Algebraic languages: a bridge between combinatorics and computer science

M. DELEST

ABSTRACT. We describe the classical Schützenberger methodology and two extensions: grammars with operators and q-grammars.

1 - Introduction

In the 1960's, motivated by the interest in writing good compilers, theoretical work was initiated on syntactic formal aspects of languages. In 1964 [30], Randell and Russel did the first complete study of a compiler (this was ALGOL 60), dealing with syntactic structures which continue to be of interest. Recognizing a computer language requires the use of language generators, called grammars. The translation of a language into one understandable by a machine is carried out by compilers, which are also built using grammars and associated rules of translation. A prevalent method is the so-called semantic attribute method due to Irons [1], [26], [27]. An automation of this process for particular grammars has been done with the very well-known programs LEX and YACC which are available free by FTP on an INTERNET site. These programs are heavily used by people interested in compilers or friendly interfaces.

In addition to having close links to computer science, languages appear to provide a powerful tool for encoding combinatorial objects. M.P. Schützenberger (see [31], [32]) has shown the connection between enumeration problems and the classification of languages. In this context, algebraic languages arise as a link between the combinatorial and analytic aspects of a problem. Let Ω be a class of combinatorial objects enumerated by a sequence of integers a_n, where n is the value of a parameter p on these objects. The Schützenberger methodology consists of three steps. First construct a bijection between Ω and the words of an algebraic language in such a way that for every object ω in Ω, the parameter p on ω becomes the number of letters in the word encoding ω. At the next step, the notion of grammar plays an essential role, leading to a noncommutative system of equations. Finally, using what is called the commutative image of the system, one deduces an equation for which the generating function of the sequence a_n is a solution. Two extensions of this method are known. The first one is due to Cori and Richard [9], who introduce operators in order to construct more intricate languages starting from a given language. Using this extension, Tutte's formulas for planar maps are proved in [7]. The second extension, due to Delest and Fédou [14], involves the use of a certain

---

1991 Mathematics Subject Classification. Primary 05A15, 05A30, 05B50.
Supported by the "PRC de Mathématiques et Informatique", the Mathematics Department of the University of Melbourne and the EC grant CHRX-CT93-0400.

translation operation on the languages. This technique comes directly from the attribute method [26], [27], and consists essentially of getting non-algebraic equations (q-equations) from an algebraic encoding of the objects.

In this paper, after introducing some definitions and notation, we describe the notion of a grammar as it originally appeared in computer science. We show then the relation between languages and formal power series, giving special attention to ambiguity problems. Then we give an outline of the Schützenberger methodology, the grammars with operators and the q-grammars. Some of these notions are implemented in the software package CalICo[1] [16], [12], which we are developing in Bordeaux.

2 - Definitions and notation

This section summarizes briefly the notions needed for understanding this paper. A more complete background can be acquired from Berstel [3], Ginsburg [23], Aho and Ullmann [1].

Let X be a nonempty set called alphabet. The elements of X are called letters. A word is a finite sequence of letters from X. The empty word is usually denoted by ε. Let u and v be two words on X, $u=u_1...u_p$ and $v=v_1....v_q$. We define the concatenation of two words to be $uv=u_1...u_pv_1....v_q$. We denote by X^* the free monoid generated by X, that is, the set of all words on X endowed with the operation of concatenation. The number of occurences of the letter x in the word u is denoted by $|u|_x$. The number of letters of w is called length of w and is denoted by $|w|$. A language is a subset of X^*. To every language L, one can associate a noncommutative formal power series

$$L = \sum_{w \in \mathbf{L}} w,$$

which is an element of the algebra $\mathbb{Z}\langle\langle X\rangle\rangle$ of noncommutative formal power series with variables in X and coefficients in $\mathbb{Z}$.

3 - Algebraic grammars

In this section we define the notion of a grammar and describe two examples which are basic in both combinatorics and computer science.

DEFINITION 1. *An algebraic grammar is a 4-uple* $G=\langle N,X,P,s\rangle$ *such that* N *and* X *are two disjoint alphabets called, respectively, the non-terminal and the terminal alphabet,* s *is an element of* N *called axiom, and* P *is a set of pairs* (α, β) *with* $\alpha\in N$ *and* $\beta\in (N \cup X)^*$ *called production rules and denoted by* $\alpha\rightarrow \beta$.

Let α be in N and u in $(N \cup X)^*$, $u=u_1\alpha u_2$. A derivation in G is a rewriting of u as $v=u_1\beta u_2$ with (α, β) in P. This will be denoted by $u\rightarrow v$. We say that a word w is deriving from a non-terminal symbol α in G if there exists a sequence of derivations which rewrites α as w. This will be denoted by $\alpha \overset{*}{\rightarrow} w$. The set L(G) of words

[1] CalICo is a software package under development at LaBRI, Université Bordeaux I. During 1994, the participants in this project were Y. Chiricota, M. Delest, J.M. Fédou, V. Gaudin, G. Mélançon, N. Rouillon.

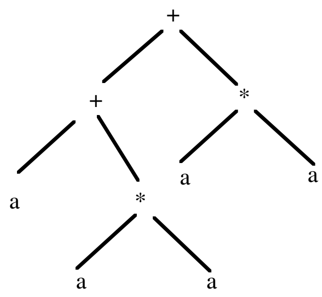


Figure 1. Binary tree associated to the expression (a+a*a)+a*a.

generated by s is called the algebraic language generated by G. In general, there may exist several grammars for a given algebraic language.

EXAMPLE 1. The most famous example in computer science is the language of well-formed arithmetic expressions over a one-letter alphabet. These words are generated by the grammar G_1 given by N={E,T,F}, X={a,+,*,(,)}, s=E, and the production rules

$$E\rightarrow E{+}T\ ,\ E\rightarrow T\ ,\ T\rightarrow T{*}F\ ,$$
$$T\rightarrow F\ ,\ F\rightarrow (E),\ F\rightarrow a\ .$$

An example of a word generated by G_1 is (a + a * a) + a * a. The abstract syntax tree, that is, the tree which comes directly from the derivation of the word in the grammar G_1, is a binary tree. This tree allows the implementation of very fast computation of such expressions in compilers (see figure 1). In the sequel, we will work with formal power series, so, in order to avoid the possibility of confusing notation, we will replace the terminal letters +, *, (,) with the letters p, m, x, $\bar{x}$, respectively.

All instructions and phrases "understandable" by a given compiler are described by an algebraic grammar. The PASCAL language, as well as its predecessor ALGOL, were described to the user by means of a grammar.

EXAMPLE 2. The deepest example in combinatorics is the Dyck language, not because of its complexity, but because of the frequency with which it occurs in different settings. It encodes numerous and diverse structures such as trees, paths, polyominos, ... Its words are generated by the grammar G_2 given by

$$N=\{D\},\ X=\{x,\ \bar{x}\},\ s=D,$$

and the production rules

$$D\rightarrow x\,D\,\bar{x}\,D\ ,\ D\rightarrow \varepsilon\ .$$

An example of a word is $x\,x\,\bar{x}\,\ x\,\bar{x}\,\bar{x}\,\ x\,\bar{x}$. Figure 2 shows three different combinatorial structures encoded by this word. In the next section we will describe briefly the three bijections suggested by this figure.

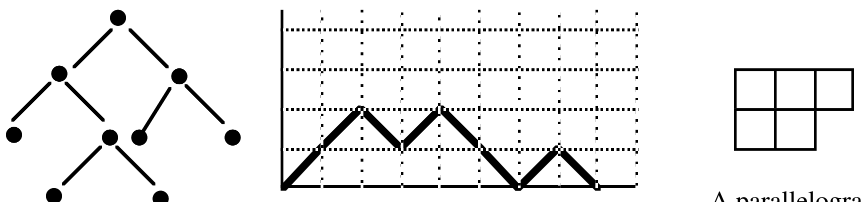


Figure 2. Combinatorial objects encoded by $x\, x\, \bar{x}\, x\, \bar{x}\, \bar{x}\, x\, \bar{x}$.

The two previous examples give rise to unambiguous algebraic grammars, that is, algebraic grammars in which every word is obtained only once from the axiom using the production rules in a left-right derivation that is deriving first the leftmost terminal. In such cases, the formal power series associated to the language verifies equations which follow directly from the production rules. For our example G_1 we obtain the system

$$E = E\mathbf{p}T + T,$$

$$T = T\mathbf{m}F + F,$$

$$F = \mathbf{x}E\bar{\mathbf{x}} + a$$

and for G_2 we obtain the equation $D=x\, D\, \bar{x}\, D + \varepsilon$. In these equations, E, T, F and D are the generating functions of the words rewritten from the corresponding non-terminal letter. In the following, all the grammars are considered to be unambigous.

4 - The Schützenberger methodology

This methodology stems from an idea of M.P. Schützenberger from 1959 (see [31], [32]). In the years that followed, it was first developed by Gross [24]. Subsequent developments are due to:

- Cori and Richard [9], leading to the enumeration of planar maps [7];
- Fliess [9], with applications to solving differential equations;
- Viennot [36], with applications in enumerative combinatorics;
- Flajolet [22], with applications to asymptotic analysis.

This method is now known as the DSV-methodology, following M.P. Schützenberger's wish expressed to Viennot [35]. In 1975, a theoretical computer science workshop was held in France, on the topic of formal power series. Its proceedings, edited by J. Berstel, contains numerous papers on this subject [3].

Let X be an alphabet, $X= \{x_1, ..., x_k\}$. The commutative image of a series produces, from a noncommutative formal power series, a commutative one, called an enumerative series of the language. This is defined by:

$$\chi_0(L) = \sum_{(i_1, i_2, \ldots, i_k)\in N^k} n_{i_1 i_2 \ldots i_k} x_1^{i_1} x_2^{i_2} \ldots x_k^{i_k}$$

where $n_{i_1 i_2 \ldots i_k}$ is the number of words w in L such that $|w|_{x_j}=i_j$ for each j in [1..k]. In this way, we obtain an application from the boolean semi-ring $\mathbb{B}\langle\langle X\rangle\rangle$ to the semi-ring $\mathbb{N}[[X]]$ of commutative formal power series with variables in X. We will often denote by L the series $\chi_0(L)$. The application χ_0 is not a morphism but the following properties hold.

PROPERTY 2. *Let* A, B, C *be three languages on* X. *Suppose that the products and sums are unambiguous. Then*
(i) *if* A=BC *then* $\chi_0(A) = \chi_0(B)\, \chi_0(C)$,
(ii) *if* A=B+C *and* B∩C *is empty then* $\chi_0(A) = \chi_0(B) + \chi_0(C)$,
(iii) *if* A *does not contain the empty word then* $\chi_0(A^*) = (1-\chi_0(A))^{-1}$.

These properties of the application χ_0 are the fundamental elements of the methodology [31] and one of their consequences is the following theorem.

THEOREM 3. *The image of an algebraic language under* χ_0 *is an algebraic series which is one of the components of the solution of the system of equations obtained via* χ_0 *from an unambiguous grammar of the language.*

EXAMPLE 3. We can compute the enumerative series associated to G_2 using the following commutative system:

$$E = \mathbf{p}ET + T\ ,\ T = \mathbf{m}TF + F\ ,\ F = \mathbf{x\bar{x}}\, E + a\ .$$

We have the following equation

$$E = \frac{(1+pE)(x+Ez^2)}{1-mx-mx\bar{x}E},$$

and we obtain the generating function for the number e_n of well-formed arithmetic expressions having n symbols by setting all the variables equal to x:

$$E(x) = \sum_{n\geq 0} e_n x^n = \frac{1-3x^2+\sqrt{1-6x^2+x^4}}{4x^3}.$$

This example is borrowed from the "ΛϒΩ cookbook" [22]. Recall that ΛϒΩ is the software developed by the Flajolet's team at INRIA. It computes automatically asymptotic expressions for the coefficients of a series. More classical is the computation on the Dyck language which leads to the equation

$$D = x\bar{x}D^2 + 1.$$

An elementary calculation shows that

$$D(x,\bar{x}) = \frac{1-\sqrt{1-4x\bar{x}}}{2x\bar{x}},$$

from which we deduce easily that the number of Dyck words of length 2n is the Catalan number

$$C_n = \frac{1}{n+1}\binom{2n}{n}.$$

Numerous results, in several areas, were obtained by this method. Here we will describe only some of these: polyominos [11], [36], and ^{t}RNA structures [33]. A general overview can be found in [34], [37].

REMARK. Some algebraic languages cannot be associated to unambiguous grammars. Flajolet [21] has shown that their generating series are related to transcendental series.

5 - Some classical results

The first step in the Schützenberger methodology, namely the encoding, requires particular insight: one must find a bijection between the objects and an algebraic language. We remark that, frequently, the language which is obtained in the bijection process turns out to be closely related to the Dyck language. This may be due in part to the fact that every algebraic language can be obtained by means of an algebraic transduction [29] applied to the Dyck language and an intersection with a rational language. So, in this section we describe bijections linked directly to Dyck words.

Let us consider the set B of binary trees. If b is in B, then it admits the following recursive description: either b = (r, L(b), R(b)) where r is an internal node called root and L(b) (R(b), resp.) is the left (right, resp.) tree, or b is a single point called leaf. To encode a tree by a Dyck word, traverse the tree in left first depth-first order (or prefix order, that is, visiting first the root, then the left subtree, then the right subtree). During the traversal, write x at each internal node and $\bar{x}$ at each leaf, except the last one. This is the classical bijection between binary trees and Dyck words. One deduces the following well-known result.

THEOREM 4. *The number of binary trees having* n+1 *leaves is the Catalan number* C_n.

We can extend this to the set T of ternary trees. Let t be a ternary tree. Then either t=(r, L(t), M(t), R(t)) where r is an internal node called root and L(b), M(b), R(b) are the left, middle, right subtrees, or b is a single point called a leaf. To encode the tree, we resort to a 3-letter alphabet, { x, y, z }, and proceed as follows:

- every internal node produces a letter x,
- every time the walk is going from a left subtree to a middle subtree, we write a letter y,
- every time the walk is going from a middle subtree to a right subtree, we write a letter z.

This yields an encoding of ternary trees by an algebraic language defined by:

$$N=\{T\},\ X=\{x,y,z\},\ s=T,$$

and the production rules

$$T \rightarrow x\,T\,y\,T\,z\,T,\ T \rightarrow \varepsilon\ .$$

Then we can enumerate, for example, the ternary trees according to the number of leaves, by solving the equation

$$T = xyzT^3 + 1$$

and we obtain

THEOREM 5. *The number of ternary trees having 2n+1 leaves is*

$$t_n = \frac{1}{2n+1}\binom{3n}{n}.$$

Let us now consider the set of paths in $\mathbb{N}\times\mathbb{N}$ which are sequences of points $(s_0, s_1, ..., s_n)$. The pairs (s_i, s_{i+1}) are called elementary steps. They are North-East (South-East, resp.) if $s_i = (x,y)$ and $s_{i+1} = (x+1,y+1)$ ((x+1,y-1), resp.). The height of the step s_i is y. These paths are clearly in bijective correspondence with Dyck words: simply follow the path from s_0 to s_n, encoding each NE step by x, and each SE step by $\bar{x}$. The word obtained in this manner is a Dyck word if and only if $s_0=(0,0)$ and $s_n= (\lfloor n/2 \rfloor, 0)$. The corresponding path is frequently referred to as a Dyck path.

A closely related class of paths is that where the elementary steps are NE, SE, and E (i.e., East: $s_{i+1} = (x+1,y)$), and where $s_0 = (0,0)$ and $s_{n+1}=(k,0)$. Such paths are encoded by the Motzkin language, which is described by the grammar

$$N=\{M\}, X=\{x, \bar{x}, y\}, s=M,$$

with the production rules

$$M\rightarrow x M \bar{x} M ,\ M\rightarrow y M ,\ M\rightarrow \varepsilon .$$

Then the generating function for the number m_n of Motzkin paths having n steps can be deduced from the equation

$$M = x\bar{x}M^2 + yM + 1$$

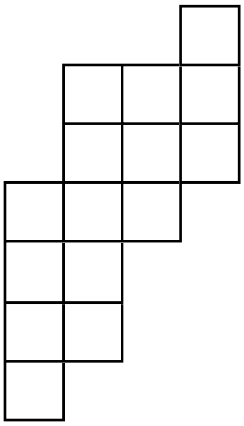
by setting $x = \bar{x} = y = t$. Thus, we obtain

THEOREM 6. *The generating function for the number of Motzkin paths of length n is given by*

$$M(t) = \frac{1 - t - \sqrt{1 - 2t - 3t^2}}{2t^2}.$$

Note that one cannot obtain a particularly nice formula counting Motzkin paths.

Another subject where the Schützenberger methodology gives good results is that of polyomino enumeration. For surveys, we refer the interested reader to [11], [25], [36]. A polyomino can be described as a finite connected union of cells (unit squares) in the plane $\mathbb{N}\times\mathbb{N}$, without cut points. A column (row, resp.) of a polyomino is the intersection of the polyomino with an infinite vertical (horizontal, resp.) strip of cells. A polyomino is column-convex (row-convex, resp.) if every column (row, resp.) is connected. A parallelogram polyomino is both row- and column-convex and for each one of its columns there is



a = (4, 5, 3, 3)

b = (2, 2, 1)

Figure 3. A parralelogram polyomino and its coding.

- no column on its right with a cell lower than its lowest cell,
- no column on its left with a cell higher than its highest cell.

An analysis of these constraints leads to an alternate definition of a parallelogram polyomino, as a pair of integer sequences $(a_1,\ldots, a_n)$ and $(b_1,\ldots, b_{n-1})$, where a_i is the number of cells belonging to the i^{th} column and (b_i+1) is the number of adjacent cells from columns i and i+1. Figure 3 shows an example of a parallelogram polyomino and the two sequences a and b. These two sequences can be viewed as the heights of the peaks (a step North-East followed by a step South-East) and the heights of the troughs (a step South-East followed by a step North-East) in a Dyck path. So encoding a parallelogram polyomino by a Dyck word is straightforward and it is easy to deduce the next result.

THEOREM 7. *The number of parallelogram polyominos whose perimeter equals 2n+2 is the Catalan number* C_n.

This result was already known a long time ago. The bijection from the Schützenberger methodology merely explains combinatorially the link between polyominos and Catalan numbers. The first new result in enumerative combinatorics obtained by this methodology pertains to convex polyominos.

THEOREM 8. *The number* p_{2n} *of convex polyominos having a perimeter 2n+8 is*

$$P_{2n} = (2n+11)4^n - 4(2n+1)\binom{2n}{n}.$$

None of the proofs of this formula, since 1984, has been entirely bijective. That is, each known proof involves a major computation. The first proof, for instance, used three grammars with about 120 production rules! Thus, a nice formula is obtained after some apocalyptic computation. It would be of interest to have a proof in which the solution arises as the difference of two sets of objects, enumerated by the two terms of the formula.

Now, we end this section with another example closely related to Dyck words, for which the Schützenberger methodology produced a new and quite non-trivial result. Consider the column-convex polyominos. Using the previous construction, one can associate to such a polyomino three sequences of integers: $(a_1,\ldots, a_n)$ and $(b_1,\ldots, b_{n-1})$ as defined earlier, and $(c_1,\ldots, c_{n-1})$ where c_i gives the way for gluing the columns i and i+1. More precisely, for $i\geq 1$, begin by attaching column i+1 to the right of column i so that their South-most cells are on the same horizontal strip ; then slide the $(i+1)^{st}$ column $|c_i|$ cells to the North if $c_i\geq 0$, or to the South if $c_i<0$. Using the first two sequences, we construct a Dyck path and then, using the third sequence, we assign colors to the valleys of the Dyck path (a valley is a maximal sequence of SE steps followed by a maximal sequence of NE steps). Each valley is colored according to the following rule: if it contains only one SE step or only one NE step, then $|c_i|$ steps are colored; otherwise, all the steps of the valley are colored, or none of them is colored. The language V which is obtained from this construction is algebraic and it is defined as follows. Let $Y=\{x,y,\bar{x},\bar{y}\}$, let d be the morphism of Y in $X=\{x,\bar{x}\}$ acting by $d(x)=d(y)=x$ and $d(\bar{x})=d(\bar{y})=\bar{x}$. The language V is the set of words w in Y^* such that d(w) is a Dyck word and w can be factored as $f=f_1x\bar{x}f_2\ldots x\bar{x}f_k$ with $k\geq 2$, f_1 in $\{x\}^*$, f_k in $\{\bar{x}\}^*$ and f_i in $\{\bar{x}\}^*\{x\}^*\cup\{\bar{x}\}^*\{\bar{y}\}^*\bar{y}\cup y\{y\}^*\{x\}^*\cup\{\bar{y}\}^*\bar{y}y\{y\}$ for each i in $]1,k[$. An example is shown in figure 4. So, the enumeration of column-convex polyominos involves an intricate algebraic language and the solution is a lengthy formula requiring an entire page (see [10]).

6 - Grammars with operators

In terms of grammars, the main interest in computer science is to succeed in conveying through the syntax a certain amount of semantics, so that the translation of the source code will be an efficient executable program. Cori and Richard [9] used the same idea for the enumeration of objects which cannot be encoded directly by an algebraic language, but such that their generating function is algebraic. They introduced the notion of equations with operators. They showed also that for certain languages, which are not algebraic, the solution for the corresponding generating function is algebraic. This approach was applied to planar maps by Cori [7]. We will describe this approach. First we define the commutative image for an operator.

DEFINITION 9. *If H is an operator from $\mathbb{B}\langle\langle X\rangle\rangle$ into $\mathbb{B}\langle\langle X\rangle\rangle$, then its commutative image H_{χ_0} is the operator from $\mathbb{N}[[X]]$ into $\mathbb{N}[[X]]$ which, if it exists, satisfies $H_{\chi_0}\circ\chi_0=\chi_0\circ H$.*

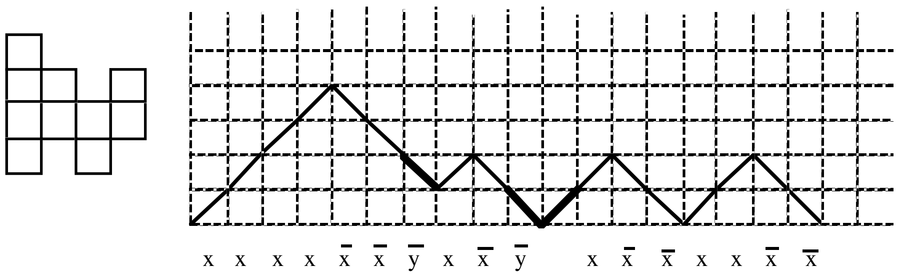


Figure 4. A column-convex polyomino and its encoding.

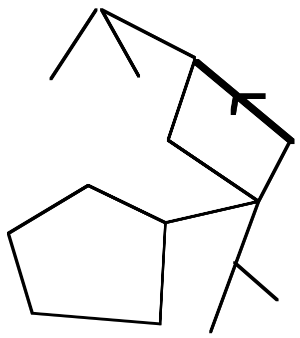


Figure 5. A simple rooted planar map.

EXAMPLE 4. An alphabetical morphism admits a commutative image. Let H_1 be the operator which preserves the empty word and maps each non-empty word to its first letter. H_1 does not have a commutative image.

In [7], Cori defined the following operator Λ on $\mathbb{B}\ll\{x,y\}\gg$ in the following way:

$$\text{if } |f|_x = 0 \text{ then } \Lambda(f)=0, \text{ else } f=y^n x g \text{ and we set } \Lambda(f)= y^{n+1}g.$$

Let $s(x,y)$ be a series in $\mathbb{N}[\![\{x,y\}]\!]$. The commutative image of the operator Λ is defined by

$$\Lambda_{\chi_0}(s(x,y)) = \frac{y}{x}\big(\chi_0(s(x,y)) - \chi_0(s(0,y))\big).$$

It is clear that if a commutative image of the operator exists, then we may extend its action to systems of equations in which the operator appears. The main issues studied for such equations are conditions under which they can be solved and conditions under which they have algebraic solutions. Chottin [6] gave a formal background and then Dulucq [19] introduced the φ-derivations for which algebraic properties can be proved. One of the main subsequent results was the bijective proof by Cori, Dulucq, Viennot [8] that C_nC_{n+1} is the number of shuffles of Dyck words. The three authors deduced the number of alternating Baxter permutations.

We give here two examples using this method. The first one comes from the work of Cori on planar maps. A simple rooted planar map is a planar map with one distinguished edge belonging to the face which is adjacent to every edge (see figure 5 for an example). Let X be the alphabet $\{x, \bar{x}\}$. Cori [7] introduced an operator D on $\mathbb{B}\ll X\gg$, which acts on a word f in X^* as follows:

$$D(f)=\{g\,\bar{x}\,h, \text{ with } f=gh \text{ and } |g|_x = |g|_{\bar{x}}\}.$$

As an example, if $f=x\,x\,\bar{x}\,x\,\bar{x}\,\bar{x}\,x\,x\,\bar{x}\,\bar{x}$, then we have

$$D(f)=\{\bar{x}f,\ f\bar{x},\ x\,x\,\bar{x}\,x\,\bar{x}\,\bar{x}\,\bar{x}\,x\,x\,\bar{x}\,\bar{x}\}.$$

It turns out that the language S encoding the rooted planar maps satisfies the equation

$$S = 1 + y\bar{y}S + xD(S) \ .$$

This equation is of the form L=A+BxD(L) with A=B=$(y\,\bar{y})^*$. Cori proved that this type of equation can be solved and deduced the following result

THEOREM 10. *The number of simple rooted planar maps having n edges is*

$$s_n = \frac{(3n+1)!}{(n+1)!(2n+1)!}.$$

More recently, the grammars with operators were used by Arnold, Delest and Dulucq [2] in the computation of the asymptotic cost of the Naïmi-Trehel algorithm. This algorithm is useful when processes are trying to access a resource under conditions of mutual exclusion, such as a printer. One of the best data structures for representing the hierarchy among the processes is a general tree, in which the process owning the resource is implemented as the root of the tree. In figure 6, we show this operation when the process i is the new owner of the resource.

The mean cost of this algorithm over the set B_n of the trees having n vertices was given by Trehel in the form

$$M_n = \sum_{A \in B_n} \pi_A \, \mathrm{N}^2(A),$$

where π_A is the probability of getting the tree A after an infinite number of transformation, and $N^2(A)$ is the degree of the root. Using two operators, we can compute the probabilities π_A. The two operators are acting on the set D of Dyck words. The first one, denoted $\square$, is defined from D$\times$D into D by

$$\varepsilon \ \square \ w = w \ ,$$
$$(u\ x\ v\ \bar{x}) \ \square \ w = u\ x\ v\ w\ \bar{x} \ ,$$

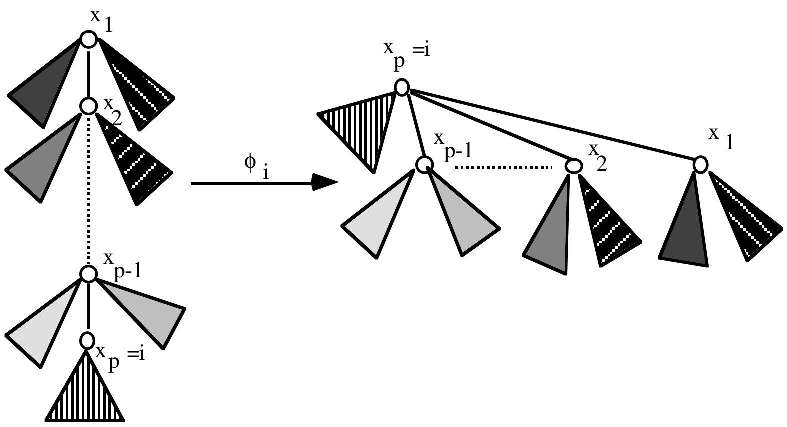


Figure 6. The operation of gripping for the process i.

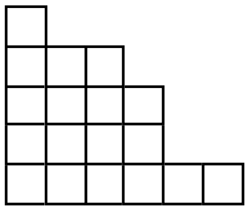


Figure 7. An integer partition represented by a Ferrers diagram.

the second, denoted μ, is defined from D into $\mathbb{N}\ll X \gg$ by

$$\mu(\varepsilon) = \varepsilon,$$
$$\mu(u\,x\,v\,\bar{x}) = u\,x\,v\,\bar{x} + (\mu(u)\backslash\{u\}) \square (xv\bar{x}) + \mu(v)xu\bar{x}.$$

Using these two operators, we can find the following result

THEOREM 11. *The mean cost of the Naïmi-Trehel algorithm for a data structure of size n is*

$$M_n = \mathrm{H}_n,$$

where $\mathrm{H}_n = \sum_{i=1}^{n} 1/i$ *is the harmonic number.*

7 - q-grammars

The study of compilers in computer science shows that the semantic attribute method described by Irons [26], [27] and then by Knuth [28] allows the translation of words from an algebraic language. Most of the resulting translations, however, are not algebraic languages. In the context of enumerative combinatorics, the same set of objects may lead to an algebraic generating function if counted according to a certain parameter, and to a non-algebraic one if counted according to another. For example, the generating function for Ferrers diagrams is algebraic according to the perimeter of the diagram

$$f(x) = \frac{x^2}{1-2x},$$

and not algebraic according to the number of cells

$$f(q) = \prod_{i=1}^{\infty} \frac{1}{1-q^i}.$$

The interest presented by the attribute method lies in the fact that translation is defined locally, on each production rule of the grammar. This is significant for writing efficient compilers since it means that the problem of translation can be decided locally, without consideration given to context, for grammars which frequently have more than one hundred rules. The interested reader may find a general study of this subject in [28]. We will confine ourselves to synthesized attributes. Before giving a formal definition, we describe an example.

EXAMPLE 5. In order to evaluate arithmetic expressions, code for loading and storing the registers of the processor must be generated in the translation. These operations are very expensive in terms of time, so their number must be minimized. So, the minimal number of registers needed for the translation is computed with the following attribute grammar.

$E \to E+T$, $\nu(E):=$if $\nu(E)=\nu(T)$ then $\nu(T)+1$ else $\max(\nu(E),\nu(T))$
$E \to T$, $\nu(E):=\nu(T)$
$T \to T*F$, $\nu(T):=$if $\nu(T)=\nu(F)$ then $\nu(T)+1$ else $\max(\nu(F),\nu(T))$
$T \to F$, $\nu(T):=\nu(F)$
$F \to (E)$, $\nu(F):=\nu(E)$
$F \to a$. $\nu(F):=1$

Figure 8 shows the computation of the attribute ν using the derivation tree from figure 1. Note that the parameter ν coincides with the Strahler number for binary trees [33] and with the parameter "order" in the ^{t}RNA structure. Figure 7 also shows the computation on the abstract syntax tree. This subject was surveyed by Viennot in [37]. We give now a rough definition of attribute grammars, sufficient for our purposes.

DEFINITION 12. *An attribute grammar is an algebraic grammar $G=\langle N,X,P,s\rangle$ having a finite set of attributes $A(Y)$ associated to each non terminal Y of V. Every attribute τ is an application from V in a set D_τ. An attribute τ is said synthesised if for every rule in P having the form*

$$Y \to u_1 Y_1 u_2 Y_2 \ldots\ldots Y_n u_{n+1}$$

such that each u_i is in X^*, $\tau(Y)$ can be computed recursively as

$$\tau(Y) = f(\gamma_1(Y_1), \ldots, \gamma_n(Y_n))$$

where γ_k is in $A(Y_k)$ for each k in $[1,n]$ and f is an application from $D_{\gamma_1} \times .. \times D_{\gamma_n}$ in D_τ. The image of a word w by τ is equal to the value of $\tau(S)$ in the derivation $S \overset{*}{\to} w$ of w in G.

For example the set D_τ can be a monoid X^* or the set of integers $\mathbb{N}$. Based on the idea of translation provided by definition 12, one can hope that, in combinatorics, the problem of finding recurrences with respect to a non-algebraic parameter may be reduced to a local translation problem on certain algebraic configurations. We will limit ourselves here to the simple case when the translation is equivalent to adjoining a letter [14].

DEFINITION 13. *A q-grammar (G,τ) is an attribute grammar where:*
(i) $G = \langle V,X,\mathcal{P},S\rangle$ is an unambiguous algebraic grammar,
(ii) τ is a synthesized attribute with values in $\mathbb{C}[X\cup\{q\}]$,
(iii) For every letter x in X and every word w in $LG)$, $|\tau(w)|_x = |w|_x$.

The translation $\tau(w)$ of a word w is called a q-analogue of the word w and is denoted by $(w;q)$. Let ${}^qG = (G,\tau)$ be a q-grammar.

DEFINITION 14. *The q-analogue $^qL(G)$ of the language $L(G)$ is defined by*

$$^{q}L(G) = \sum_{w \in L(G)} (w;q).$$

DEFINITION 15. *The generating function of the language $^qL(G)$ is defined by*

$$^{q}l = \sum_{w \in L(G)} \chi_1(\tau(w))\, \chi_0(w)$$

where χ_1 is the morphism which suppresses from a word of $(X \cup \{q\})^*$ all the letters belonging to X.

It can be shown (see [17]) that if an attribute is linear (in a sense which we will not explain here), then a system of q-equations can be obtained directly from the q-grammar. The addition of the attribute to the grammar introduces non-algebraic substitutions in the commutative equations. The general case for several attributes has been treated by P. Duchon and the results are similar.

EXAMPLE 6. The language encoding Ferrers diagrams is the language encoding their profile by means of words of the form w=aub written on the alphabet {a,b}. A grammar for this language is

$$G = \langle \{S,L\},\{a,b\},\{S \rightarrow aLb,\ L \rightarrow aL;\ L \rightarrow bL;\ L \rightarrow \varepsilon\},\ S \rangle.$$

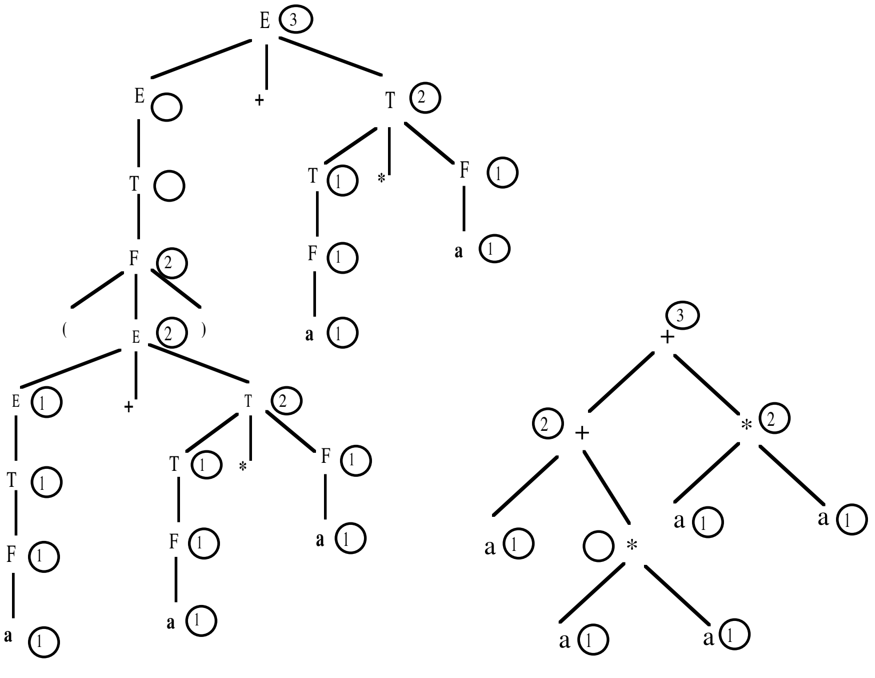


Figure 8. Attribute derivation tree and Strahler number of the abstract syntax tree.

The attribute τ defined below computes the number of cells based on this encoding:

$$S \to a\,L\,b\ ,\ \tau(S) = q^{|\tau(L)|_a + |\tau(L)|_b + 1}\ a\ \tau(L)\ b$$

$$L \to a\,L,\ \tau(L) = q^{|\tau(S)|_b}\ a\ \tau(L),$$

$$L \to b\,L,\ \tau(L) = \ b\ \tau(L)\ ,$$

$$L \to \varepsilon,\ \tau(L) = \ \varepsilon.$$

It is easy to show that the system of q-equations is

$${}^{q}S(a,b) = q\ a\ {}^{q}L(aq,bq)\ b,$$

$${}^{q}L(a,b) = a\ {}^{q}L(a,bq) + b\ {}^{q}L(a,b) + \varepsilon\ ,$$

from which one can deduce the well-known generating function

$${}^{q}l(a,b) = \sum_{n=0}^{\infty} \frac{a^n q^{n+1}}{(1-qb)(1-q^2 b)\ldots(1-q^{n+1}b)}.$$

Numerous q-systems can be obtained by this method, but solving them remains challenging even in cases when they give very nice results (see [13]).

Conclusion

In the case of the classical DSV-methodology the equations to be solved have algebraic solutions, and these can be extracted from the equation(s). The situation is very different for the two extensions which we described, grammars with operators and q-grammars, for which a general solving method is still out of reach. Another extension of the DSV-methodology, to object grammars, is due to Dutour and Fédou [20]. We have not discussed this third extension, which also lacks a general method for solving the resulting equations. M. Bousquet-Melou gives in [5] a solution in a special case.

Acknowledgment

The author thanks Tony Guttmann for fruitful discussions and his warm welcome in Australia and the referee for his patient help in improving style and correctness.

References

[1] A. Aho, J. Ullman, *The theory of parsing, translation and compiling*, Prentice Hall, Englewood Cliffs, NJ, 1973.

[2] A. Arnold, M. Delest, S. Dulucq, Complexité de l'algorithme d'exclusion mutuelle de Naïmi-Trehel, Actes du 37ème Congrès de l'ACFAS, UQUAM, Montréal, Canada, May 1989.

[3] J. Berstel et al., *Séries formelles en variables non commutatives et applications*, Actes de la cinquième Ecole de Primptemps d'Informatique Théorique, Vieux-Boucau les Bains, 1977.

[4] J. Betrema, J.G. Penaud, Animaux et arbres guingois, Theor. Comp. Sci. 117 (1993), 67-89.

[5] M. Bousquet-mélou, A method for the enumeration of various classes of column-convex polygons, Rapport LaBRI No. 578-93, submitted for publication.

[6] L. Chottin, Etude syntaxique de certains langages solutions d'équations avec opérateurs, Theor. Comp. Sci. 5 (1977), 51-84.

[7] R. Cori, *Un code pour les graphes planaires et ses applications*, Astérisque 27 (1975).

[8] R. Cori, S. Dulucq, X. Viennot, Shuffle of parenthesis systems and Baxter permutations, J. of Comb. Th. A, 43 (1986), 1-22.

[9] R. Cori, J. Richard, Enumération des graphes planaires à l'aide des séries formelles en variables non commutatives, Discrete Math., 2 (1972), 115-162.

[10] M. Delest, Generating function for column-convex polyominoes, J. of Comb. Th. A, 48,1 (1988), 12-31.

[11] M. Delest, Polyominoes and animals: some recent results, J. of Math. Chemistry, 8 (1992), 3-18.

[12] M. Delest, J.M. Fédou, G. Melançon, N. Rouillon, Faire des mathématiques avec CalICo, Bulletin des Annales du Québec, to appear.

[13] M. Delest, J.M. Fédou, Enumeration of skew Ferrers diagrams, Discrete Maths, 112 (1993), 65-79.

[14] M. Delest, J.M. Fédou, Attribute grammars are useful for combinatorics, Theor. Comp. Sci. 98 (1992) 65-76.

[15] M. Delest, X. Viennot, Algebraic languages and polyominoes enumeration, Theor. Comp. Sci., 34 (1984), 169-206.

[16] M. Delest, N. Rouillon, CalICo a software for combinatorics, DIMACS Series in Discrete Mathematics and Theoretical Computer Science, 15 (1994), 327-333.

[17] J.P. Dubernard, *q-grammaires et polyominos parallélogrammes*, Thèse de l'Université Bordeaux 1, 1993.

[18] J.P. Dubernard, I. Dutour, Enumération de polyominos convexes dirigés, proceedings SFCA'94, L. Billera, C. Greene, R. Simions, R. Stanley Eds, Mai 1994, 109-118.

[19] S. Dulucq, *Equations avec opérateurs: un outil combinatoire*, Thèse de 3ième cycle, Bordeaux, 1981.

[20] I. Dutour, J.M. Fédou, Grammaires d'objets, preprint, LaBRI 1994.

[21] P. Flajolet, Ambiguity and transcendence, Research Report 397, INRIA, 1985.

[22] P. Flajolet, B. Salvy, P. Zimmerman, Automatic average-case analysis of algorithms, Theor. Comp. Sci. 79, 1 (1991), 37-109.

[23] S. Ginsburg, *The mathematical theory of context free languages*, McGraw-Hill, New-York, 1966.

[24] M. Gross, Applications géométriques des langages formels, I.C.C. Bul., 5 (1961), 141-168.

[25] A.J. Guttmann, Planar Polygons: Regular, convex, almost convex, staircase and row convex, AIP Conference Proceedings 248, "Computer-aided Statistical Physics", Taipei, Taiwan 1991. Printed 1992.

[26] E.T. Irons, A syntax directed compiler for Algol 60, Comm. ACM 4 (1961) 51-55.

[27] E.T. Irons, Towards more versatile mechanical translations, Proc. Symp. Appl. Math., 15 (1963) 41-50.

[28] D.E. Knuth, Semantics of context-free languages, Maths. Sys. 2 (1968), 127-145.

[29] M. Nivat, Transductions des langages de Chomsky, Ann. Inst. Fourier, 18 (1968) 339-456.

[30] B. Randell, L.J. Russel, *Algol 60 implementation*, Academic Press, New-York, 1964.

[31] M.P. Schützenberger, Certain elementary families of automata, in Proc. Symp. on Mathematical Theory of Automata, Polytechnic Institute of Brooklyn, 1962, 139-153.

[32] M.P.Schützenberger, Context-free langages and pushdown automata, Information and Control 6 (1963), 246-264.

[33] M. Vauchaussade, X. Viennot, Enumeration of RNA secondary structures by complexity, Actes Mathematics in Biology and Medecine, Bari (1985), Lecture Notes in Biomathematics, 1985, 360-365.
[34] X. Viennot, Enumerative combinatorics and algebraic languages, Actes FCT'85, ed. L. Budach, Lecture Notes in Computer Science, 199, 450-464.
[35] X. Viennot, private communication with M.P. Schützenberger, May 1991.
[36] X. Viennot, A survey of polyominoes enumeration, Actes SFCA'92, Ed. P. Leroux and C. Reutenauer, Montréal, 1992, 399-420.
[37] X. Viennot, Trees in *Mots, mélange offert à M.P. Schützenberger*, M. Lothaire (ed.), Hermes, Paris, 1990, 265-291.

LaBRI, Université BORDEAUXI, FRANCE
Current adress : LaBRI, 351 Cours de la Libération, 33405 TALENCE CEDEX, FRANCE
E-mail adresss : Maylis.Delest@labri.u-bordeaux.fr