Serdica
Bulgariacae mathematicae publicationes

Сердика
Българско математическо списание

A TECHNIQUE OF AUTOMATIC DATA EXTRACTION FROM SPECIALIZED TEXTS

STEPHAN M. KERPEDJIEV

The process of automatic data extraction is presented as a translation from the language of specialized texts into the internal language of the system. In order to illustrate that process analogy with program translation has been made. A functional scheme of the translator and the principles of its operation are proposed. Detailed abstract models of some of the modules are developed. A certain universality is achieved by means of describing the lexis, syntax and structure of the text class and by tuning the modules appropriately. Since the text class description plays an important role for the application of this translator, a suitable environment for facilitating that activity is proposed and some suggestions on its automation are made. The translator applications are outlined on the basis of three text classes given as examples.

1. Introduction. The great difficulties connected with data input force the specialists to pay more attention to the automation of this activity. The idea is to extract the data directly from their original text automatically thus avoiding the participation of people in that process. Many tasks in the office automation, economics, publishing and editing spheres require extracting data from texts having a specific structure and a limited set of phases used. Texts of that type will be called specialized ones [2]. Their restrictions enable the formal description of different text classes. Thus the automatic analysis of texts becomes feasible and an automatic data extraction could be performed.

Nowadays that problem is approached to by means of a "quasinatural language understanding". This aspect, however, pays attention mainly to the ways of complex knowledge representation (semantic networks, frames, predicate calculus, etc.) [6] and assumes that the quasinatural language is a very common restriction of a given natural language based primarily on its grammar. This consideration suggests that the texts should contain very universal information which is supposed to be understood by the system as well as a man would do. In fact, in the greatest deal of applications we are interested only in some data dispersed in entirely specialized texts and we can consider them as a more human oriented form of data representations (namely, by explanations, by standard phrases and by a proper arrangement). That point of view allows to describe the text class (structure and phrases used) by simpler means.

The frequent changes of the text classes which are to be analyzed presuppose the presence of convenient tools of the system for its tuning to a corresponding class.

A similar problem is presented in [3] where the analyzed objects are dates and postal addresses. The approach used to analyze them could be applied to more complex texts as well. Any text class could be characterized as mentioned before by a set of data items, a structure, a set of phrases and an arrangement [2].

We distinguish three kinds of languages for a specialized text representation. The natural language used will be called an F-language. The formally described subset of F will be called an M-language. The language used for internal representation will characterize only the data structures and the data formats and will be called an I-language.

SERDICA Bulgaricae mathematicae publicationes. Vol. 13, 1987, p. 239—251.

In the terms defined the problem considered could be reformulated as a computer-aided $F \rightarrow I$ translation. The formalization of M would enable the automatic translation $M \rightarrow I$ while the translation $F \rightarrow M$ should be performed personally when a deviation from the M-language is discovered by the $M \rightarrow I$ translator.

In section 2 the general scheme of the $M \rightarrow I$ translator is presented and the input and output of each module are described. In sections 3, 4, 5, 6 and 7 the basic modules of the $M \rightarrow I$ translator together with the principles which their construction follows are considered in details. Three text classes, the results from one concrete implementation of the system, some possibilities of automating the process of tuning the $M \rightarrow I$ translator and some specific features of the approach adopted are discussed in section 8.

2. General scheme of the $M \rightarrow I$ translator. The functional flow-chart of the $M \rightarrow I$ translator is shown in Fig. 1. Its basic modules are denoted by circles and the input and the resulting information are denoted by rectangulars.

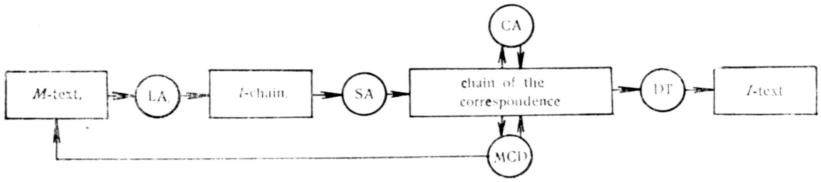

Fig. 1. Functional scheme of the $M \rightarrow I$ translator

2.1. Lexical analysis (LA). A text in M-language (M-text) is the input of thy scanner and an l-chain is its output. The l-chain represents sequences of lexemes. And lexeme belongs to one of the following types: nonabbreviated words (WO); abbreviate words (AW); Roman numbers (RN); numbers (NU); punctuation marks and editing symbols (OL); notions (KNO).

Any lexeme contains information about its type, position in the text and the value obtained in a way characteristic of each type of lexemes. For example, the values of the notions are extracted from a dictionary. The organization of the l-chain is shown in Fig. 2. More than one lexemes may begin from one and the same position. In such a case all the lexemes beginning from that position are given in a linked list.

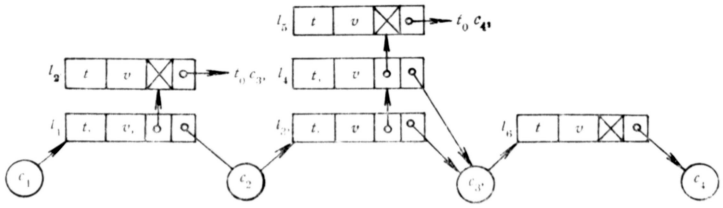

Fig. 2. Organization of the l-chain and lexeme structure
(l_i — lexeme, t-type; v-value; c_j — lexeme beginning)

2.2. Syntactic analysis (SA). The parser groups the lexemes of the l-chain into phrases and marks the positions of the data items occurring in those phrases. One substring may be interpreted in different sequences of phrases. The output of the parser is a structure called a chain of correspondence. It presents the set of the interpretations of the text. An example of such a chain is shown in Fig. 3. d_i is the data item identifier in the I-language, b_j, $j=1, 2, \ldots, 7$ mark the substrings representing the data items in the text.

2.3. Context analysis (CA). This module solves automatically some contradictions and ambiguities in the chain of correspondence. For instance, the substring from b_1 to b_4 in Fig. 3 may be interpreted in the following three ways: 1) d_1, d_2 and an insignificant substring; 2) d_1, d_3 and an insignificant substring; 3) d_1, d_4.

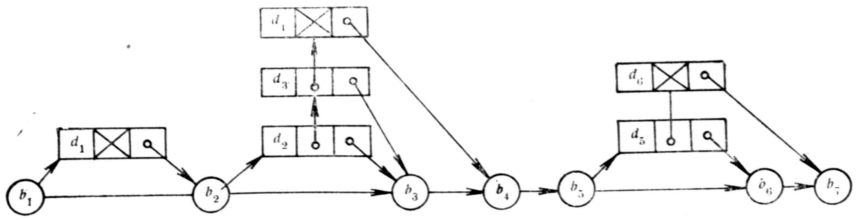

Fig. 3. Organization of the chain of correspondence

CA relies on the text class structure description which comprises the eventual relations between the representations of different groups of data items. CA aims at obtaining a reduced chain of correspondence characterized by the lack of ambiguously recognized substrings (Fig. 4).

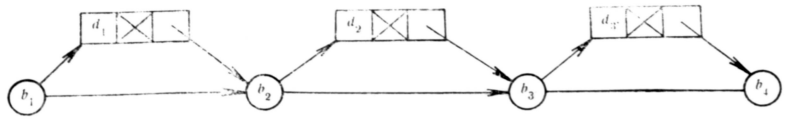

Fig. 4. Reduced chain of correspondence (there are no ambiguously recognized substrings)

2.4. Man — computer dialogue (MCD). This phase aims at correcting the ambiguities, contradictions and some errors left after the CA. The correction under consideration is performed by the user himself.

2.5. Data translation (DT). First the data are localized by the previously mentioned modules. Then the module under consideration transforms each data item from its free form in the text into a format strictly defined by the *I*-language. Because of the relatively complex structure of the data items DT consists of the following three stages: analysis, control and synthesis.

With the intention to ensure fast and easy tuning of the $M \rightarrow I$ translator to different classes of specialized texts or different *M*-languages, the main modules in the scheme are controlled by tables of parameters describing the lexis, phrase syntax, text structure and data formats in the *M*- and *I*-languages. That opportunity enables the change of the *I*-language when the *M*-language is constant.

3. Lexical analysis. LA is carried out by a deterministic finite automaton similar to the one described in [4]. The automaton is extended by the following procedures:

gc — reads the consecutive character of the text and makes it a current character;

mb(*x*), where *x* is a lexeme type — opens a new lexeme of type *x* and marks the current character as a beginning of that lexeme;

me(*x*), where *x* is a lexeme type — marks the previous character as an end of the lexeme of type *x* and includes it into the *l*-chain;

error — informs for the appearance of a wrong character and ignores it afterwards.

When two consecutive procedures *mb* having one and the same argument occur the effect of the first one is ignored. The same is valid for the procedure *me*.

One of the transitions from a given state may not be specified by a set of characters. Passing through such a transition is allowed when the current character does not belong to any of the sets specifying the other transitions from that state.

An automaton describing the scanner is presented in Fig. 5. The sets of characters specifying a given transition are denoted by capital letters while the procedures are

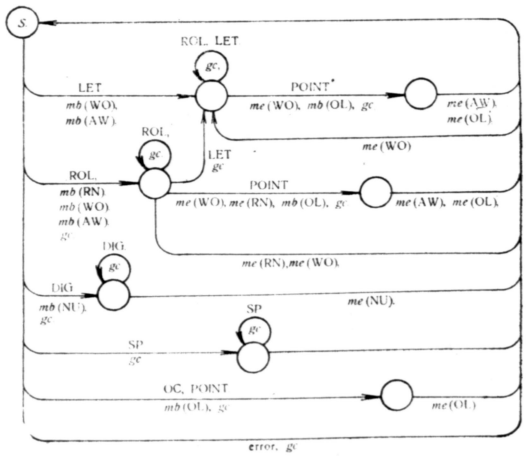

Fig. 5. An exemplary finite automaton for LA

denoted by small letters. Procedures are performed following the order of their occurrence in the automaton. The initial and the unique final state is *S*. The following sets of characters are used: letters (LET) — {A, B, E, F, G, H, J, K, N, O, P, Q, R, S, T, U, W, Y, Z}; letters and Roman numerals (ROL) — {I, V, X, L, C, D, M}; digits (DIG) — {0, 1, 2, 3, 4, 5, 6, 7, 8, 9}; a point (POINT) — {.}; a space (SP) — {␣}; other characters (OC) — {', —, ;, !, ?, ", :, /}.

The automaton described is rather universal and could be applied to a wide scope of specialized text classes.

In the next stage of LA the KNO lexemes are formed and the values of all the lexemes are obtained. The values of the OL lexemes are their ASCII codes, the values of the WO and AW lexemes are the numbers of characters in the corresponding lexeme, the values of the NU and RN lexemes are their numerical values.

The KNO lexemes represent words and word groups organized in a dictionary. The words and word groups having a similar meaning (e. g. "assistant professor", "researcher", etc.) form a notion which is represented in the dictionary and in the lexemes by a fixed numerical value. Any word or word group may be an element of different notions. Consequently any entry of the dictionary may be associated with more than one different values and as a result of the occurrence of such a word or word group the corresponding number of the KNO lexemes will appear in the *l*-chain.

4. Syntactic analysis. 4.1. Parsing automata. Automata described by means of transition graphs are applied to describing and parsing the phrases of a given text class. The transition from one state to another depends on the type of the current lexeme and probably on the result of the comparison between its value and a given constant. The six comparison operations grouped into three pairs of complementary operations are given below:

$= \neq$

$< \geqq$

$> \leqq$

An automaton with two transitions (the second one with a comparison) is shown in Fig. 6. A procedure *gl* may ba performed during any transition. It provides the following lexeme thus making it a current one. So it is possible to model arbitrary logical expressions by means of the three basic logical operations — conjunction, disjunction and negation. The negation is modelled by replacing the negating operation with its complementary operation. Modelling the conjuction $a \wedge b \wedge c$, the disjunction $a \vee b \vee c$ and the logical expression $(a=b) \vee (c<d) \wedge \overline{(e \geqq f)}$ is shown in Fig. 7 a), b) and c), respectively.

Fig. 6. Transitions with and without comparisons

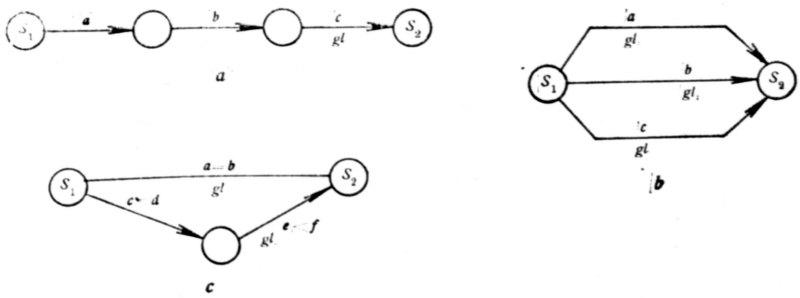

Fig. 7. Modelling logical expressions

The automata described enable transitions which could be passed through in spite of the type of the current lexeme. They are called unconditional transitions and are denoted by unmarked arcs. An unconditional transition having no *gl* procedure is called an empty one.

4.2. Subdescriptions. Usually the phrases are constructed of naturally separated parts — data items, subphrases, etc. It is more convenient to describe such parts in spite of the form of subdescription and then to include them into the phrase description as generalized transitions denoted by double arcs (see Fig. 8).That kind of automata could be used for the subdescriptions. Suppose that the double arcs X leads from state S_1 to state S_2. Then the real description (having no double arcs) is obtained by substituting X, namely connecting S_1 and the initial state of X with an empty transition and connecting each of the final states of X and S_2 again with some empty transitions. For example, the result of substituting X and Y by their subdescriptions (the final states are hatched) is shown in Fig. 8.

4.3. Nondeterministic automata. The proposed automata are nondeterministic. For instance, if the current state is S_{11} (see Fig. 8 d) and the current lexeme satisfies h, then S_3 and S_{12} are accessible.

It is known that any nondeterministic automaton can be transformed into a deterministic one. However, this statement is true only for pure automata, i. e. automata

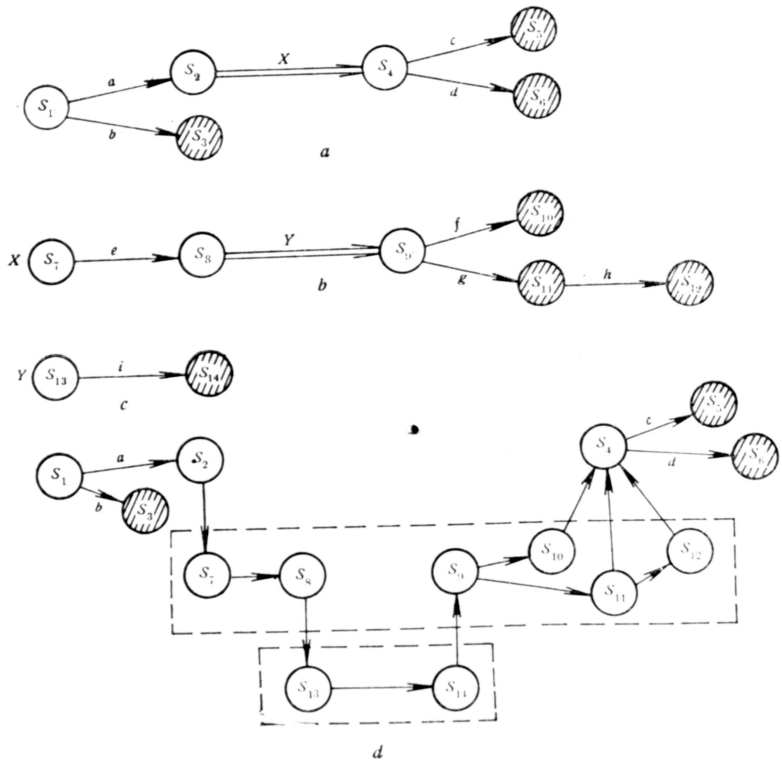

Fig. 8. Semantics of the subdescriptions (the description (d) is obtained in result of the inclusion of X and Y into the description (a))

whose unique aim is recognition. In our case some additional procedures are performed at certain states in order to provide the necessary information for CA and DT. These procedures are:

$bd(d)$ — marks the current lexeme as a beginning of the data item d;

ed — marks the previous lexeme as an end of the data item initialized by the procedure bd;

Since the transformation of a nondeterministic automaton to a deterministic one leads to uniting some states into generalized ones as well as to appearance of new transitions, then the specified procedures may mix and probably they would not mark appropriately the beginnings and the ends of the data items.

4.4. Dynamic model of SA. The nondeterminism and the possibility of activating more than one automata leads to working out of a suitable dynamic model of their operation. The characteristic features of the model proposed are:

a) Whenever activating an automaton which describes an arbitrary phrase a different agent of that automaton will be used. Any of its agents contains information on: the current state; the stack of the returned states; the current data item (the identifier, the initial and probably the final lexeme); the pointer to the list of data items being identified in the phrase up to a certain moment (moment under consideration).

When a new data item is initialized by the procedure *bd* or an agent is included into the chain of correspondence, then the current data item will be included into the list of the identified data items.

b) Whenever an agent reaches any of the states where a double arc originates from, then the automaton representing the corresponding subdescription is activated. For this reason the inclusion is never performed preliminarily and therefore static resulting automata like that in Fig. 8d do not exist. The mechanism of subroutines is applied to that run time inclusion, i. e. when an agent at state S_1 tries to pass to state S_2 through the subdescription X then the initial state of X becomes a current state of that agent and S_2 is pushed into the stack of the returned states. When the agent reaches the final state of X the state at the top of the stack (in this case S_2) is popped and the current state of the agent is replaced by it.

c) When an agent reaches a final state of an automaton describing a certain phrase the data items of that phrase should be included into the chain of correspondence.

d) When an agent cannot pass from the current state to another one it is terminated.

e) When an agent can pass through more than one transition it is multiplied into as many agents as the number of the transitions it should pass through. Then the new formed agents pass through those transitions.

4.5. An example. The description of phrase from the class "Autobiography" with a complete subdescription of the data item DATE is shown in Fig. 9 (for clarity the real values of some lexemes are replaced by their meanings put in quotation marks).

5. Context analysis. CA aims at reducing the chain of correspondence obtained by SA. It determines whether the interpretations embedded in the chain of correspondence satisfy the preliminarily described text structure. Since the methods and the algorithms for CA were presented in [5] only some basic features of CA are outlined here.

The structure of the text class is represented by a tree. Any node of the tree represents a set of identifiers of data items. The set of a nonterminal node is a union of the sets of its successors. The set of a terminal node contains the identifier of one data item only determined by the one-to-one correspondence between the leaves of the tree and the data items of the class considered. A specific feature of that type of tree is that the nearest successors of any node are linked by one of the following three kinds of links: a) free link (Fig. 10a); b) one-way link (Fig. 10b); c) incompatibility link (Fig. 10c).

A free link means that the data items with identifiers belonging to F may appear in the text in an arbitrary order but only in a group, i. e. no data item which does not belong to F could appear between any two data items belonging to F. A one-way link means that the data items belonging to the successors of F should appear in the text not only in a group but also in a predetermined order $S_1, S_2, \ldots, S_k$. An incompatibility link means that only data items belonging to no more than one successors of F could appear in the text.

The structure of the data item "address" is presented in Fig. 11 by a tree of that type. This description reflects the address structure in Bulgaria. The tree describing the text class "A historic personality entry of the Encyclopaedia "Bulgaria" is shown in Fig. 12.

CA reduces the chain of correspondence by eliminating the data items breaking the text structure.

6. Man-computer dialogue. The chain of correspondence obtained by CA has one of the following properties: a) there is no interpretation of the text; b) there are more than one interpretations of the text; c) there is only one interpretation of the text.

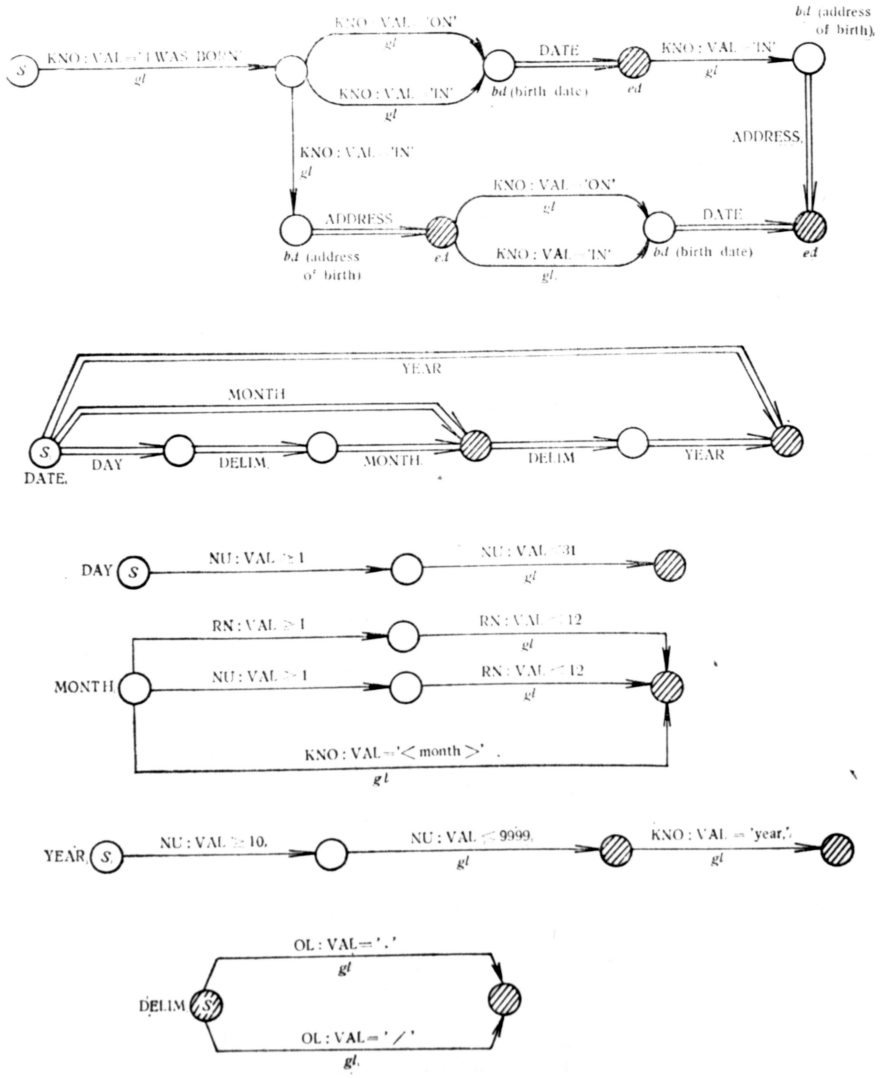

Fig. 9. An example of phrase description from the text class "Autobiography" with a complete subdescription of the data item "DATE" (the descriptions of the dates follow their structure in Bulgarian but the words are given in English)

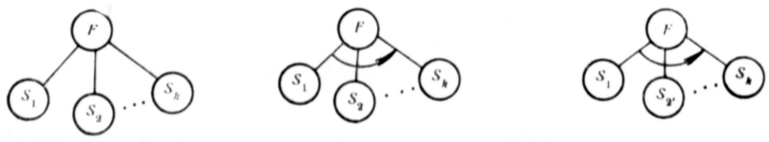

Fig. 10. The three kinds of links used in text structures

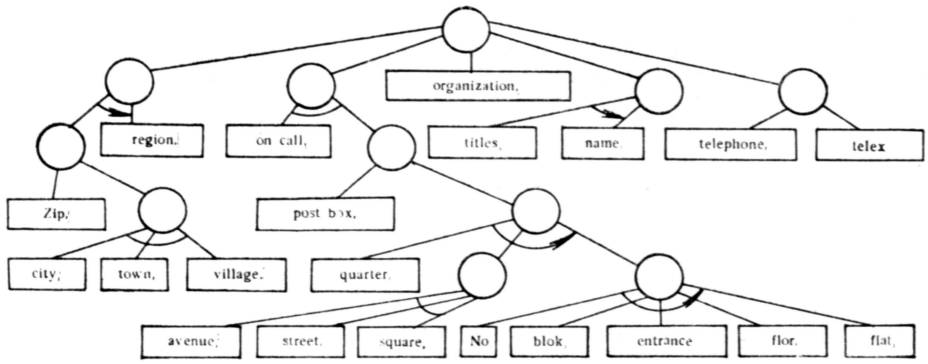

Fig. 11. Structure description of the data item "ADDRESS"

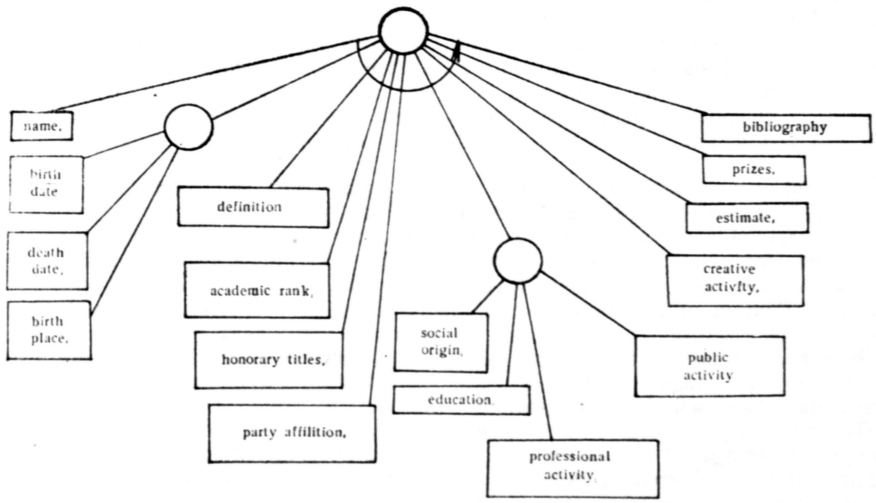

Fig. 12. Structure description of the text class 'A historic personality entry of the Encyclopaedia "BULGARIA"'

Evidently the first two cases are unsatisfactory. In the case c) the data translation may begin but a final confirmation for the correctness of the interpretation could not be given before the data control is realized over the internal data representation. The purpose of the MCD is to ensure a proper connection between the user and the analysis process in cases a) and b) in order to accomplish the text analysis and to obtain a chain with the property c). MCD has two main functions: to present the results of the analysis to the user in a suitable form and to receive his instructions aimed at correcting the analysis.

The user can intervene in the translator's work in the following ways: to make corrections in the original text by substituting the substring wrongly analyzed by the right one; to choose the right interpretation from the set of interpretations for a sub-

string ambiguously recognized; to indicate directly the identifiers and to represent some data items in the text. The following dialogue illustrates how the MCD could be applied when choosing the right interpretation.

An excerpt from the text:

```
. . . MY HOME ADDRESS IS:
    SOFIA — 1120
    RAKOVSKY STREET 26
    DR. ASSENOV  . . .
MCD: THE SUBSTRING "DR. ASSENOV" IS AMBIGUOUS:
    WHICH OF THE FOLLOWING INTERPRETATIONS IS VALID:
        1. ACADEMIC STATUS: DR.
           NAME          : ASSENOV
        2. NAME          : DR. ASSENOV
        3. NONE OF THE PREVIOUS TWO
```

The user: 2.

7. Data translation. In principle translation is not only transformation but it rather consists of analyzing the announcement in the source language and its synthesis in the target language representing the same information. This applies to data translation as well. The analysis of a compound data item aims at checking its syntactic and structural correctness. The techniques described in sections 3, 4, 5 and 6 could be used to obtain its elements (simpler data items, e. g. a date consists of the elements: day, month and year). The synthesis of the internal data representation has the following stages:

a) transformation of each element of the data item translated from the format appears in *M*-text into the format specified by the *I*-language;

b) control of the values of the data item elements as well as the internal structure of the data item (the internal structure is represented by the relations between the values of the data item elements);

c) reunification of the data item elements into the structure required by the *I*-language.

Since the formats in the *M* and *I*-languages vary to a great extent only a general view of the three stages of the synthesis will be presented and an example of one concrete data type will be discussed.

The values of the simple (not compound) data items are represented in the *M*-language by strings of characters while in the *I*-language they may be represented either by strings of characters or by numerical values. Some standard algorithms for transforming the representations in the *M*-language into those in the *I*-language could be applied.

The paper [1] is devoted to the data control problem. A table language describing the conditions which should be satisfied by the single values or groups of values has been proposed. A proper translator has been developed for the practical application of that language. Since its use requires unified data representations it could hardly be applied before stage a).

The reunification of the elements of a certain data item into a complete structure may vary from simple unification of the elements into one of the known structures (array, record, etc.) to obtaining a new value as a function of the values of the elements.

Translation of dates will be discussed as an illustration. Let us suppose that in the *M*-language the dates are described as it is shown in Fig. 9 and in the *I*-language they are represented in two bytes as it is shown in Fig. 13.

Only 128 years can be presented in seven bits. Since a much more limited interval has been used with many other tasks this range is not considered a great restriction.

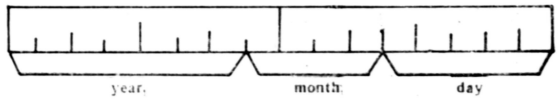

Fig. 13. Internal representation of the dates

The interval [1901, 2000] has been accepted in this example and the year has to be diminished by 1900 prior to its unification with the other two elements.

Let us denote by x, y and z the numeric values of day, month and year respectively. If any of these elements is absent its value will be considered 0. According to the scheme described in the first stage x, y and z are obtained from the substrings representing the values of the corresponding elements in the text. Values represented by numbers or Roman numbers are taken directly from the corresponding lexeme. When a month is represented by a word one of its values in the dictionary should indicate the number of the month. For example, the interval [1101, 1112] could be determined for representing the months in the dictionary. Then the subdescription "MONTH" in Fig. 9 should be specified by replacing the transition KNO: VAL = ⟨month⟩' with the proper modelling of the expression KNO: $(\mathrm{VAL} \geqq 1101) \wedge (\mathrm{VAL} \leqq 1112)$. In this case y is calculated by subtracting 1100 from the value of the corresponding KNO lexeme.

Taking into account that by 15.03.85 is meant 15.03.1985, when the number representing the year belongs to the interval [1, 99] this number should be increased by 1900 in order to obtain z.

The compatibility of x, y and z is checked by the following logical expression $((y=0 \vee y=1 \vee y=3 \vee y=5 \vee y=7 \vee y=8 \vee y=10 \vee y=12) \wedge (x \leqq 31)) \vee ((y=4 \vee y=6 \vee y=9 \vee y=11) \wedge x \leqq 30) \vee (y=2 \wedge (x<29 \vee (x=29 \wedge z=0 \bmod 4)))$. If its value is false then evidently this date will not be correct.

The unification of x, y, and z into a number having the structure shown in Fig. 13 is accomplished according to the formula $x+32y+512(z-1900)$.

8. Discussion. An experimental system was worked out following the model described. In order to check its operation the system was tuned up to the class of postal addresses considered as a relatively simple one. The tuning was aided by additionally developed modules of creating and modifying lexical, syntactic and structural descriptions. The system was complemented with modules for maintaining an archive of addresses and their transformation from the *I*-language into a form suitable for human perception. The last function is quite universal because it enables the parametric concretization of strictly defined output formats.

The formal description of specialized text classes is of great interest itself and deserves special attention. Here we shall restrict in the limits of enumerating only three text classes and mention their data items. The structure of the text class "biographical sketch" is presented in Fig. 12. The announcements for scientific events form another class. Its specific data items are: event (congress, conference, symposium, etc.); time; place; program and organizing committees; preliminary programme; instruction for authors; registration fee; etc. Recipes appear to be another interesting class. Their data items are divided into two groups — a) products and quantities; b) ways of the meal preparation (i. e. boiling, baking, mixing, straining, etc.), the order of the operations and some of the parameters characterizing those operations (e. g. its duration).

The fact that the chain of correspondence tolerates unrecognized substrings shows that some insignificant parts of the texts of a given class may not be described. That is the reason why the strategy of syntax description by means of a set of automata controlled by the text structure has been chosen. For example, in the case described in Fig. 14 a no substring may appear between the data items a and b while in the case

of Fig. 14 b this is possible. Describing the syntax and the structure separately leads to higher flexibility.

Some phrases have well specified fixed parts (e. g. "I WAS BORN" in Fig. 9) which identify reliably the corresponding phrase. This feature makes possible the location of some errors or devaitions from the M-language and facilitates the user in cor-

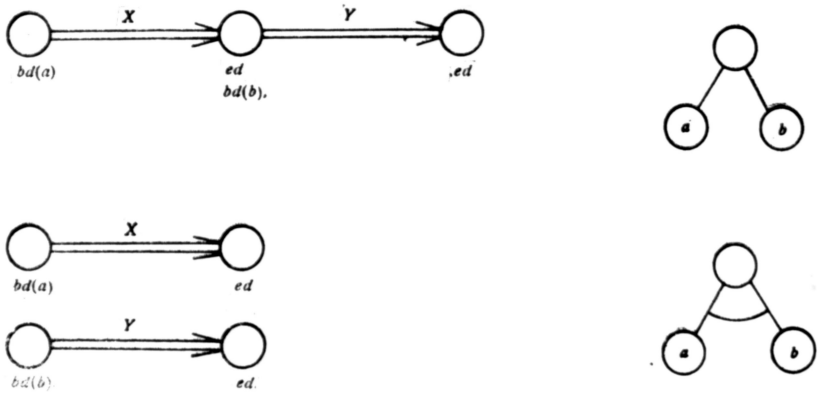

Fig. 14. Impossibility (a) and possibility (b) for occurrence of undescribed parts between two describe data items

recting the text when necessary. For this reason phrases consisting entirely of fixed parts and containing no data items have to be described syntactically and included in the text structure as dummy data items, thus creating some "milestones" of the CA.

The integration of the additional modules of tuning into a complete system would provide a powerful tool for automating the description of text classes. A link between the editor of the syntax descriptions and the dictionary would be very helpful. When creating syntax descriptions we may not be concerned of the exact values of the notions specifying certain transitions but we could work with their meanings (as in Fig. 9). Libraries of descriptions and subdescriptions could be organized and used in order to describe different classes by analogy with the methods of program development automation. Separate translators of data items being frequently used could be developed as proposed in [3].

9. Conclusion. The paper considers the principles which should be observed when constructing the basic modules of the $M \rightarrow I$ translator. A translator universality is achieved by a separate description of the lexis, syntax and structure of the text class supposed to be analyzed and the consequent tuning of the translator to that class. An appropriate environment facilitating the process of description is proposed. A series of experiments should be carried out in order to check how the translator will operate when texts of different classes are analysed and to estimate how it would work in practice.

This work was accomplished at the section of Software at the Centre of Mathematics and Mechanics. The microcomputer IBM PC/XT was used.

I would like to express my gratitude to Prof. Barnev for his help in the working process when dealing with those problems.

REFERENCES

1. P. Barnev, P. Dumkov. Analyzing the Correctness of Data Prior to Processing. *Serdica*, 2, 1976, 350-359.
2. P. Barnev, S. Kerpedjiev. Automatic Data Extraction from Specialized Texts. (in print).
3. П. Барнев, С. Керпеджиев. Подход к вводу данных в свободном формате, примененный для дат и почтовых адресов. Математика и математическо образование (13-та прол. конф. на СМБ). С., 1984, 244-255.
4. Д. Грис. Конструирование компиляторов для цифровых вычислительных машин. М., 1975.
5. С. Керпеджиев. Использование контекстной информации для распознавания составных объектов. Математика и математическо образование (14-та прол. конф. на СМБ). С., 1985, 435-441.
6. П. Уинстон. Искусственный интеллект. М., 1980.

Centre for Mathematics and Mechanics
Sofia 1090 *P. O. Box 373*

Received 15. 11. 1985
Revised 26. 11. 1986