Discrete Mathematics and Theoretical Computer Science **2**, 1998, 49–63

Object grammars and random generation

I. Dutour[1] and J. M. Fédou[2]

[1]*LaBRI, Université Bordeaux I, 351 cours de la Libération, 33405 Talence Cedex, France*
E-mail: `dutour@labri.u-bordeaux.fr`
[2]*I3S-ESSI, Université de Nice, 650 Route des Colles, B.P. 145, 06903 Sophia-Antipolis Cedex, France*
E-mail: `fedou@unice.fr`

This paper presents a new systematic approach for the uniform random generation of combinatorial objects. The method is based on the notion of object grammars which give recursive descriptions of objects and generalize context-free grammars. The application of particular valuations to these grammars leads to enumeration and random generation of objects according to non algebraic parameters.

Keywords: Uniform random generation, object grammars, q-equations

1 Introduction

An *object grammar* defines classes of objects by means of terminal objects and certain types of operations applied to the objects. It is most often described with pictures. For instance, the standard decomposition of complete binary trees is an object grammar (Figure 1). The formalism given here for object grammars [6, 7] generalizes the one for context-free grammars. An important application of these grammars is a systematic approach for the specification of bijections between sets of combinatorial objects (see [7]).

The paper outlines another important application of the object grammars: a *systematic* method for generating combinatorial objects *uniformly at random*. The work lies in the *recursive method* framework; this method is to generate recursively random objects by endowing a recurrence formula with a probabilistic interpretation. This generation process has been first formalized by Nijenhuis and Wilf [11, 14, 15]. They have a general approach. They base the recursive procedure on an acyclic directed rooted graph with a terminal vertex and numbered edges, graph which depends on the family of objects. The recursive method has been also formalized by Hickey and Cohen in the special case of context-free languages [10], and by Greene within the framework of the labelled formal languages [9].

Recently, Flajolet, Zimmermann and Van Cutsem have given a *systematic* approach for this method with specifications of structures by grammars involving *set*, *sequence* and *cycle* constructions [8]. The methods that they have examined enable to start from any hight level specification of decomposable class and compile *automatically* procedures that solve the corresponding random generation problem. They have presented two closely related groups of methods: the *sequential algorithms* (linear search) which have worst case time complexity $O(n^2)$, when applied to objects of size n, and the *boustrophedonic algorithms* (bidirectional fashion) which have worst case time complexity $O(n \log n)$.

The present work is a continuation of the research of these authors. It is a systematization of the recursive method based on the object grammars. It then extends the field of structures which can be

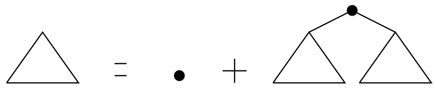

Fig. 1: Complete binary trees.

generated using such method. Another important contribution of this work is to consider the random generation of objects according to *several* parameters simultaneously, and to consider not only algebraic parameters (i.e. that lead to algebraic generating functions), but also parameters that lead to generating functions satisfying q-*equations*. The other methods have rarely dealed with these latest parameters.

In section 2, we review the necessary definitions for object grammars. We then provide in section 3 the notion of q-*linear valuations*. They formalize the behaviour (on objects defined by an object grammar) of parameters that lead automatically to q-equations satisfied by the corresponding generating functions.

Section 4 introduces our systematic random generation method. Given an unambigous object grammar and a corresponding q-linear valuation, it allows to construct automatically the enumeration and uniform generation procedures according to the valuation. These procedures use sequential algorithms and have worst case time complexity $O(kl(k+l))$, when applied to objects of valuation $x^k q^l$, assuming the enumeration tables have been computed once for all in $O(k^2 l^2)$ time (see [6] for the general case of valuation). If one only considers an algebraic parameter (x^k), the complexity is the same as in [8], and the boustrophedonic search can be used. Note that, as in [8], the complexity is related to the *number* of arithmetic operations, unit cost is taken for the manipulation of a large integer.

The path taken here is eminently praticable and the method has been implemented in the *Maple* language (package named *qAlGO*). Section 5 gives some results obtained with this program concerning the uniform random generation of convex polyominoes according to the area and planar trees according to the internal path length. The package *CombStruct*, written by P. Zimmermann, cannot study these objects and this type of parameter. The packages *qAlGO* and *CombStruct* complement each others. In section 6, we finish by discussing some ideas and directions of research.

2 Object Grammars

Let $\mathcal{E}$ be a family of sets of objects. An *object operation (in $\mathcal{E}$)* is a mapping $\phi : E_1 \times \ldots \times E_k \longrightarrow E$, where $E \in \mathcal{E}$ and $E_i \in \mathcal{E}$ for i in $[1, k]$. It describes the way of building an object of E from k objects belonging to $E_1, \ldots, E_k$, respectively.

The *domain* of ϕ is $E_1 \times \ldots \times E_k$, denoted by $dom(\phi)$, the *codomain* is E, denoted by $codom(\phi)$ and the *image* is denoted by $Im(\phi)$. The i-th projection E_i of $dom(\phi)$ is called a *component* of ϕ.

Example 2.1 . A parallelogram polyomino can be defined as the surface lying between two North-East paths that are disjoint, except at their common ending points (see Figure 2) [5]. Let E_{pp} be the set of parallelogram polyominoes.

The mappings ϕ_1, ϕ_2 and ϕ_3 illustrated in Figure 3 are object operations in $\mathcal{E} = \{E_{pp}\}$. The operations ϕ_1 and ϕ_2 are operations of arity 1 ($E_{pp} \longrightarrow E_{pp}$). The operation ϕ_1 glues a new cell at the left of the lowest cell of the first column of a polyomino. The operation ϕ_2 adds a new cell at the bottom of each column of a polyomino. The operation ϕ_3 is an operation of arity 2 ($E_{pp} \times E_{pp} \longrightarrow E_{pp}$) ; it takes two polyominoes as argument, applies ϕ_2 to the first one and glues

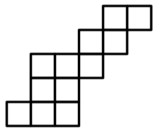

Fig. 2: A parallelogram polyomino.

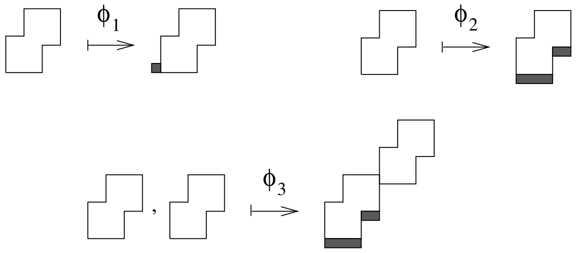

Fig. 3: Object operations on parallelogram polyominoes.

them by one cell: the top-cell of the last column of the first polyomino facing the bottom-cell of the first column of the second.

Definition 2.1 *An* object grammar *is a* 4*-tuple* $< \mathcal{E}, \mathcal{T}, \mathcal{P}, S >$ *where:*

$\mathcal{E}$	$= \{E_i\}_{i \in I}$	*is a finite family of sets of objects. (I is a finite subset of* $\mathbb{N}$*).*
$\mathcal{T}$	$= \{T_{E_i}\}_{i \in I}$	*is a finite family of finite subsets of sets of* $\mathcal{E}$*,* $T_{E_i} \subset E_i$*, whose elements are called* terminal objects.
$\mathcal{P}$		*is a set of object operations* ϕ *in* $\mathcal{E}$*.*
S		*is a fixed set of* $\mathcal{E}$ *called the* axiom *of the grammar.*

The *dimension* of an object grammar is the cardinality of $\mathcal{E}$.
Remark. Sometimes a 3-tuple $< \mathcal{E}, \mathcal{T}, \mathcal{P} >$ is called also object grammar. The axiom is chosen later in $\mathcal{E}$.
In the following, the terms *grammar* and *operation* will often be used for *object grammar* and *object operation* respectively.

The construction of an object can be described by its *derivation tree* : *internal nodes* are labelled with object operations and *leaves* with terminal objects. These derivation trees are comparable to the abstract trees within the theory of Compiling.

Let $G = < \mathcal{E}, \mathcal{T}, \mathcal{P} >$ be an object grammar and $E \in \mathcal{E}$ a set of objects. An object o is said to be *generated* in G by E, if there is a derivation tree of G on E (i.e. the codomain of the label of the root is E) whose evaluation is o.
The set of objects generated by E in G is denoted by $\mathcal{O}_G(E)$. If S in $\mathcal{E}$ is chosen as the axiom of G, then $\mathcal{O}_G(S)$ is called the set of objects generated by G.

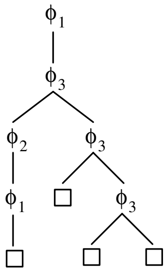

Fig. 4: A derivation tree in G_2.

Example 2.2 . Let's note $\square$ the one-cell polyomino. Here are two examples of object grammars:

$$\begin{aligned} G_1 &= \ < \{E_{pp}\}, \{\{\square\}\}, \{\phi_1, \phi_2\}, E_{pp} > \\ \text{and} \quad G_2 &= \ < \{E_{pp}\}, \{\{\square\}\}, \{\phi_1, \phi_2, \phi_3\}, E_{pp} > \ . \end{aligned}$$

The parallelogram polyomino of Figure 2 belongs to $\mathcal{O}_{G_2}(E_{pp})$, its derivation tree in G_2 is given in Figure 4. The set $\mathcal{O}_{G_2}(E_{pp})$ is the set of parallelogram polyominoes.
The set $\mathcal{O}_{G_1}(E_{pp})$ is the set of *Ferrers diagrams*; it is a proper subset of parallelogram polyominoes.

By analogy to context-free grammars, an object grammar G is *unambiguous* if every object in $\mathcal{O}_G(S)$ has exactly one derivation tree. Unambiguity is an important property for building bijections.

One can also define several *normal forms* for object grammars: *reduced*, *1-2* or *complete*. The *reduced* and *1-2* forms extend usual normal forms of context-free grammars: the reduced and Chomsky normal form. A grammar is said to be reduced if every set of objects E in $\mathcal{E}$ is accessible from the axiom and $\mathcal{O}_G(E) \neq \emptyset$; it is said to be in 1-2 form if all its operations are of arity 1 or 2. The *complete* form is specific for object grammars. A grammar is said to be complete if $\mathcal{O}_G(E) = E$ for every set of objects E in $\mathcal{E}$ (generally $\mathcal{O}_G(E) \subseteq E$). For example, the grammar G_2 previously defined is complete while G_1 is not. The details on transformations of object grammars into normal forms are given in [6].

Another Definition

A complete, unambiguous object grammar $G =< \mathcal{E}, \mathcal{T}, \mathcal{P}, S >$ can be described as a *system of equations* Σ involving sets of objects, terminal objects and object operations, or as a *system of graphic equations*. The equations describe the decomposition of a set of objects into a disjoint union of terminal objects and images of operations:

$$\Sigma = \left\{ E_i = \sum_{e_i \in T_{E_i}} e_i \ + \sum_{codom(\phi) = E_i} \phi(E_{i_{1,\phi}}, \ldots, E_{i_{k,\phi}}) \right\}_{i=1,\ldots,n} .$$

For example, the equation for the grammar G_2 generating parallelogram polyominoes previously defined is

$$E_{pp} = \square + \phi_1(E_{pp}) + \phi_2(E_{pp}) + \phi_3(E_{pp}, E_{pp}).$$

A schematic representation of this grammar is given in Figure 5.

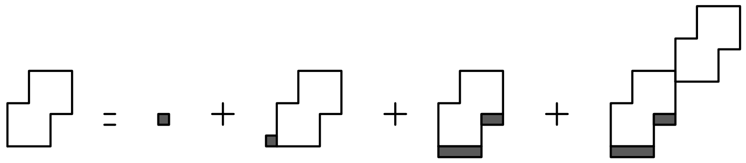

Fig. 5: Schematic object grammar for parallelogram polyominoes.

Expanded 1-2 form

The automatic method of random generation presented in the paper is based on the *expanded* 1-2 form of object grammars.

An object grammar $G =< \mathcal{E}, \mathcal{T}, \mathcal{P} >$ is called in *expanded 1-2 form* if, for every E in $\mathcal{E}$, the equation that defines it has one of the forms

$$E = e;\ E = E_1 + E_2;\ E = \phi(E_1);\ E = \phi(E_1, E_2).$$

Proposition 2.2 *Every object grammar has an equivalent expanded 1-2 form.*

Proof. To transform an object grammar into an expanded 1-2 form, it suffices to change all the sums and domains of the object operations having arity > 2 by adding sets of objects and identity object operations of arity 2. Thus, the equation $E = \phi(E_1, \ldots, E_k)$ is replaced by the set of equations

$$E = \phi_{E_1}(E_1, F_{E_2}),\, F_{E_2} = \phi_{E_2}(E_2, F_{E_3}),\, \ldots,\, F_{E_{k-1}} = \phi_{E_{k-1}}(E_{k-1}, E_k).$$

□

In the following, we will often use the term *1-2 form* for *expanded 1-2 form.*

3 Enumeration

Let K be a ring and $X = \{x_1, \ldots, x_n\}$ a set of variables. Then $K[X]$ (resp. $K[[X]]$) denotes the set of *polynomials* (resp. *formal power series*) in the variables $x_1, \ldots, x_n$ having coefficients in K.

Given a set of objects E, an *object valuation* (on E) is a mapping $V_E : E \longrightarrow K[X]$ satisfying

$$\forall\, (k_1, \ldots, k_n) \in \mathbb{N}^n; \{e \in E / < {x_1}^{k_1} \ldots {x_n}^{k_n},\, V_E(e) >, \neq 0\} \text{ is finite.}$$

Consequently, the generating function associated with E, $\sum_{e \in E} V_E(e)$, is a formal power series which lies in $K[[X]]$. It will be denoted by $V_E(E)$.

Theorem 3.1 *Let $G =< \mathcal{E}, \mathcal{T}, \mathcal{P} >$ be a complete, unambiguous object grammar, and $\nu = \{V_E : E \longrightarrow K[X], E \in \mathcal{E}\}$ a set of object valuations. For all E in $\mathcal{E}$, from its equation in G*

$$E \;=\; \sum_{e \in T_E} e \;+ \sum_{codom(\phi)=E} \phi(E_{i_{1,\phi}}, \ldots, E_{i_{k,\phi}})\,, \tag{1}$$

one can directly obtain the following equation:

$$V_E(E) \;=\; V_E(T_E) \;+ \sum_{codom(\phi)=E} V_E(\phi(E_{i_{1,\phi}}, \ldots, E_{i_{k,\phi}}))\,. \tag{2}$$

Proof. The object grammar G is unambiguous and complete, given equation (1). Equation (2) is obvious, since we have disjoint unions.

□

The objective is to obtain a system of equations for the generating functions of the sets of the grammar. Then, one has to express $V_E(\phi(E_{i_{1,\phi}},\ldots,E_{i_{k,\phi}}))$ in terms of $V_{E_{i_{1,\phi}}}(E_{i_{1,\phi}})$, ..., $V_{E_{i_{k,\phi}}}(E_{i_{k,\phi}})$. This depends on the nature of object valuations considered.

Example 3.1 . Let E_{pp} be the set of parallelogram polyominoes and consider the following valuation:

$$\begin{array}{llll} V_{wa}: & E_{pp} & \longrightarrow & \mathbb{N}[x,q] \\ & e & \longmapsto & x^{width(e)}q^{area(e)} \end{array}$$

It is well-known that the generating function $V_{wa}(E_{pp})$, denoted here by $f_{wa}(x,q)$, satisfies the q-equation (see for example [3])

$$f_{wa}(x,q) = xq + xqf_{wa}(x,q) + f_{wa}(xq,q) + f_{wa}(xq,q)f_{wa}(x,q)\ .$$

The object valuation V_{wa} is called *q-linear*. The general definition of such a valuation is detailed below.

q-linear Object Valuations

Let $X = \{x_1,\ldots,x_n\}$ and $Q = \{q_1,\ldots,q_r\}$ be two disjoint sets of variables.

Notations. x denotes the n-tuple $(x_1,\ldots,x_n)$ and q the r-tuple $(q_1,\ldots,q_r)$. If A is a matrix having coefficients in $\mathbb{N}$ ($A = (a_{ij})$, $1 \le i \le n$, $1 \le j \le r$), then

- $xq^A = (x_1{q_1}^{a_{11}}\ldots{q_r}^{a_{1r}},\ldots,x_n{q_1}^{a_{n1}}\ldots{q_r}^{a_{nr}})$,
- for $f(x,q) \in K[[X,Q]]$, $f(x,q)|_{x\leftarrow xq^A} = f(xq^A,q)$.

Let $E, E_1,\ldots,E_k$ be sets of objects, $V_E, V_{E_1},\ldots,V_{E_k}$ object valuations on $E, E_1,\ldots,E_k$ respectively, and ϕ an object operation with $codom(\phi) = E$ and $dom(\phi) = E_1\times\ldots\times E_k$.

V_E is called *q-linear* with respect to ϕ if it exists a polynomial λ_ϕ in $K[X,Q]$ and matrices ${A_i}^\phi$ for $i\in[1,k]$ such that

$$V_E(\phi(e_1,\ldots,e_k)) = \lambda_\phi \prod_{i=1}^{k} V_{E_i}(e_i)|_{x\leftarrow xq^{{A_i}^\phi}}\ \text{, for every } (e_1,\ldots,e_k)\in dom(\phi).$$

If ϕ is *injective*, then

$$V_E(\phi(E_1,\ldots,E_k)) = \lambda_\phi \prod_{i=1}^{k} V_{E_i}(E_i)|_{x\leftarrow xq^{{A_i}^\phi}}\ .$$

Corollary 3.2 *If all the object valuations of ν are q-linear, equation (2) of Theorem 3.1 becomes:*

$$V_E(E) \quad = \quad V_E(T_E) \ + \sum_{codom(\phi)=E} \lambda_\phi \prod_{i=i_{1,\phi}}^{i_{k,\phi}} V_{E_i}(E_i)|_{x\leftarrow xq^{{A_i}^\phi}}\ . \qquad (3)$$

This is a system of q-equations were the unknowns are the generating functions $V_E(E)$ for the sets E in $\mathcal{E}$.

Example 3.2 . The object valuation V_{wa} is q-linear with respect to the object operations ϕ_1, ϕ_2 and ϕ_3. Then the equation

$$V_{wa}(E_{pp}) = V_{wa}(\square) + V_{wa}(\phi_1(E_{pp})) + V_{wa}(\phi_2(E_{pp})) + V_{wa}(\phi_3(E_{pp}, E_{pp}))$$

becomes the q-equation seen before ($f_{wa}(x,q) = V_{wa}(E_{pp})$)

$$f_{wa}(x,q) = xq + xqf_{wa}(x,q) + f_{wa}(xq,q) + f_{wa}(xq,q)f_{wa}(x,q)\ .$$

Special Case: Linear Object Valuations

The *linear* object valuations are q-linear object valuations such that, for every object operation ϕ and all i, $A_i{}^{\phi} = (0)$. These linear valuations are exactly in the DSV methodology framework [2, 13], they yield algebraic generating functions (see [6]).

Object Valuations and 1-2 form

The proof of Proposition 2.2 has shown how to reduce object grammars in 1-2 form. The q-linear object valuations are very well preserved through this transformation.

Proposition 3.3 *If ν is a set of q-linear object valuations associated with an object grammar, it is possible to construct an equivalent set of object valuations associated with its 1-2 form.*

Proof. Recall that an equation $E = \phi(E_1, \ldots, E_k)$ in the grammar is replaced by the set of equations

$$E = \phi_{E_1}(E_1, F_{E_2}),\ F_{E_2} = \phi_{E_2}(E_2, F_{E_3}),\ \ldots,\ F_{E_{k-1}} = \phi_{E_{k-1}}(E_{k-1}, E_k).$$

Concerning the object valuations, if we have

$$V_E(\phi(E_1,\ldots,E_k)) = \lambda_\phi \prod_{i=1}^{k} V_{E_i}(E_i)\Big|_{x \leftarrow xq^{A_i^\phi}},$$

then we define

$$\begin{cases} V_E(\phi_{E_1}(E_1, F_{E_2})) & = \ \lambda_\phi\ V_{E_1}(E_1)\Big|_{x\leftarrow xq^{A_1^\phi}} . V_{F_{E_2}}(F_{E_2}) \\ for\ \ i = 2\ldots k-2, & \\ V_{F_{E_i}}(F_{E_i}) & = \ V_{F_{E_i}}(\phi_{E_i}(E_i, F_{E_{i+1}})) \\ & = \ V_{E_i}(E_i)\Big|_{x\leftarrow xq^{A_i^\phi}} . V_{F_{E_{i+1}}}(F_{E_{i+1}}) \\ & \\ V_{F_{E_{k-1}}}(F_{E_{k-1}}) & = \ V_{F_{E_{k-1}}}(\phi_{E_{k-1}}(E_{k-1}, E_k)) \\ & = \ V_{E_{k-1}}(E_{k-1})\Big|_{x\leftarrow xq^{A_{k-1}^\phi}} .\ V_{E_k}(E_k)\Big|_{x\leftarrow xq^{A_k^\phi}} \end{cases}$$

$\square$

4 Enumeration and Random Generation Procedures

Not all object grammars G and possible corresponding sets of functions ν lead to random generation. The couples (G, ν) considered here are *well-founded*, i.e. each set of objects generates at least one object, and it generates a finite number of objects having the same valuation's value (it is the definition of an object valuation). An algorithm performing this task is detailed in [6]. It is inspired by works of Zimmermann [16].

In the following, the study is limited to the case of q-linear object valuations having values in the set of monomials denoted by $Mon[X, Q]$. Moreover, X and Q contain only one variable: $X = \{x\}$ and $Q = \{q\}$. The complete case is detailed in [6].

4.1 Enumeration Procedures

Let $G =< \mathcal{E}, \mathcal{T}, \mathcal{P} >$ be an object grammar in expanded 1-2 form and $\nu = \{V_E : E \longrightarrow Mon[x, q], E \in \mathcal{E}\}$ a set of q-linear valuations such that the couple (G, ν) is well-founded.

The generating function of a set of objects E is denoted by

$$V_E(E) = \sum_{K,L=0}^{\infty} c_E(K, L)\; x^K q^L.$$

Theorem 4.1 *(i) The coefficients $c_E(K, L)$ are given by the following formulas :*

- $E = e$, *then*
$$\textit{if } V_E(e) = x^K q^L \;\textit{ then }\; c_E(K, L) = 1 \;\textit{ else }\; 0,$$
- $E = E_1 + E_2$, *then*
$$c_E(K, L) = c_{E_1}(K, L) + c_{E_2}(K, L),$$
- $E = \phi(E_1)$ *with* $V_E(\phi(E_1)) = x^{k_0} q^{l_0}\, V_{E_1}(E_1)|_{x \leftarrow x q^{a_1}}$, *then*
$$c_E(K, L) = c_{E_1}(K - k_0, L - l_0 - (K - k_0).a_1),$$
- $E = \phi(E_1, E_2)$ *with* $V_E(\phi(E_1, E_2)) = x^{k_0} q^{l_0}\, V_{E_1}(E_1)|_{x \leftarrow x q^{a_1}}\, V_{E_2}(E_2)|_{x \leftarrow x q^{a_2}}$, *then*
$$c_E(K, L) = \sum_{i=0}^{K} \sum_{j=0}^{L} c_{E_1}(i, j).c_{E_2}(K - k_0 - i, L - l_0 - i.a_1 - (K - k_0 - i).a_2 - j).$$

(ii) The computation of all the coefficients up to the value $x^k q^l$ needs $O(k^2 l^2)$ arithmetic operations.

Proof. The details of the algorithm computing the coefficients in $O(k^2 l^2)$ time are given in [6]. □

4.2 Generation Procedures

With each set of objects E of the grammar is associated a procedure g_E having parameters k and l, and generating (uniformly at random) an object of E having the valuation $x^k q^l$. More precisely, this procedure constructs the derivation tree in the grammar. It depends on the form of the equation which defines E in the object grammar:

- $E = e$. The procedure g_E is trivial :

g_E :=**Proc**(k,l)
  If $V_E(e) = x^k q^l$ **Then** Return e
 End Proc

- $E = E_1 + E_2$. The procedure g_E must generate an object belonging either to E_1 or to E_2. The probability that this object belongs to E_1 is equal to $c_{E_1}(k,l)/c_E(k,l)$:

g_E :=**Proc**(k,l)
  U := Uniform($[0,1]$);
  If $U < c_{E_1}(k,l)/c_E(k,l)$ **Then** Return $g_{E_1}(k,l)$
  Else Return $g_{E_2}(k,l)$
 End Proc

- $E = \phi(E_1)$, with $V_E(\phi(E_1)) = x^{k_0} q^{l_0} V_{E_1}(E_1)|_{x \leftarrow x q^{a_1}}$. Then the procedure g_E is very simple. It returns an object of E obtained by ϕ from an object of E_1 having the valuation $x^{k-k_0} q^{l-l_0-(k-k_0).a_1}$:

g_E :=**Proc**(k,l)
  kk := $k - k_0$; ll := $l - l_0$;
  Return $\phi(\ g_{E_1}(kk, ll - kk.a_1)\ )$
 End Proc

- $E = \phi(E_1, E_2)$, with $V_E(\phi(E_1,E_2)) = x^{k_0} q^{l_0} V_{E_1}(E_1)|_{x \leftarrow x q^{a_1}} V_{E_2}(E_2)|_{x \leftarrow x q^{a_2}}$. The procedure g_E must generate an object of E obtained by ϕ from an object of E_1 and an object of E_2 which respect the definition of the valuations. The probability for the object of E_1 having the valuation $x^K q^L$ and those of E_2 having the valuation $x^{k-k_0-K} q^{l-l_0-K.a_1-(k-k_0-K).a_2-L}$ is $c_{E_1}(K,L).c_{E_2}(k-k_0-K, l-l_0-K.a_1-(k-k_0-K).a_2-L)/c_E(k,l)$. The procedure is:

g_E :=**Proc**(k,l)
  kk := $k - k_0$; ll := $l - l_0$;
  S := $c_{E_1}(0,0).c_{E_2}(kk, ll - kk.a_2)/c_E(k,l)$;
  K := 0;
  U := Uniform($[0,1]$);
  While $U > S$ **Do**
   K := $K + 1$;
   T := $ll - K.a_1 - (kk - K).a_2$;
   L := 0;

Random generation algorithm:
Input: a couple (G, ν).
Output: procedures for generating the objects generated by G at random.

- $\star$ Transform G into 1-2 form $\Longrightarrow (G_0, \nu_0)$.
- $\star$ Verify that (G_0, ν_0) is well-founded, else error.
- $\star$ For each set of objects E in G_0, create the enumeration procedure c_E, then compute all the coefficients up to rank (k, l).
- $\star$ For each set of objects E in G_0, create the generation procedures g_E as indicated above.

Fig. 6: A random generation procedure.

```
    While U > S And L ≤ l Do
      L := L + 1;
      S := S + c_E1(k,l).c_E2(kk − K, T − L)/c_E(k,l);
    End While
  End While
  Return φ( g_E1(k,l), g_E2(kk − K, T − L) )
End Proc
```

Theorem 4.2 *The worst case time complexity of the generation procedures is of $O(kl(k+l))$ arithmetic operations.*

Proof. A random generation procedure consists in constructing recursively a derivation tree in G. This tree is binary because G is in 1-2 form. The size of the derivation tree of an object having the valuation $x^k q^l$ is proportional to $k + l$. The generation of a vertex of the tree has a maximal cost of $O(kl)$ (the loops of the procedure). Thus, the complexity of the generation of the derivation tree in the worst case is $O(kl(k+l))$. □

4.3 Algorithm for Uniform random generation

One can now describe an uniform random generation procedure for the objects of an object grammar according to a set of q-linear object valuations (Figure 6).

The obtained generation procedures give the derivation trees of objects in G_0, but not directly in G. A simple transformation (linear cost in $O(k+l)$) gives the derivation trees in G. This postprocessing does not affect the conclusions of the complexity studies. Futhermore, at the expense of some programming effort, it can be effected 'on the fly'.

5 Maple package qAlGO

The program *qAlGO* (in *Maple* language) implements the method developed in the previous sections. The package *qAlGO* builds automatically the enumeration and generation procedures from a unambiguous object grammar and a set of corresponding q-linear valuations (see the annex of [6] for syntax and use).

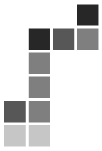

Fig. 7: A parallelogram polyomino of area 10.

The automatic nature of the software *qAlGO* gives a very useful tool which makes easy the experimental study of various statistics on combinatorial objects. In the following, we present relevant examples of random generation.

Convex Polyominoes According to the Area

Here is an example of experimental studies using *qAlGO*. It concerns the random generation, according to the area, of different classes of convex polyominoes: parallelogram polyominoes, convex directed polyominoes and convex polyominoes.

First, the example of parallelogram polyominoes. It suffices to give as input to *qAlGO* an object grammar that generates them and the corresponding object valuations: the grammar in Figure 5 and the valuation V_{wa} defined in example 3.1. Thus one obtains the enumeration and the uniform generation according to the width (in x) and the area (in q).

```
> with(qalgo);
              [compile, countgo, drawgo, drawgoall]
  # definition of the object grammar and valuations
> paralgo  := { P = cell + phi1(P) + phi2(P) + phi3(P,P) }:
> paralval := [[cell, 1, 1], [phi1, 1, 1, [0]],
               [phi2, 0, 0, [1]], [phi3, 0, 0, [1, 0]]]:
  # construction of the procedures
> compile( paralgo, paralval, qlinear, Identity):
```

We are then able to generate these objects at random. More precisely, *qAlGO* returns the derivation tree of a random parallelogram polyomino. For example,

```
  # generation of a polyomino having area 10
> drawgo( paralgo, paralval, qlinear, P, 10);
         phi3(phi1(phi2(phi2(phi2(cellp)))), phi1(phi2(cellp)))
```

It then suffices to evaluate this derivation tree according to the object operations, and to write the polyomino under a form understood by the interface *XAnimal* of *CalICo*† [4, 12]; so we can visualize the polyomino Figure 7.

For the convex polyominoes, we use a much more complicated object grammar (of dimension 9 and with 34 object operations !), but the principle is exactly the same as before for the parallelogram polyominoes.

† *CalICo* offers a software environment for manipulations and visualizations of combinatorial objects; it allows the communication of graphical interfaces and computer algebra software such as *Maple*.

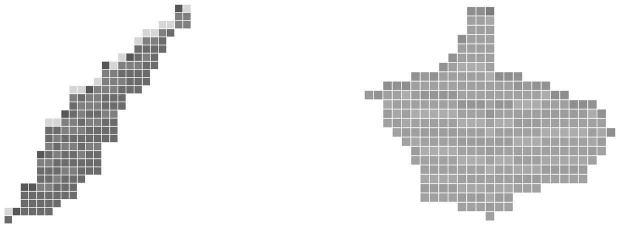

Fig. 8: Random parallelogram and convex polyominoes of perimeter 100.

Fig. 9: Random parallelogram and convex polyominoes of area 100.

Such experiments showed us how thin the random convex polyominoes according to the area is. According to the perimeter, they look more thick. Examples of such polyominoes are given Figures 8 and 9. We also find out that convex polyominoes, random according to the area, have either a north-east, or north-west orientation (as Figure 9), with same probability one half.

To understand the thin look of such random polyominoes according to the area, we computed experimentally the average value of two parameters: the *height of a column* and the *gluing number between two adjacent columns*, which is the number of cells by which two adjacent columns are in contact. After generating 1000 convex polyominoes having area 100 and 1000 parallelogram polyominoes having area 200, we obtain the following average values: $2, 37$ for the height of the columns, and $1, 37$ for the gluing number between two adjacent columns.

Remark. The result for the average height of a column $(2, 37)$ coincides exactly with what Bender [1] has obtained using asymptotic analysis methods.

The difference of 1 between these two average parameters can be explained simply by noticing that this difference has for limit the quotient between the height and the width of the polyomino, which is 1 by symmetry.

Planar Trees According to the Internal Path Length

A simple unambiguous object grammar for the planar trees is given in Figure 10. The *internal path*

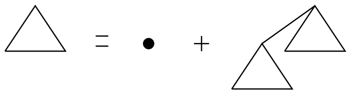

Fig. 10: An object grammar for planar trees.

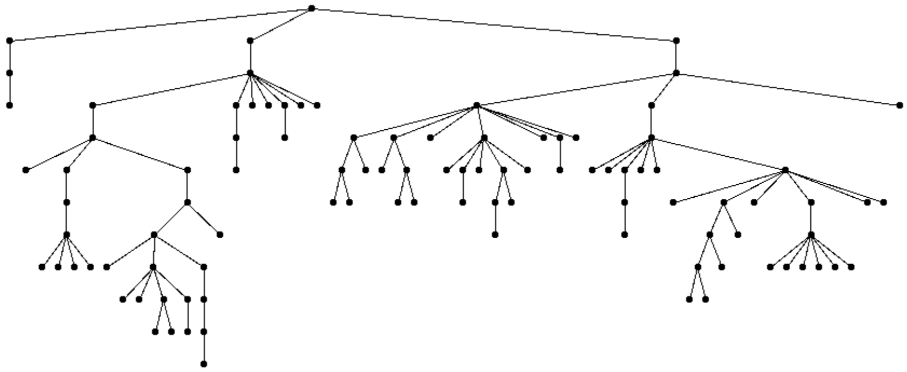

Fig. 11: A random planar tree of size 100.

length (ipl) of a tree is the sum of the distances of all its nodes from its root. The q-linear valuation $V_{ipl} : e \longmapsto x^{nodes(e)} q^{ipl(e)}$ leads to the following q-equation for the generating function

$$T(x, q) = x + T(xq, q)T(x, q) .$$

Setting q to 1, we obtain the linear valuation for the *size* of the trees (the number of nodes).

Using *qAlGO* with the above grammar and these two valuations, we are able to generate at random planar trees according to the size (see Figure 11), and also, according to the internal path length (see Figure 12). The latter have a remarkable look: they have a very small height.

6 Conclusions and Perspectives

The interest of our approach lies in its generality and simplicity. Time complexity are 'computable' and, at the same time, one gains access to the random generation of arbitrarily complex objects according to

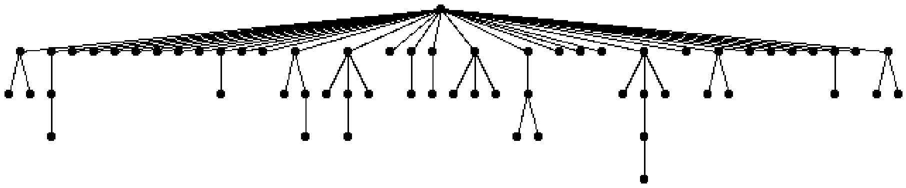

Fig. 12: A random planar tree of internal path length 100.

several parameters, algebraic or not. In the previous section, we showed how it is possible to use the package *qAlGO* in order to get conjectures on some parameters of objects. A lot of studies can be done for other objects as paths (according to the area), different classes of trees (according to the internal path length),… The automatic nature of *qAlGO* makes such studies easy.

Concerning the decomposable structures, Flajolet, Zimmermann and Van Cutsem [8] have shown that the generation of a structure of size n is in $O(n \log n)$ by using a *boustrophedonic* algorithm instead of $O(n^2)$ by using a sequential one. It would be interesting to make a similar analysis to see if the *boustrophedonic* principle can improve the complexity of our algorithms of generation in the same way. But we have here more than one parameter, then the strategy is not obvious to determine (and to analyse).

Acknowledgements

The authors thank Jacques Labelle and the anonymous referees for very helpful comments.

References

[1] Bender, E. A. (1974). Convex n-ominoes. *Discrete Math.* **8**: 219–226.

[2] Delest, M. (1994). Langages algébriques: à la frontière entre la Combinatoire et l'Informatique. *Actes du* $6^{ème}$ *Colloque Séries Formelles et Combinatoire Algébrique*, 69–78. DIMACS, Rutgers University, USA.

[3] Delest, M. and Fédou, J. M. (1992). Attribute grammars are useful for combinatorics. *Theoretical Computer Science* **98**: 65–76.

[4] Delest, M., Fédou, J. M., Melançon, G. and Rouillon, N. (1998). *Computation and images in Combinatorics*. Springer-Verlag.

[5] Delest, M. and Viennot, X. G. (1984). Algebraic languages and polyominoes enumeration. *Theoretical Computer Science* **34**: 169–206.

[6] Dutour, I. (1996). *Grammaires d'objets: énumération, bijections et génération aléatoire*. Université Bordeaux I.

[7] Dutour, I. and Fédou, J. M. (1997). *Object grammars and bijections*. LaBRI, Université Bordeaux I, 1164-97. (Submitted to *Theoretical Computer Science*).

[8] Flajolet, P., Zimmermann, P. and Van Cutsem, B. (1994). A Calculus for the Random Generation of Combinatorial Structures. *Theoretical Computer Science* **132**: 1–35.

[9] Greene, D. H. (1983). *Labelled formal languages and their uses*. Stanford University.

[10] Hickey, T. and Cohen, J. (1983). Uniform Random Generation of Strings in a Context-Free Language. *SIAM. J. Comput.* **12**(4): 645–655

[11] Nijenhuis, A. and Wilf, H. S. (1975). *Combinatorial Algorithms, 2nd ed.* Academic Press.

[12] Rouillon, N. (1994). *Calcul et Image en Combinatoire*. Université Bordeaux I.

[13] Viennot, X. G. (1985). Enumerative combinatorics and algebraic languages. *Proceedings FCT'85*, 450–464. *Lecture Notes in Computer Science 199*. Springer-Verlag.

[14] Wilf, H. S. (1977). A unified setting, ranking, and selection algorithms for combinatorial objects. *Advances in Mathematics*, **24**: 281–291.

[15] Wilf, H. S. (1978). A unified setting for selection algorithms (II). *Ann. Discrete Math.* **2**: 135–148.

[16] Zimmermann, P. (1991). *Séries génératrices et analyse automatique d'algorithmes*. Ecole Polytechnique de Palaiseau.