$ istnieje $B \in B$ i $A \subset B$.

Piszemy $A \overset{\Gamma}{=} B$, jeżeli $A \rightarrow B$ i $B \rightarrow A$. Piszemy $A \not\rightarrow B$, jeżeli nie jest prawdą, że $A \rightarrow B$. Relacja $\rightarrow$ jest zwrotna i przechodnia.

Symbol $[R_1] \equiv [R_2]$ oznacza, że z prawdziwości relacji R_1 wynika prawdziwość relacji R_2, i odwrotnie. Natomiast symbol $(R_1) \Rightarrow (R_2)$ oznacza, że z prawdziwości relacji R_1 wynika prawdziwość relacji R_2.

Mamy

$$\left[A \overset{\Gamma}{=} B\right] \equiv \left[\eta(A) = \eta(B)\right] . \qquad (2)$$

3.3. Rodzina typu m

$S(A)$ jest to zbiór, do którego x należy wtedy i tylko wtedy, gdy istnieje $A \in A$ i $x \in A$.

Rodzinę A nazywamy rodziną typu m, jeżeli

$$\eta(A) = m\left(S(A), [A]^2\right)^{1)}. \tag{3}$$

P R Z Y K Ł A D Y

1. Rodziny $\{A\}$ i $m(A, B^2)$ są typu m.
2. Rodzina zbiorów rozłącznych jest typu m.
3. Rodzina 2^A jest typu m, bo $\eta(2^A) = \{A\}$.

T w i e r d z e n i e 1. Jeżeli A jest rodziną typu m, to dla dowolnego B^2 rodzina $\bar{m}(A, B^2)$ jest także rodziną typu m, to znaczy, że jeśli zachodzi (3), to

$$\bar{m}(A, B^2) = m\left(S(A), [A]^2 \setminus B^2\right).$$

D o w ó d. Niech zachodzi (3). Na mocy (1) mamy: jeśli $D \in \bar{m}(A, B^2)$, to $D \in m(A, (A)^2 \setminus B^2)$ dla pewnego $A \in A \overset{\Gamma}{=} \eta(A)$. Tak więc $A \subset A_1 \in \eta(A)$. Wobec (3) oznacza to, że $D \subset A_1 \in m(S(A), [A]^2)$, ponieważ $D \subset A \subset A_1$. Z drugiej strony $(D)^2 \subset (A)^2 \setminus B^2 \subset (A_1)^2 \setminus B^2 \subset [A]^2 \setminus B^2$, bo $A \subset A_1$ oraz $(A_1)^2 \subset [A]^2 = [\eta(A)]^2$. Zatem $D \subset S(A)$ i $(D)^2 \subset [A]^2 \setminus B^2$. Warunki te oznaczają, że $\{D\} \to m(S(A), [A]^2 \setminus B^2)$, ponieważ $[A]^2 \setminus B^2 \subset (S(A))^2$. Wobec dowolności zbioru $D \in \bar{m}(A, B^2)$ mamy $\bar{m}(A, B^2) \to m(S(A), [A]^2 \setminus B^2)$.

D o w ó d w odwrotną stronę, Jeżeli $D \in m(S(A), [A]^2 \setminus B^2)$, to $D \subset S(A)$ i $(D)^2 \subset [A]^2 \setminus B^2 \subset [A]^2$. Warunki $D \subset S[A]$ i $(D)^2 \subset [A]^2$ oznaczają wobec (3), że $D \subset A \in \eta(A) \subset A$, skąd $D \subset A \in A$. Warunki $D \subset A \in A$ i $(D)^2 \subset [A]^2 \setminus B^2$ oznaczają, że $\{D\} \to m(A, (A)^2 \setminus B^2) \to \bar{m}(A, B^2)$, bo $(D)^2 \subset (A)^2 \setminus B^2$. Wobec dowolności zbioru D mamy więc $m(S(A), [A]^2 \setminus B^2) \to \bar{m}(A, B^2)$. Stąd i z poprzedniej części dowodu mamy $\bar{m}(A, B^2) \overset{\Gamma}{=} m(S(A), [A]^2 \setminus B^2)$. Rodziny te są złożone z elementów maksymalnych, więc ostatnia relacja na mocy (2) kończy dowód twierdzenia.

---

1) Zauważmy, że $S(\eta(A)) = S(A)$ i $[\eta(A)]^2 = [A]^2$.

3.4. Algorytm znalezienia rodziny $m(A, B^2)$

Przez card (A) (card (A)) oznaczamy moc zbioru A (moc rodziny A).

Zapowiedziany algorytm jest następujący: niech card(A) = n. Ustawmy wszystkie elementy zbioru A w ciąg $x_1, x_2, \ldots, x_n \in A$. Dla j = 1,2,..., n-1 tworzymy ciąg zbiorów

$$A_j = \{x \in A : \bigvee_{\{x_j, x\} \in (A)^2 \setminus B^2} [x \neq x_t \text{ dla } t < j]\} \qquad (4)$$

oraz ciągi rodzin $R_0 = \{A\}$, A_j i R_j takich, że:

1. jeżeli $D \in R_{j-1}$ i $x_j \in D$ oraz $D \cap A_j \neq \emptyset$, to $D \setminus \{x_j\}$, $D \setminus A_j \in A_j$, a w przeciwnym przypadku $D \in A_j$,

2. $R_j = \eta(A_j)$.

Wówczas dla j = n-1 otrzymamy $R_{n-1} = m(A, B^2)$.

Algorytm ten nazywamy algorytmem I. Jest to również algorytm znalezienia rodziny $\bar{m}(A, D^2)$, ponieważ dla każdego $B \in A$ znajdziemy odpowiednią rodzinę $m(B, (B)^2 \setminus D^2)$, a więc rodzinę $\bar{m}(A, D^2)$ znajdziemy następnie na mocy wzoru (1)

Zbiór $(A)^2 \setminus B^2$, biorący udział w ciągu (4), zapisujemy równoważnie za pomocą ciągu (4) w postaci ciągu

$$(A)^2 \setminus B^2 \equiv \{x_1(A_1), x_2(A_2), \ldots, x_{n-1}(A_{n-1})\}. \qquad (5)$$

Wtedy odpowiednikiem ciągu (4) dla zbioru B^2 jest ciąg

$$C_j = \{x \in A \setminus A_j : x \neq x_t \text{ dla } t < j\}, \quad j = 1,2,\ldots,n-1, \qquad (6)$$

skąd otrzymamy, jako odpowiednik ciągu (5), następujący zapis zbioru B^2 :

$$B^2 \equiv \{x_1(C_1), x_2(C_2), \ldots, x_{n-1}(C_{n-1})\}. \qquad (7)$$

Zapis zbiorów (5) i (7) jest wygodny w praktyce, a ponadto posiada pewne znaczenie gdy chodzi o sprawne znalezienie zarazem rodziny $m(A, B^2)$ i $m(A,(A)^2 \setminus B^2)$.

4. ZBIÓR DOMKNIĘTY W GRAFIE

Głównym celem rozdziału 4 jest pojęcie grafu skończonego zorientowanego $X|f$ oraz algorytm znalezienia zbioru (zbiorów) domkniętego w tym grafie, tj. takiego zbioru $A \subset X$, że $f(A) \subset A$, jak również algorytm znalezienia zbiorów domkniętych w dowolnym grafie częściowym grafu $X|f$. Wymienione pojęcie, tj. pojęcie zbioru domkniętego w grafie, ma duże znaczenie dla minimalizacji automatów. Właśnie metody minimalizacji automatów podane w rozdziale 5 oparte są o wymienione pojęcie.

4.1. Graf zorientowany w zbiorze X

Graf zorientowany skończony jest to uporządkowana dwójka $X|f$, w której X jest dowolnym niepustym skończonym zbiorem elementów, zwanych wierzchołkami grafu, a f jest dowolną funkcją, której argumenty x przebiegają zbiór X, zaś wartości $f(x)$ są podzbiorami zbioru X, przy czym nie wykluczamy $f(x) = \emptyset$ dla pewnych (go) $x \in X$. O takim grafie $X|f$ mówimy, że jest to graf pierwotnie dany.

$f^{-1}(x)$ jest to zbiór, do którego x_1 należy wtedy i tylko wtedy, gdy $x \in f(x_1)$.

$f(A)/f^{-1}(A)$ / jest to zbiór, do którego x_1 należy wtedy i tylko wtedy, gdy istnieje $x \in A$ i $x_1 \in f(x)/x_1 \in f^{-1}(x)$ /.

Przyjmujemy $f(\emptyset) = f^{-1}(\emptyset) = \emptyset$.

Uporządkowaną parę $\langle x_1, x_2 \rangle$ nazywamy łukiem elementarnym w grafie $X|f$, jeżeli $x_2 \in f(x_1)$. Wierzchołek x_1 nazywamy początkiem łuku $\langle x_1, x_2 \rangle$, a wierzchołek x_2 - końcem łuku $\langle x_1, x_2 \rangle$.

Każdy ciąg $U_k = [a_1, a_2, \ldots, a_k, a_{k+1}, \ldots]$, złożony z niepustych łuków elementarnych $a_1, a_2, \ldots, a_k, a_{k+1}, \ldots$, nazywamy łańcuchem w grafie $X|f$, jeżeli koniec łuku a_k jest identyczny z początkiem łuku a_{k+1}, przy czym $[a_1]$ jest łańcuchem w w grafie $X|f$, to znaczy, że jeśli $a_1 = \langle x_1, x_2 \rangle$, to może być $f(x_2) = \emptyset$, ale $x_2 \in f(x_1)$. Początek łuku a_1 nazywamy początkiem łańcucha U_k. Jeżeli k jest liczbą skończoną, to łańcuch U_k jest skończony. Jeżeli łańcuch $[a_1, a_2, \ldots, a_m]$ jest skończony, to koniec łuku a_m nazywamy końcem tego łańcucha.

4.2. Funkcje β i β^{-1}

Mówimy, że wierzchołek x położony jest na łańcuchu E w grafie $X|f$, jeżeli istnieje łuk $\langle a, b \rangle \in E$ i $x = a$ lub $x = b$.

D e f i n i c j a 1. $\beta(x)/\beta^{-1}(x)/$ jest to zbiór o własnościach:

1. $x \in \beta(x)/\ x \in \beta^{-1}(x)/$,

2. $x_1 \in \beta(x) \setminus \{x\}/x_1 \in \beta^{-1}(x) \setminus \{x\}/$ wtedy i tylko wtedy, gdy w grafie $X|f$ istnieje łańcuch o początku x /o końcu x/ i x_1 jest różnym od x wierzchołkiem położonym na tym łańcuchu.

Definicja 1 równoważna jest następującej:

D e f i n i c j a 1a. $\beta(x)/\beta^{-1}(x)/$ jest zbiorem o własnościach:

1. $x \in \beta(x)/\ x \in \beta^{-1}(x)/$,

2. jeżeli $x_1 \in f(x)/\ x_1 \in f^{-1}(x)/$, to $x_1 \in \beta(x)/x_1 \in \beta^{-1}(x)/$

3. jeżeli $x_1 \in \beta(x)/x_1 \in \beta^{-1}(x)/$ i $x_2 \in f(x_1)/x_2 \in f^{-1}(x_1)/$, to $x_2 \in \beta(x)/x_2 \in \beta^{-1}(x)/$,

4. zbiór $\beta(x)/\beta^{-1}(x)/$ innych elementów nie zawiera poza tymi wszystkimi, które można otrzymać na mocy 1., 2. i 3. w skończonej liczbie kroków.

$\beta(A)/\beta^{-1}(A)/$ jest to zbiór, do którego x_1 należy wtedy i tylko wtedy, gdy istnieje $x \in A$ i $x_1 \in \beta(x)/x_1 \in \beta^{-1}(x)/$. Przyjmujemy $\beta(\emptyset) = \beta^{-1}(\emptyset) = \emptyset$.

4.3. Graf częściowy

Oznaczamy $\delta(A) = f(A) \setminus A$. Mamy więc

$$[\delta(A) = \emptyset] = [f(A) \subset A].$$

D e f i n i c j a 2. Przy założeniu, że $\delta(A) = \emptyset$, zbiorowi A przyporządkowujemy graf $A|\hat{f}$, w którym mamy:

$$\left.\begin{array}{l} 1.\ \hat{f}(D) = f(D) \\ 2.\ \hat{f}^{-1}(D) = f^{-1}(D) \cap A \end{array}\right\} \text{ dla } D \in A\,|\,D \subset A|,$$

3. dla podkreślenia, że $D \subset A$ /$D \in A$/ i $\delta(A) = \emptyset$ piszemy $\hat{\beta}^{-1}(D, A)$ zamiast $\beta^{-1}(D) \cap A$.

D e f i n i c j a 3. Przy założeniu, że $\delta(A) \neq \emptyset$, zbiorowi A przyporządkowujemy graf $(A \cup f(A)) \,|\, \hat{\hat{f}} \equiv A|\hat{\hat{f}}$, w którym mamy:

1. $\hat{\hat{f}}(x) = \emptyset$ dla $x \in \delta(A)$,

2. $\hat{\hat{f}}(D) = f(D)$ dla $D \in A$ /$D \subset A$/,

3. $\hat{\hat{f}}^{-1}(D) = f^{-1}(D) \cap A$ dla $D \in A \cup f(A)$ /$D \subset A \cup f(A)$/ ,

4. piszemy $\hat{\hat{\beta}}^{-1}(A)$ zamiast $\beta^{-1}(\delta(A)) \cap A$, przy czym w tym przypadku nie wykluczamy $\delta(A) = \emptyset$.

Graf $A|\hat{f}$ /$A|\hat{\hat{f}}$/ nazywamy grafem częściowym grafu $X|f$.

4.4. Zbiór domknięty w grafie

Zbiór $A \subset X$ nazywamy domkniętym, jeżeli $\delta(A) = \emptyset$. Czasami będziemy mówić, że A jest zbiorem domkniętym w grafie $B|\hat{f}$ /$B|\hat{\hat{f}}$/, jeżeli $A \subset B$ i $\delta(A) = \emptyset$.

T w i e r d z e n i e 1. Zbiór A jest domknięty wtedy i tylko wtedy, gdy $\beta(x) \subset A$ dla każdego $x \in A$, lub też

$$[\sigma(A) = \emptyset] \equiv [\beta(A) = A].$$

Przez $\delta(A)$ oznaczamy podzbiór zbioru A maksymalny domknięty w grafie $A|\hat{f}/A|\hat{f}/$, tj. taki zbiór $D \subset A$, że jeśli $D \subset B \subset A$ i $\sigma(B) = \emptyset$, to $D = \delta(A) = B$.

Mamy więc

$$[\delta(A) = A] \equiv [\sigma(A) = \emptyset].$$

T w i e r d z e n i e 2. Zachodzą relacje:

$$f(A \setminus \hat{\beta}^{-1}(A)) \subset A \setminus \hat{\beta}^{-1}(A); \tag{1}$$

$$f(B \setminus \hat{\beta}^{-1}(D, B)) \subset B \setminus \hat{\beta}^{-1}(D, B). \tag{2}$$

D o w ó d. Udowodnimy relację (1). Dowód relacji (2) jest analogiczny.

Załóżmy, że dla pewnego $x \in f(A \setminus \hat{\beta}^{-1}(A))$ jest $x \notin A \setminus \hat{\beta}^{-1}(A)$. Istnieje więc $x_1 \in A \setminus \hat{\beta}^{-1}(A)$ i $x \in f(x_1)$. Warunki $x \notin A \setminus \hat{\beta}^{-1}(A)$ i $x \in f(x_1)$ oznaczają, że $x \in \sigma(A)$ lub $x \in \beta^{-1}(\sigma(A)) \cap A = \hat{\beta}^{-1}(A)$. Jeżeli $x \in \sigma(A)$, to $x \in \beta^{-1}(x) \subset \beta^{-1}(\sigma(A))$ i $x_1 \in \beta^{-1}(x)$, bo $x \in f(x_1)$. Zatem $x_1 \in \hat{\beta}^{-1}(A)$, ponieważ $x_1 \in A$. Jeżeli $x \in \hat{\beta}^{-1}(A)$, to $x \in \beta^{-1}(\sigma(A))$. Warunki $x \in f(x_1)$ i $x \in \beta^{-1}(\sigma(A))$ oznaczają, że $x_1 \in \beta^{-1}(\sigma(A))$, więc $x_1 \in \hat{\beta}^{-1}(A)$. W każdym przypadku otrzymaliśmy, że $x_1 \in \hat{\beta}^{-1}(A)$ wbrew temu, że $x_1 \in A \setminus \hat{\beta}^{-1}(A)$, o b d d.

T w i e r d z e n i e 3. Zbiór $\delta(A)$ wyraża się w grafie $A|\hat{f}$ wzorem

$$\delta(A) = A \setminus \hat{\beta}^{-1}(A). \tag{3}$$

D o w ó d. Niech $\sigma(A) \neq \emptyset$. Wobec (1) wystarczy dowieść, że $\delta(A) \subset A \setminus \hat{\beta}^{-1}(A)$, bo $A \setminus \hat{\beta}^{-1}(A) \subset \delta(A)$. W tym celu załóżmy, że dla pewnego $x \in \delta(A)$ jest $x \notin A \setminus \hat{\beta}^{-1}(A)$. Zatem

$x \in A$, bo $x \in \delta(A) \subset A$. Warunki $x \in A$ i $x \notin A \setminus \hat{\beta}^{-1}(A)$ oznaczają, że $x \in \beta^{-1}(\sigma(A)) \cap A = \hat{\beta}^{-1}(A)$. Ponieważ $x \in \delta(A)$, to $f(\beta(x)) \subset \beta(x) \subset \delta(A)$. Z drugiej strony $(x \in \hat{\beta}^{-1}(A)) \Rightarrow$ $\Rightarrow x \in \beta^{-1}(\sigma(A))$. Istnieje więc $x_1 \in \sigma(A)$ i $x_1 \in \beta(x) \subset \delta(A)$. Nie może być jednocześnie $x_1 \in \delta(A) \subset A$ i $x_1 \in \sigma(A)$, bo $\sigma(A) \cap A = \emptyset$. Otrzymana niedorzeczność oznacza, że jeśli $x \in \delta(A)$, to nie może być $x \notin A \setminus \hat{\beta}^{-1}(A)$. Zatem $(x \in \delta(A)) \Rightarrow$ $\Rightarrow x \in A \setminus \hat{\beta}^{-1}(A)$, c b d d.

Mówimy, że graf $A|\hat{f}$ posiada własność $\mathfrak{T}$, jeżeli w grafie tym istnieje łańcuch, którego początek i koniec są identyczne i zbiór wszystkich wierzchołków położonych na tym łańcuchu jest zbiorem A.

T w i e r d z e n i e 4. Jeżeli graf $A|\hat{f}$ posiada własność $\mathfrak{T}$, to $\beta(x) = A$ i $\hat{\beta}^{-1}(x,A) = A$ dla każdego $x \in A$.

WN1. Jeżeli graf $A|\hat{f}$ posiada własność $\mathfrak{T}$, to każdy zbiór domknięty w tym grafie, różny od zbioru $\emptyset$, jest zbiorem A (na mocy twierdzenia 4).

4.5. Algorytm znalezienia zbiorów $\beta^{-1}(A)$, $\beta(A)$, $\hat{\beta}^{-1}(D,B)$ i $\hat{\beta}^{-1}(R)$

Weźmy ciąg zbiorów

$$A_j = f(A_{j-1}) \setminus \bigcup_{t=0}^{j-1} A_t \neq \emptyset \quad \text{dla } j = 1,2,\ldots, p \text{ oraz} \qquad (4)$$

$$A_{p+1} = \emptyset.$$

Oznaczamy

$$\beta'(A_0) = \begin{cases} A_0 & \text{, jeżeli } A_1 = \emptyset \\ \bigcup_{j=0}^{p} A_j & \text{, jeżeli } A_1 \neq \emptyset. \end{cases} \qquad (5)$$

a. Przyjmując w (4) $f' = f^{-1}/f/$ oraz $A_0 = A$ - otrzymamy zbiór $\beta'(A_0) = \beta^{-1}(A)\ /\beta(A)/$ wyrażający się wzorem (5). Zatem także znajdziemy zbiór $X \setminus \beta^{-1}(A)$.

b. Przyjmując w (4) $A_0 = D \subset B$, $\delta(B) = \emptyset\ /A_0 = \delta(R) \neq \emptyset/$ oraz $f'(A_{j-1}) = f^{-1}(A_{j-1}) \cap B\ /f'(A_{j-1}) = f^{-1}(A_{j-1}) \cap R/$ - otrzymamy zbiór $\beta'(A_0) = \hat{\beta}^{-1}(D,B)\ /\ \hat{\beta}^{-1}(R) = \beta'(A_0) \setminus A_0/$ wyrażający się wzorem (5). Zatem także znajdziemy zbiory $B \setminus \hat{\beta}^{-1}(D,B)$ i $\delta(R) = R \setminus \hat{\beta}^{-1}(R)$.

Powyższy algorytm nazywamy algorytmem II.

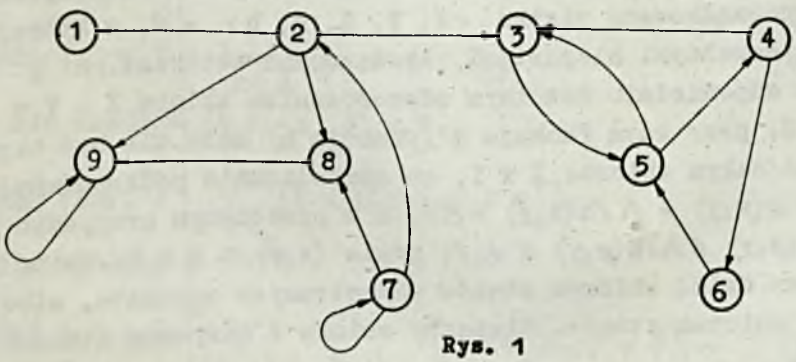
W praktyce ciąg (4) zapisujemy

$$A_0 \overset{\beta'}{\Longrightarrow} A_1 \Longrightarrow A_2 \Longrightarrow \dots \Longrightarrow A_p \Longrightarrow A_{p+1} = \emptyset, \qquad (6)$$

przy czym symbol $A_0 \overset{\beta'}{\Longrightarrow}$ zastępujemy w przypadku a. i b. odpowiednio symbolem:

$$A \overset{\beta^{-1}}{\Longrightarrow} /A \overset{\beta}{\Longrightarrow} /;\ (D,B) \overset{\hat{\beta}^{-1}}{\Longrightarrow} /R \xrightarrow{f} f(R) \Longrightarrow \delta(R) \overset{\hat{\beta}^{-1}}{\Longrightarrow} /.$$

P R Z Y K Ł A D. Graf $X|f$ dla $X = \{1,2,\dots,9\}$ pokazano na rys. 1.



Rys. 1

Z rys. 1 otrzymujemy:

1. Dla $A_0 = \{1,2,3\}$ z funkcją β^{-1}, ciąg (6) jest $A_0 \overset{\beta^{-1}}{\Longrightarrow} \{7,4,5\} \Longrightarrow \{6\} \Rightarrow \emptyset$, skąd $\beta^{-1}(A_0) = \{1,2,3,7,4,5,6\}$,

2. Dla $A = \{1,2,8,9,3\}$ z funkcją $\hat{\beta}^{-1}$, ciąg (6) jest
$A \xrightarrow{f} f(A) = \{3,8,1,9,5\} \Longrightarrow \delta(A) = \{5\} \overset{\hat{\beta}^{-1}}{\Longrightarrow} \{3\} \Longrightarrow \{2\} \Longrightarrow \emptyset$,
skąd $\hat{\beta}^{-1}(A) = \{3,2\}$, a więc $\delta(A) = \{1,8,9\}$.

5. RODZINY DOMKNIĘTE AUTOMATU

Głównym celem niniejszego rozdziału jest podanie algorytmów i metod znalezienia rodzin domkniętych automatu. Zagadnienie to pozostaje w ścisłym związku z pojęciem minimalizacji automatu. Znalezienie bowiem odpowiedniej rodziny domkniętej A równoważne jest, na mocy zawartych tu twierdzeń 6 i 7, znalezieniu automatu, który jest uproszczeniem danego automatu. Taką rodzinę A nazywamy rozwiązaniem automatu. Z drugiej strony pojęcie domkniętej rodziny automatu jest pojęciem zbioru domkniętego w odpowiednim grafie. Zatem niniejszy rozdział stanowi zasadniczy trzon artykułu poświęcony minimalizacji automatów.

5.1. Automat. Określenie grafu w zbiorze 2^X za pośrednictwem odwzorowania g (automatu)

Automat skończony częściowo określony albo niezupełny jest to uporządkowana piątka $\langle X, Y, Z, g, h \rangle = \mathcal{A}$, w której X, Y i Z są dowolnymi niepustymi skończonymi zbiorami, a g i h jest odpowiednio dowolnym odwzorowaniem zbioru X x Y w zbiór X i Z, przy czym funkcja g (funkcja h) może nie być określona na całym zbiorze X x Y, co symbolicznie podkreślamy, pisząc $g(x,y) = \Lambda$ /$h(x,y) = \Lambda$/, a w przeciwnym przypadku piszemy $g(x,y) \neq \Lambda / h(x,y) \neq \Lambda$ /, gdzie $(x,y) \in X \times Y$. Zbiór X nazywamy dalej zbiorem stanów wewnętrznych automatu, albo krótko - zbiorem stanów. Elementy zbioru X nazywamy stanami automatu.

Odwzorowanie g i h, albo automat, opisujemy równoważnie w postaci pewnej tabeli automatu: wiersze tej tabeli oznaczamy

symbolami różnych elementów zbioru X, a kolumny - symbolami różnych elementów zbioru Y. W punkcie przecięcia się x-tego wiersza z y-tą kolumną tabeli automatu znajduje się para uporządkowana $(g(x,y), h(x,y))$ elementów $g(x,y)$ i $h(x,y)$ oddzielonych przecinkiem. Jeżeli $g(x,y)=\Lambda / h(x,y) = \Lambda /$, to zamiast symbolu Λ wpisujemy równoważny mu symbol -, czyli kreskę.

P R Z Y K Ł A D. Tabela T1 jest opisem automatu, w którym $X = \{1,2,3,4,5,6\}$; $Y = \{y_1, y_2, y_3\}$ i $Z = \{c,d\}$. Z tabeli T1 wynika, że $g(1, y_1) = 1$, $g(4, y_2) = \Lambda$, $h(2, y_1) = c$, $h(4, y_2) = \Lambda$ itd.

	y_1	y_2	y_3
1	1,-	2,-	3,-
2	-,c	3,-	-,-
3	4,-	6,-	5,c
4	-,d	-,-	2,-
5	2,-	5,c	2,d
6	6,-	-,d	-,-

$\Rightarrow$

	y_1	y_2	y_3
1	{1}	∅	∅
2	{5}	{1}	{4,5}
3	∅	{2}	{1}
4	{3}	∅	∅
5	∅	{5}	{3}
6	{6}	{3}	∅

T1 - Tabela automatu

T1.1 - Tabela odwzorowania λ

Odwzorowanie U i φ określamy następująco:

$$U(A,y) \overset{\text{def}}{=\!=} \{x_1 \in X : \bigvee_{x \in A} [x_1 = g(x,y)]\} \tag{1}$$

dla każdego $(A,y) \in 2^X \times Y$,

$$\varphi(A) \overset{\text{def}}{=\!=} \{B \subset X : \bigvee_{y \in Y} [B = U(A,y) \neq \emptyset]\} \tag{2}$$

dla każdego $A \in 2^X$,

$$\varphi(\overline{A}) \overset{\text{def}}{=\!=} \{B \subset X : \bigvee_{(A,y) \in \overline{A} \times Y} [B = U(A,y) \neq \emptyset]\} = \tag{3}$$

$$= \{B \subset X : \bigvee_{A \in \overline{A}} [B \in \varphi(A)]\} \text{ dla każdego } \overline{A} \subset 2^X.$$

Przyjmujemy $\varphi(\emptyset) = \emptyset$.

W rozdziale 4. mieliśmy dany graf $X|f$. Z uwagi na dowolność zbioru X w grafie $X|f$ przyjmijmy zamiast zbioru X - zbiór 2^X, gdzie, oczywiście, X jest zbiorem stanów wewnętrznych automatu $\mathcal{A}$. Otrzymamy graf $2^X|f$ o zbiorze wierzchołków 2^X. Tak więc nie wprowadzając nowych symboli umawiamy się dalej, że wszystkie funkcje oraz ich wartości wynikające z określenia funkcji f w zbiorze X, albo wynikające z określenia grafu $X|f$ w rozdziale 4, są tymi samymi symbolami i tak samo są określone w przypadku grafu $2^X|f$ z tym, że obecnie przez symbol f rozumiemy funkcję określoną następująco:

$$f(A) \overset{\text{def}}{=\!=} \varphi(A) \setminus U(X) \quad \text{dla każdego } A \in 2^X. \tag{4}$$

Zauważmy, że $2^X|\varphi$ też jest grafem, ale w tym przypadku interesuje nas tylko funkcja φ i dlatego umowa przyjęta w przypadku grafu $2^X|f$ nie dotyczy grafu $2^X|\varphi$.

Wobec (4) mamy, np.:

$$f(A) = \Big\{B \subset X : \bigvee_{(A,y) \in A \times Y} \Big[B = U(A,y) \neq \emptyset \tag{5}$$

$$\text{i } B \notin U(X)\Big]\Big\}$$

Z (4) wynika, że $f((X)^2) \subset (X)^2 \ / \sigma((X)^2) = \emptyset /$.

Graf $(X)^2|\hat{f}$ nazywamy grafem wszystkich par stanów[1] automatu $\mathcal{A}$.

5.2. Związek pomiędzy funkcjami f^{-1} i λ w przestrzeni $(X)^2$

Odwzorowanie λ określamy następująco:

$$\lambda(x,y) \overset{\text{def}}{=\!=} \{x_1 \in X : g(x_1,y) = x\} \text{ dla każdego} \tag{6}$$

$$(x,y) \in X \times Y.$$

[1] Pod słowem "para stanów" rozumiemy element zbioru $(X)^2$ w przypadku automatu $\mathcal{A}$.

Z tabeli automatu łatwo otrzymamy na mocy (6) następującą tabelę odwzorowania λ: jeżeli dla każdego $(x,y) \in X \times Y$ w punkcie przecięcia się x-tego wiersza z y-tą kolumną tabeli automatu umieścimy zamiast pary $(g(x,y), h(x,y))$ odpowiedni zbiór, tj. zbiór $\lambda(x,y)$, to tak otrzymana nowa tabela jest tabelą odwzorowania λ. Tabelę tę nazywamy równoważnie antytabelą stanów automatu.

P R Z Y K Ł A D. Tabela T1.1 jest antytabelą stanów automatu zadanego tabelą T1. Z tabeli T1.1 mamy:

$$\lambda(1,y_1) = \{1\}, \quad \lambda(4,y_2) = \emptyset \quad \lambda(2,y_3) = \{4,5\} \text{ itd.}$$

T w i e r d z e n i e 1. Niech $K_1 = \{x_1,x_2\}$, $K = \{x', x''\} \in (X)^2$. Wówczas $K \in f^{-1}(K_1)$ wtedy i tylko wtedy, gdy istnieje $y \in Y$ taki, że $x' \in \lambda(x_1,y)$ i $x'' \in \lambda(x_2,y)$ lub $x' \in \lambda(x_2,y)$ i $x'' \in \lambda(x_1,y)$.

D o w ó d. Jeśli $K \in f^{-1}(K_1)$, to $K_1 \in f(K)$. Istnieje więc $y \in Y$ i $K_1 = U(K,y)$ (patrz wzór (1) i (4)). Oznacza to, wobec (1), że $x_1 = g(x', y)$ i $x_2 = g(x'', y)$ lub $x_1 = g(x'',y)$ i $x_2 = g(x', y)$, więc na mocy (6) mamy $x' \in \lambda(x_1,y)$ i $x'' \in \lambda(x_2,y)$ lub $x'' \in \lambda(x_1,y)$ i $x' \in \lambda(x_2,y)$.

Dowód w odwrotną stronę. Z definicji odwzorowania g i λ wynika, że: jeżeli $x_1 \neq x_2$ i $y \in Y$ oraz $x' \in \lambda(x_1,y)$ i $x'' \in \lambda(x_2,y)$, to także $x' \neq x''$. Zatem, jeśli $x' \in \lambda(x_1,y)$ i $x'' \in \lambda(x_2,y)$ dla pewnego $y \in Y$, to $x' \neq x''$ i na mocy (6) mamy $x_1 = g(x', y)$ i $x_2 = g(x'', y)$, więc $K_1 = U(K,y)$, skąd $K_1 \in f(K)$. Zatem $K \in f^{-1}(K_1)$. Wobec poprzedniej części dowodu twierdzenie jest więc udowodnione.

Z twierdzenia 1 mamy wzór:

$$f^{-1}(A^2) \cap (X)^2 = \left\{\{x', x''\} \in (X)^2 : \bigvee_{(K,y) \in A^2 \times Y} \left[x' \in \lambda(x_1,y) \right.\right. \quad (7)$$

$$\left.\left. \text{i } x'' \in \lambda(x_2,y)\right]\right\}, \text{ gdzie } K = \{x_1,x_2\}.$$

Zauważmy podobieństwo (symetrię) wzoru (7) i wzoru

$$f(A^2) = \left\{\{x',x''\} \in (X)^2 : \bigvee_{(K,y) \in A^2 \times Y} \left[x' = g(x_1,y) \text{ i } x'' = g(x_2,y)\right]\right\}.$$

5.3. Relacja zgodności i pokrywania stanów automatu

W dowolnym niepustym zbiorze A każdy ciąg złożony z elementów tego zbioru nazywamy słowem.

Przez $y, y_1, y_2, \ldots$ oznaczamy elementy zbioru Y. Słowo $S_y = y_1\, y_2 \ldots y_m$ nazywamy dopuszczalnym dla stanu $x_1 \in X$, jeżeli ciąg $x_{j+1} = g(x_j, y_j)$, $j = 1,2,\ldots, m$, posiada tę własność, że $x_k \neq \Lambda$ dla $k = 1,2,\ldots, m$.

Zauważmy, że dla każdego $(x,y) \in X \times Y$ słowo złożone tylko z elementu y jest dopuszczalne dla stanu $x \in X$ bez względu na to, czy $g(x,y) \neq \Lambda$, czy też $g(x,y) = \Lambda$.

Przez A_1 oznaczamy dowolny automat niezupełny postaci $\langle Q, Y, Z, g', h' \rangle = A_1$.

D e f i n i c j a 1. Stan $x_1 \in X$ i stan $q_1 \in Q$ nazywamy zgodnymi, jeżeli z tego, że $S_y = y_1\, y_2 \ldots y_m$ jest dowolnym słowem dopuszczalnym jednocześnie dla stanu x_1 i q_1 wynika $z_j = z'_j$ dla każdego takiego wskaźnika j $(1 \leq j \leq m)$, że $z_j \neq \Lambda$ i $z'_j \neq \Lambda$, gdzie:

$$\begin{cases} x_{j+1} = g(x_j, y_j), \\ z_j = h(x_j, y_j), \\ q_{j+1} = g'(q_j, y_j), \\ z'_j = h'(q_j, y_j), \end{cases}$$

dla $j = 1,2,\ldots, m$.

D e f i n i c j a 2. Mówimy, że stan $q_1 \in Q$ pokrywa stan $x_1 \in X$, jeżeli:

1. każde słowo w zbiorze Y dopuszczalne dla stanu x_1 jest dopuszczalne dla stanu q_1,

2. jeżeli z tego, że $S_y = y_1 y_2 \dots y_m$ jest dowolnym słowem dopuszczalnym jednocześnie dla stanu x_1 i q_1 wynika $z_j = z'_j$ dla każdego takiego wskaźnika j $(1 \leqslant j \leqslant m)$, że $z_j \neq \Lambda$, gdzie ciągi z_j, z'_j, x_{j+1} i q_{j+1} dla słowa S_y są określone w definicji 1.

Relacja zgodności stanów jest zwrotna i symetryczna. Relacja pokrywania stanów jest zwrotna i przechodnia.

Pod słowem "automat" rozumiemy dalej automat $\mathcal{A}$, jeżeli nie będzie powiedziane inaczej. Przez $\bar{D}_{\mathcal{A}}$ i $D_{\mathcal{A}}$ oznaczamy odpowiednio zbiór wszystkich par stanów zgodnych i zbiór wszystkich par stanów niezgodnych (które nie są zgodne) automatu.

W oparciu o definicje 1 i 2 łatwo dowodzi się następujące dwa twierdzenia:

T w i e r d z e n i e 2. Jeżeli A jest zbiorem stanów automatu $\mathcal{A}$ pokrytych przez pewien stan $q \in Q$ (stan $x \in X$) i $K \in (A)^2$, to $K \in \bar{D}_{\mathcal{A}}$. Zatem $(A)^2 \subset \bar{D}_{\mathcal{A}}$.

T w i e r d z e n i e 3. Jeżeli stan $q \in Q$ pokrywa stan $x \in X$ i $g(x,y) \neq \Lambda$ dla pewnego $y \in Y$, to:

1. $g'(q,y) \neq \Lambda$,
2. stan $g'(q,y)$ pokrywa stan $g(x,y)$.

5.4. Pierwszy krok minimalizacji automatu

Do pierwszego kroku minimalizacji automatu zaliczamy znalezienie zbiorów D_h, $D_{\mathcal{A}}$ i $\bar{D}_{\mathcal{A}}$ oraz rodzin $m(X, \bar{D}_{\mathcal{A}})$ i $m(X, D_{\mathcal{A}})$, jak niżej.

Oznaczamy

$$D_h \overset{\text{def}}{=\!=} \left\{\{x_1, x_2\} \in (X)^2 : \bigvee_{y \in Y} \left[h(x_1, y) \neq h(x_2, y) \text{ oraz } h(x_1, y) \neq \Lambda \text{ i } h(x_2, y) \neq \Lambda\right]\right\}. \tag{8}$$

Z definicji 1 wynika, że

$$\bar{D}_A = (X)^2 \setminus D_A \text{ i } D_h \subset D_A. \tag{9}$$

T w i e r d z e n i e 4. Zbiór D_A w grafie $(X)^2 | \hat{f}$ wyraża się wzorem

$$D_A = \hat{\beta}^{-1}(D_h, (X)^2). \tag{10}$$

D o w ó d. Niech $A = A_1$, czyli $Q = X$, $g' = g$ i $h' = h$. Załóżmy, że $S_y = y_1 y_2 \ldots y_m$ jest dowolnym słowem dopuszczalnym jednocześnie dla stanu $x_1 \in X$ i dla stanu $q_1 \in X$. Wobec definicji 1 mamy następujące dwa ciągi albo przyporządkowanie:

$$\text{(I)} \quad \begin{cases} x_1 \xrightarrow{y_1/z_1} x_2 \xrightarrow{y_2/z_2} x_3 \ldots \xrightarrow{y_m/z_m} x_{m+1}, \\ q_1 \xrightarrow{y_1/z'_1} q_2 \xrightarrow{y_2/z'_2} q_3 \ldots \xrightarrow{y_m/z'_m} q_{m+1}. \end{cases}$$

Załóżmy, że $K_1 = \{x_1, q_1\} \in D_A$. Na mocy definicji 1 istnieje wskaźnik j /$1 \leq j < m$/ taki, że $z_j \neq z'_j$ oraz $z_j \neq \Lambda$ i $z'_j \neq \Lambda$, bo w przeciwnym przypadku mielibyśmy, z uwagi na dowolność słowa S_y, że $K_1 \in \bar{D}_A$ wbrew założeniu. Dowodzi to, że dla każdego $K_1 \in D_A \setminus D_h$ istnieje $K \in \beta(K_1)$ i $K \in D_h$, więc $K_1 \in \hat{\beta}^{-1}(K, (X)^2) \subset \hat{\beta}^{-1}(D_h, (X)^2)$, skąd $K_1 \in \hat{\beta}^{-1}(D_h, (X)^2) \setminus D_h$. Mamy więc $D_A \subset \hat{\beta}^{-1}(D_h, (X)^2)$.

Dowód w odwrotną stronę wynika z przyporządkowania (I), bo jeżeli $K_m = \{x_m, q_m\} \in D_h$, to $\hat{\beta}^{-1}(K_m, (X)^2) \subset D_A$, a także $\hat{\beta}^{-1}(D_h, (X)^2) \subset D_A$. Wobec poprzedniej części dowodu równość (10) jest więc udowodniona.

P R Z Y K Ł A D. Zwróćmy uwagę, że na mocy wzoru (7) można korzystać z antytabeli stanów automatu tylko w zbiorze $(X)^2$, czyli w grafie $(X)^2|\hat{f}$ lub w grafie częściowym tego grafu. Łatwo więc znajdziemy zbiory $A^2 \setminus \hat{\beta}^{-1}(B^2, A^2)$ i $\delta(D^2) = D^2 \setminus \hat{\beta}^{-1}(D^2)$, mając antytabelę stanów automatu z jednej strony, a z drugiej - tabelę automatu, w oparciu o wzór (7) i algorytm II.

Zbiór D_h znajdziemy na mocy (8) mając tabelę automatu, a następnie znajdziemy zbiór D_A na mocy (10) w oparciu o wzór (7) i algorytm II, a więc korzystając z antytabeli stanów automatu. Na mocy (9) znajdziemy więc także zbiór $\bar{D}_A$. Mając zbiory D_A i $\bar{D}_A$ znajdziemy na mocy algorytmu I rodziny $m(X, \bar{D}_A)$ i $m(X, D_A)$.

Umówmy się ponadto, że zbiory, jako elementy pewnej rodziny **A**, np. zbiór $A = \{1,2,3\} \in \mathbf{A}$, będziemy w przykładach zapisywać w postaci słowa złożonego ze wszystkich elementów zbioru A, czyli $123 \equiv A = \{1,2,3\}$. Podobnie będziemy zapisywać zbiory, jako wierzchołki pewnego grafu na rysunku tego grafu. Dwuznaczność może wystąpić w takim przypadku, gdy A składa się z jednego elementu $x \in X$, ale wówczas, jeżeli to będzie budzić wątpliwości, będziemy pisać $A = \{x\}$.

1. Z tabeli T1 mamy:

$$D_h \equiv \{24, 56, 35\} . \qquad /1/$$

2. Z tabeli T1.1 dla zbioru /1/ mamy

$$(D_h, (X)^2) \xrightarrow{\hat{\beta}^{-1}} \{25,13\} \Longrightarrow \{15,34\} \Longrightarrow \emptyset, \text{ skąd}$$

$$D_A = \hat{\beta}^{-1}(D_h, (X)^2) \equiv \{25,13,15,34,24,56,35\} \equiv \qquad /2/$$

$$\equiv \{1(35), 2(45), 3(45), 4(\emptyset), 5(6)\} \ ^{1)}.$$

3. Ze wzoru (9), wobec /2/ i $X = \{1,2,\ldots,6\}$ mamy

---

1) W zależności od wygody stosujemy zapis podzbiorów zbioru $(X)^2$ zgodnie z umową przyjętą w rozdziale 3, patrz algorytm I

$$\bar{D}_{\mathcal{A}} = (X)^2 \setminus D_{\mathcal{A}} \equiv \{1(246), 2(36), 3(6), 4(56), 5(\emptyset)\}. \qquad /3/$$

4. Na mocy algorytmu I, wobec /2/ i /3/ otrzymamy:

$$m(X, \bar{D}_{\mathcal{A}}) \equiv \{146, 126, 236, 45\}, \qquad /4/$$

$$m(X, D_{\mathcal{A}}) \equiv \{135, 24, 25, 34, 56\}. \qquad /5/$$

5.5. Relacja pokrywania automatów

D e f i n i c j a 3. Będziemy mówić, że automat $\mathcal{A}_1$ pokrywa automat $\mathcal{A}$, albo że jest równoważny automatowi $\mathcal{A}$, jeżeli dla każdego stanu $x \in X$ istnieje pokrywający go stan $q \in Q$.

Relacja pokrywania automatów niezupełnych jest zwrotna i przechodnia.

Przez $\mathbf{R}_{\mathcal{A}}$ oznaczamy rodzinę wszystkich zbiorów stabilnych symbolu: $\langle X, \bar{D}_{\mathcal{A}} \rangle$.

Odwzorowanie ε określamy:

$$\varepsilon(A,y) = \{z \in Z : \bigvee_{x \in A} [z = h(x,y)]\} \text{ dla każdego } (A,y) \in 2^X \times Y.$$

T w i e r d z e n i e 5. Jeżeli $(A,y) \in \mathbf{R}_{\mathcal{A}} \times Y$, to

$$\varepsilon(A,y) = \emptyset \text{ lub } \varepsilon(A,y) = \{z\} \text{ i } z \in Z.$$

D o w ó d. Przypuśćmy, że $\{z_1, z_2\} \subset \varepsilon(A,y)$ dla pewnego $(A,y) \in \mathbf{R}_{\mathcal{A}} \times Y$. Zatem istnieją $x_1, x_2 \in A$ $/x_1 \neq x_2/$ takie, że $z_1 = h(x_1,y) \neq h(x_2,y) = z_2$ oraz $z_1 \neq \Lambda$ i $z_2 \neq \Lambda$. Wobec (8) oznacza to, że $\{x_1, x_2\} \in D_h \subset D_{\mathcal{A}}$ wbrew temu, że $\{x_1, x_2\} \in \bar{D}_{\mathcal{A}}$, bo $\{x_1, x_2\} \in (A)^2 \subset \bar{D}_{\mathcal{A}}$. Otrzymana sprzeczność kończy dowód.

Przez $\mathcal{Z}(\mathbf{R}_{\mathcal{A}})$ oznaczamy zbiór (rodzin) wszystkich takich $\mathbf{A} \subset \mathbf{R}_{\mathcal{A}}$, że $S(\mathbf{A}) = X$ i $\varphi(\mathbf{A}) \to \mathbf{A}$.

T w i e r d z e n i e 6. Jeżeli automat $\mathcal{A}_1$ pokrywa automat $\mathcal{A}$, to istnieje $\mathbf{A} \in \mathfrak{X}(\mathbf{R}_{\mathcal{A}})$ i card $(\mathbf{A}) \leq$ card (Q).

D o w ó d. Jeżeli automat $\mathcal{A}_1$ pokrywa automat $\mathcal{A}$, to dla każdego stanu $x \in X$ istnieje pokrywający go stan $q \in Q$.

Niech A_q, gdzie $q \in Q$, będzie zbiorem wszystkich stanów automatu $\mathcal{A}$ pokrytych przez stan q. Niech $A_q \in \mathbf{A}$ wtedy i tylko wtedy, gdy istnieje $q \in Q$ taki, że $A_q \neq \emptyset$.

Mamy więc $S(\mathbf{A}) = X$ i $\mathbf{A} \subset \mathbf{R}_{\mathcal{A}}$, bo $A_q \in \mathbf{R}_{\mathcal{A}}$ na mocy twierdzenia 2. Ponadto card $(\mathbf{A}) \leq$ card (Q).

Wystarczy dowieść, że $\varphi(\mathbf{A}) \rightarrow \mathbf{A}$. W tym celu niech $B \in \varphi(\mathbf{A})$. Istnieje więc $(A_q, y) \in \mathbf{A} \times Y$ i $B = \bigcup (A_q, y)$ oraz istnieją $x \in A_q$ i $x' \in B$ takie, że $g(x,y) = x' \neq \Lambda$, bo $B \neq \emptyset$. Wobec twierdzenia 3 mamy $g'(q,y) = q_1 \neq \Lambda$ i stan q_1 pokrywa stan x'. Oznacza to, że B jest zbiorem stanów automatu $\mathcal{A}$ pokrytych przez stan $q_1 \in Q$, a więc $A_{q_1} \in \mathbf{A}$. Wobec definicji zbioru A_{q_1} mamy $B \subset A_{q_1}$, bo q_1 pokrywa każdy stan $x \in B$. Udowodniliśmy zatem, że $\{B\} \rightarrow \mathbf{A}$, co wobec $B \in \varphi(\mathbf{A})$ daje $\varphi(\mathbf{A}) \rightarrow \mathbf{A}$ na mocy dowolności zbioru B, obdd.

T w i e r d z e n i e 7. Jeżeli $\mathbf{A} \in \mathfrak{X}(\mathbf{R}_{\mathcal{A}})$, to automat (niezupełny) $\mathcal{A}_{\mathbf{A}} = \langle \mathbf{A}, Y, Z, \varepsilon_1, \varepsilon_2 \rangle$ pokrywa automat $\mathcal{A}$, gdzie:

1. funkcja ε_1 przyporządkowuje każdemu $(R,y) \in \mathbf{A} \times Y$ określony element $\varepsilon_1(R,y) = A \in \mathbf{A}$ taki, że $A \supset \bigcup (R,y) = P$, jeżeli $P \neq \emptyset$, w przeciwnym przypadku $\varepsilon_1(R,y)$ nie jest elementem określonym, czyli $\varepsilon_1(R,y) = \Lambda$,

2. funkcja ε_2 przyporządkowuje każdemu $(R,y) \in \mathbf{A} \times Y$ określony element $\varepsilon_2(R,y) = z \in \varepsilon(R,y)$, jeżeli $\varepsilon(R,y) \neq \emptyset$, a w przeciwnym przypadku $\varepsilon_2(R,y)$ nie jest elementem określonym, czyli $\varepsilon_2(R,y) = \Lambda$.

D o w ó d. Niech $S_y = y_1 y_2 \dots y_m$ będzie dowolnym słowem dopuszczalnym dla stanu $x_1 \in X$. Istnieją zatem ciągi $x_{j+1} = g(x_j, y_j)$, $z_j = h(x_j, y_j)$, $j = 1,2,\dots,m$, przy czym $x_k \neq \Lambda$ dla $k = 1,2,\dots, m$.

Jeżeli $S(A) = X$, to dla każdego x_j /$1 \leq j \leq m$/ istnieje $A_j \in A$ i $x_j \in A_j$, a ponieważ $\varphi(A) \to A$, to wobec 1. mamy

$$A \ni \varepsilon_1(A_j, y_j) = A_{j+1} \supset U(A_j, y_j) \ni g(x_j, y_j) = x_{j+1}\big((A_j, y_j) \in A \times Y\big),$$

$j = 1,2,\dots, m$. Zatem $A_k \neq \Lambda$ dla $k = 1,2,\dots, m$. Oznacza to, że słowo S_y jest dopuszczalne dla stanu $A_1 \in A$.

Jeśli $A \subset R_A$, to wobec twierdzenia 5, na mocy 2. mamy $\varepsilon_2(A_j, y_j) = z_j' \in Z$, jeżeli tylko $\varepsilon(A_j, y_j) \neq \emptyset\big(1 \leq j \leq m\big)$, a ponieważ $x_j \in A_j$, to $z_j = z_j'$, jeżeli tylko $z_j \neq \Lambda$, co wobec poprzedniej części dowodu oznacza, że stan $A_1 \in A$ pokrywa stan $x_1 \in X$. Wobec dowolności stanu $x_1 \in X$ twierdzenie jest udowodnione.

Z tabeli automatu łatwo otrzymamy następującą tabelę rodziny A: niech $(A,y) \in A \times Y$. W punkcie przecięcia się A-tego wiersza z y-tą kolumną tabeli rodziny A znajduje się para uporządkowana $(U(A,y), \varepsilon(A,y))$ elementów $U(A,y)$ i $\varepsilon(A,y)$ oddzielonych przecinkiem. Jeżeli $U(A,y) = \emptyset$ $(\varepsilon(A,y) = \emptyset)$, to zamiast symbolu $\emptyset$ wpisujemy symbol $-$, czyli kreskę, a w przeciwnym przypadku najlepiej jest zbiór $U(A,y)$ $(\varepsilon(A,y))$ zapisać w postaci słowa.

Mając tabelę rodziny $A \in \mathfrak{A}(R_A)$ łatwo znajdziemy tabelę automatu A_A w oparciu o twierdzenie 7. Przykładem jest tabela T2 (dla automatu zadanego tabelą T1), z której otrzymuje się tabelę T2.1, tj. tabelę odpowiedniego automatu A_A dla $A = \{k_1, k_2, k_3, k_4\}$.

	y_1	y_2	y_3
k_1 146	k_1 16,d*)	k_2 2,d	k_3 23,-
k_2 126	k_2 16,c	k_3 23,d	k_3 3,-
k_3 236	k_1 46,c	k_3 36,d	k_4 5,c
k_4 45	k_3 2,d	k_4 5,c	k_3 2,d

T2 - Tabela rodziny /4/

$\Rightarrow$

	y_1	y_2	y_3
k_1	k_1,d	k_2,d	k_3,-
k_2	k_2,c	k_3,d	k_3,-
k_3	k_1,c	k_3,d	k_4,c
k_4	k_3,d	k_4,c	k_3,d

T2.1 - Tabela automatu

Jeżeli automat A_1 pokrywa automat A i card $(Q) <$ card(X), to automat A_1 nazywamy uproszczeniem automatu A, a sam proces znalezienia takiego automatu A_1, gdy mamy zadany automat A, nazywamy minimalizacją automatu A, a ogólniej - minimalizacją automatów.

W klasie (zbiorze) wszystkich automatów pokrywających automat A istnieje (istnieją) automat złożony z najmniejszej liczby stanów wewnętrznych. Automat taki nazywamy automatem minimalnym automatu A. Jeżeli moc zbioru wszystkich stanów wewnętrznych każdego automatu minimalnego automatu A nie jest mniejsza od liczby card (X), to mówimy wówczas, że automat A nie jest upraszczalny.

5.6. Rozwiązanie. Zbiór $\mathfrak{D}(\mathbf{R}_A)$

Przez $\mathcal{R}(\mathbf{R}_A)$ oznaczamy zbiór wszystkich takich $\mathbf{A} \in \mathcal{X}(\mathbf{R}_A)$, że card $(\mathbf{A}) <$ card (X).

Każdy element $\mathbf{A} \in \mathcal{R}(\mathbf{R}_A)$ nazywamy rozwiązaniem automatu A, albo krótko - rozwiązaniem. Rozwiązanie złożone z najmniejszej liczby zbiorów spośród wszystkich rozwiązań nazywamy minimalnym.

Na mocy twierdzenia 6 i 7 zagadnienie znalezienia automatu, który jest uproszczeniem automatu A, jest równoważne zagadnieniu znalezienia rozwiązania, tj. rodziny $\mathbf{A} \in \mathcal{R}(\mathbf{R}_A)$.

*) Mamy $U(146,y_1) = \{1,6\} \equiv 16$ i $\varepsilon(146,y_1) = \{d\} \equiv d$, gdzie $146 \equiv \{1,4,6\}$.

Jeśli bowiem znana jest taka rodzina $\mathbf{A}$, to na mocy twierdzenia 7 znajdziemy automat $\mathcal{A}_{\mathbf{A}}$. Automat ten jest oczywiście uproszczeniem automatu $\mathcal{A}$. Jeśli przy tym $\mathbf{A}$ jest rozwiązaniem minimalnym, to $\mathcal{A}_{\mathbf{A}}$ jest automatem minimalnym automatu $\mathcal{A}$. Zamiast więc mówić dalej o automacie, który jest uproszczeniem automatu $\mathcal{A}$ - będziemy mówić o rozwiązaniu

Mamy $(A \subset B \text{ i } y \in Y) \Longrightarrow \left(U(A, y) \subset U(B,y)\right)$. Zatem także

$$(\mathbf{A} \dashv \mathbf{B}) \Longrightarrow \left(\varphi(\mathbf{A}) \dashv \varphi(\mathbf{B}) \text{ i } f(\mathbf{A}) \dashv f(\mathbf{B})\right); \tag{11}$$

$$(\mathbf{A} \overset{r}{=} \mathbf{B}) \Longrightarrow \left(\varphi(\mathbf{A}) \overset{r}{=} \varphi(\mathbf{B}) \text{ i } f(\mathbf{A}) \overset{r}{=} f(\mathbf{B})\right). \tag{12}$$

Ponieważ $\mathbf{A} \overset{r}{=} \eta(\mathbf{A})$, to wobec (12) także $\varphi(\mathbf{A}) \overset{r}{=} \varphi(\eta(\mathbf{A}))$. Stąd wynika, że spośród dwóch rozwiązań $\mathbf{A}$ i $\eta(\mathbf{A})$, jako rozwiązanie bierzemy rodzinę $\eta(\mathbf{A})$, gdyż w praktyce zależy nam na rozwiązaniu o mocy możliwie najmniejszej, a oczywiście $\operatorname{card}(\mathbf{A}) \geq \operatorname{card}(\eta(\mathbf{A}))$, bo $\eta(\mathbf{A}) \subset \mathbf{A}$. Dlatego dalej umawiamy się, że jeśli $\mathbf{A} \in \mathcal{R}(\mathbf{R}_{\mathcal{A}})$, to $\mathbf{A} = \eta(\mathbf{A})$.

T w i e r d z e n i e 8. Zachodzi relacja $\varphi(\mathbf{R}_{\mathcal{A}}) \subset \mathbf{R}_{\mathcal{A}}$.

D o w ó d. Jeśli $B \in \varphi(\mathbf{R}_{\mathcal{A}})$, to $B = U(A,y)$ dla pewnego $(A,y) \in \mathbf{R}_{\mathcal{A}} \times Y$. Załóżmy, że $(B)^2 \not\subset \bar{D}_{\mathcal{A}}$. Zatem istnieje $K \in (B)^2$ i $K \in D_{\mathcal{A}}$ oraz istnieje $K_1 \in (A)^2$ i $K = U(K_1, y)$, więc $K \in f(K_1)$. Ponieważ $K \in D_{\mathcal{A}}$, to wobec wzoru (10) mamy $K \in \hat{\beta}^{-1}(D_h, (X)^2) = D_{\mathcal{A}}$. Warunki $K \in f(K_1)$ i $K \in D_{\mathcal{A}}$ oznaczają, że $K_1 \in D_{\mathcal{A}}$ wbrew temu, że $K_1 \in (A)^2 \subset \bar{D}_{\mathcal{A}} = [\mathbf{R}_{\mathcal{A}}]^2$. Otrzymana sprzeczność dowodzi, że $B^2 \subset \bar{D}_{\mathcal{A}}$. Ponieważ $S(\mathbf{R}_{\mathcal{A}}) = X$ i $(B)^2 \subset \bar{D}_{\mathcal{A}}$, to $B \subset R \in m(X, \bar{D}_{\mathcal{A}}) \overset{r}{=} \mathbf{R}_{\mathcal{A}}$, więc $B \in 2^R \subset \mathbf{R}_{\mathcal{A}}$, skąd $B \in \mathbf{R}_{\mathcal{A}}$, cbdd.

Z twierdzenia 8, wobec wzoru (4), mamy także $f(\mathbf{R}_{\mathcal{A}}) \subset \mathbf{R}_{\mathcal{A}}$, bo $f(\mathbf{A}) \subset \varphi(\mathbf{A})$.

Otrzymaliśmy zatem graf $\mathbf{R}_{\mathcal{A}} \mid \hat{f} / \mathbf{R}_{\mathcal{A}} \mid \hat{\varphi} /$.

Na mocy twierdzenia 2 z rozdziału 4. mamy $\left(\mathbf{A} \subset \mathbf{B} \text{ i } \sigma(\mathbf{B}) = \emptyset\right) \Longrightarrow \sigma\left(\mathbf{B} \setminus \hat{\beta}^{-1}(\mathbf{A}, \mathbf{B})\right) = \emptyset$, skąd własność $\sigma(\bar{D}_{\mathcal{A}}) = \emptyset$ otrzymamy, jeśli przyjmiemy $\mathbf{A} = D_h$ i $\mathbf{B} = (X)^2$,

bo $\sigma\left((X)^2\right) = \emptyset$ i zachodzi wzór (10) i (9). Mamy więc graf $\bar{D}_A \mid \hat{f}$.

Przez $\mathcal{D}(R_A)$ oznaczamy zbiór wszystkich takich rodzin $A \subset R_A$, dla których prawdziwa jest przynajmniej jedna z następujących czterech relacji:

1. $f(A) \subset A$;
2. $\varphi(A) \subset A$;
3. $f(A) \rightarrow A$;
4. $\varphi(A) \rightarrow A$.

Jeśli $A \in \mathcal{D}(R_A)$ i zachodzi relacja 1., to będziemy mówić o zbiorze A domkniętym w grafie $R_A \mid \hat{f}$, przy czym jeśli $A = B^2 \subset \bar{D}_A$, to mówimy o zbiorze B^2 domkniętym w grafie $\bar{D}_A \mid \hat{f}$. Jeśli zachodzi relacja 3. lub 4., to mówimy, że A jest domkniętą rodziną względem relacji $\rightarrow$.

W ogólnym przypadku każdą rodzinę $A \subset 2^X$ nazywamy domkniętą rodziną automatu A, albo krótko - domkniętą rodziną, jeżeli zachodzi przynajmniej jedna z relacji od 1. do 4.

Metoda znalezienia rozwiązania jest więc jednoznaczna, a mianowicie polega ona na znalezieniu odpowiedniego zbioru domkniętego A w grafie $R_A \mid \hat{f}$, tj. takiego zbioru, że $\eta(A) \in \mathcal{R}(R_A)$. Tak więc, ze względu na interesujące nas zagadnienie znalezienia rozwiązania, warunkiem koniecznym jest, aby argumenty wszystkich funkcji przebiegały przestrzeń $R_A \subseteq m(X, \bar{D}_A)$, a wówczas wartości tych funkcji nie wychodzą poza przestrzeń R_A. Jednakże własności na ogół będziemy podawać dalej w odniesieniu do przestrzeni 2^X. Czytelnik, chcący korzystać z podanych dalej własności przy minimalizacji automatu, powinien nieodzownie przestrzegać powyższej uwagi.

Algorytmy znalezienia zbiorów domkniętych w grafie $R_A \mid \hat{f}$ podane są w rozdziale 4. Dlatego dalej podajemy tylko algorytmy (metody) znalezienia rodzin domkniętych względem relacji $\rightarrow$. Tak więc, w ogólnym przypadku będziemy mogli wybrać ze zbioru $\mathcal{D}(R_A)$ takie rodziny A, że $A \in \mathcal{R}(R_A)$. Podkreśla-

my przy tym, że znalezienie rozwiązania, nawet minimalnego, trudniejsze jest tylko w bardzo nielicznych przypadkach automatów spotykanych w praktyce inżynierskiej, o ile dany automat jest upraszczalny. W przeciwnym przypadku łatwo na ogół stwierdzimy nieupraszczalność danego automatu.

Szczególną uwagę należy zwrócić na oczywistą własność

$$\left(S(A) = X\right) \Rightarrow \left(\left[\varphi(A) \rightarrowtail A\right] \equiv \left[f(A) \rightarrowtail A\right]\right) \qquad (13)$$

i na oczywiste

T w i e r d z e n i e 9. $\mathfrak{D}(R_A)$ jest addytywną rodziną, to znaczy taką, że

$$\left(A,\ B \in \mathfrak{D}(R_A)\right) \Rightarrow \left(A \cup B \in \mathfrak{D}(R_A)\right).$$

Ponadto z (11) i (12) wynika np., że jeśli $A \in \mathfrak{D}(R_A)$ i $f(A) \rightarrowtail A$, to $f(A)$, $\eta(f(A)) \in \mathfrak{D}(R_A)$. Jeśli $\varphi(A) \rightarrowtail A$, to $\varphi(A)$, $\eta(\varphi(A)) \in \mathfrak{D}(R_A)$. Własności te zachodzą tym bardziej w przypadku relacji $\subset$.

Dla ułatwienia pracy Czytelnikowi wymieniamy niżej niektóre elementy zbioru $\mathfrak{D}(R_A)$, zwłaszcza wynikające z materiału zawartego w rozdziale 4:

$$\left.\begin{array}{l} \bar{D}_A,\ m(X, D_A),\ U(A) \\ \beta(A),\ \delta(D),\ B \setminus \beta^{-1}(A, B) \end{array}\right\} \in \mathfrak{D}(R_A).$$

Podkreślamy raz jeszcze, że np. zapis $\beta(A) \in \mathfrak{D}(R_A)$ pociąga za sobą, że $A \rightarrowtail m(X, D_A) \overset{\Gamma}{=} R_A$.

Oznaczamy liczby:

$$\theta(A) = \min_{D \subset A \cdot \text{i } S(D) = S(A)} \operatorname{card}(D),$$

$$\nu(A) = \max_{A \in A} \operatorname{card}(A).$$

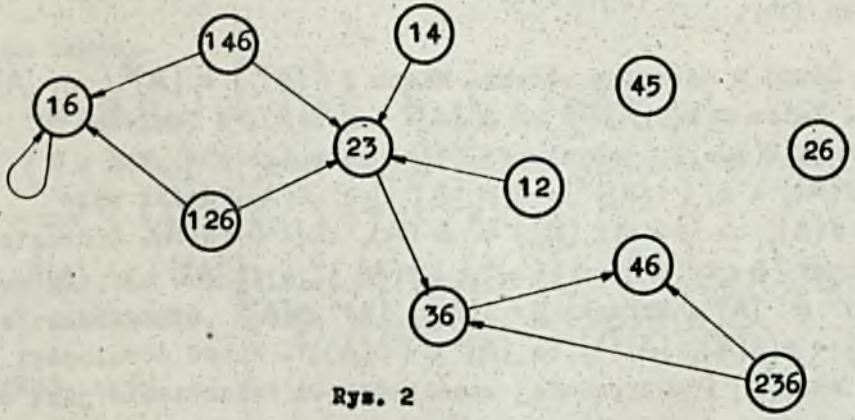
Z warunku $S(A) = X$ mamy

WS1. $\left(A \in \mathcal{R}(R_A)\right) \Rightarrow \text{card}\ (A) \geqslant \vartheta\left(m(X, \bar{D}_A)\right) \geqslant \nu\left(m(X, D_q)\right)$.

P R Z Y K Ł A D Y

1. Zbiór $\beta(A)$ możemy znaleźć, poza rysunkiem grafu, za pośrednictwem tabeli automatu na mocy wzoru (5) i w oparciu o algorytm II, a więc także narysujemy graf $\beta(A)|\hat{f}$ dla każdej rodziny A - taki graf pokazany jest na rys. 2 dla $A = m(X, \bar{D}_A) \cup \bar{D}_A \equiv \{146, 126, 236, 45, 14, 16, 46, 12, 26, 23, 36\}$ - oraz automatu zadanego tabelą T1.

Z rysunku grafu, np. grafu $\beta(A)|\hat{f}$, łatwo można znaleźć rodziny domknięte automatu, a więc także i rozwiązanie. Metodę tę należy zaliczyć do jednej z najłatwiejszych metod znalezienia rozwiązania, zwłaszcza dla $A = \left(m(X, \bar{D}_A) \cup \bar{D}_q\right) \setminus U(X)$. Na przykład z rys. 2



Rys. 2

widzimy, że dla $B \equiv \{146, 236, 45\}$ mamy $B \in \mathcal{R}(R_A)$, gdyż $B \overset{\beta}{\Longrightarrow} \{23, 16, 46, 36\} \Longrightarrow \emptyset$, skąd $\beta(B) \equiv B \cup \{23, 16, 46, 36\}$, a więc $\eta(\beta(B)) = B$ i $S(B) = X$ oraz $\text{card}\ (B) = 3 < \text{card}\ (X) = 6$.

2. Z WS1. wynika, że w pierwszym kroku minimalizacji automatu zawsze trzeba sprawdzić moc rodziny $m(X, \bar{D}_A)$. Jeśli więc się zdarzy, że $\text{card}\left(m(X, \bar{D}_A)\right) < \text{card}\ (X)$, to

$m(X, \bar{D}_A) \in \mathcal{R}(R_A)$, a jeżeli card $(m(X, \bar{D}_A)) = \nu(m(X, D_A))$, to $m(X, \bar{D}_A)$ jest rozwiązaniem minimalnym. Właśnie $\mathbf{B}$ jest rozwiązaniem minimalnym wobec rodziny /5/ - patrz 5.4.

5.7. Rodziny domknięte typu m ($\bar{m}$)

Rodziny typu m (rozdział 3.) są to rodziny postaci $\mathbf{A} = m(S(A), [A]^2)$ lub $\eta(A) = m(S(A), [A]^2)$.

W oparciu o wzory od (1) do (5) łatwo dowodzi się, że

$$[\varphi(A)]^2 = [f(A)]^2 = f([A]^2) = \varphi([A]^2) \setminus U(X). \qquad (14)$$

T w i e r d z e n i e 10. Zachodzi relacja

$$[f(A) \to m(S(A), [A]^2)] \equiv [\sigma([A]^2) = \emptyset]. \qquad (15)$$

D o w ó d. Oczywiście $(B \to D) \Rightarrow [B]^2 \subset [D]^2$. Mamy więc $(f(A) \to m(S(A), [A]^2)) \Rightarrow [[f(A)]^2 \subset [A]^2] \equiv [\sigma([A]^2) = \emptyset]$, gdyż $[m(S(A), [A]^2)]^2 = [A]^2$ i $[f(A)]^2 = f([A]^2)$ na mocy (14).

Dowód w odwrotną stronę. Niech $f([A]^2) \subset [A]^2$ $(\sigma([A]^2) = \emptyset)$. Zatem $S(f([A]^2)) \subset S([A]^2) \subset S(A)$, a ponieważ $f(A) \cap U(X) = \emptyset$ (patrz wzór (4)) i zachodzi (14), to $S(f(A)) = S([f(A)]^2) = S(f([A]^2)) \subset S(A)$. Jeśli więc $B \in f(A)$, to $B \subset S(f(A)) \subset S(A)$, skąd $B \subset S(A)$. Z drugiej strony $(B \in f(A)) \Rightarrow (B)^2 \subset [f(A)]^2 = f([A]^2) \subset [A]^2$, skąd $(B)^2 \subset [A]^2$. Warunki $B \subset S(A)$ i $(B)^2 \subset [A]^2$ oznaczają, że $\{B\} \to m(S(A), [A]^2)$, bo $[A]^2 \subset (S(A))^2$. Wobec dowolności zbioru $B \in f(A)$ i poprzedniej części dowodu twierdzenie jest udowodnione.

Zauważmy, że relacja (15) pozostaje w związku z operacją m na rodzinie $\mathbf{A}$ z jednej strony, a z drugiej mówi nam, że $m(A, B^2)$ jest domkniętą rodziną automatu wtedy i tylko wtedy, gdy B^2 jest zbiorem domkniętym w grafie $(X)^2 | \hat{f}$ i $B^2 \subset (A)^2$, co łatwo widać przez podstawienie do (15) symbolu $\mathbf{A} = m(A, B^2)$. Otrzymujemy więc ważny następujący związek:

$$\left[\sigma\left(B^2\right) = \emptyset \quad \text{i} \quad B^2 \subset (A)^2\right] \equiv \left[f\left(m\left(A,B^2\right)\right) \to m\left(A,B^2\right)\right]. \qquad (16)$$

Z (16) mamy, między innymi, następujące elementy zbioru $\mathfrak{D}\left(R_A\right)$:

$$\left.\begin{array}{l} m\left(S\left(\beta\left(B^2\right)\right), \beta\left(B^2\right)\right); \; m\left(X, \; \beta\left(B^2\right)\right); \\ m\left(X, \; A^2 \setminus \hat{\beta}^{-1}\left(B^2, A^2\right)\right); \; m\left(X, \; \delta\left(D^2\right)\right); \\ m\left(A, \; \delta\left((A)^2\right)\right); \; m\left(X, \; \delta\left((A)^2\right)\right) \end{array}\right\} \in \mathfrak{D}\left(R_A\right).$$

Wobec (16) mamy oczywiste

T w i e r d z e n i e 11. Wszystkich rozwiązań typu m, tj. rozwiązań postaci $m\left(X, B^2\right)$, istnieje dokładnie tyle, ile jest różnych zbiorów B^2 domkniętych w grafie $\bar{D}_A|\hat{f}$ takich, że $\text{card}\left(m\left(X, B^2\right)\right) < \text{card}(X)$ /$B^2 \subset \bar{D}_A$ i $\sigma\left(B^2\right) = \emptyset$/.

Wobec (14) mamy

$$\left(\varphi(A) \to A \quad \text{lub} \quad f(A) \to A\right) \Rightarrow \sigma\left([A]^2\right) = \emptyset,$$

a więc także:

WS2. $\left(A \in \mathcal{R}\left(R_A\right)\right) \Rightarrow \left([A]^2 \neq \emptyset \quad \text{i} \quad \sigma\left([A]^2\right) = \emptyset\right)$,

WS3. $\left(A \in \mathcal{R}\left(R_A\right) \quad \text{i graf} \quad \bar{D}_A \,|\, \hat{f} \text{ posiada własność } \mathcal{T}\right) \Rightarrow$
$\Rightarrow \left([A]^2 = \bar{D}_A \quad \text{i} \quad S(A) = X\right)$.

Powiemy, że w rodzinie $\mathbf{A}$, gdzie $S(\mathbf{A}) = X$ i $\mathbf{A} \subset R_A$, istnieje rozwiązanie, np. $\mathbf{B}$, jeżeli $\mathbf{B} \to \mathbf{A}$ i $\mathbf{B} \in \mathcal{R}\left(R_A\right)$.

Z WS3. mamy

W n i o s e k 1. Jeżeli graf $\bar{D}_A|\hat{f}$ posiada własność $\mathcal{T}$, to rozwiązania należy poszukiwać tylko w takich $\mathbf{A} \subset m\left(X, \bar{D}_A\right)$, że $[A]^2 = \bar{D}_A$ i $S(A) = X$.

Z wniosku 1, wobec twierdzenia 11, mamy

T w i e r d z e n i e 12. Jeżeli graf $\bar{D}_A \,|\, \hat{f}$ posiada własność $\mathcal{T}$ i $\mathbf{A}$ jest rozwiązaniem typu m, to $\mathbf{A} = m\left(X, \bar{D}_A\right)$ (gdyż przyjęliśmy, że $\mathbf{A} = \eta \quad (\mathbf{A})$).

T w i e r d z e n i e 13. Załóżmy, że:

1. dla każdego $A \in m\left(X, \bar{D}_A\right)$ istnieje $x \in A$ lub istnieje $K \in (A)^2$ i $x \notin B \in m\left(X, \bar{D}_A\right)$ lub $K \notin (B)^2$ dla $B \neq A$,

2. graf $\bar{D}_A \mid \hat{f}$ posiada własność $\mathfrak{N}$,

3. $A \in \mathcal{R}(R_A)$.

Wówczas ważna jest t e z a, że $m\left(X, \bar{D}_A\right) \in \mathcal{R}(R_A)$ i $m\left(X, \bar{D}_A\right)$ jest rozwiązaniem minimalnym.

D o w ó d. Wobec 1. mamy $\left(A \subset m\left(X, \bar{D}_A\right) \setminus U(X)\right) \Rightarrow$ $\Rightarrow [A]^2 \subset \bar{D}_A$. Oznacza to wobec 3. i wniosku 1, że card $(A) \geqslant$ card $\left(m(X, \bar{D}_A)\right)$. Z drugiej strony $m\left(X, \bar{D}_A\right) \in \mathcal{D}(R_A)$ i $\varphi\left(m(X, \bar{D}_A)\right) \rightarrow m(X, \bar{D}_A)$, więc $m\left(X, \bar{D}_A\right) \in \mathcal{R}(R_A)$. Zatem $m\left(X, \bar{D}_A\right)$ jest rozwiązaniem minimalnym, cbdd.

T w i e r d z e n i e 14. Niech

1. $\varphi^{\bar{D}_A} \overset{\text{def}}{=\!=} \left\{K \in \bar{D}_A : f(K) = \emptyset \text{ lub } f(K) = \{K\}\right\}$.

T e z a. Zachodzi:

(I) $f(B) \rightarrow \{B\}$ dla każdego $B \in m\left(X, \varphi^{\bar{D}_A}\right)$, więc $\{B\} \in \mathcal{D}(R_A)$,

(II) $\varphi\left(m\left(X, \varphi^{\bar{D}_A}\right)\right) \rightarrow m\left(X, \varphi^{\bar{D}_A}\right)$, więc $m\left(X, \varphi^{\bar{D}_A}\right) \in \mathcal{D}(R_A)$.

D o w ó d. Niech $B \in m\left(X, \varphi^{\bar{D}_A}\right)$. Więc $(B)^2 \subset \varphi^{\bar{D}_A}$. Mamy $\left(K \in (B)^2\right) \Rightarrow \left(f(K) = \emptyset \text{ lub } f(K) = \{K\}\right)$, więc $6\left((B)^2\right) = \emptyset$. Podobnie $6\left(\varphi^{\bar{D}_A}\right) = \emptyset$. Własność (I) otrzymamy, jeśli w (15) przyjmiemy $\bar{A} = \{B\}$, bo $6\left((B)^2\right) = \emptyset$ i $m\left(S(\{B\}), [\{B\}]^2\right) = m\left(B, (B)^2\right) = \{B\}$ oraz $f(\{B\}) = f(B)$. Podobnie otrzymamy własność (II), bo zachodzi (13), cbdd.

T w i e r d z e n i e 15. Jeżeli $\bar{A}$ jest rodziną typu m, to

$$\left[f\left(\bar{m}\left(\bar{A}, B^2\right)\right) \rightarrow \bar{m}\left(\bar{A}, B^2\right)\right] = \left[6\left([A]^2 \setminus B^2\right) = \emptyset\right]. \quad (17)$$

D o w ó d. Jeśli $\bar{A}$ jest rodziną typu m, to na mocy twierdzenia 1 z rozdz. 3. mamy $\bar{m}\left(\bar{A}, B^2\right) = m\left(S(\bar{A}), [A]^2 \setminus B^2\right)$.

Zatem relację (17) otrzymamy, jeśli w (15) w miejsce symbolu A podstawimy symbol $\bar{m}(A, B^2)$, obdd.

T w i e r d z e n i e 16.

(I) Jeżeli

$$1.\ \bar{m}(f(A), \sigma([A]^2)) \setminus U(X) \subset (X)^2,$$

to

$$f(\bar{m}(A, \hat{\beta}^{-1}([A]^2))) \to \bar{m}(A, \hat{\beta}^{-1}([A]^2)). \qquad (18)$$

(II) Jeżeli:

$$1.\ \sigma([A]^2) = \emptyset \text{ i } B^2 \subset f([A]^2),$$
$$2.\ \bar{m}(f(A), B^2) \setminus U(X) \subset (X)^2,$$

to

$$f(\bar{m}(A, \beta^{-1}(B^2, [A]^2))) \to \bar{m}(A, \hat{\beta}^{-1}(B^2, [A]^2)). \qquad (19)$$

D o w ó d. W przypadku (I) warunek 1. oznacza, że $f(\bar{m}(A, \hat{\beta}^{-1}([A]^2))) \subset (X)^2$. Jeśli więc $B \in f(\bar{m}(A, \hat{\beta}^{-1}([A]^2)))$, to $B \in f(\delta([A]^2)) \subset \delta([A]^2) \to \bar{m}(A, \hat{\beta}^{-1}([A]^2))$. Zatem $\{B\} \to \bar{m}(A, \hat{\beta}^{-1}([A]^2))$. Stąd wynika relacja (18).

Dowód relacji (19) jest analogiczny, jak w przypadku (I), obdd.

Podkreślmy od razu, że w praktyce tak się dzieje, że nie sprawdzając tej własności, czy A jest rodziną typu m - relacja

$$f(\bar{m}(A, \hat{\beta}^{-1}([A]^2))) \to \bar{m}(A, \hat{\beta}^{-1}([A]^2)) \qquad (20)$$

na ogół zawsze zachodzi. Relację (20) otrzymamy, jeśli w (17) przyjmiemy $B^2 = \hat{\beta}^{-1}([A]^2)$. Tego faktu nie można jednak przyjąć jako twierdzenia, gdyż łatwo można podać przykład taki, że jeśli A nie jest rodziną typu m, to relacja (20) nie zachodzi, pomimo że $[A]^2 \setminus \hat{\beta}^{-1}([A]^2) = \delta([A]^2)$. W szczególności może być $f(A) \not\to A$, mimo że $\sigma([A]^2) = \emptyset$.

Ale w tych ostatnich przypadkach, np. gdy $\delta([A]^2) = \emptyset$, lecz $f(A) \not\subset A$ - twierdzenie 10 mówi, że wtedy $m(S(A), [A]^2) \in \mathfrak{D}(R_A)$ (przez podstawienie w (15) symbolu $m(S(A), [A]^2)$ zamiast A).

W ogólności twierdzenie 15, poza twierdzeniem 10, ma duże znaczenie dla minimalizacji automatów, co wynika z łatwego stosowania operacji $\bar{m}$ z jednej strony, a z drugiej - z łatwego znalezienia zbioru $\hat{\beta}^{-1}([A]^2)$ $(\hat{\beta}^{-1}(B^2, [A]^2))$ za pośrednictwem antytabeli stanów automatu bądź rysunku grafu $\bar{D}_A \mid \hat{f}$, jak również łatwość sprawdzenia zachodzenia lub nie zachodzenia relacji (20) za pośrednictwem tabeli automatu. Ponadto umiemy znaleźć rodziny domknięte typu m. Na przykład z (17) mamy $\bar{m}(m(X, \bar{D}_A), \hat{\beta}^{-1}(B^2, \bar{D}_A)) \in \mathfrak{D}(R_A)$ itp. Z relacji (20) korzystamy w szczególności dla takich A, kiedy $S(A) = X$.

P R Z Y K Ł A D. Niech $A = \{236, 126, 45\}$. Z rys. 2 mamy:

$$A \xrightarrow{f} f(A) = \{46, 36, 23, 16\} \Rightarrow \delta([A]^2) = \{46\} \Rightarrow$$
$$\xrightarrow{\hat{\beta}^{-1}} \{36\} \Rightarrow \{23\} \Rightarrow \{12\} \Rightarrow \emptyset, \text{ skąd}$$

$\bar{m}(A, \hat{\beta}^{-1}([A]^2)) = \{16, 26, 3, 45\} \in \mathfrak{R}(R_A)$ (dla automatu zadanego tabelą T 1).

5.8. Rodzina $C(A)$

$\hat{2}^A$ jest to rodzina, do której zbiór B należy wtedy i tylko wtedy, gdy istnieje $A \in A$ i $B \in 2^A$.

Oznaczamy

$$C(A) \overset{\text{def}}{=} \eta(\delta(\hat{2}^A)). \tag{21}$$

Oczywiście

$$[C(A) = \eta(A)] \equiv [f(A) \rightarrow A] \equiv [f(\hat{2}^A) \subset \hat{2}^A].$$

W odróżnieniu od zbioru $\delta(\mathbf{A})$, zbiór $\delta\left(\hat{2}^{\mathbf{A}}\right)$ posiada tę własność, że

$$S\left(\delta\left(\hat{2}^{\mathbf{A}}\right)\right) = S(\mathbf{A}) = S\left(C(\mathbf{A})\right), \tag{22}$$

gdyż w grafie $\hat{2}^{\mathbf{A}} \mid \hat{\hat{f}}\left(\hat{2}^{\mathbf{A}} \mid \hat{f}\right)$ mamy $U\left(S(\mathbf{A})\right) \subset \delta\left(\hat{2}^{\mathbf{A}}\right)$ oraz $S\left(U\left(S(\mathbf{A})\right)\right) = S(\mathbf{A})$.

T w i e r d z e n i e. Zachodzą równości

$$\left[C(\mathbf{A})\right]^2 = \left[\delta\left(\hat{2}^{\mathbf{A}}\right)\right]^2 = \delta\left([\mathbf{A}]^2\right).$$

D o w ó d. Mamy $[\mathbf{A}]^2 \subset \hat{2}^{\mathbf{A}}$, więc także $\delta\left([\mathbf{A}]^2\right) \subset \delta\left(\hat{2}^{\mathbf{A}}\right)$, skąd $\delta\left([\mathbf{A}]^2\right) \subset \left[\delta\left(\hat{2}^{\mathbf{A}}\right)\right]^2$, bo $\left[\delta\left([\mathbf{A}]^2\right)\right]^2 = \delta\left([\mathbf{A}]^2\right)$.

Dowód w odwrotną stronę. Mamy $\left[f\left(\delta\left(\hat{2}^{\mathbf{A}}\right)\right)\right]^2 = f\left(\left[\delta\left(\hat{2}^{\mathbf{A}}\right)\right]^2\right) \subset \left[\delta\left(\hat{2}^{\mathbf{A}}\right)\right]^2 \subset [\mathbf{A}]^2$, gdyż $f\left(\delta\left(\hat{2}^{\mathbf{A}}\right)\right) \subset \delta\left(\hat{2}^{\mathbf{A}}\right)$. Zatem $\left[\delta\left(\hat{2}^{\mathbf{A}}\right)\right]^2 \subset \delta\left([\mathbf{A}]^2\right)$, ckd.

T w i e r d z e n i e 17. Zachodzi relacja

$$C(\mathbf{A}) \rightarrow \bar{m}\left(\mathbf{A}, \hat{\beta}^{-1}\left([\mathbf{A}]^2\right)\right). \tag{23}$$

D o w ó d. Wobec (22) wystarczy dowieść, że jeśli $B \in C(\mathbf{A})$ i $(B)^2 \neq \emptyset$, to $\{B\} \rightarrow \bar{m}\left(\mathbf{A}, \hat{\beta}^{-1}\left([\mathbf{A}]^2\right)\right)$. Zatem mamy $\left(B \in C(\mathbf{A})\right) \Rightarrow (B)^2 \subset \delta\left([\mathbf{A}]^2\right) = [\mathbf{A}]^2 \setminus \hat{\beta}^{-1}\left([\mathbf{A}]^2\right) = \left[\bar{m}\left(\mathbf{A}, \hat{\beta}^{-1}\left([\mathbf{A}]^2\right)\right)\right]^2$. Z drugiej strony $B \subset A \in \mathbf{A}$, gdyż $C(\mathbf{A}) \rightarrow \mathbf{A}$. Załóżmy, że $K \in (B)^2$ i $K \notin (A)^2 \setminus \hat{\beta}^{-1}\left([\mathbf{A}]^2\right)$. Ponieważ $K \in (B)^2 \subset (A)^2$, to $K \in (A)^2$ i $K \notin (A)^2 \setminus \hat{\beta}^{-1}\left([\mathbf{A}]^2\right)$, skąd $K \in \hat{\beta}^{-1}\left([\mathbf{A}]^2\right)$, i stąd $K \notin \delta\left([\mathbf{A}]^2\right)$ wbrew temu, że $K \in (B)^2 \subset \delta\left([\mathbf{A}]^2\right)$. Otrzymana sprzeczność dowodzi, że $(B)^2 \subset (A)^2 \setminus \hat{\beta}^{-1}\left([\mathbf{A}]^2\right)$. Warunki $B \subset A$ i $(B)^2 \subset (A)^2 \setminus \hat{\beta}^{-1}\left([\mathbf{A}]^2\right)$ oznaczają, że $\{B\} \rightarrow m\left(A, (A)^2 \setminus \hat{\beta}^{-1}\left([\mathbf{A}]^2\right)\right) \rightarrow \bar{m}\left(\mathbf{A}, \hat{\beta}^{-1}\left([\mathbf{A}]^2\right)\right)$, skąd $\{B\} \rightarrow \bar{m}\left(\mathbf{A}, \hat{\beta}^{-1}\left([\mathbf{A}]^2\right)\right)$, a stąd wynika (23), ckd.

Wobec twierdzenia 15, z (23) mamy

W n i o s e k 2. Jeżeli $\mathbf{A}$ jest rodziną typu m, to

$$C(A) = \bar{m}\left(A, \hat{\beta}^{-1}\left([A]^2\right)\right). \tag{24}$$

Mamy także

$$\left(f\left(\bar{m}\left(A, \hat{\beta}^{-1}\left([A]^2\right)\right)\right) \to \bar{m}\left(A, \hat{\beta}^{-1}\left([A]^2\right)\right)\right) \Rightarrow \Rightarrow C(A) = \bar{m}\left(A, \hat{\beta}^{-1}\left([A]^2\right)\right). \tag{25}$$

Z (24) mamy np., że $C(\{A\}) = m\left(A, \delta\left((A)^2\right)\right)$.

Mamy więc ważną własność

WS4. $\left(A \in \mathcal{R}\left(R_A\right) \text{ i } A \to B\right) \Rightarrow \left(A \to C(B) \text{ i } C(B) \neq U(X)\right)$.

Oczywiste jest, że $C(A) \in \mathcal{O}\left(R_A\right)$.

5.9. Algorytm znalezienia rozwiązania minimalnego

O zadawanych rodzinach, np. o rodzinie A zakładamy, że $S(A) = X$ i $\eta(A) = A \to m\left(X, \bar{D}_A\right)$.

Moc rozwiązania $B \to A$ pozostaje w związku zawartym w następujących dwóch oczywistych twierdzeniach:

T w i e r d z e n i e 18. Jeżeli $B \in \mathcal{R}\left(R_A\right)$ i $B \to A$, to $\text{card}(B) \geqslant \Theta\left(m\left(X, [A]^2\right)\right) \geqslant \nu\left(m\left(X, (X)^2 \setminus [A]^2\right)\right)$;

T w i e r d z e n i e 19. Niech

$$e(A) \overset{\text{def}}{=} \left\{A \in A : \bigvee_{x \in A}\left[x \notin B \in A \text{ dla } B \neq A\right]\right\}.$$

T e z a. Jeżeli $B \in \mathcal{R}\left(R_A\right)$ i $B \to A$, to

$\text{card}(B) \geqslant \text{card}\left(e(A)\right)$ $\left(\text{card}(\emptyset) = 0\right)$.

Zakładamy, że $\bar{D}_A \neq \emptyset$ i $\bar{D}_A \neq (X)^2$.

Algorytmem III nazywamy następujące postępowanie dla dowolnej pary uporządkowanej (A, r), gdzie $A \subset m\left(X, \bar{D}_A\right)$ i $S(A) = X$ oraz $\text{card}(A) = r = \Theta\left(m\left(X, \bar{D}_A\right)\right)$. Należy stwierdzić istnienie lub nieistnienie w rodzinie A rozwiązania B o

mocy r i jeśli takie istnieje, to należy je znaleźć. Zapis analityczny odpowiedniego algorytmu jest następujący: jeśli $\varphi(A) \rightarrow A$, to A jest rozwiązaniem o mocy r, (wtedy rozwiązanie to jest minimalne) i proces na tym kończymy.

Załóżmy, że $\varphi(A) \not\rightarrow A$ i niech $D_0 = \{C(A)\}$. Istnieją następujące ciągi:

1. $A_j = \left\{ B : \bigvee_{K \in D_{j-1}} [B \subset K,\ S(B) = X \text{ i } \operatorname{card}(B) = r] \right\}$,

2. $B_j = \left\{ B \in A_j : \varphi(B) \rightarrow B \right\}$,

3. $D_j = \left\{ R : \bigvee_{S \in A_j \setminus B_j} [R = C(S)] \right\}$,

dla $j = 1, 2, \ldots$ Wówczas mamy:

a. Jeśli $A_1 = \emptyset$, to w rodzinie A w ogóle nie istnieje rozwiązanie o mocy r,

b. Załóżmy, że $A_1 \neq \emptyset$. Istnieje więc wskaźnik $j = p \geqslant 1$ taki, że $A_j \neq \emptyset$ dla $j = 1, 2, \ldots, p$ oraz $A_{p+1} = \emptyset$. Wówczas albo istnieje wskaźnik j, $1 \leqslant j \leqslant p$, taki że $B_j \neq \emptyset$, albo $B_j = \emptyset$ dla $j = 1, 2, \ldots, p$.

Jeśli więc $B_j = \emptyset$ dla $j = 1, 2, \ldots, p$, to w rodzinie A w ogóle nie istnieje rozwiązanie o mocy r, w przeciwnym przypadku istnieje najmniejszy wskaźnik j, $1 \leqslant j \leqslant p$, taki że $B_j \neq \emptyset$. Wtedy $B \in B_j$ jest rozwiązaniem o mocy r. W ten sposób znajdziemy rozwiązanie $B \rightarrow A$ o mocy r lub stwierdzimy jego nieistnienie. Oczywiście, jeśli takie rozwiązanie $B \rightarrow A$ o mocy r istnieje, to B jest rozwiązaniem minimalnym, co wynika z definicji pary (A, r) i procesu wyżej opisanego.

W ogólnym przypadku rozwiązanie minimalne możemy znaleźć lub stwierdzić jego nieistnienie, jak następuje: przyjmijmy w algorytmie III zamiast pary (A, r) parę uporządkowaną $(A, r+k)$, gdzie $A \subset m(X, D_A)$ i $S(A) = X$ oraz $\operatorname{card}(A) =$

$r + k$, $k \in \{0, 1, 2, \ldots, t\}$, przy czym $r + t = \mathrm{card}(X) - 1$ $\left(r = \Theta\left(m\left(X, \mathfrak{D}_A\right)\right)\right)$. Zatem w 1. zamiast liczby r trzeba przyjąć liczbę r+k. Proces opisany w algorytmie III prowadzimy dla pary $(A, r+k)$ kolejno dla $k = 0, 1, 2, \ldots$ t i dla wszystkich A, to jest dla $k = 0$ i wszystkich A, a następnie dla $k = 1$ i wszystkich A itd.

Jeśli więc dla każdej pary $(A, r + k)$ stwierdzimy na mocy algorytmu III nieistnienie rozwiązania o mocy $r + k$ w rodzinie A, to rozwiązanie minimalne w ogóle nie istnieje, co dowodzi, że wtedy automat nie jest upraszczalny, w przeciwnym przypadku znajdziemy rozwiązanie o mocy $r + k$ dla najmniejszej liczby $k \in \{0, 1, 2, \ldots, t\}$. Wtedy, oczywiście, znalezione rozwiązanie B o najmniejszej mocy $r + k$ jest minimalne, co wynika z definicji pary $(A, r + k)$ i procesu opisanego w algorytmie III.

Uwaga. W procesie występującym w algorytmie III na ogół trzeba znaleźć wiele rodzin postaci $C(A)$. Niemal każdą z nich (na ogół każdą) znajdziemy poprzez sprawdzenie zachodzenia relacji (20), a więc jeśli relacja (20) zachodzi, to $C(A) = \bar{m}\left(A, \beta^{-1}\left([A]^2\right)\right)$, w przeciwnym przypadku rodzinę $C(A)$ znajdziemy na mocy wzoru

$$C(A) = \eta\left(\delta\left(\hat{2}^D\right)\right) = \eta\left(\hat{2}^D \setminus \hat{\beta}^{-1}\left(\hat{2}^D\right)\right), \text{ gdzie}$$

$D = \bar{m}\left(A, \hat{\beta}^{-1}\left([A]^2\right)\right)$, mając rysunek grafu $\hat{2}^D \mid \hat{f}$.

Podobnie znajdziemy rodzinę $C(A)$ w przypadku, gdy $\varphi(A) \not\to A$, mimo, że $\sigma\left([A]^2\right) = \emptyset$.

Ponadto bardzo często korzystamy z twierdzenia 19.

P R Z Y K Ł A D. Automat zadany jest tabelą T3. Tabela T3.1 jest antytabelą stanów automatu zadanego tabelą T3.

	y_1	y_2
1	5,c	-,-
2	1,-	-,-
3	-,-	1,c
4	3,d	2,-
5	4,-	-,d

T3 - Tabela automatu

	y_1	y_2
1	{2}	{3}
2	$\emptyset$	{4}
3	{4}	$\emptyset$
4	{5}	$\emptyset$
5	{1}	$\emptyset$

T3.1 - Tabela odwzorowania

Znaleźć rozwiązanie minimalne automatu zadanego tabelą T3 lub stwierdzić jego nieistnienie.

R o z w i ą z a n i e. Dla automatu zadanego tabelą T3 mamy:

/6/ $D_h \equiv \{14, 35\}$,

/7/ $D_A \equiv \{14, 35, 25\} \equiv m(X, D_A)$,

/8/ $\bar{D}_A \equiv \{1,(235), 2(34), 3(4), 4(5)\}$,

/9/ $m(X, \bar{D}_A) \equiv \{123, 234, 45, 15\}$.

Niech $A_{r+k} = \{A \subset m(X, \bar{D}_A) : S(A) = X \text{ i } \text{card}(A) = r + k\}$, gdzie $k \in \{0, 1, 2, \ldots\}$ oraz $r = \theta(m(X, \bar{D}_A))$.

Z /9/ mamy $r = 2$. Zatem dla $k = 0$ mamy $A_2 = \{A_{2,1}, A_{2,2}\}$ (co łatwo można sprawdzić), gdzie: $A_{2,1} \equiv \{123, 45\}$ i $A_{2,2} \equiv \{15, 234\}$.

Rozwiązanie minimalne o mocy $r = 2$ w ogóle nie istnieje, bo:

$A_{2,1} \Rightarrow \bar{m}(A_{2,1}, \hat{\beta}^{-1}([A_{2,1}]^2)) = C(A_{2,1}) \equiv$

$\equiv \{13, 23, 4, 5\}$ [1] - nie istnieje rozwiązanie o mocy $r = 2$,

$A_{2,2} \Rightarrow \bar{m}(A_{2,2}, \hat{\beta}^{-1}([A_{2,2}]^2)) = C(A_{2,2}) \equiv$

$\equiv \{23, 4, 1, 5\}$ - nie istnieje rozwiązanie o mocy $r = 2$.

Zatem mamy:

$m(X, \bar{D}_A) \supset A_{3,1} \equiv \{123, 234, 15\} \Rightarrow \bar{m}(A_{3,1}, \hat{\beta}^{-1}([A_{3,1}]^2)) =$

$= C(A_{3,1}) \equiv \{13, 23, 24, 5\} \Rightarrow \{13, 24, 5\} \equiv B \subset C(A_{3,1})$ oraz $S(B) = X$, $\text{card}(B) = 3$ i $\varphi(B) \to B$. Więc $B \in \mathcal{R}(R_A)$ i B jest rozwiązaniem minimalnym, co było do wykonania.

---

[1] Oczywiście, że $\{13, 23, 4, 5\}$ oznacza to samo, co $\{\{1, 3\}, \{2, 3\}, \{4\}, \{5\}\}$.

Literatura

[1] GŁUSZKOW W.M.: Sintez cifrovych avtomatov. Fizmatgiz, 1962, Moskwa 1962.

[2] ŁAZARIEW W.G., PIJL E.I.: Sintez asinchronnych koniecznych avtomatov. Akademija Nauk SSSR. Institut Periedaczi Informacji. Izdatielstvo "Nauka". Moskwa 1964.

[3] GŁUSZKOW W.M.: Wstęp do cybernetyki. Książka i Wiedza 1967.

[4] WAWIŁOW E.N., PORTNOJ G.P.: Sintez schem elektronnych cifrovych maszin. Pod redakcijej E.N. Wawiłowa, Izdatielstvo "Sovietskoje Radijo". Moskwa 1963.

[5] Akademija Nauk Ukrainskoj SSSR: Kibernietika, Nr 1. Kijev 1967.

[6] BERGE C.: Théorie des graphes et ses applications. Dunod Paris 1958. Collection Universitaire de Mathématiques.

[7] JAROŃ J.: Zastosowania grafów do układów cybernetycznych. Łódź 1961. Zeszyty Naukowe Uniwersytetu Łódzkiego. Seria II. NAUKI MATEMATYCZNO-PRZYRODNICZE. Zeszyt 11.

[8] KURATOWSKI K.: Wstęp do teorii mnogości i topologii. Wydanie drugie zmienione. PWN. Warszawa 1962.

[9] GRZEGORCZYK A.: Zarys logiki matematycznej. PWN. Warszawa 1961.

ON AUTHOMATA MINIMIZATION

Summary

The paper presents algorithms and methods of minimization of any finite authomata partly determined or incomplete on the basis of the notion of closed sets in a graph oriented finite and an algorithm of finding minimal authomata for the given one - Chapter 5.

We emphasize that the notions comprised are formulated from the viewpoint of authomata minimization in engineer practice. Proofs of many properties and theorems are not given; it is assumed they are evident or easy to be found.

К вопросу:

МИНИМИЗАЦИЯ АВТОМАТОВ

Резюме

В статье приведены алгоритмы и методы минимизирования любых конечных автоматов, частично определенных или неполных, на основе понятия замкнутых множеств в ориентированном завершенном графе, а также алгоритм отыскания минимального автомата для заданного автомата - Раздел 5.

Подчеркивается, что указанные понятия формулированы с точки зрения минимализирования автоматов в инженерской практике. Не поданы многие свойства и теоремы, так как принимается, что они очевидны или что их легко найти.

Warszawa, ul. Krzywickiego 34
A.M.
K.M.

Cena zł 40,-