SYNTHON APPROACH AND GRAPH GRAMMAR*

Vladimír Kvasnička

Department of Mathematics,
Slovak Institute of Technology, 812 37 Bratislava

Received July 14th, 1982

The Corey's synthon approach is formalized by a special type of graph grammar. The graph language induced by this grammar is composed of all possible graphs (molecules) that can be constructed from the so-called starting graphs. The retro-synthetic analysis is treated as the parser which provides an answer to the question „is a graph element of the given language?".

In our previous communication[1] we have suggested the graph-theory formalism of the organic chemistry. The molecular system is considered as a multigraph with loops, its vertices are evaluated by surjective mapping onto the vocabulary of vertex labels (*e.g.* atomic symbols). In the framework of this formalism it is possible to give unambiguous determination of many notions and concepts that are naturally in the computer simulation of organic chemistry. In particulat, the Corey's concept of synthon (which plays a fundamental role in the so-called first-generation programs) was determined[1,2] as a special subgraph of the given graph. The synthons allow to overcome many problems and pitfalls with proper coding of chemical reactivity, a synthon is assigned directly to a preselected type of reaction. What is slightly discouraging here, to cover all most important chemical reactions we have to use few hundreds different synthons, it may give rise to very serious technical difficulties in an actual implementation of this technique. This unpleasant feature of the pure synthon approach is partially sormounted by making use of the mechanistic approach[3-12] (the second-generation programs). It means that the pertinent synthons are not *apriori* prescribed but they are generated following simple chemical heuristics and/or qualitative (semi-quantitative) calculations. Hence, the mechanistic approach needs a simple as well as efficient mathematical model of chemical reactivity of organic molecules. To our knowledge, this is the main problem not yet resolved in the implementation of second-generation programs.

The purpose of this communication is to formalize the Corey's synthon approach in such a way that a formal graph grammar is elaborated. It means that the organic chemistry (of course, only in the framework of the synthon model) may be formally

* Part II in the series Mathematical Model of Organic Chemistry; Part I: This Journal *48*, 2097 (1983).

considered as a graph language induced by the graph grammar. This offers very powerful formal tool to support our tries to implement a computer program which will simulate the organic chemistry in whole its diversity.

The graph grammars are extensively used in the syntactic methods of pattern recognition[13]. They are usually formed in such a way that there exists a simple formal correspondence with the standard string grammars[14,15]. This is not very appropriate for the present purposes, therefore we shall formulate initially a special type of string grammar which may be formally considered as a realization of the context-sensitive grammar[14]. Then we formulate a graph grammar with direct correspondence with our string grammar. Usually, the string grammars are defined by making use of an auxiliary device of the so-called non-terminal symbols that are serving for the generation of the proper strings. The concept of non-terminal symbols is now fully omitted, in some extent they are substituted by the notion of initial strings (see below).

Recently, the web grammar of Phaltz and Rosenfeld[17,13] (a special type of graph grammar) has been used in chemistry by Balaban and coworkers[16] in the generation of acyclic isopreniod structures.

String Grammar

The *alphabet* $\sum = \{a, b, c, \ldots\}$ is a finite set of symbols $a, b, c, \ldots$. The set $\sum^*$ is composed of all possible strings that can be formed over the alphaber $\sum$, including the so-called empty string λ,

$$\sum^* = \{\lambda, a, b, \ldots, aa, ab, ba, \ldots, abc, bca, \ldots\}\,, \qquad (1)$$

where the strings that are composed of the same symbols but different in their order are considered as distinct (*e.g.* the string *ab* and *ba* are distinct). The strings are denoted by lower-case Greek letters. The length of string $\sigma \in \sum^*$ is the number of symbols in the string, it will be denoted by $|\sigma|$. By definition, the length of empty string is zero, $|\lambda| = 0$. The concatenation of two strings $\sigma_1, \sigma_2 \in \sum^*$ is a string $\sigma = \sigma_1\sigma_2$ from $\sum^*$. In particular, for the empty string we have $\lambda\sigma = \sigma\lambda = \sigma$, for an arbitrary $\sigma \in \sum^*$. The string β is a substring of the string σ (formally $\beta \subseteq \sigma$) iff the string σ may be written as $\sigma = \alpha\beta\gamma$.

The central concept of a grammar (introduced over the alphabet $\sum$) is a finite (non-empty) set $\mathfrak{P}$ of the so-called productions, which describe how the strings of a language are to be generated. In order to specify the set of production we have to introduce the following two non-empty sets:

(*1*) The set of left strings

$$V_{1s} = \{\alpha_1, \alpha_2, \ldots\} \subseteq \sum^*\,, \qquad (2a)$$

(2) the set of right strings

$$V_{rs} = \{\beta_1, \beta_2, \ldots\} \subseteq \sum^* . \qquad (2b)$$

Then the set of productions, $\mathfrak{P}$, is defined as a subset of the direct product of V_{ls} and V_{rs},

$$\mathfrak{P} = \{P_1, P_2, \ldots\} \subseteq V_{ls} \times V_{rs} . \qquad (3)$$

Hence, a production $P \in \mathfrak{P}$ is a pair of strings (α, β) [we shall use the descriptive convention $\alpha \to \beta$), for $\alpha \in V_{ls}$ and $\beta \in V_{rs}$. We say that a string σ' is directly derived from a string $\sigma (\sigma \overset{P}{\Rightarrow} \sigma')$ by making use of the production $P : \alpha \to \beta$ if the strings σ and σ' may be written as $\sigma = \omega\alpha\chi$ and $\sigma' = \omega\beta\chi$. In general, the term $\sigma \overset{*}{\Rightarrow} \sigma'$ denotes that a string σ' is derived from a string σ, then we have a sequence of strings $\sigma_1, \sigma_2, \ldots, \sigma_n$ such that $\sigma = \sigma_1$, $\sigma' = \sigma_n$, and σ_{i+1} is directly derived from σ_i . $(\sigma_i \overset{P}{\Rightarrow} \sigma_{i+1})$, for $i = 1, 2, \ldots, n-1$. The sequence $\sigma_1, \sigma_2, \ldots, \sigma_n$ is called the derivation of σ' from σ.

Let us introduce a non-empty set of starting strings

$$V_{ss} = \{\mu_1, \mu_2, \ldots\} \subseteq \sum^* , \qquad (4)$$

where we require that each string $\mu \in V_{ss}$ has at least a one substring from the set V_{ls}, *i.e.* for an arbitrary $\mu \in V_{ss}$ there exists a substring $\alpha \subseteq \mu$ such that $\alpha \in V_{ls}$. Now we are ready to define a grammar G as the ordered 2-tuple

$$G = (V_{ss}, \mathfrak{P}) . \qquad (5)$$

The language $L(G)$ induced by this grammar G is the set of all possible strings that can be derived from the starting strings in V_{ss},

$$L(G) = \{\sigma; \text{ there exists a starting string } \mu \in V_{ss} \text{ such that } \mu \overset{*}{\Rightarrow} \sigma\} \subseteq \sum^* . \qquad (6)$$

Example 1. The alphabet $\sum$ is determined by

$$\sum = \{a, b, c\} .$$

The sets V_{ls} and V_{rs} are

$$V_{ls} = \{aa, ab, ac, cb\} ,$$

$$V_{rs} = \{aab, baa, caa, aac, bbc\} .$$

The set of productions $\mathfrak{P}$ is a subset of $V_{ls} \times C_{rs}$, *e.g.* we put

$$P_1 : aa \to aab ,$$

$$P_2 : ab \to aab ,$$

$$P_3\colon\ ac \to aac\ ,$$

$$P_4\colon\ cb \to bbc\ .$$

Finally, the set of starting strings is

$$V_{ss} = \{\mu_1 = aab,\ \mu_2 = acb\}\ .$$

Let us try to construct a few illustrative samples of the language $L(G)$ induced by the grammer $G = (V_{ss}, \Psi)$. For instance, we start from μ_1, it contains two substrings aa and ab that are left members of the productions P_1 and P_2, respectively. We get

$$\mu_1 = aab \begin{cases} \xRightarrow{P_1} aabb = \sigma_1\ , \\ \xRightarrow{P_2} aaab = \sigma_2\ . \end{cases}$$

In similar way, for the starting string μ_2 we get

$$\mu_2 = acb \begin{cases} \xRightarrow{P_3} aacb = \sigma_3\ , \\ \xRightarrow{P_4} abbc = \sigma_4\ . \end{cases}$$

The strings σ_1 to σ_4 can serve for the production of next strings that are belonging to the language $L(G)$,

$$\sigma_1 \begin{cases} \xRightarrow{P_1} aabbb = \sigma_5\ , \\ \xRightarrow{P_2} aaabb = \sigma_6\ , \end{cases}$$

$$\sigma_2 \begin{cases} \xRightarrow{P_1} aabab = \sigma_7\ , \\ \xRightarrow{P_1} aaabb = \sigma_6\ , \\ \xRightarrow{P_2} aaaab = \sigma_8\ , \end{cases}$$

$$\sigma_3 \begin{cases} \xRightarrow{P_1} aabcb = \sigma_9\ , \\ \xRightarrow{P_3} aaacb = \sigma_{10}\ , \\ \xRightarrow{P_4} acbbc = \sigma_{11}\ , \end{cases}$$

$$\sigma_4 \xRightarrow{P_2} aabbc = \sigma_{12}\ .$$

The language $L(G)$ is composed of the following strings with the length 4 or 5,

$$L(G) = \{aabb,\ aaab,\ aacb,\ abbc,\ aabbb,\ aapb,\ aabab,\ aaaab,\ aabcb,\ aaacb,\ acbbc, \ldots\}\ .$$

After a grammar is constructed to induce a language, the next step standing before

us is to suggest a recognizer that will recognized the strings generated by the grammar. Formally, let us have a string $\sigma \in \sum^*$, then the recognizer solve the following problem: $\sigma \in L(G)$? The process that would result in an answer to such a question with respect to the given grammar G is, in general, called syntax, analysis or parsing. In addition to get an answer "yes" or "no", the process can also provide the generation of the so-called derivation tree of σ.

A very simple and straightforward form of implementing the recognizer is to look for in the input string σ a substring β ($\beta \subseteq \sigma$) which is equal to the right string of a production $P : \alpha \to \beta$. We arrive at the string σ', symbolically $\sigma' \xleftarrow{P} \sigma$ (this is called the retro-production). The whole process is repeated for the produced string σ', *etc.*,

$$\tilde{\sigma} \xleftarrow{\tilde{P}} \ldots \sigma'' \xleftarrow{P'} \sigma' \xleftarrow{P} \sigma \,. \tag{7}$$

The process is stopped if the resulting $\tilde{\sigma}$ does not contain a substring which is the right string β of an arbitrary production P. Finally, if the string $\tilde{\sigma}$ belongs to the set of starting strings V_{ss} (*i.e.* $\tilde{\sigma} \in V_{ss}$), then the input string σ is recognized from the language $L(G)$ [*i.e.* $\sigma \in L(G)$], in the opposite case the input string σ is not an element of $L(G)$.

Example 2. Let us recognize the input string $\sigma = aabbbc$ whether it belongs or not to the language $L(G)$ induced by the grammar G describe in example 1. The derivation tree is illustrated in Fig. 1. We see that the input string $\sigma \in L(G)$ can be derived from the starting string $\mu_2 = acb$ by three independent ways,

$$\mu_2 \xrightarrow{P_4} abbc \xrightarrow{P_2} aabbc \xrightarrow{P_1} \sigma \,,$$

$$\mu_2 \xrightarrow{P_3} aacb \xrightarrow{P_4} aabbc \xrightarrow{P_1} \sigma \,,$$

$$\mu_2 \xrightarrow{P_3} aacb \xrightarrow{P_1} aabcb \xrightarrow{P_4} \sigma \,.$$

Hence, the present simple recognizer offers not only an answer to the question "is $\sigma \in L(G)$?" but produces as a by-product the derivation tree from which it is easy to see the retro-derivation of σ.

Graph Grammar

Consider the graph[1]

$$G = (V, E, L, \varphi, \mathfrak{B}) \,, \tag{8}$$

its synthon is

$$G(Q) = (V_Q, E_Q, L_Q, \varphi, \mathfrak{B}) \subseteq G \,. \tag{9}$$

The graph G can be written as a union of the synton $G(Q)$ and the synthon comple-

ment $\tilde{G}(Q) = (\tilde{V}_Q, \tilde{E}_Q, \tilde{L}_Q, \varphi, \mathfrak{B})$,

$$G = G(Q)\,\mu\,\tilde{G}(A)\,, \tag{10}$$

where

$$V = V_Q \cup \tilde{V}_Q\,, \tag{11a}$$

$$E = E_Q \cup \tilde{E}_Q \quad \text{and} \quad E_Q \cap \tilde{E}_Q = \emptyset\,, \tag{11b}$$

$$L = L_Q \cup \tilde{L}_Q \quad \text{and} \quad L_Q \cap \tilde{L}_Q = \emptyset\,. \tag{11c}$$

The synthon $G(Q)$ and the synthon complement $\tilde{G}(Q)$ are edge and loop disjoint [Eqs (*11b*) and (*11c*)], while, in general, the vertex sets V_Q and $\tilde{V}_Q$ have common vertices.

A production P is determined as the ordered 3-tuple

$$P = (G_1, G_r, \pi)\,, \tag{12}$$

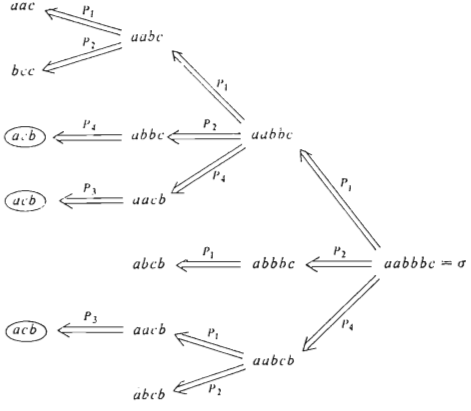


FIG. 1

The derivation tree of the string $\sigma = aabbbc$. The oval block indicates that the given leftmost string belong to the set $\mathfrak{A}$ composed of the starting strings

where

$$G_l = (V_l, E_l, L_l, \varphi_l, \mathfrak{B}), \tag{13a}$$

$$G_r = (V_r, E_r, L_r, \varphi_r, \mathfrak{B}), \tag{13b}$$

are the so-called left and right synthons, respectively. The term π in (12) is called the matching rule (see below).

Now we specify what it means when one says that the production P is aplied to the graph G. The matching rule π classifies some vertices in G_l and G_r as the matching vertices. Formally, we put

$$\pi = (V_{l,match}, V_{r,match}, \omega), \tag{14}$$

where the vertex subsets $V_{l,match} \subseteq V_l$ and $V_{r,match} \subseteq V_r$ are composed of the same number of vertices (*i.e.* $|V_{l,match}| = |V_{r,match}|$). The symbol ω is a bijective mapping of $V_{r,match}$ onto $V_{l,match}$, $\omega : V_{r,match} \to V_{l,match}$, which conserves the evaluation of vertices, *i.e.* $\omega(v) = v'$ implies $\varphi_r(v) = \varphi_l(v')$. In the initial stage of the application of P to G we have look for a synthon $G(Q) \subseteq G$ which is isomorphic with the left synthon $G_l \in P$. According to this isomorphism some part of vertices in G are unambiguously classified as the matching vertices, they form a subset of V denoted by $V_{G(Q),match}$. It can be alternatively determined [knowing the decomposition of G in $G(Q)$ and $\tilde{G}(Q)$] as a subset composed of those vertices that are simultaneously appearing in V_Q and $\tilde{V}_Q$,

$$V_{G(Q),match} = V_Q \cap \tilde{V}_Q \subseteq V. \tag{15}$$

Since the synthons G_l and $G(Q)$ are isomorphic we may formally construct a bijective mapping $\psi : V_{l,match} \to V_{G(Q),match}$, it is realized as a restriction of the original bijective mapping of V_l onto $V_{G(Q)}$ introduced in the framework of this isomorphism. Composing the mappings ω and ψ we get the bijective mapping $\chi = \omega\psi : V_{r,match} \to V_{G(Q),match}$, *i.e.* if $v \in V_{r,match}$ then $\omega[\psi(v)] = v' \in V_{G(Q),match}$ and $\varphi_r(v) = \varphi(v')$. By using this new bijective mapping χ we are able to assign to the right synthon G_r an isomorphic graph G'_r in such a way that the matching vertices in G_r are substituted by the corresponding ones in $G(Q)$. Finally, we determine the application of the production P to the graph G as the union of the graph G'_r (defined above) and the synthon complement $\tilde{G}(Q)$,

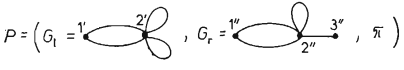
$$G \xRightarrow{P} G' = G'_r \cup \tilde{G}(Q). \tag{16}$$

Simply speaking, the application of the production P to the graph G means that a subgraph in G is substituted by another subgraph, these subgraphs are isomorphic

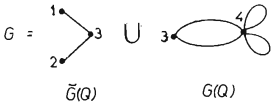
with the right and left synthons, respectively, and the matching rule π ensures the unambiguity of whole process.

Example 3. Let us consuder the graph[1]

The production P is determined by



It is easy to see, with respect to the left synthon G_1, that the graph G may be decomposed on the synthon $G(Q)$ and the synthon complement $\tilde{G}(Q)$ as follows

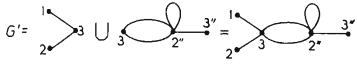


The matching rule is specified by

$$\pi = (V_{l,\mathrm{match}} = \{1'\}\,, \quad V_{r,\mathrm{match}} = \{1''\}\,, \quad \omega\colon 1'' \to 1')\,.$$

The mapping ψ induced by the isomorphism of G_1 and $G(Q)$ is simply determined by $\psi\colon 1' \to 3$. Hence, the composed mapping $\chi = \omega\psi\colon 1'' \to 3$. Of course, the evaluation of vertices $1'$, $1''$, and 3 should be of the same type. By using the mapping χ we construct

which is isomorphic with the original right synthon G_r. Finally, the result of the production P applied to the graph G is



We define the so-called graph grammar in similar way to the string grammar. The non-empty set $\mathfrak{P}$ is composed of the productions,

$$\mathfrak{P} = \{P_1, P_2, \ldots\}\,, \tag{17}$$

where each production $P \in \mathfrak{P}$ is determined by the formulae (*12*) and (*13*). The graph G' is directly derived from the graph G if there exists such a production $P \in \mathfrak{P}$ that the relation (*16*) is satisfied, *i.e.* $G \xRightarrow{P} G'$. This can be simply extended to the following more general notion: The graph G' is derived from the graph G (formally $G \xRightarrow{*} G'$) if we have a sequence of graphs $G_1, G_2, \ldots, G_n$ such that $G = G_1$, $G_n = G'$, and G_{i+1} is directly derived from G_i ($G_i \xRightarrow{P} G_{i+1}$), for $i = 1, 2, \ldots, n - 1$. The sequence of graphs $G_1, G_2, \ldots, G_n$ is called the derivation of G' from G.

Let us introduce a non-empty set of starting graphs

$$\mathfrak{A} = \{G_{sg}^{(1)}, G_{sg}^{(2)}, \ldots\} . \tag{18}$$

We require that each starting graph $G_{sg} \in \mathfrak{A}$ has at least a one synthon which is isomorphic with the left synthon G_l of a production $P \in \mathfrak{P}$. Finally, the graph grammar is is determined as the ordered 2-tuple

$$\mathfrak{G} = (\mathfrak{A}, \mathfrak{P}) . \tag{19}$$

The language $L(\mathfrak{G})$ induced by the grammar $\mathfrak{G}$ is the set of all possible graphs derived starting graphs in $\mathfrak{A}$,

$$L(\mathfrak{G}) = \{G; \text{ there exists a starting graph } G_{sg} \in \mathfrak{A} \text{ such that } G_{sg} \xRightarrow{*} G\} . \tag{20}$$

We give the following simple interpretation of the above formalism: The molecules are identified (formally) with the graphs, they represent unambiguously their constitutional formulae. The reactions that can run over the molecules are determined by the productions $P_1, P_2, \ldots$, the set $\mathfrak{P}$ is a formal representation of a set of reactions that are available for the chemical transformations. The grammar $\mathfrak{G}$ then means a chemistry over the starting molecules performed by the reactions from the set $\mathfrak{P}$. The language $L(\mathfrak{G})$ is composed of all possible molecules that can be synthetized from the starting molecules.

The recognizer of graphs from the language $L(\mathfrak{G})$ may be constructed in similar way to our illustrative string grammar. Here the main problem is to suggest a method providing an answer to the following question: Is $G \in L(\mathfrak{G})$?

Assume that we have a production $P = (G_l, G_r, \pi) \in \mathfrak{P}$. Then we look for in the analyzed (parsed) graph a synthon $G(Q)$ which is isomorphic with the right synthon $G_r \in P$. The graph G is equal to

$$G = G(Q) \cup \tilde{G}(Q) , \tag{21}$$

where $\tilde{G}(Q)$ is the synthon complement. The matching vertices in G are

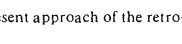
$$V_{G(Q),match} = V_{G(Q)} \cap \tilde{V}_{G(Q)} . \tag{22}$$

Employing the matching rule π and the isomorphism between G_r and $G(Q)$ we are able unambiguously to construct bijective mapping $\chi' : V_{1,\text{match}} \to V_{G(Q),\text{match}}$ which conserves the evaluation of matching vertices. From the left synthon G_1 we construct an isomorphic graph G_1' in such a way that the matching vertices in G_1 are substituted by the corresponding (from the standpoint of mapping χ') ones in $G(Q)$. The resulting graph G' of this inverse application of the production P to the graph G is determined as the union of the synthon complement $\tilde{G}(Q)$ and the graph G_1',

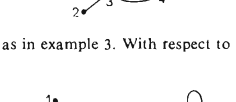
$$G' = G_1' \cup \tilde{G}(Q) \overset{P}{\Longleftarrow} G , \tag{23}$$

where $\overset{P}{\Longleftarrow}$ denotes the inverse application of the production P (which will be called the retro-production).

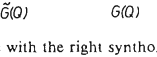
Example 4. We illustrate the present approach of the retro-production notion on the example 3. The input graph G is



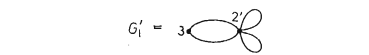
The production P is the same as in example 3. With respect to the right synthon G_r the graph G is expressed as follows



where the synthon $G(Q)$ is isomorphic with the right synthon G_r. The matching vertices in $G(Q)$ are determined by $V_{G(Q),\text{match}} = \{3\}$, and the mapping $\chi' : 3 \to 1'$. It means that the graph G_1' (isomorphic with G_1) is



Finally, the result of the inverse application of P to the graph G is



The resulting graph G' is isomorphic with the starting graph G in example 3, which was to be obtained.

After this simple example illustrating the notion of retroproduction we turn our attention again to the construction of a parser for the language $L(\mathfrak{G})$. We would like

to get an answer to the question whether or not a graph G belongs to the language $L(\mathfrak{G})$. The above procedure is repeatly applied to the graph G,

$$\tilde{G} \xleftarrow{\tilde{P}} \ldots G'' \xleftarrow{P'} G' \xleftarrow{P} G . \qquad (24)$$

The process is stopped if the resulting graph $\tilde{G}$ does not contain a synthon which is isomorphic with the right synthon of an arbitrary production $P \in \mathfrak{P}$. Finally, if the graph $\tilde{G} \in \mathfrak{A}$, then the input graph G is recognized from the language $L(\mathfrak{G})$, in the opposite case the input graph G is not an element of $L(\mathfrak{G})$. The present simple recognizer offers not only an answer to the question "is $G \in L(\mathfrak{G})$?" but produces as a by-product the derivation tree of the graph G.

Conclusions

We have demonstrated that the Corey's concept of synthons can be formalized in such a way that a special kind of graph grammar is suggested. The graph language induced by this grammar is composed of all possible graphs that can be constructed from the starting graphs. The retro-synthetic analysis is treated as the parser which gives an answer to the question "is a graph element of the given grammar?". If "yes", then we are sure that the considered graph may be constructed from the starting graph, and moreover, the parser has provided also its derivation tree. Consequently, one can say, the organic chemistry (at least, in framework of synthon approach) is formally equivalent to the graph language, its syntax and some part of the theory of chemical reactivity are covered by the corresponding graph grammar.

The author wishes to express his appreciation for many useful and stimulating discussions with Professor M. Kratochvíl and Dr J. Koča.

REFERENCES

1. Kvasnička V.: This Journal *48*, 2097 1983
2. Corey E. J.: Pure Appl. Chem. *14*, 19 (1967).
3. Dugundji J., Ugi I.: Topics Current Chem. *39*, 19 (1973).
4. Ugi I., Bauer J., Brandt J., Friedrich J., Gasteiger J., Jochum C., Schubert W.: Angew. Chem. Int. Ed. Engl. *18*, 111 (1979).
5. Hendrickson J. B.: Topics Current Chem.: *62*, 49 (1976).
6. Hendrickson J. B., Braun-Keller E., Toczko G. A.: Tetrahedron (Supplement) *37*, 319 (1981).
7. Weise A.: Z. Chem. *15*, 333 (1975); *17*, 108 (1977).
8. Weise A., Qcharow H.-G.: Z. Chem. *19*, 49 (1979).
9. Salatin T. D., Jorgensen W. L.: J. Org. Chem. *45*, 2043 (1980).
10. Kratochvíl M.: Chem. Listy *75*, 673 (1981).
11. Kratochvíl M.: Chem. Listy, in press.
12. Kratochvíl M., Koča J., Kvasnička V.: Chem. Listy, in press.

13. Fu K. S.: *Syntactic Methods in Pattern Recognition.* Academic Press, New York 1974.
14. Hopcroft J. E., Ullman J. D.: *Formal Languages and their Relation to Automata.* Addison-Wesley, Reading, Mass. 1969.
15. Aho A. V., Ullman J. D.: *The Theory of Parsing, Translation and Compiling*, Vol. I.: *Parsing.* Prentice-Hall, Englewood Cliffs, New Jersey 1972.
16. Phalz J. L., Rosenfeld A.: Proc. Int. Joitn Conf. Artificial Inteligence 1st, Washington D. C. 1969, p. 609.
17. Balaban A. T., Barasch M., Marcus S.: Match (Comm. Math. Chem.) *8*, 193 (1980).

Translated by the author.