A linear Time Algorithm for the Generation of Trees

Laurent Alonso, Jean-Luc Rémy, René Schott

INRIA

INSTITUT NATIONAL DE RECHERCHE EN INFORMATIQUE ET EN AUTOMATIQUE

A linear time algorithm for the generation of trees

Laurent Alonso Jean-Luc Rémy René Schott

N° 2934

Juillet 1996

THÈME 2

ISSN 0249-6399

A linear time algorithm for the generation of trees

Laurent Alonso Jean-Luc Rémy René Schott

Thème 2 — Génie logiciel
et calcul symbolique
Projet Eureca

Rapport de recherche no 2934 — Juillet 1996 — 32 pages

Abstract: We present a linear algorithm which generates randomly and with uniform probability many kinds of trees: binary trees, ternary trees, arbitrary trees, forests of p k-ary trees, ... The algorithm is based on the definition of generic trees which can be coded as words. These words, in turn, are generated in linear time.

Key-words: generation, uniform, tree, grammar, combinatory

(Résumé : tsvp)

Cet article doit paraître dans Algorithmica en 1996

* INRIA-Lorraine, BP. 101, 54602 Villers-lès-Nancy, France, alonso@loria.f
† CRIN, CNRS, BP. 239, 54506 Vandœuvre-lès-Nancy, France, remy@loria.fr
‡ CRIN, UHP (Université Henri Poincaré), BP. 239, 54506 Vandœuvre-lès-Nancy, France, schott@loria.f

Unit´e de recherche INRIA Lorraine
Technôpole de Nancy-Brabois, Campus scientifique,
615 rue de Jardin Botanique, BP 101, 54600 VILLERS LÈS NANCY (France)
T´el´ephone : (33) 83 59 30 30 – T´el´ecopie : (33) 83 27 83 19
Antenne de Metz, technopôle de Metz 2000, 4 rue Marconi, 55070 METZ
T´el´ephone : (33) 87 20 35 00 – T´el´ecopie : (33) 87 76 39 77

Un algorithme linéaire pour la génération d'arbres

Résumé : Nous présentons ici un algorithme linéaire pour engendrer aléatoirement et uniformément de nombreuses variétés d'arbres : arbres binaires, arbres ternaires, arbres quelconques, forêts de p arbres k-aires, ... Cet algorithme est basé sur la définition d'arbres génériques qui peuvent être codés facilement sous forme de mots, mots qui peuvent à leur tour être engendrés en temps linéaire.

Mots-clé : génération, uniforme, arbre, grammaire, combinatoire

1 Introduction

The problem of generating ordered trees has been studied extensively in recent literature, motivated by applications to the generation of synthetic pictures and plants. Some problems have been solved in linear time (for example the generation of binary trees by J.L.Rémy [RE]). These methods are hard to extend to other problems.

Here we begin the generation of more complex tree structures with another tool. We generate sequences of letters that are in 1-1 correspondence with certain classes of trees. This is a classical approach (cf. [SC]). We will code here a class of objects defined by Dershowitz and Zaks [DZ2] to determine the number of certain kinds of trees. We are thus able to present an algorithm which generates forests of trees decomposed into patterns. This class is very large and contains the trees with n vertices and l leaves, the forests of k-ary trees, the trees with n inner vertices, an arity larger than k, and l leaves, etc...

This algorithm uses very simple principles. First, we code the class of trees that we want to generate by words of a language $\mathcal{E}$, to be defined later. Then we generate a word of this language randomly and with uniform probability in three steps: "mixing of patterns", "introduction of missed edges", and "application of the cyclic lemma". Finally, we relate the word to the desired tree structure by an appropriate transformation.

The definition of forests of trees decomposed into patterns is given in Section 2. Section 3 shows how to code a forest of trees by a word of $\mathcal{E}$. In Section 4 we explain how to generate such a word in linear time and how to transform it into the corresponding forest of trees decomposed into patterns. Finally, in Section 5, we show how to use this algorithm for generating certain classes of trees: binary trees, k-ary trees, ... At this point we should mention that we want to generate uniformly one tree of the corresponding family (not all trees).

2 Forests of trees decomposed into patterns

In this section we use the notion of forests of trees decomposed into patterns defined by N.Dershowitz and S.Zaks[DZ1]. Each forest is defined as a combination of the structures $\langle F, \mathcal{M}, f\rangle$. The first, F, is a forest of trees; this term is understood as a list of trees. The second, $\mathcal{M}$, presents a list of patterns:

tree structures composed of four symbols, namely vertices and three kinds of edges. The last, f, is a mapping that sends the components of the patterns in $\mathcal{M}$ to a vertex, an edge or a set of edges of F.

First, we give the definition of the patterns, then we explain the conditions on $\langle F, \mathcal{M}, f \rangle$ required for a forest of trees decomposed into patterns.

2.1 Definition of patterns

We define four symbols which we will call *components*. These symbols are then used for the definition of a pattern.

Definition 1 *These components are*

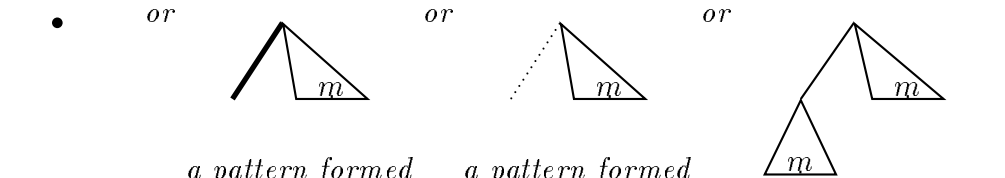
- *the vertices represented by* $\bullet$,
- *the classical-edges represented by* $|$,
- *the semi-edges represented by* $\vdots$,
- *the multi-edges represented by* $\mathbf{|}$.

We continue by defining the patterns:

Definition 2 *A pattern is an isolated vertex or a vertex r that has as child a list of semi-edges, multi-edges and patterns. In case a vertex r has a pattern M as a child, there exists a classical-edge which connects r to the root of M; this classical-edge will also be considered as a child of r.*

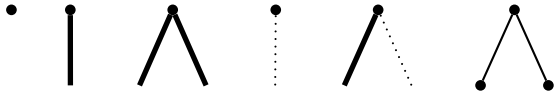
This definition can also be expressed diagrammatically:

Definition 3 *A pattern is*

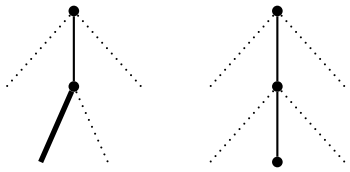


a vertex — *a pattern formed by adding a semi-edge to the root of the pattern* — *a pattern formed by adding a multi-edge to the root of the pattern* — *a pattern formed by connecting two patterns by a classical-edge*

We give several examples of patterns:
Some simple patterns are:



Two, more complicated, patterns:



Remark :
The two first symbols (vertex and classical-edge) are the same as the items which permit the construction of normal trees: we will see later that these symbols will be in correspondence with the vertices and edges of the forest F, in a forest decomposed into patterns $\langle F, \mathcal{M}, f\rangle$. The semi-edges will also be mapped to edges of F; they *differ* from the classical-edges by the fact that they are connected to only one vertex. A multi-edge will correspond to a sequence of consecutive edges of arbitrary cardinality, including zero.

2.2 Definition of cutting

Definition 4 *We say that the function f cuts the pattern M into a forest of trees F if and only if*

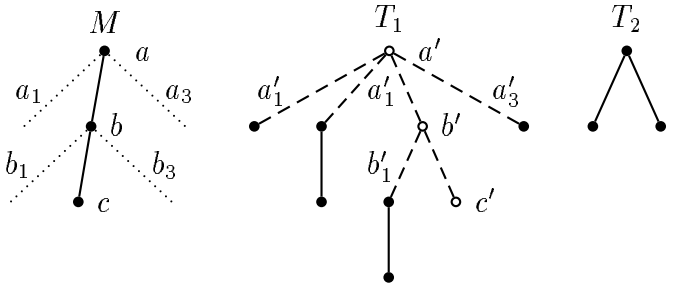
- *f associates*
  - *to each vertex of M, a vertex of F,*
  - *to each classical-edge or semi-edge of M, an edge of F,*
  - *to each multi-edge of M, a set of edges of F that have the same parent v and that are consecutive children of v. This set can be of arbitrary cardinality, including zero.*

- *f preserves the "parent-child" relation. This means that if v and w are two items of M and if v is the parent of w in M, then $f(v)$ is the parent of $f(w)$ in F.*

- *f preserves the ordering relations "left-right" order, i.e. if a vertex v of M has for edges (classical-edges, semi-edges or multi-edges) v_1, v_2, ..., v_p as children ordered from left to right, then the vertex $f(v)$ of F has the edges $f(v_1)$, ..., $f(v_p)$ from left to right, and it has* no other *edges.*

We will call $\langle F, M, f\rangle$ a forest split by the pattern M if and only if f cuts the pattern M into F.

We give now some examples: Let M and $F = \langle T_1, T_2\rangle$ be defined by



(The letters a, a_1, a_3, b, ..., c' are used to name the elements of M and F; they are not part of M or F. Certain edges and vertices of F are represented by circles or dashed lines to indicate that they belong to the set of images of the mapping f.) The mapping f that maps:

- the vertices a, b, c of M to a', b', c',

- the classical-edge that connects a to b to the edge a' b', and the edge which connects b to c to the edge $b'c'$,

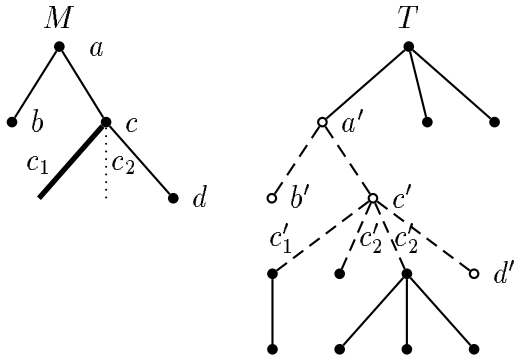
- the multi-edge b_3 to the empty set,

- the multi-edges b_1 and a_3 to the sets consisting of the edge b'_1 and of the edge a'_3, respectively,

- the multi-edge a_1 to the set consisting of the two edges denoted a'_1

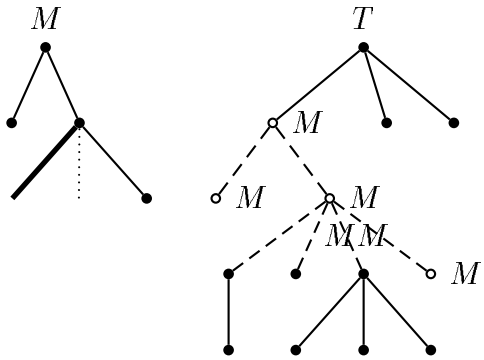
cuts the pattern M into the forest F.

The following is also a forest split by a pattern M when $F = \langle T \rangle$ with



if f is the function which maps a, b, c, c_1, c_2, d to a', b', c', c'_1, c'_2, d', respectively and the classical-edges ab, ac, cd to $a'b'$, $a'c'$ and $c'd'$.

Another way of representing a forest split by a pattern $\langle F, M, f \rangle$ is sometimes helpful. To indicate the mapping f, we mark in F the vertices and edges of F which correspond to the vertices and multi-edges of the initial pattern. The previous example can also be shown as follows:



We will use this representation in the future if it defines the function f unambiguously.

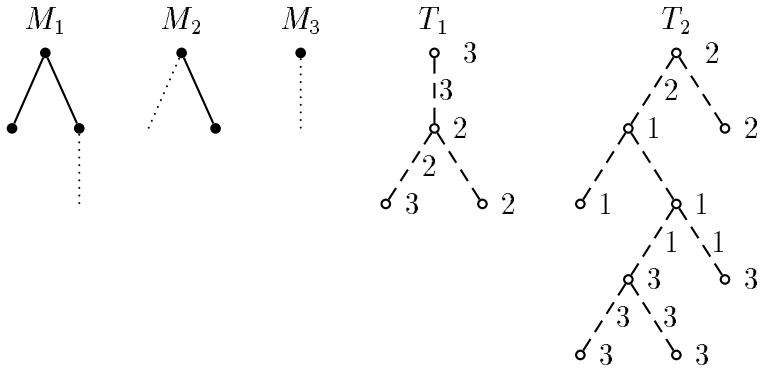
2.3 Forests of trees decomposed into patterns

We will now define what it means for a forest to be decomposed into patterns.

Definition 5 *Let M_1, ..., M_k be k patterns, a_1, a_2, ..., a_k a set of non-negative integers and $\mathcal{M} = \{a_1 \times M_1, \cdots, a_k \times M_k\}$ the multiset formed by the set of patterns M_i whose multiplicity is a_i. We call $\langle F, \mathcal{M}, f\rangle$ a forest of trees decomposed into patterns if and only if*

- *$\forall i \in [\![1, k]\!]$, the restriction of f to the components of the patterns found in $a_i \times M_i$ cuts one by one the a_i patterns M_i into F,*
- *f is a bijective function (i.e. f maps two distinct symbols of $\mathcal{M}$ to two distinct elements (or groups of elements) of T and the image set of f is the set of elements of F).*

Example :
Consider M_1, M_2, M_3, $F = \langle T_1, T_2\rangle$:



$\langle F, \mathcal{M}, f\rangle$ is a forest decomposed into patterns if we take $\mathcal{M} = \{1 \times M_1, 2 \times M_2, 6 \times M_3\}$ and indicate by 1, 2 and 3 the vertices and edges of F which correspond to the vertices and the multi-edges of M_1, M_2 and M_3.

The forests of trees decomposed into patterns have the advantage that they can be enumerated easily (cf [DZ1]). We will see that if we denote by:

- n the number of vertices in $\mathcal{M}$,
- e the number of classical-edges in $\mathcal{M} = \{a_1 \times M_1, \cdots, a_k \times M_k\}$,
- c the number of semi-edges of $\mathcal{M}$,
- d the number of multi-edges of $\mathcal{M}$,

- $s = \sum_{j=1}^{k} a_j$,

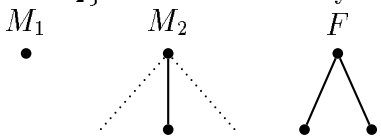
the number of forests decomposed into patterns $\langle F, \mathcal{M}, f\rangle$ such that the forest F has p trees is equal to:

$$\frac{p}{s}\left(\begin{array}{c} s \\ a_1, \ldots, a_k \end{array}\right)\left(\begin{array}{c} n+d-p-e-c-1 \\ d-1 \end{array}\right).$$

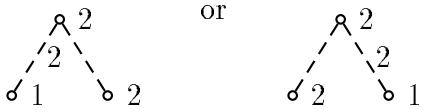
Further on we will see how a forest decomposed into patterns can be generated in linear time.

Remark :
Sometimes when we get a forest F and a multiset of patterns $\mathcal{M}$ there exists a unique mapping f satisfying the conditions of Definition 5. This property can be very useful for generating the more common types of forests. Indeed suppose that we have a multiset $\mathcal{M}$ and that for all forests decomposed into patterns $\langle F, \mathcal{M}, f\rangle$ such that F has p trees, the mapping f is uniquely defined by F and $\mathcal{M}$. In this case we have a 1-1 correspondence between the forests decomposed into patterns $\langle F, \mathcal{M}, f\rangle$ and the forests F. Therefore, if we have an algorithm that builds a forest decomposed into patterns, we will have an algorithm that builds most forests of interest.

However in some cases we have several possible mappings. For instance, consider $\mathcal{M} = \{1 \times M_1, 1 \times M_2\}$ and F defined by



In this case, we have two possible forests decomposed into patterns



3 Coding of forests of trees decomposed into patterns as words

First, we define two languages:

- the language $\mathcal{A}$ consists of the words formed with the symbols x, y, o, f,
- the language $\mathcal{C}$ consists of the words formed with the symbols x, y, (,), [,].

Next we define a language $\mathcal{B}$ that is a subset of $\mathcal{A}$. $\mathcal{B}$ will allow us to code the patterns.

There are also two subsets of $\mathcal{C}$ called $\mathcal{D}$ and $\mathcal{E}$ which are in 1-1 correspondence with trees decomposed into patterns and with forests decomposed into patterns, respectively.

We define a mapping sending each pattern to a word of the language $\mathcal{B}$. Then we see that this mapping yields a bijection from the set of forests of trees decomposed into patterns to a subset of $\mathcal{C}$: the language $\mathcal{E}$.

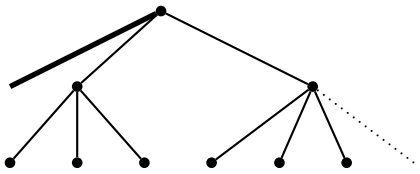
3.1 Coding of a pattern by a word

A mapping t is defined recursively by the following rules:

- a classical-edge is mapped to y,
- a semi-edge is mapped to f,
- a multi-edge is mapped to o,
- a vertex v that has from left to right $(a_1, a_2, \ldots, a_p)$ as classical-edges, semi-edges or multi-edges and $(v_1, \ldots, v_{p'})$ as subtrees such that the roots of the trees v_1, ..., $v_{p'}$ are children of v, is mapped by t to:

$$t(v_1)t(v_2)\cdots t(v_{p'})x\ t(a_p)\cdots t(a_1).$$

We obtain, for example, for the pattern M_1:



the word:

$$t(M_1) = xxxxyyy\,xxxxoyyy\,xyyf.$$

Remark :
If the pattern does not contain any multi-edge or semi-edge, we find the classical coding of a tree in the form of a sequence of the letters x, y.

Definition 6 *We denote by $\mathcal{B}$ the language defined by the grammar*

$$\mathcal{B} = x + \mathcal{B}f + \mathcal{B}o + \mathcal{B}\mathcal{B}y.$$

First we have two propositions

Proposition 1 *If $w \in \mathcal{B}$, then $|w|_x = |w|_y + 1$.*

Proof. This proposition is shown by induction using the definition of $\mathcal{B}$. □

This leads to the following proposition:

Proposition 2 *If we remove the letters o and f from a word w of $\mathcal{B}$, we obtain a 1-dominated sequence (i.e. a letter x followed by a Dyck prefix) having one more x than y's.*

Proof. We proceed again by induction on the size of the word w. Using the definition of $\mathcal{B}$, we show that, if the letters f and o are removed from a word w, we get a 1-dominated sequence. Then we use Proposition 1 to conclude the result. □

Using the recursive definition of the patterns (Definition 3), it is easy to see that the mapping t maps all patterns to words of $\mathcal{B}$. We give now two propositions for reconstructing the pattern M given a word $w = t(M)$ of $\mathcal{B}$.

Proposition 3 *Let M be a pattern and $w = t(M)$. If the last occurrence of the letter x in the word w is followed by the symbols $y_1 y_2 \dots y_p$, (each being a y, f or o), then*

- *the root r of M has p edges as children (classical-edges, semi-edges or multi-edges),*

- *the i^{th} edge (classical-edge, multi-edgeand semi-edge) from the right child of r is obtained by replacing the letter y_i by a classical-edge, a multi-edge or a semi-edge if y_i equals y, o or f, respectively.*

Proof. This proposition follows directly from the definition of t. □

Now we reconstruct from a word $w = t(M)$ the edges $(a_1, \ldots, a_p)$ that are children of the root of M. Next, we want to find the subtrees that correspond to the classical-edges a_i among the collection of edges a_1, ..., a_p.

We define the height of a letter l in the sequence $w = t(M)$ as the difference of the number of letters x and the number of letters y that are located before l in w (the letter l inclusive).

We have the following proposition.

Proposition 4 *Let $w = t(M)$ and denote x'_i the last letter x of w that has height i. The subtree corresponding to the j^{th} classical-edge of the root of M is coded between the letter x'_j inclusive and the letter x'_{j+1} exclusive.*

Proof. This proposition follows from Proposition 2. □

We prove now that:

Theorem 1 *The mapping t defines a coding of patterns by words of the language $\mathcal{B}$.*

Proof. Propositions 3 and 4 allow us to construct a recursively defined mapping that maps each word w of $\mathcal{B}$ to a pattern M such that $t(M) = w$. □

3.2 Coding of a tree decomposed into patterns by a word

We define now a coding b of the trees decomposed into patterns, then we will extend this coding to forests of trees with patterns.

Let $\mathcal{M} = \{a_1 \times M_1, \ldots, a_k \times M_k\}$ be a multiset, T be a tree, and f be a function whose restriction to the components of a pattern $M = M_i$ of $\mathcal{M}$ splits this pattern into T. We define a mapping t' which depends on four arguments: a tree T, a pattern M, a mapping f.

Definition 7 *We denote by t' the function of the variables T, M, f where $t'(T, M, f)$ is a word of the language $\mathcal{C}$ determined by*

- *calculating $t(M)$,*
- *then replacing in $t(M)$ all the letters o that correspond to a multi-edge v of M by $card\,(f(v))$ letters f surrounded by the symbols " (" and ") ",*
- *and finally surrounding the obtained word by the symbols " [" and "] ".*

Noting by "." the concatenation of two sequences, we can define the function b recursively on the trees decomposed into patterns as follows:

Definition 8 *Let $\langle T, \mathcal{M}, f\rangle$ be a tree decomposed into patterns and denote by i the number of patterns such that f maps the roots of the patterns M_i to the root of T. Then*

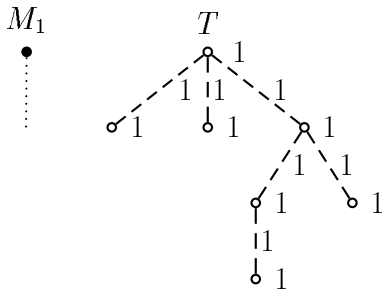
- *If T is a leaf, $b(\langle T, \mathcal{M}, f\rangle) = t'(T, M_i, f)$.*
- *Otherwise, we divide the tree T in two parts: the vertices and edges of T that have as inverse image a vertex, a classical-edge, a semi-edge or a multi-edge of the same pattern M_i, and the other vertices and edges of T. This gives a set of vertices and edges T' and a set of trees that can be ordered according to the appearance of their roots in T: T'_1, ..., T'_p in postfix traversal. Therefore we have:*

$$b(\langle T, \mathcal{M}, f\rangle) = b(\langle T'_1, \mathcal{N}_1, f_1\rangle).\cdots.b(\left\langle T'_p, \mathcal{N}_p, f_p\right\rangle).t'(T, M_i, f)$$

where $\mathcal{N}_j$ is the multiset $\{b_{1,j} \times M_1, \ldots, b_{k,j} \times M_k\}$ with $b_{e,j}$ the number of patterns M_e into which f splits T_j and f_j is the restriction of the function f to the components that form the patterns of $\mathcal{N}_j$.

Examples :

We start with a simple example, let M_1 be a pattern, T be a tree:

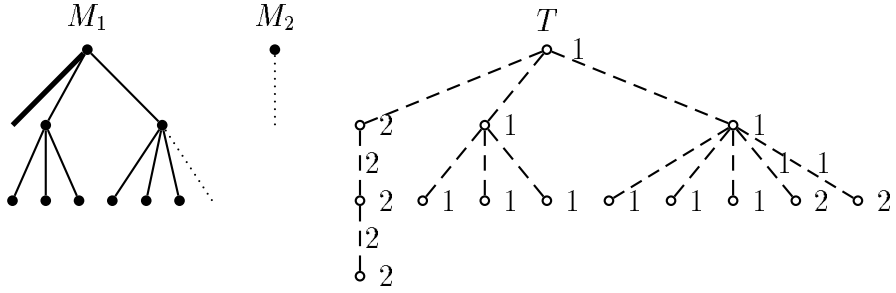


$\mathcal{M} = \{7 \times M_1\}$ and f be the mapping which maps seven times the root of M_1 to a node of T. Then

$$b(\langle T, \mathcal{M}, f\rangle) = [x()][x()][x()][x(f)][x()][x(ff)][x(fff)].$$

We meet here the classical representation of a tree coded by a sequence in postfix order when we remove the letters [,], (and) and when we replace the letters f by y's.

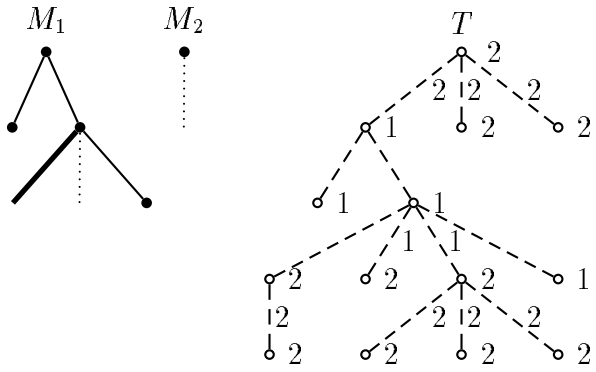
Taking the pattern M_1 and M_2 as in the preceding section, and T and f such that:



where $\mathcal{M} = \{1 \times M_1, 5 \times M_2\}$, we get:

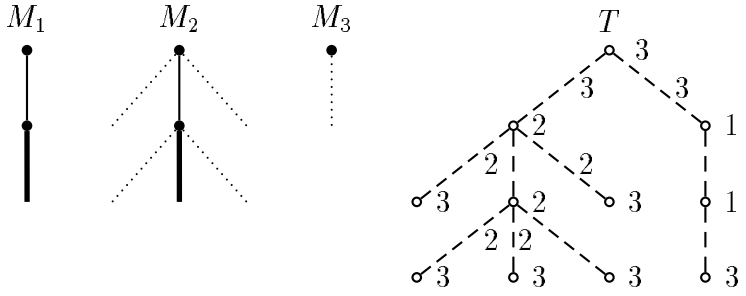
$$b(\langle T, \mathcal{M}, f\rangle) = [x()][x(f)][x(f)][x()][x()][xxxxyyyxxxx(ff)yyyxyyf].$$

With $\mathcal{M} = \{1 \times M_1, 10 \times M_2\}$, T, and f defined by:



$$b(\langle T, \mathcal{M}, f\rangle) = [x()][x(f)][x()][x()][x()][x()][x(fff)] \\ [xxxy(ff)fxyy][x()][x()][x(fff)].$$

Finally, we look at a more complicated example with several patterns. Let us take the tree decomposed into patterns $\langle T, \mathcal{M}, f\rangle$ with $\mathcal{M} = \{1 \times M_1, 1 \times M_2, 7 \times M_3\}$ and M_1, M_2, M_3, T, f defined by:



For this tree we get:

$$b(\langle T, \mathcal{M}, f\rangle) = [x()][x()][x()][x()][x()][x()f(ff)x(f)f(f)][x()][xfxy][x(ff)].$$

We can now define the language $\mathcal{D}$:

Definition 9 *The language $\mathcal{D}$ is equal to the image of the set of trees decomposed into patterns under the mapping b.*

Obviously, the language $\mathcal{D}$ is a subset of the language $\mathcal{C}$. We will show that b defines a $1-1$ correspondence between the trees decomposed into patterns and the words of $\mathcal{D}$.

Definition 10 *We say that a word of $\mathcal{C}$ satisfies the property of the patterns if:*

- *the numbers of letters " [" and "]" contained in the sequence are equal,*
- *the i^{th} " [" comes before the i^{th} "]",*
- *the word obtained by*
  - *taking the letters between the i^{th} " [" and the i^{th} "]",*
  - *and replacing all the edges f surrounded by the symbols (and) with the letter o,*

  corresponds to a pattern M_j,
- *there are no symbols appearing outside the representation of the patterns.*

Proposition 5 *All words of $\mathcal{D}$ satisfy the property of the patterns.*

Proof. This proposition follows directly from the constructive definition of the language $\mathcal{D}$. We only introduce letters " [" and "]" when we want to code a pattern; in this case, we replace in $t(M_i)$ all o's that represent a multi-edge v by a word formed by $card\,(f(v))$ letters f surrounded by the two letters (and). □

We define now a projection mapping Φ that maps a word of the language $\mathcal{C}$ to a sequence of the letters x and y:

- $\Phi(u.v) = \Phi(u).\Phi(v)$ if u and v are words of the language $\mathcal{C}$,
- $\Phi(u) = \epsilon$: the empty word, if u is one of the following symbols (,), [or],
- $\Phi(x) = x$,

- $\Phi(y) = \Phi(f) = y$.

We will say that

Definition 11
- *A sequence s satisfies the* dominance property *if and only if $\Phi(s)$ is a 1-dominated sequence (i.e., $\Phi(s)$ is a word formed by a letter x followed by a left factor of a Dyck word),*
- *A sequence s satisfies the* strict dominance property *if and only if s satisfies the dominance property and if $\Phi(s)$ has one letter x more than y's (i.e. $\Phi(s)$ is formed by a letter x followed by a Dyck word)*

Proposition 6 *All words of $\mathcal{D}$ satisfy the strict dominance property.*

Proof. The proof is by induction on n, the number of vertices of T.

For $n = 1$, the proposition is true, since we must have $\mathcal{M} = \{1 \times M_1\}$ where $M_1 =$ • or M_1 is a pattern formed by a node whose children are multi-edges. This gives the sequences $[x]$, $[x()]$, $[x()()]$, ...

We suppose now that the proposition is true for all trees decomposed into patterns $\langle T, \mathcal{M}, f\rangle$ such that $_{\#}T < N$ ($_{\#}$ means the cardinality) and take a tree decomposed into patterns $\langle T, \mathcal{M}, f\rangle$ such that $_{\#}T = N$. We denote by i the index of the pattern that has the inverse image of the root of T as root.

Now as in Definition 8 of b, we let $T'_1, \ldots, T'_p$ be the subtrees formed by the components which do not belong to the image set of M_i and T' the image under f of M_i's components and we define the multisets $\mathcal{N}_1, \ldots, \mathcal{N}_k$ and the mappings $f_1, \ldots, f_k$. It follows from the induction hypothesis that $\Phi(b(\langle T'_1, \mathcal{N}_1, f_1\rangle)), \ldots, \Phi(b(\left\langle T'_p, \mathcal{N}_p, f_p\right\rangle))$ are p 1-dominated sequences, that have each a single letter x more than y's.

The sequence $\Phi(b(\langle T'_1, \mathcal{N}_1, f_1\rangle)).\ldots.\Phi(b(\left\langle T'_p, \mathcal{N}_p, f_p\right\rangle))$ is therefore a 1-dominated sequence that has p letters x more than y's. We saw in the previous section that the letters x's and y's contained in $t(M_i)$ also form a 1-dominated word. Remarking that there are exactly p letters f in $t'(T, M_i, f)$, finishes the proof. □

We denote by $\mathcal{D}'$ the language that contains the words w of the language $\mathcal{C}$ which satisfy the pattern and the strict dominance properties.

We prove now the following theorem:

Theorem 2 *If w is a word of $\mathcal{D}'$, then there exists a tree decomposed into patterns $\langle T, \mathcal{M}, f\rangle$ such that $b(\langle T, \mathcal{M}, f\rangle) = w$.*

Proof. Let us take a word w of $\mathcal{D}'$. We first extract from w the patterns M_1, M_2, ..., M_k that are encoded. This is possible in a unique way because w satisfies the property of the patterns.

The words $[x]$, $[x()]$, $[x()()]$, ... are the only words of $\mathcal{D}'$ with one letter x. So if w has one letter x, we get $w = b(\langle T, \mathcal{M}, f\rangle)$ using a leaf for T and $\mathcal{M} = \{1 \times M_1\}$ where M_1 is a pattern formed by a root that has as many multi-edges as children as w has letters '(', and f the function which maps the root of the pattern M_1 to the root of T. The theorem is therefore true for all words of $\mathcal{D}'$ with one letter x.

Assume that the theorem is true for all words of $\mathcal{D}'$ that have less than N letters x and that w is a word of $\mathcal{D}'$ with N letters x. We define a height h for each symbol r of w. This height is given by the difference of the number of letters x, and the number of letters y, f that are located before the symbol r (inclusive). We call $[_l$ and $]_l$ the last symbols [and] that have height l.

First we search for the letter l' (the letter " [") corresponding to the beginning of this pattern. Then we split w in two parts:

- w_1 the symbols located before l' (l' exclusive),
- w_2 the symbols located after l' (l' inclusive).

Let p be the number of letters f in w_2. Then the first letter of w_2 has height $p + 1$. This allows us to decompose the word w_1 in p words of $\mathcal{D}'$: $w_1 = m_1.m_2 \cdots .m_p$ where the words m_l are formed by the letters between $[_{l-1}$ (inclusive) and $]_l$ (inclusive).

We apply now the induction hypothesis on the words $m_1, \ldots, m_p$, which are words of $\mathcal{D}'$, to obtain p trees decomposed into patterns $\langle T'_1, \mathcal{N}_1, f_1\rangle, \ldots, \left\langle T'_p, \mathcal{N}_p, f_p\right\rangle$ such that $b(\langle T'_1, \mathcal{N}_1, f_1\rangle) = m_1, \ldots, b(\left\langle T'_p, \mathcal{N}_p, f_p\right\rangle) = m_p$.

To obtain a tree decomposed into patterns $\langle T, \mathcal{M}, f\rangle$ that is an inverse image of w under b, we only have to map the word w_2 to a pattern M_i and then to connect the p edges that correspond to a letter f to the trees T'_1, ..., T'_p. This gives a tree T. The multiset $\mathcal{M}$ is obtained by adding a pattern M_i to the multiset formed by concatenating the multisets $\mathcal{N}_1$, ..., $\mathcal{N}_p$. The function

f is the function which acts on the components of $\mathcal{N}_j$ as did f_j for each j in $[\![1, p]\!]$, and maps correctly the components of M_i that were just added. □

Corollary 1 *The words of $\mathcal{D}$ are the words of $\mathcal{C}$ which satisfy the property of the patterns and the strict dominance property.*

Proof. Propositions 5 and 6 prove that all words of $\mathcal{D}$ satisfy the strict dominance property and pattern property.

Theorem 2 proves that for each word w of $\mathcal{C}$ which satisfies the pattern property and the strict dominance property, there exists a tree decomposed into patterns such that $b(\langle T, \mathcal{M}, f\rangle) = w$. To prove it, we define a mapping b^{-1} which maps such a word w to a tree decomposed into patterns $\langle T, \mathcal{M}, f\rangle$ such that $b(\langle T, \mathcal{M}, f\rangle) = w$. This function is such that for each tree decomposed into patterns $\langle T, \mathcal{M}, f\rangle$, we get $b^{-1}(b(\langle T, \mathcal{M}, f\rangle)) = \langle T, \mathcal{M}, f\rangle$. This proves that b is bijective. □

3.3 Coding a forest of trees decomposed into patterns by a word

We will now extend the coding of trees decomposed into patterns to forests of trees decomposed into patterns.

Definition 12 *With the language $\mathcal{D}$ we can define the language $\mathcal{E}$ as the set of words of $\mathcal{D}^\star$ which satisfy the property of patterns (i.e., any word of $\mathcal{E}$ can be obtained by concatenating a finite number of words belonging to $\mathcal{D}$ and it satisfies the property of patterns).*

Now we can easily code a forest of trees decomposed into patterns $\langle F, \mathcal{M}, f\rangle$ into a word of the language $\mathcal{E}$. We denote

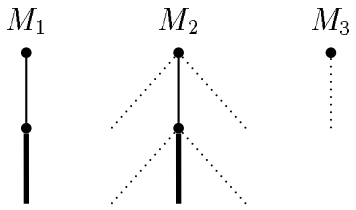
- $T_1, \ldots, T_p$ the list of trees which defines the forest F,
- $\mathcal{N}_i$ the multiset formed by the patterns of $\mathcal{M}$ that are mapped by f to the elements of T_i,

- f_i the restriction of f to the basic items of $\mathcal{N}_i$.

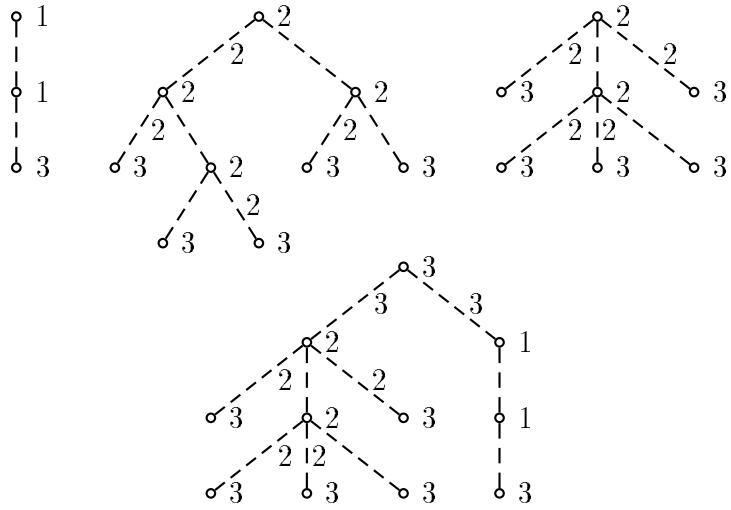
Then it is sufficient, in fact, to return the word

$$b'(\langle F, \mathcal{M}, f\rangle) = b(\langle T_1, \mathcal{N}_1, f_1\rangle).\cdots.b(\langle T_p, \mathcal{N}_p, f_p\rangle).$$

For example, let $\mathcal{M} = \{2 \times M_1, 4 \times M_2, 18 \times M_3\}$ with:



and F and f



we get therefore

$$\begin{aligned} b'(\langle F, \mathcal{M}, f\rangle) = {} & [x()][xfxy] \\ & .[x()][x()][x()][x(f)f()x()f(f)][x()][x()][x()f(f)x()f(f)] \\ & .[x()][x()][x()][x()][x()][x()f(ff)x(f)f(f)] \\ & .[x()][x()][x()][x()][x()][x()f(ff)x(f)f(f)][x()][xfxy][x(ff)]. \end{aligned}$$

Using Proposition 6 it is easy to see that we obtain a coding of forests of trees decomposed into patterns by words of $\mathcal{E}$.

4 The generation algorithm

We show now how a word of $\mathcal{E}$ corresponding to a forest of p trees whose patterns are $\mathcal{M} = \{a_1 \times M_1, \cdots, a_k \times M_k\}$ can be generated randomly and with uniform distribution.

We proceed in four steps: we begin by mixing the patterns, then we add a certain number of edges, apply the cycle lemma [DZ2] in order to get a word of $\mathcal{E}$, and finally we build the forest decomposed into patterns that corresponds to this word. All these steps have a complexity in $O(n)$ where n is the number of nodes which are in $\mathcal{M}$.

4.1 Mixing the patterns

Let n be the number of vertices in our patterns.

We mix a_1 patterns M_1, ..., a_k patterns M_k, this gives a number of possibilities equal to

$$\left(\begin{array}{c} s \\ a_1, a_2, \ldots, a_k \end{array} \right)$$

with $s = \sum_{j=1}^{k} a_j$. This can be easily done with the following algorithm:

```
position = 1 // this variable helps fill the array list
for j = 1 to k
        for i = 1 to a_j
                list [position] = j
                position = position+1

// the array list contains now all non-mixed patterns

for i = position − 1 to 1 (in decreasing order)
        N = 1+random (i) // a random number of [[ 1, i ]]
        output[i] = list[N]
        list[N] = list[i]
// the array "output" contains the desired mixture of the patterns
```

After the mixing has been done, we replace the patterns M_i by their representations as words of $\mathcal{C}$ and their letters o by the word (). Thus we obtain a word of $\mathcal{C}$ which satisfies the property of patterns.

The algorithm mixes the s patterns uniformly (see [VIT] for a proof of the validity of this algorithm).

4.2 Insertion of the missed edges

In this step we add certain edges to the sequence w just obtained. We introduce the following variables:

- n the number of vertices in $\mathcal{M}$,
- e the number of classical-edges in $\mathcal{M}$,
- c the number of semi-edges in $\mathcal{M}$,
- d the number of multi-edges in $\mathcal{M}$.

The forest that we wish to obtain has p trees. Therefore the corresponding word of $\mathcal{E}$ has to have $n-p$ symbols y and f. The sequence w has already $e+c$ of these symbols. We consider two cases:

If $n-p < e+c$, then we stop. There is no forest decomposed into patterns $\langle F, \mathcal{M}, f\rangle$ such that F is a forest of p trees that has n vertices.

If $n-p \geq e+c$, then we add $n-p-e-c$ symbols y or f to the sequence w. But we can only add some letters f, and since these letters f can be only inserted between two letters " (" and ")" which correspond to a multi-edge, we have d possible places.

We distribute the $n-p-e-c$ missed letters f on the d places with repetitions allowed. Therefore, we get:

$$A = \left(\begin{array}{c} s \\ a_1, a_2, \ldots, a_k \end{array} \right) \left(\begin{array}{c} n+d-p-e-c-1 \\ d-1 \end{array} \right)$$

different words of the language $\mathcal{C}$ (recall that $s = \sum_{j=1}^{k} a_j$).

This can be done linearly with the following algorithm:

pos = 1
number-edges $= n - p - e - c$ *// number of edges which remain to be placed*
while number_ edges ≥ 1
    if random $(d - pos + number\_edges) \geq number\_edges$
      pos = pos + 1
    else
      place a supplementary letter f in the pos^{th} *place*
      number_ edges = number_ edges - 1

Theorem 3 *The sequences thus produced are words of* $\mathcal{C}$ *that*

- *are composed of* n *letters* x *and* $n - p$ *letters* y *and* f,
- *satisfy the property of patterns,*
- *contain the patterns of the multiset* $\mathcal{M}$,
- *and begin with the representation of one of the patterns of* $\mathcal{M}$.

The algorithm generates each of the sequences that satisfy these properties with probability $\frac{1}{A}$.

Proof. We start with a sequence that has n letters x and $e + c$ letters y and f and add $n - p - e - c$ letters f. Therefore the resulting sequence has n letters x and $n - p$ letters y and f.

The sequence is formed by concatenating the s patterns of $\mathcal{M}$, then by adding f letters between the letters () in the positions that correspond to multi-edges. Therefore the sequence can only

- begin with a symbol that represents the beginning of a pattern,
- satisfy the property of patterns,
- contain the subsequences which code the s patterns of $\mathcal{M}$.

Since we just showed that the algorithm generates A sequences with uniform probability, we only have to verify that it does not leave out any sequences. We take therefore a sequence that satisfies the properties of Theorem 3: it has subsequences that represent the a_1 patterns M_1, ..., a_k patterns M_k. Also it has n letters x and satisfies the property of patterns. This sequence can therefore be produced by first mixing the s patterns of $\mathcal{M}$, then by adding $n-p-e-c$ letters f in the d places that correspond to the d multi-edges of $\mathcal{M}$. So it can be constructed with our algorithm. □

4.3 Application of the Cycle Lemma

The sequence w that we just constructed in the previous section satisfies all properties of the words of $\mathcal{E}$ except for one: the dominance property.

We will now show how a sequence with n letters x, and $n-p$ letters y, f can be transformed into a word of $\mathcal{E}$.

For this, we recall the definition of cyclic permutations:

Definition 13 *A cyclic permutation is a mapping which maps a word v formed by the symbols a_1, a_2, ..., a_{r-1}, a_r, ..., $a_{|v|}$ to a word $v' = a_r \cdots a_{|v|} a_1 \cdots a_{r-1}$, where r is an integer from $[\![1, |v|]\!]$ (i.e. the i^{th} symbol of v' is equal to the $((i+r-2) mod\, |v|+1)^{th}$ symbol of v).*

First, we prove the following lemma often called the Cycle Lemma

Lemma 1 *There exist p cyclic permutations that transform a sequence of n letters x and $n-p$ letters y into a 1-dominated sequence.*

Proof. This lemma appeared first in [DM]. Since then many proofs have been found (see also [DZ2] for the history). We will give only one here: the proof of [DZ2] which was inspired by the paper of Silberger [SI].

It consists in showing that removing two consecutive letters, x and y, does not change the number of cyclic permutations that we are looking for. By repeated application of this procedure, we get a sequence that consists of p letters x. This sequence can be transformed into a 1-dominated sequence (i.e., one x followed by a left Dyck factor) by p cyclic permutations. □

We have then the following theorem

Theorem 4 *There exist p cyclic permutations that transform the sequence w into a word of $\mathcal{E}$.*

Proof. Consider the given sequence w and denote $u = \Phi(w)$ (where Φ is the mapping defined in the previous paragraph that projects the letters x to the letter x and the letters y and f to y). Theorem 3 implies that the sequence u consists of n letters x and $n-p$ letters y. We apply Lemma 1 to show that there exist p cyclic permutations which transform u into a 1-dominated sequence.

Now we look for cyclic permutations that transform w into a sequence that satisfies the dominance property and begins with a letter [. Let t be one of these permutations. The word $\Phi(t(w))$ is then a 1-dominated sequence that can be obtained by a cyclic permutation t_1 acting on the word u. If we require also that the second letter of $t(w)$ (a letter x) corresponds via Φ to the first letter of $t_1(u)$, then the cyclic permutation t_1 is uniquely determined by the choice of t.

We can also show that the choice of a cyclic permutation t_1 determines uniquely a cyclic permutation t that transforms the word w into a word $t(w)$ which satisfies the dominance property. We have therefore as many cyclic permutations that transform w into a word satisfying the dominance property as we have cyclic permutations that transform u into a 1-dominated word, i.e., p.

It only remains to be seen that these p cyclic permutations indeed give words of $\mathcal{E}$. We denote by w' a word of $\mathcal{C}$ obtained by applying to w one of the p cyclic permutations. The sequence w' satisfies the dominance property. We still have to check that w' satisfies the property of patterns and that the patterns contained in w' correspond to the list $\mathcal{M}$. We use Proposition 2 for this. This proposition shows that the representation of a pattern in the form of a word of $\mathcal{C}$ satisfies the property of strict dominance. Thus, the representation of a pattern can not be cut in two by the transformation of w into w'. □

The third step of our algorithm is to choose randomly one of the p cyclic permutations that transform w into a word of $\mathcal{E}$ (with probability $\frac{1}{p}$). Then this transformation is applied to w. This gives:

// we look first for the minimal position of the sequence
pos = 1
height = 0

```
min = 0
pos_min = 1
run through the sequence from left to right
    if the letter read is x
        if height ≤ min
            pos_min = pos
            min = height
        height = height + 1
    if the letter read is y, f
        height = height-1
    pos = pos+1
// the variable pos now points to the last letter x of minimal height
// and min indicates this height
// we choose one of the p possible cyclic permutations,
height_chosen = min+random(p)
// find the first letter of the new sequence
pos = 1
height = 0
run through the sequence from left to right
    if the letter read is x
        if height = height_chosen
            beginning = pos
        height = height+1
    if the letter read is y, f
        height = height-1
    pos = pos+1
// we realize the chosen cyclic permutation
pos = beginning - 1
posw = 1
do
    pick the pos^th symbol of w and put it in the posw^th position of the new sequence
    pos = 1 + pos mod |w|
    posw = posw + 1
while pos ≠ beginning
```

Theorem 5 *We obtain each word of $\mathcal{E}$ which corresponds to a forest of p trees $\langle F, \mathcal{M}, f\rangle$ with probability $\frac{1}{A}$, where:*

$$A = \frac{p}{s}\left(\begin{array}{c} s \\ a_1, a_2, \ldots, a_k \end{array}\right)\left(\begin{array}{c} n+d-p-e-c-1 \\ n-p-e-c \end{array}\right).$$

Proof. Here it is sufficient to show that to each word of $\mathcal{E}$ corresponding to a forest of p trees $\langle F, \mathcal{M}, f\rangle$ correspond s sequences which satisfy the properties of Theorem 3 and then to apply Theorem 4.

We take a word w' of $\mathcal{E}$ that corresponds to a forest decomposed into patterns $\langle F, \mathcal{M}, f\rangle$ such that the forest F has p trees. This word has s letters " [" which indicate the beginning of a pattern. We denote these letters by l_1, ..., l_s. Then we define the s cyclic permutations that transform the sequence w' into a sequence that begins with one of the s letters l_1, ..., l_s. These s cyclic permutations transform w' into sequences w that satisfy the properties of Theorem 3. They are also the only cyclic permutations that transform w' into a word whose first symbol corresponds to the beginning of a pattern. □

We recover therefore the results of [DZ1]:

Corollary 2 *The number of forests of $\langle F, \mathcal{M}, f\rangle$, where F is a forest of p trees, is equal to:*

$$A = \frac{p}{s}\left(\begin{array}{c} s \\ a_1, a_2, \ldots, a_k \end{array}\right)\left(\begin{array}{c} n+d-p-e-c-1 \\ d-1 \end{array}\right).$$

Proof. We use the 1-1 mapping between the forests decomposed into patterns and the words of $\mathcal{E}$ and then Theorem 5. □

4.4 Mapping words of $\mathcal{E}$ to forests decomposed into patterns

In this section we present a decoding algorithm for mapping a word w of $\mathcal{E}$ to a forest F that is the first ingredient of the forest decomposed into patterns $\langle F, \mathcal{M}, f\rangle = b^{-1}(w)$. Indeed, this is the unique ingredient of the forest decomposed into patterns that we need when we want to build some common kinds

of forests. This algorithm reads a word of $\mathcal{E}$ and builds the forest F. It is linear and uses two stacks. The first *globalstack* stores the constructed trees which we construct when we read w while the second *patternstack* stores the trees which we construct when we read a pattern.

create two empty stacks globalstack *and* patternstack
read the word of $\mathcal{E}$ *to be transformed from left to right*
    if the next symbol is " [*"*
        read the next symbol
    create a new vertex v
    while the next symbol of the sequence differs from x
        read this symbol
        if the symbol is equal to f
            pop a tree from globalstack *and add it to the children of* v
        if the symbol is equal to y
            pop a tree from patternstack *and add it to the children of* v
    if the next symbol is "]*"*
        read the next symbol
        push the created subtree to globalstack
    else
        push the created subtree to patternstack
in the end globalstack *contains the subtrees* $T_1, \ldots, T_p$ *that form the forest* F.

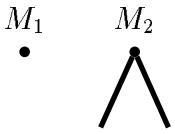
5 Applications

We show now how to generate some common classes of trees with the algorithm designed in the previous sections.

5.1 Generating binary trees of size $2n+1$

Proposition 7 *There exists a* 1-1 *mapping between binary trees with* $2n+1$ *vertices and the trees decomposed into patterns* $\langle \langle T \rangle, \mathcal{M}, f \rangle$ *such that:*

- $\mathcal{M} = \{(n+1) \times M_1, n \times M_2\}$ *with*



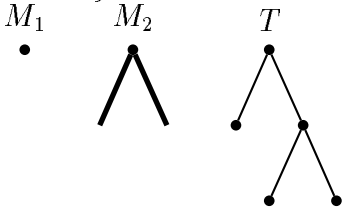
- *T is a tree of size $2n+1$.*

Proof. Let T be a binary tree with $2n+1$ vertices. Then it is easy to see that T has n inner vertices and $n+1$ leaves.

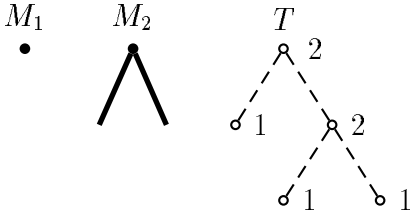
If we try to define a function f that splits the patterns of $\mathcal{M}$ in T, then we have only one choice. We have to associate $n+1$ times a leaf of T to the vertex of the pattern M_1 and n times an inner vertex of T and its children to the components of the patterns M_2.

Example :

If we take $\mathcal{M} = \{3 \times M_1, 2 \times M_2\}$ and T as indicated below:



then there exists only one tree decomposed into patterns $\langle \langle T \rangle, \mathcal{M}, f \rangle$:

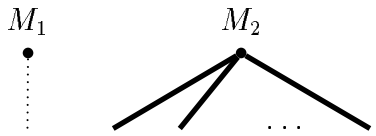


□

We obtain a linear algorithm that generates uniformly a binary tree of size $2n+1$. Indeed, we only have to generate a tree decomposed into patterns $\langle \langle T \rangle, \mathcal{M}, f \rangle$ with $\mathcal{M} = \{(n+1) \times M_1, n \times M_2\}$ and then keep only the tree T.

5.2 Generating forests with p k-ary trees and $kn+p$ vertices

We use the fact that these forests are in 1-1 correspondence with the forests decomposed into patterns $\langle F, \mathcal{M}, f\rangle$ where F is a forest with p trees and $\mathcal{M} = \{((k-1)n+p) \times M_1, n \times M_2\}$ with



Here M_2 stands for a pattern corresponding to a k-ary vertex.

5.3 Generating trees with n vertices

This time we use the pattern list $\mathcal{M} = \{n \times M_1\}$ where M_1 is the pattern representing a vertex of arbitrary arity

5.4 Generating forests of trees with n inner vertices and l leaves

We take $\mathcal{M} = \{n \times M_1, l \times \bullet\}$ with

and for F a forest of p trees.

5.5 Generating forests of p unary-binary-ternary trees with a leaves, b unary vertices, c binary vertices, d ternary vertices

We use, this time, $\mathcal{M} = \{a \times M_1, b \times M_2, c \times M_3, d \times M_4\}$ with

and for F a forest of p trees.

6 Conclusion

We have designed a linear algorithm that randomly generates a forest of trees decomposed into patterns. This algorithm allows us to generate most of the trees of interest: k-ary trees, arbitrary trees, trees with n internal vertices and l leaves, ... All forests with simple, closed enumeration formulae are within the scope of this method. Unfortunately, the generation of a unary-binary tree requires a different approach (see [AL]). Our algorithm can be implemented on parallel computers, and allows then the generation of a word of $\mathcal{E}$ corresponding to a tree split by a pattern with worst case complexity in $O(Log^2(n))$ [AS] . The generation of strings is done in [FZC] with different tools .

7 Acknowledgments

The authors thank N.Dershowitz, P.Feinsilver, C.L.Liu, J.G.Penaud and E.M.Reingold for many helpful suggestions.

References

[AL] L.Alonso, Uniform generation of a Motzkin word, TCS, 134(2), 529-536, 1994.

[AS] L.Alonso, R.Schott, A Parallel Algorithm for the Generation of Permutations, Congrès Gascom, Bordeaux, 1994, to appear in TCS.

[DZ1] N.Dershowitz, S.Zaks, Patterns in Trees, Discrete Applied Math. 25, 241-255, 1989.

[DZ2] N.Dershowitz, S.Zaks, The Cycle Lemma and some applications, Europ. J. of Comb. 11, 35-40, 1990.

[DM] A. Dvoretzky, Th. Motzkin, A problem of arrangements, Duke Math. J., 24, 305-313, 1947.

[FZC] P.Flajolet, P.Zimmermann, B.V.Cutsem, A calculus for the generation of combinatorial structures, TCS, 132, 1-35, 1994.

[RE] J.L.Rémy, Un procédé itératif de dénombrement d'arbres binaires et son application à leur génération aléatoire, R.A.I.R.O. Informatique Théorique, 19, 2, 179-195, 1985.

[SC] M.P.Schützenberger, Context-free languages and pushdown automata, Information and Control, 6, 246-261, 1963.

[SI] D.M.Silberger, Occurrences of the integer $\frac{(2n-2)!}{n!(n-1)!}$, Roczniki Polskiego Towarzystwa Math. I, vol. 13, 91-96, 1969.

[VIT] J.S.Vitter, Optimum algorithms for two random sampling problems, Proc. F.O.C.S.-83, 65-75, 1983.

Unit´e de recherche INRIA Lorraine, Technop^ole de Nancy-Brabois, Campus scientifique,
615 rue du Jardin Botanique, BP 101, 54600 VILLERS LÈS NANCY
Unit´e de recherche INRIA Rennes, Irisa, Campus universitaire de Beaulieu, 35042 RENNES Cedex
Unit´e de recherche INRIA Rh^one-Alpes, 46 avenue F´elix Viallet, 38031 GRENOBLE Cedex 1
Unit´e de recherche INRIA Rocquencourt, Domaine de Voluceau, Rocquencourt, BP 105, 78153 LE CHESNAY Cedex
Unit´e de recherche INRIA Sophia-Antipolis, 2004 route des Lucioles, BP 93, 06902 SOPHIA-ANTIPOLIS Cedex

Éditeur
INRIA, Domaine de Voluceau, Rocquencourt, BP 105, 78153 LE CHESNAY Cedex (France)
ISSN 0249-6399