Serdica
Bulgariacae mathematicae publicationes

Сердика
Българско математическо списание

AUTOMATIC DRAWING OF SYNTAX DIAGRAMS

MARGARITA R. BARNEVA, KAMEN D. KANEV

1. The Problem. Quite many programming languages are described by context-free grammars. The syntax of such languages is usually expressed in Backus-Naur metalanguage [1]. Recently in a number of instances the so-called graph diagrams or syntax-diagrams are used for visualisation of the most important syntax units [3]. With regard to this, the problem of automatization of the drawing process arises, mainly for languages described by context-free grammars, aiming to avoid labour-consuming activity, making mistakes and to provide common-style drawing. For that purpose it is appropriate a microcomputer with graphic capabilities to be used interactively. In this way syntax editing and error correction becomes easier in the course of the session.

Experimental drawing and some others conviniences are also provided. The method of syntax description is specifyed in Section 2. In Section 3 a command language is presented by which the syntax of user language is defined and Automatic Syntax Diagrams Drawing System (ASDDS) is controlled. A brief description for an implementation of the system and some exemplary results are given in Section 4.

2. Syntax representation of programming languages. As the theory of formal languages states [1, 2], a context-free grammar may be defined as $G=(N, T, P, S)$ where:

— N is a finite set of nonterminal symbols, called also metalinguistic variables (MLV) or notions;

— T is a finite set of terminal symbols, such that TUN=0;

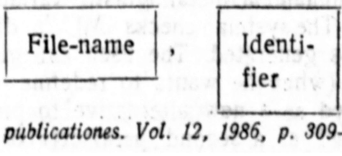
— P is a finite set of productions or derivation rules of type $A \longrightarrow \alpha$ or $(A ::= \alpha)$, where $A \in N$, and α is a string of terminal and nonterminal symbols, i. e. $\alpha \in (\mathrm{NUT})^*$. Elements of P are also called metalinguistic formulae.

— S is a special symbol of N, called also starting symbol or axiom. By the rules $p \in P$ are constructed admissible for language $L(G)$ sentences, i. e. language syntax is defined by set P.

2.1. Denoting terminal and nonterminal symbols. Most often the elemets of the set N are denoted by elements of set T, so it is necessary some agreements to be accepted, concerning the representation of terminal symbols and metalinguistic variables. In this paper we will denote N and T elements as follows:

a) Elements of T are enclosed in apostrophes. When the apostrophe is element of T, it is doubled. For example: '.', 'BEGIN', ''''. In syntax diagrams the terminal symbols are printed in elliptical blocks.

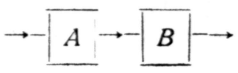
b) Elements of N are denoted by strings of letters, digits and underscore (—) sings, begining by letter. For example: File-name, Integer, $k2$. As an additional symbol the minus (—) sign may be used, which means that the line continues below. For example: Identi-fier. In syntax diagrams the nonterminal symbols are printed in rectangular blocks. For example:



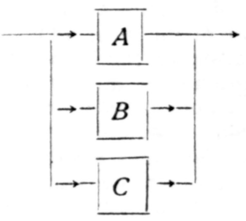
2.2. Representation of derivation rules. Each derivation rule $p_i \in P$ has the following form: $n_i ::= \alpha$, where $n_i \in N$ and α is an expression, built up of components and operators. Terminal and nonterminal symbols are components. Operators used in the derivation rules are:

a) Concatenation ($\sqcup$) and alternation ($|$) operators.

Strings are formed by concatenations. For example: $A \sqcup B$. The corresponding graph diagram has the form:



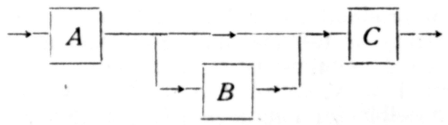
Two or more equal in rights possibilities are indicated by alternations. For example: $A | B | C$. The corresponding graph-diagram is:



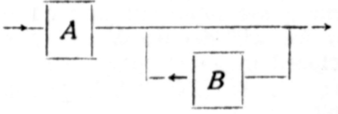
The priority of the concatenation is higher, i. e. $A \sqcup B | C$ means: $(A \sqcup B)$ or (C) but not $(A \sqcup B)$ or $(A \sqcup C)$.

b) Parenthesis by which the order of operations is altered. For example: $A \sqcup (B | C)$ means $A \sqcup B$ or $A \sqcup C$. This may also be expressed as $A(B | C)$.

c) Square brackets "[" and "]" denote zero or one occurrence of the enclosed substring. For example: $A [B] C$ means $A \sqcup B \sqcup C$ or $A \sqcup C$. The corresponding graph-diagram is:



d) Braces "{" and "}" denote zero or more occurrences of a substring. For example $A\{B\}$ means A or $A \sqcup B$ or $A \sqcup B \sqcup B \ldots$ The corresponding graph-diagram is



The number of components and operators in derivation rules is not restricted.

3. Command language. The command language of ASDDS consists of set of commands which determine all possible actions. Each command has its own mnemonic code and parameters. The commands are grouped as follows:

3.1. Input commands.

a) DEFINE. By this command a metalinguistic variable is defined. The MLV's name is the first parameter. The system checks MLV's dictionary and if the name is already defined a message is generated. The user can give up defining current MLV (in case of error), confirm (when he wants to redefine current MLV) or require the current definition to be added as a new alternative to previous one The right side of the derivation rule is entered as a second parameter of the command. The program checks each entered string for syntax errors and if correct, it is included in the dictionary. When errors are detected they are underlined and the user is prompted to reenter the string.

b) RDEF. By this command the user can remove the derivation rule for a selected metalinguistic variable. Its name is entered as a parameter of the RDEF command.

3.2. Syntax diagrams modification commands.

a) SUBSTITUTE. By this command the user can substitute the derivation rule for a selected metalinguistic variable. As first parameter the name of the substituted MLV is given and as second parameter the name of the MLV in which derivation rule substitutions are to be made is given. For example let's have the rules: $A::=C[D]$ and $B::=A\sqcup E$. After execution of the command SUBSTITUTE A IN B the second derivation rule will be $= B::C[D]E$. Generally by command SUBSTITUTE some components in derivation rules are represented by more detailed drawings.

b) RSUBST. This command restores selected derivation rule to the state before application of command SUBSTITUTE, i. e. RSUBST has an action opposite to SUBSTITUTE. In case that a variable is substituted more than once in depth for selected derivation rule, the restoration of previous state is accomplished by only one execution of the command RSUBST. For example, let's have the derivation rules: $B::=A\sqcup C$ and $A::=K|A\sqcup K$. When the command SUBSTITUTE A IN B is executed twice the rule for B will be $B::=(K|(K|A\sqcup K)K)C$. RSUBST A IN B produces $B::=A\sqcup C$.

3.3. Commands providing data transfer to/from external devices.

a) SAVE. By this command all derivation rules are stored on external device. It could be used when the session is to be interrupted and the current state of syntax description should be preserved.

b) LOAD. This command loads the data preserved by SAVE command. It can also be used for loading of before-hand prepared text of derivation rules in standard BNF or EBNF.

c) CSAVE. It is similar to the SAVE command. The only difference is that the current state of syntax-description is preserved in internal representation. The data preserved by CSAVE command is compressed and occupies less space on the external device.

d) CLOAD. The same as LOAD but the syntax-description processed by CLOAD could be prepared by CSAVE only. The command CLOAD inputs the data faster than LOAD.

3.4. Command providing drawing of syntax diagrams.

a) DRAW. By this command the corresponding to the selected MLV syntax-diagram is drawn. MLV's name is given as the only parameter of the command. All syntax diagrams are drawn on graphic display. The DRAW command operates in two ways selected by commands SON and SOFF.

b) SON. Enables substitutions and all already made substitutions are represented in the drawn syntax-diagrams.

c) SOFF. Disables substitutions and all substitutions are skipped off during the drawing process.

3.5. Additional commands.

a) SYN. This command displays all synonyms of a given MLV, which name is a parameter of the command. When no parameter is entered a full table of synonyms for all MLVs is printed.

b) LIST. This command displays the syntax of user's language, entered by that moment.

c) XREF. This command displays full cross-reference table for currently entered syntax.

d) HELP. Prints some operating instructions for the user and explanations of command language syntax.

The results (when using commands in groups 3.5 and 3.6) could be copied on paper by functions of the operating system.

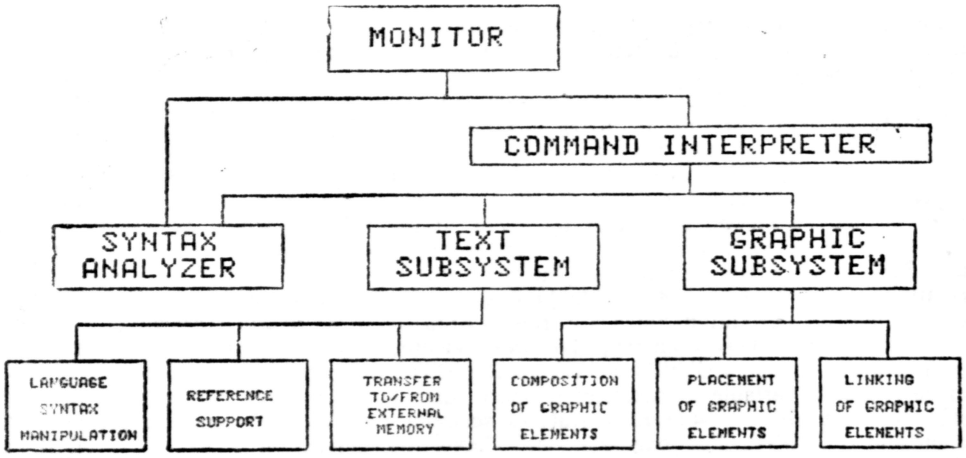

MONITOR
COMMAND INTERPRETER
SYNTAX ANALYZER
TEXT SUBSYSTEM
GRAPHIC SUBSYSTEM
LANGUAGE SYNTAX MANIPULATION
REFERENCE SUPPORT
TRANSFER TO/FROM EXTERNAL MEMORY
COMPOSITION OF GRAPHIC ELEMENTS
PLACEMENT OF GRAPHIC ELEMENTS
LINKING OF GRAPHIC ELEMENTS

Fig. 1

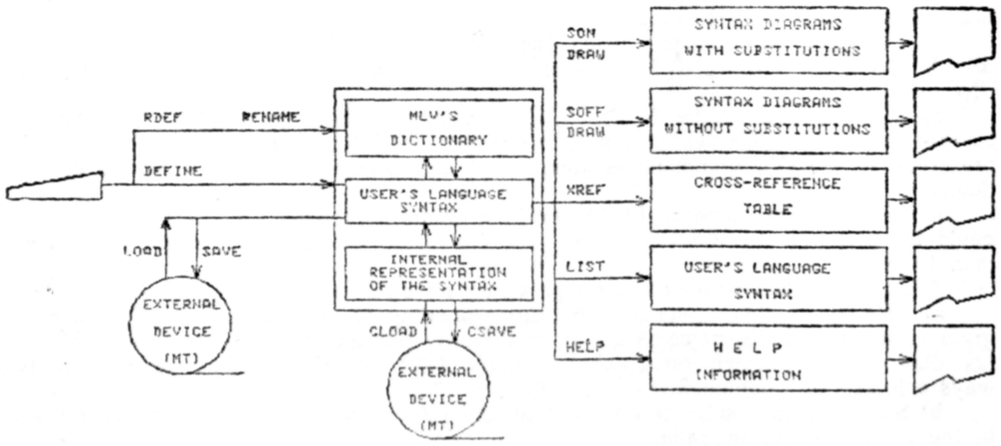


Fig. 2

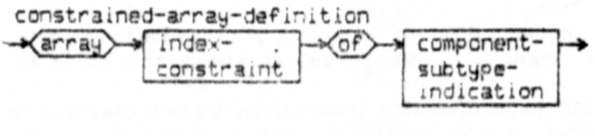


Fig. 3

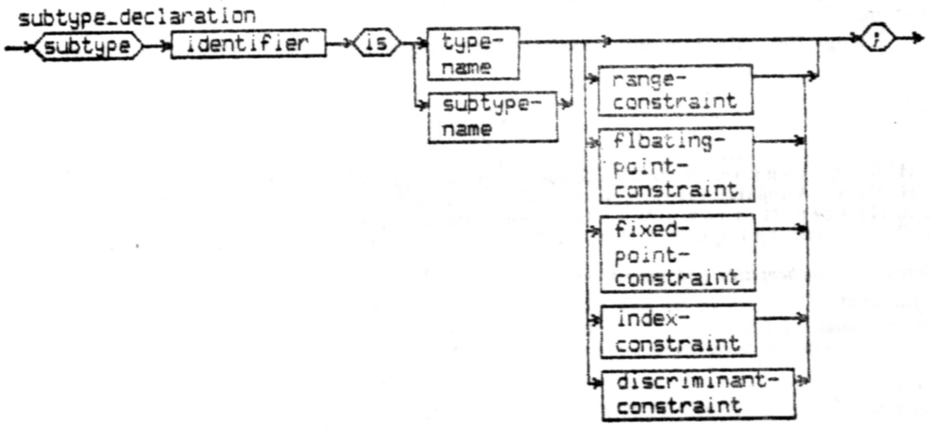


Fig. 4

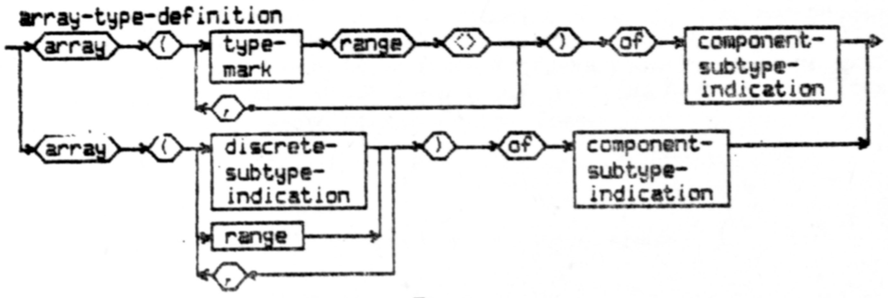

array-type-definition
array
(
type-mark
range
<>
)
of
component-subtype-indication
,
array
(
discrete-subtype-indication
range
,
)
of
component-subtype-indication

Fig. 5

4. Implementation. The described system is experimentally implemented by program modules written for the HP 2647A — Intelligent Graphics Terminal. The terminal is used in the local operating mode. All program modules are written in BASIC. In the graphic modules some commands of the graphic extension AGL are used.

The structure of ASDDS is presented in Fig. 1. The MONITOR carryes out a dialogue with the user. It is arranged in such a way, that the user is continuously directed by help messages. He is also asked some additional questions when neccessary (e. g. missing parameter, illegal name et al.). The SYNTAX ANALYZER performs two base functions. It checks the syntax of each entered command line and also analyzes all metalinguistic formulae. The COMMAND INTERPRETER processes error-free command lines only. In Fig. 2 the functional scheme of the system is shown. The names of commands are printed nearby corresponding arrows, pointing to appropriate functional blocks.

The implemented ASDDS is fully portable. The second version of the system is implemented on DEC-Professional/300 Graphic Terminal. The only few changes were made in modules dealing with graphics.

In Figs 5,3 and 4 are shown some exemplary syntax diagrams, drawn by described system.

Using the ASDDS all base syntax diagrams of ADA language [4] were drawn.

REFERENCES

1. И. Л. Братчиков. Синтаксис языков программирования. М., 1978.
2. Н. Уирт. Алгоритми+структури от данни=програми. С., 1980.
3. К. Йенсен, Н. Вирт. ПАСКАЛЬ. Руководство для пользователя и описание языка. М., 1983.
4. The programming language ADA reference manual. (*Lecture Notes in Comp. Sci.*, Vol. 155), Berlin, 1981.

Centre for mathematics and mechanics
Sofia 1090 P. O. Box 373

Received 8. 5. 1984