match no. 8 pp. 193 - 213 1980

PICTURE GRAMMARS IN CHEMISTRY. GENERATION OF ACYCLIC ISOPRENOID STRUCTURES

Alexandru T.Balaban[a], Mariana Barasch[b] and Solomon Marcus[b]

[a]Department of Organic Chemistry, The Polytechnic,Str.Polizu 1, Bucharest, Roumania

[b]Department of Mathematics, University of Bucharest, Roumania

(received: March 1980)

Abstract. The generation of the infinite modalities for combining isoprene units into acyclic polyisoprenoid structures can be performed using picture grammars, in this case web grammars. They consist of a finite set of rules which, with restrictions concerning the mode of linking (regular i.e. head-to-tail, or standard i.e. head-to-tail, tail-to-head, head-to-head, tail-to-tail), afford the desired infinite collection of isoprenoid structures. Vice-versa, the parsing algorithm enables an analysis of a given structure in order to find out if it is isoprenoid of the required type. Future work is directed towards generalizing this approach to include cyclic isoprenoid structures.

1. Introduction

Given a molecular formula of a substance and knowing that the substance has an isoprenic skeleton, the problem raised is to find all the isoprenoid structural formulas corresponding to the molecular formula. In other words, the problem is to build all the isoprenoid graphs which can be decomposed into n isoprene units. Thus, we are confronted with a problem of generating a potentially infinite set of isoprenoid structures. The natural tool suitable to approach such a problem is the theory of automata and formal grammars, which has already been used in some problems of information retrieval in the field of chemistry.

But the existing applications of formal grammars in chemistry (see, for instance [1] and [2]) are all concerned with their classical form, due to Noam Chomsky [3]; this form was built having in mind natural languages, which are essentially of a concatenative, linear structure. Chomskian grammars are equally efficient in the study of the syntax of programming languages, because programming languages share with natural languages the property to be linear.

Structural formulas in chemistry are no longer built in a linear way. Similar situations arise in many other fields and the problem was raised fifteen years ago whether one cannot build some generative devices which, on the one hand, are similar to Chomskian grammars, and on the other hand, are suitable to generate polydimensional structures (i.e., structures which are represented in euclidian spaces with several dimensions). The answer to this question was the beginning of the study of a new type of generative devices, called picture grammars, where the elements of the terminal alphabet are graphic (usually bidimensional) entities, whereas concatenation of these elements consists of geometric operations. So far, many types of picture grammars are used in many different fields of research, especially in Pattern Recognition related to Physics, Biology, Genetics, Architecture, Visual Arts etc. Curiously, no application of picture grammars in Chemistry appears to be known so far in the literature, despite of the fact that such an application seems to be very natural. Even usual generative grammars were used very seldom in Chemistry ; as we have mentioned, this was done only in the field of information processing; no theoretical problem concerning chemical structures was approached from a generative

point of view. Thus, it is necessary to explain the relation between the generative approach and the already known combinatorial approach which is usually done by means of Graph Theory.

The generative approach is concerned with the process of formation of chemical structures of a definite type. We are looking for a kind of (logical) machine which is able to produce exactly the structures we are considering. The way in which this machine works is the most essential information we can obtain with respect to the considered structures. Thus, we can say that the generative approach is an extrapolation of the combinatorial one. Some combinatorial tendencies are transformed into recursive (self embedding) rules, by means of which our machine has an infinite generative capacity, in contrast with the combinatorial approach, which is usually dealing with finite graphs. Among the rich literature devoted to picture grammars, two types of such grammars were used as suitable to approach acyclic isoprenoid structures : web grammars [4-6] (they are defined in the next paragraph) and array grammars. The terminal alphabet will be the set of isoprenoid units, whereas head-to-tail isoprenoid acyclic structures are obtained by means of head-to-tail linkings of isoprenoid units. The strings which will be generated in this way correspond to all head-to-tail isoprenoid acyclic structures. The set of these structures is infinite, but we are able to read them in a finite device, which is just the picture grammar generating them. This grammar is the *deep structure* of the considered type of chemical formulas, in contrast with their *surface structure*, which consists of the syntactic structure of formulas generated by the grammar.

One of our reasons to prefer web grammars follows from the possibility they offer to analyse the given structures by using reduction rules (obtained by inversion of rewriting rules of the grammar). This corresponds to testing whether a given structure is or is not of a given type. In this way, the same grammar is used to solve both the problem of generation and of checking.

Picture grammars, like classical generative grammars, are particular forms of algorithms. Thus, they are algorithmic descriptions which allow to transfer the problems to a computer. Picture grammars allow to build a compiler and to achieve by means of a computer both the construction and the verification of structural formulas. Thus, the problem of isomeric structures (to each molecular formula correspond several structural formulas) can be easily solved.

Array grammars will show their advantage when dealing with generation of isoprenoid acyclic chemical structures with an odd number of isoprenoid units, with head-to-tail linkage, which admit a symmetry axis.

Finally, we remark that the generative approach allows to introduce into chemistry the distinction between chemical competence and chemical performance. When an infinite class of chemical structures is generated by a grammar, we can better understand the common denominator of these structures, which is given by the structure of the grammar. The chemical competence is just this infinite potentiality defined by the grammar. Only a finite part of this potentiality is actualized in each case; it defines the chemical performance. The problem of generation of acyclic isoprenoid structures with regular (i.e.

head-to-tail) linking and those with standard (i.e. head-to--tail, head-to-head, tail-to-head, tail-to-tail) linking becomes very interesting when new theories are appealed to, namely those which endorse picture languages. For this problem we found a solution using web languages.

2. Web grammars. Introductive notions

In 1969 Pfaltz and Rosenfeld introduced the notion of *web grammar*, whose language is a set of labelled graphs ("webs") and whose productions replace subwebs by other subwebs.

Let V be a finite nonempty set, whose elements will be called *labels*. A *web on V* is a triple $\omega = (N_\omega, E_\omega, f_\omega)$, where : N_ω is a finite, nonempty set, its elements are called the *vertices* of ω ; E_ω is a set of unordered pairs of distinct elements of N_ω, these pairs are called the *arcs* (or *edges*) of ω ; f_ω is a function from N_ω into V. The pair $(N_\omega, E_\omega) = G_\omega$ is a graph, called the *underlying graph* of ω. The distinctness of the terms of the pairs in E_ω implies that the G_ω is loop-free. The vertices m and n are called *neighbours*, if (m,n) is in E_ω. We call m and n *connected* if there exists a *path* $m=m_1,\ldots,m_k=n$, $k \geq 1$, such that $m_i \in N_\omega$, $1 \leq i \leq k$, and (m_i, m_{i+1}) E, $1 \leq i < k$. If any two nodes of ω are connected, ω itself is called connected. Let α, ω be webs on V. We say that α is a *subweb* of ω if : N_α is a subset of N_ω; E_α is the restriction of E_ω to N_α; f_α is the restriction of f_ω to N_α. The complement of α in ω is the subweb $\omega-\alpha$ defined by restricting E and f to the subset $N_\omega - N_\alpha$ of N_ω.

A *web grammar* is a 4-tuple $G = (V, V_T, S, P)$, where V is a finite, nonempty set, called the *vocabulary*; V_T is a non-

empty subset of V, called the *terminal vocabulary* ; $S \in V - V_T$ is called the *initial symbol* ; P is a finite, nonempty set of triplets $\pi = (\alpha,\beta,\varphi)$, called *productions*, where α and β are connected webs on V, φ is a function from $N_\beta \times N_\alpha$ into 2^V (the set of subsets of V). The significance of the function φ will be discussed below.

We say that ω' is *directly derivable* from ω in G if ω, ω' are webs on V, and for some (α,β,φ) in P : (i) α is a subweb of ω ; (ii) for each $m \in N_\alpha$, all labels of neighbours of m in $\omega-\alpha$ are in $\varphi(n,m)$ for some $n \in N_\beta$; (iii) for all m_1, m_2 in α and all $n \in N_\beta$, any neighbour of m_1 in $\omega-\alpha$ that has a label in $\varphi(n,m_1) \cap \varphi(n,m_2)$ is also a neighbour of m_2; (iv) ω' is obtained from ω by replacing α by β, and joining each $n \in N_\beta$ to all the neighbours (in $\omega'-\beta = \omega-\alpha$) of each $m \in N_\alpha$ whose labels are in $\varphi(n,m)$. Formally : $N_{\omega'} = (N_\omega - N_\alpha) \cup N_\beta = N_{\omega-\alpha} \cup N_\beta$;
$E_{\omega'} = E_{\omega-\alpha} \cup E_\beta \cup \{(n,p), n \in N_\beta,\ p \in N_{\omega-\alpha};\ (p,m) \in E_\omega$ and $f_\omega(p) \in \varphi(n,m)$ for some $m \in N_\alpha$ } ; $f_{\omega'} = f_{\omega-\alpha} \cup f_\beta$, $f_{\omega-\alpha} = f_\omega | N_{\omega-\alpha}$. Condition (ii) above provides that every neighbour of (some vertices of) α in $\omega-\alpha$ will be joined to (some nonempty set of vertices of) β. Thus no edges are "lost" or end up "dangling". Condition (iii) eliminates labelling conflicts which arise between edges which coincide under the replacement of α by β. It also guarantees that all productions and, hence, derivations are "reversible". The function φ in a production specifies the *embedding* associated with the production : i.e., it specifies how to join the vertices of β to the neighbours of each vertex of α. This too is done in terms of the neighbours' labels. Each $n \in N_\beta$ in joined to those neighbours (in $\omega-\alpha$) of each $m \in N_\alpha$

whose labels lie in a certain set $\varphi(n,m)$. Condition (ii) thus insures that, for each $m \in N_\alpha$, *some* $n \in N_\beta$ will in fact be joined to every neighbour of m in $\omega-\alpha$, since each such neighbour has a label in at least one of the sets $\varphi(n,m)$.

We say that ω' is *derivable* from ω in the web grammar G if there exists $\omega = \omega_1, \ldots, \omega_k = \omega'$, $k \geq 1$, such that ω_{i+1} is directly derivable from ω_i, $i \leq 1 < k$. By the *language* L(G) of G is meant the set of webs on V_T ("terminal webs") that are derivable in G from the "initial web" consisting of a single vertex labelled S. Any web derivable in G from the initial web is called a *sentential form* of G.

Proposition 1. Any sentential form of a web grammar is a connected web.

It will be noted that by our conventions, webs always have nonempty sets of vertices, so that the "null web" ε never occurs in any web language. If desired, it can be explicitly adjoined to a language. Alternatively, its presence in a language can be made possible by allowing productions in which $\beta = \varepsilon$. To ensure that such productions can never disconnect derived webs, one can simply require that φ is empty (i.e., $\varphi(n,m) = \emptyset$ for all m,n) in any such production, so that by condition (ii), the production can be applied only if α has no neighbours.

In string grammars, L(G) can equivalently be defined as the set of terminal webs which can be "parsed" (that is : from which the initial web can be derived) by applying productions of G in reverse (β is replaced by α, rather than α by β). In the web case, a production $\pi = (\alpha,\beta,\varphi)$ is applicable only if conditions (ii) and (iii) hold, and its application makes use

of the embedding function φ ; thus to apply π in reverse, we must specify how the reverse embedding function is defined, and we must be able to guarantee that conditions (ii) and (iii)hold for the reversal of π. Let φ_R be the function from $N_\alpha \times N_\beta$ into 2^V defined by $\varphi_R(m,n) = \varphi(n,m)$ for all $m \in N_\alpha$, $n \in N_\beta$, and let $\pi_R = (\beta,\alpha,\varphi_R)$; then we have :

Proposition 2. Let ω' be directly derived from ω by applying π to a particular instance of α as a subweb of ω. Then π_R is applicable to the resulting instance of β in ω' and the result of its application is ω.

We say that a terminal web can be ***parsed*** (analysed) in G if the initial web can be derived from it by applying productions of $P_R = \{ \pi_R \mid \pi \in P \}$. By Proposition 2 and induction, if ω is a terminal web of G, it can be parsed in G by simply reversing its derivation. The converse follows similarly, since clearly $(\pi_R)_R = \pi$ for all π. We thus have :

Proposition 3. The set of terminal webs that can be parsed in G is exactly L(G).

In practice, the function φ specifies how to replace α by β in terms of the vertex labels of α and β. Thus, φ is a function from $f_\beta(N_\beta) \times f_\alpha(N_\alpha)$ into 2^V. "$\varphi(M,N) = \{Q\}$ " would mean : "Join vertices labelled M from β to the neighbours labelled Q (in $\omega-\alpha$) of the vertices labelled N from α".

3. The generation of regular acyclic isoprenoid structures using web grammars

Any grammar, particularly a web grammar, can be regarded as a mechanism which, using a finite number of rules, can generate an infinite number of strings which form a language. Taking into consideration the graphic aspect of these kinds of

chemical structures we could construct a web grammar generating a language which is the set of all acyclic regular isoprenoid structures, i.e. with regular (head-to-tail superscript R) linking [7] :

$$G_W^R = (V, V_T, S, P), \quad \text{where} \quad V = V_N \cup V_T$$

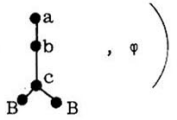
$$V_N = \{ S, B, D \}, \qquad V_T = \{ a, b, c, d, e \}$$

$$P = \{ \pi_i, \ 1 \le i \le 4 \ / \ \pi_i = (\alpha, \beta, \varphi), \alpha, \beta, \ \text{webs on } V,$$

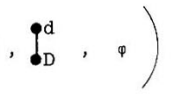
$$\varphi : f_\beta(N_\beta) \times f_\alpha(N_\alpha) \rightarrow 2^V \}$$

(Elements of V_N are of a generic, categorial nature, whereas elements of V_T are individuals. This is the reason why the former ones are represented by capitals, like sets in Set Theory, whereas the latter ones are represented by small letters of the latin alphabet, like elements of a set).

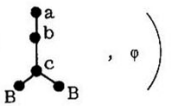
The productions π_i, $1 \le i \le 4$ used by G_W^S are the following :

$\pi_1 =$ 

, where $\varphi(n,S) = \emptyset, \ \forall n \in f_\beta(N_\beta)$

$\pi_2 =$ 

, where $\begin{cases} \varphi(d,B) = \{ c \} \\ \varphi(D,B) = \emptyset \end{cases}$

$\pi_3 =$ 

, where $\begin{cases} \varphi(a,D) = \{d\} \\ \varphi(n,D) = \emptyset, \forall n \neq a, n \in f_\beta(N_\beta) \end{cases}$

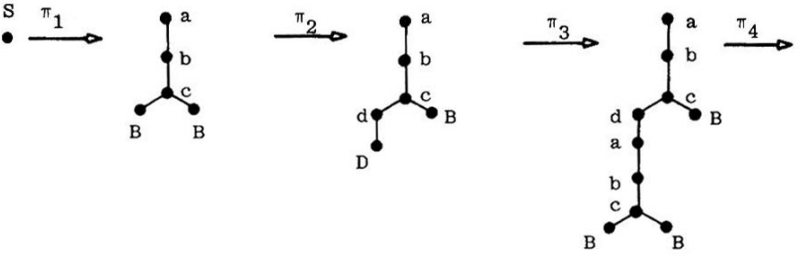
$\pi_4 =$ 

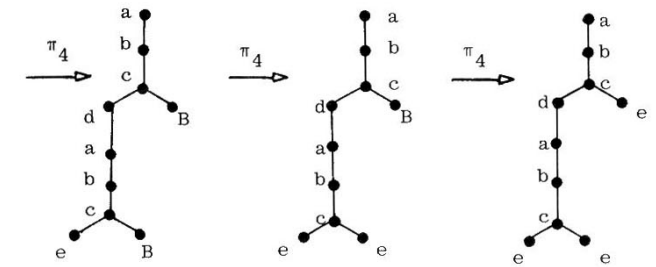
, where $\varphi(e,B) = \{c\}$

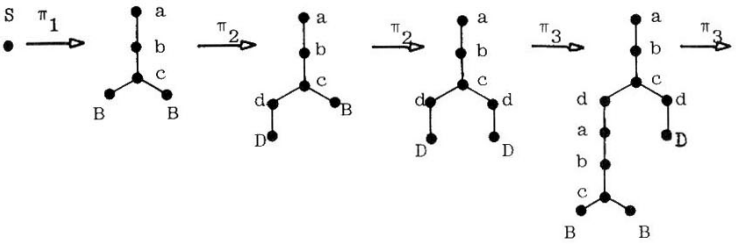
In production π_1, the existence of two vertices labelled B for the web β shows the possibility of linking a new isoprene unit either to the left or to the right of the existing isoprenoid structure. Production π_2 represents the linking edge which always appears between two isoprene units. Production π_3 takes into account that the link which is formed between two isoprene units is a regular one, and that we intend to obtain acyclic structures. The generation process of regular acyclic isoprenoid structures takes place from top to bottom and can continue by using the productions π_2 and π_3. Production π_4 represents the end of the concatenation process on the branch on which it is used. We can observe that in any case a rule of type π_4 is the last production which is applied in generating structures of the required type.

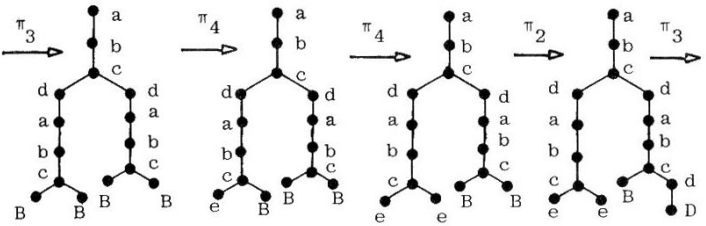
When we intend to generate structures with p isoprene units ($p \geq 1$), we apply (p-1) times each rule π_2 and π_3, and (p+1) times rule π_4.

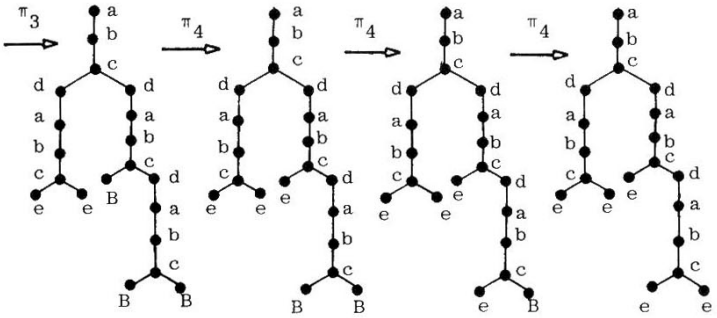
Example of generating an acyclic regular isoprenoid structure




π4
π4
π4

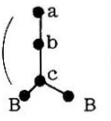

S
π1
π2
π2
π3
π3

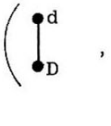

π3
π4
π4
π2
π3

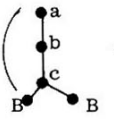

π3
π4
π4
π4

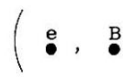
In the process of analysis of an acyclic regular isoprenoid structure one uses the reduction rules obtained by inverting the rewriting rules (productions).

If by application of the reduction rules to a given structure we obtain the initial web consisting of a single vertex labelled S, then the analysis ends successfully, and this means that the given structure is of the required type. The reduction rules associated to the grammar G_W^R are the following :

$\pi_1^R = \left(\right.$  $, S, \varphi_R \left. \right)$, where $\varphi_R(S,n) = \varphi(n,S) = \emptyset$ $\forall n \in f_\beta(N_\beta)$

$\pi_2^R = \left(\right.$  $, B, \varphi_R \left. \right)$, where $\begin{cases} \varphi_R(B,d) = \varphi(d,B) = \{c\} \\ \varphi_R(B,d) = \varphi(D,B) = \emptyset \end{cases}$

$\pi_3^R = \left(\right.$  $, D, \varphi_R \left. \right)$, where $\begin{cases} \varphi_R(D,a) = \varphi(a,D) = \{d\} \\ \varphi_R(D,n) = \varphi(n,D) = \emptyset \\ \quad \forall n \neq a,\ n \in f_\beta(N_\beta) \end{cases}$

$\pi_4^R = \left(\right.$  $, B, \varphi \left. \right)$, where $\varphi_R(B,e) = \varphi(e,B) = \{c\}$

The language generated by G_W^R consists in the set of all acyclic regular isoprenoid structures along whose paths one may proceed following the generating direction, namely from top to bottom. The following algorithm gives the labelling with elements from the terminal vocabulary V_T.

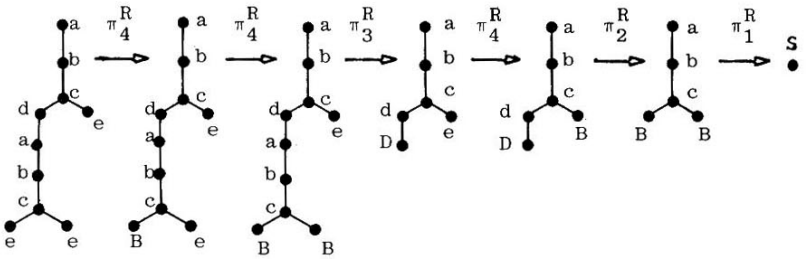
Algorithm of labelling

1. We label "e" all the vertices of degree 1 of the graph except the first vertex of this type (which will be labelled "a") we meet when proceeding(going) along the graph. We will take as a starting point in going along the graph the vertex of degree 1 which is situated at a distance equal with 2 to a vertex of degree 3. If there is no such vertex, then the structure is not of required type.
2. We label "c" all the vertices of degree 3.
3. All the unlabelled direct descendants (taking into account the direction we have chosen) of the vertices labelled "c" will be labelled "d".
4. All the unlabelled direct descendants of the vertices labelled "d" will be labelled "a".
5. All other unlabelled vertices will be marked "b".

This algorithm is the basis of the analyses of a given structure by means of the constructed web grammar G_W^R. Thus, the given structure will be labelled or relabelled with elements from V_T according to the algorithm, after which the analysis starts. If the analysis ends successfully, then the number of the isoprene units that compose the structure is given by the number of applications of the reduction rule π_3^R plus one.

Example of analysis of a given structure

We consider that the structure given for analysis has been labelled with elements from V_T in agreement with the given algorithm.



The analysis ends successfully, therefore the given structure is a regular acyclic isoprenoid one formed of two isoprene units. The sequence π_4^R, π_4^R, π_3^R, π_4^R, π_2^R, π_1^R represents the analysis made.

4. The generation of standard acyclic isoprenoid structures using web grammars

Standard acyclic regular isoprenoid structures (superscript S) have head-to-head, head-to-tail, tail-to-head, or tail-to-tail linking [8] :

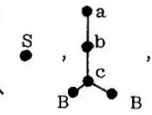
$$G_W^S = (V, V_T, S, P), \quad \text{where} \quad V = V_N \cup V_T$$

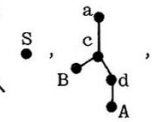
$$V_N = \{S, A, B, D\}, \quad \text{where} \quad V_T = \{ a, b, c, d, e \}$$

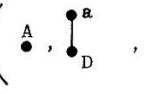
$$P = \{\pi_i, 1 \le i \le 10 \ / \pi_i = (\alpha,\beta,\varphi),\alpha,\beta \text{ webs on } V,$$

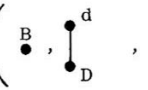
$$\varphi : f_\beta(N_\beta) \times f_\alpha(N_\alpha) \to 2^V \}$$

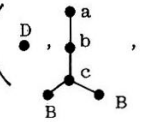
The productions π_i, $1 \le i \le 10$ used by G_W^S are the following :

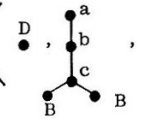
$\pi_1 =$  , where $\varphi(n,S) = \emptyset, \forall n \in f_\beta(n_\beta)$

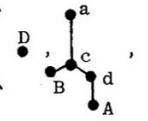
$\pi_2 =$  , where $\varphi(n,S) = \emptyset, \forall n \in f_\beta(N_\beta)$

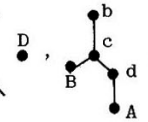
$\pi_3 =$  , where $\begin{cases} \varphi(a,A) = \{d\} \\ \varphi(D,A) = \emptyset \end{cases}$

$\pi_4 =$  , where $\begin{cases} \varphi(d,B) = \{c\} \\ \varphi(D,B) = \emptyset \end{cases}$

$\pi_5 =$  , where $\begin{cases} \varphi(a,D) = \{d\} \\ \varphi(n,D) = \emptyset, \forall n \neq a, \\ \qquad\qquad n \in f_\beta(N_\beta) \end{cases}$

$\pi_6 =$  , where $\begin{cases} \varphi(a,D) = \{a\} \\ \varphi(n,D) = \emptyset, \forall n \neq a, \\ \qquad\qquad n \in f_\beta(N_\beta) \end{cases}$

$\pi_7 =$  , where $\begin{cases} \varphi(a,D) = \{d\} \\ \varphi(n,D) = \emptyset, \forall n \neq a, \\ \qquad\qquad n \in f_\beta(N_\beta) \end{cases}$

$\pi_8 =$  , where $\begin{cases} \varphi(b,D) = \{a\} \\ \varphi(n,D) = \emptyset, \forall n \neq a, \\ \qquad\qquad n \in f_\beta(N_\beta) \end{cases}$

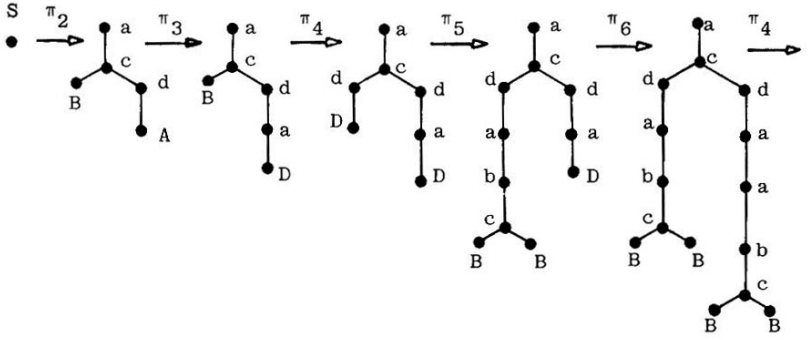
$\pi_9 = \left(\overset{A}{\bullet}, \overset{e}{\bullet}, \varphi \right)$, where $\varphi(e,A) = \{d\}$

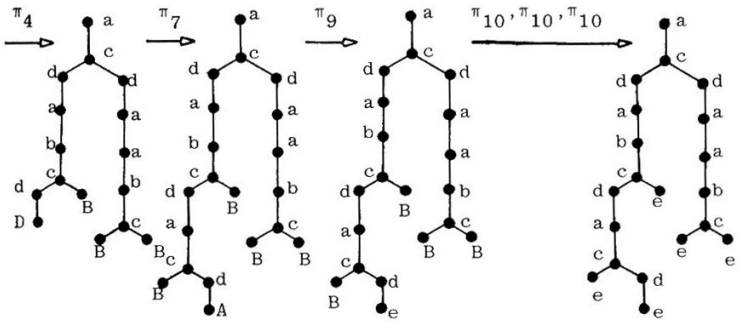
$\pi_{10} = \left(\overset{B}{\bullet}, \overset{e}{\bullet}, \varphi \right)$, where $\varphi(e,B) = \{c\}$

The observations related to the significance of the productions in the web grammar G_W^R are valid also in the case of the web grammar G_W^S if we take into account the following correspondences between the elements of the two grammars :

G_W^R	G_W^S
B	A, B
π_1	π_1, π_2
π_2	π_3, π_4
π_3	π_5, π_6, π_7, π_8
π_4	π_9, π_{10}

Exemple of generating an acyclic isoprenoid structure with standard linking





The reduction rules using in analyses associated to the grammar G_W^S are the following :

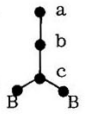
$$\pi_1^R = \left(\begin{array}{c} a \\ b \\ c \\ B \quad B \end{array} \;,\; \overset{S}{\bullet} \;,\; \varphi_R \right) \;, \text{ where } \varphi_R(S,n) = \varphi(n,S) = \emptyset, \; \forall n \in f_\beta(N_\beta)$$

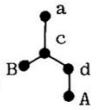
$$\pi_2^R = \left(\begin{array}{c} a \\ c \\ B \quad d \\ \quad\; A \end{array} \;,\; \overset{S}{\bullet} \;,\; \varphi_R \right) \;, \text{ where } \varphi_R(S,n) = \varphi_R(n,S) = \emptyset, \; \forall n \in f_\beta(N_\beta)$$

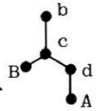
$$\pi_3^R = \left(\begin{array}{c} a \\ D \end{array} \;,\; \overset{A}{\bullet} \;,\; \varphi_R \right) \;, \text{ where } \begin{cases} \varphi_R(A,a) = \varphi(a,A) = \{d\} \\ \varphi_R(A,D) = \varphi(D,A) = \emptyset \end{cases}$$

$$\pi_4^R = \left(\begin{array}{c} d \\ D \end{array} \;,\; \overset{B}{\bullet} \;,\; \varphi_R \right) \;, \text{ where } \begin{cases} \varphi_R(B,d) = \varphi(d,B) = \{c\} \\ \varphi_R(B,D) = \varphi(D,B) = \emptyset \end{cases}$$

$$\pi_5^R = \left(\begin{array}{c} a \\ b \\ c \\ B \quad B \end{array} \;,\; \overset{D}{\bullet} \;,\; \varphi_R \right) \;, \text{ where } \begin{cases} \varphi_R(D,a) = \varphi(a,D) = \{d\} \\ \varphi_R(D,n) = \varphi(n,D) = \emptyset \\ \quad \forall n \neq a, \; n \in f_\beta(N_\beta) \end{cases}$$

$\pi_6^R = \left(\right.$  $, \overset{D}{\bullet}, \varphi_R \left. \right)$, where $\begin{cases} \varphi_R(D,a) = \varphi(a,D) = \{a\} \\ \varphi_R(D,n) = \varphi(n,D) = \emptyset \\ \forall n \neq a,\ n \in f_\beta(N_\beta) \end{cases}$

$\pi_7^R = \left(\right.$  $, \overset{D}{\bullet}, \varphi_R \left. \right)$, where $\begin{cases} \varphi_R(D,a) = \varphi(a,D) = \{d\} \\ \varphi_R(D,n) = \varphi(n,D) = \emptyset \\ \forall n \neq a,\ n \in f_\beta(N_\beta) \end{cases}$

$\pi_8^R = \left(\right.$  $, \overset{D}{\bullet}, \varphi_R \left. \right)$, where $\begin{cases} \varphi_R(D,b) = \varphi(b,D) = \{a\} \\ \varphi_R(D,n) = \varphi(n,D) = \emptyset \\ \forall n \neq b,\ n \in f_\beta(N_\beta) \end{cases}$

$\pi_9^R = \left(\overset{e}{\bullet}, \overset{A}{\bullet}, \varphi_R \right)$, where $\varphi_R(A,e) = \varphi(e,A) = \{d\}$

$\pi_{10}^R = \left(\overset{e}{\bullet}, \overset{B}{\bullet}, \varphi_R \right)$, where $\varphi_R(B,e) = \varphi(e,B) = \{c\}$

On going along a standard acyclic isoprenoid structure which forms the language of the web grammar G_W^S the direction of its generation is followed, namely from top to bottom. The labelling of vertices with elements from V_T of a given structure which is to be analysed by using the reduction rules, is indicated by the following algorithm :

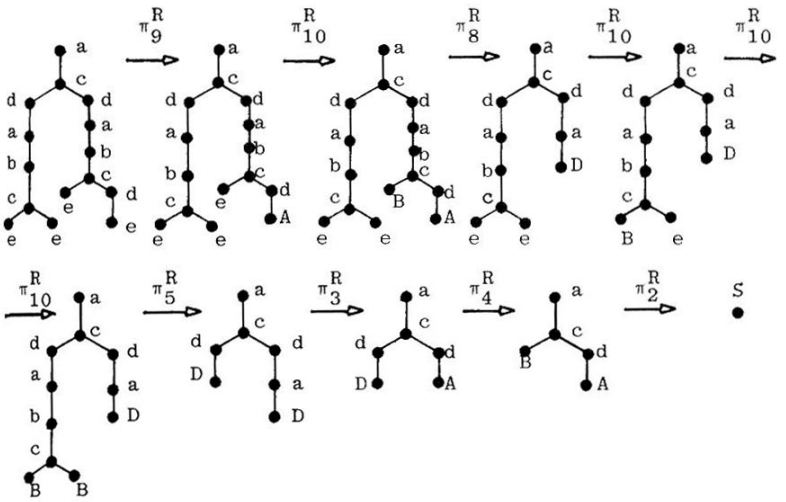
Algorithm of labelling

1. We label "e" all vertices of degree 1 in the graph with the exception of the first vertex of this type met when going along the structure, which will be labelled "a".
2. We label "c" all vertices of degree 3.

3. All the unlabelled direct descendants (taking into account the direction we have chosen) of the vertices marked "c" will be labelled "d".
4. All the unlabelled direct descendants of the vertices labelled "d" will be marked "a".
5. If the distance between a vertex marked "a" and the first vertex (met when going along) labelled "c" is longer than 2, then the direct descendant of the vertex marked "a" will be marked "a" too. This step is applied once in each case.
6. All the other unlabelled vertices will be marked "b".

Example of analysis of a given structure

We suppose that the given structure for analysis has been labelled with elements from V_T in agreement with the given algorithm.



The analysis ended successfully, therefore the given structure is an acyclic isoprenoid structure with standard linking, formed from three isoprene units. The sequence π_9^R, π_{10}^R, π_8^R, π_{10}^R, π_{10}^R, π_5^R, π_3^R, π_4^R, π_2^R represents the analysis made.

5. Discussions and conclusions

It is easy to observe that the language generated by G_W^S includes the language generated by G_W^R .

Pfaltz and Rosenfeld, continuing the study of the web grammars, reached an outstanding result. Thus, the relation between the web grammars and the web acceptors is formulated in the following theorem :

For each web grammar G there is an acceptor M, so that the set of terminal webs on V_T accepted by M, marked T(M), coincides with L(G), the language generated by G.

Thus, the problem of generating acyclic isoprenoid structures with regular or standard linking can be continued in the sense of constructing a compiler and of obtaining by means of the computer a constructive process of these structures. The web grammars G_W^R and G_W^S are the basis of the construction of context free grammars for the linear codification of the structures under examination, a fundamental problem in the representation of the structures.

The generation of a certain subset of acyclic regular isoprenoid structures (mentioned in the introduction) by means of array grammars will be discussed in a further paper.

REFERENCES

1. E.I.Burkart and W.E.Underwood, "A grammar for the line formulas of organic compounds", University of Maryland, Computer Science Center, Report TR-530, April 1977.
2. G.S.Mancill and W.E.Underwood, "A grammar for the nomenclature of acyclic compounds", University of Maryland, Computer Science Center, Report TR-529, April 1977.
3. N.Chomsky, "Syntactic Structures", Monton, The Hague, 1957.
4. A.Rosenfeld and D.L.Milgram, "Web automata and web grammars", University of Maryland, Computer Science Center, Report TR 181, March 1972.
5. D.L.Milgram, "Web automata", University of Maryland, Computer Science Center, Report TR 271, October 1973.
6. D.L.Milgram, "Web automata", University of Maryland, Computer Science Center, Report TR 182, April 1972.
7. A.T.Balaban, M.Barasch and S.Marcus, Math.Chem. 5, 239 (1979).
8. A.T.Balaban, M.Barasch and S.Marcus, Math.Chem. (in press).