ANALYTIC VARIATIONS ON THE COMMON SUBEXPRESSION PROBLEM

Philippe Flajolet
INRIA, Rocquencourt
F–78150 Le Chesnay (France)

Paolo Sipala
Università degli Studi di Trieste
Via A. Valerio, 10
34127 Trieste (Italy)

Jean-Marc Steyaert
LIX
Ecole Polytechnique
F– 92128 Palaiseau (France)

Abstract. *Any tree can be represented in a maximally compact form as a directed acyclic graph where common subtrees are factored and shared, being represented only once. Such a compaction can be effected in linear time. It is used to save storage in implementations of functional programming languages, as well as in symbolic manipulation and computer algebra systems. In compiling, the compaction problem is known as the "common subexpression problem" and it plays a central rôle in register allocation, code generation and optimisation. We establish here that, under a variety of probabilistic models, a tree of size n has a compacted form of expected size asymptotically*

$$C\frac{n}{\sqrt{\log n}},$$

where the constant C is explicitly related to the type of trees to be compacted and to the statistical model reflecting tree usage. In particular the savings in storage approach 100% on average for large structures, which overperforms the commonly used form of sharing that is restricted to leaves (atoms).

Introduction. A tree can be compacted by representing occurrences of repeated subtrees only once. In that case, several pointers will point to the representation of any common subtree, and the original tree becomes a *directed acyclic digraph* also called a *dag*. The process itself is diversely known as *"sharing"* or *"common subexpression recognition"*. Obviously some storage is saved in this way, and our purpose is to estimate the expected gain attained by such a representation. (See Fig. 1.)

Trees that we consider are plane rooted trees [21] as commonly occur in a variety of contexts: in Lisp systems or as a representation of expressions in compiling; as syntax trees in parsing and code generation or structured programme editing; as the representation of terms in symbolic manipulation and computer algebra systems.

In the programming language Lisp [24], all programmes and data are represented in the form of "symbolic expressions" (S-expressions) having a binary branching tree structure. Under the tree representation, the external nodes are labelled with primitive symbols or atoms while the internal nodes, that are unlabelled, only reflect the hierarchical structure of the expression.

In most Lisp implementations, external nodes of (S-expression) trees are kept separately, in a storage area called "atom space", and multiple instances of them are stored uniquely. This is a restricted form of sharing that presents obvious storage saving advantages. Following an original suggestion by McCarthy, this sharing scheme can also be extended to internal nodes. Shared representations of Lisp trees introduce greater complexity in the management

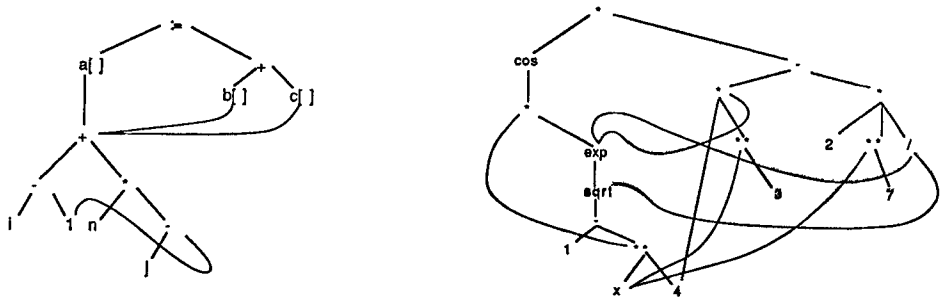

Figure 1. Compacted representations corresponding to: (i) an intermediate syntax tree for the assignment statement $a[i,j] := b[i,j] + c[i,j]$ with $x[i,j]$ being an abbreviation for $x[i-1+(j-1)*n]$; (ii) a tree representation of $\frac{d}{dx}\sin(x^4\exp(\sqrt{1-x^4}))$.

of memory, but also provide greater savings in the use of storage. Therefore, the extended sharing scheme was adopted in some implementations of Lisp [19], [32].

The syntax structure of a programme is also described by a tree. Such a representation is invariably used in parsers and structured editors. For instance, the syntax oriented Mentor system [12] uses some amount of sharing to save space when storing the representation of a programme on disk. In another context, common subexpression recognition leads to improved register allocation efficiency, a well known fact in compiler design [1]; this in turn results in code that is faster to execute. Alternatively, recognition of common programme fragments is used to generate more compact compiled code by some compilers.

Unification itself, which is at the heart of logic programming systems is usually implemented using sharing in order to avoid the combinatorial explosion that would arise from repeated duplications of subtrees.

In a related context, a symbolic manipulation system like Maple [8, 7] manages storage using pointers and hashing in such a way that subexpressions exist uniquely in main memory. With this representation, the effect of applications of expansion rules that cause subtree duplications, like distributivity or symbolic differentiation, is somewhat decreased: For instance in [18], it is shown that symbolic differentiation has expected time and storage costs of $O(n^{3/2})$ while the cost reduces to $O(n)$ when sharing is used. Furthermore, in the Maple system, functions have a "remember" option (memo functions) and in conjunction with sharing, this feature appreciably improves time performances for many applications. This can be seen when computing, for instance

$$\frac{d^3}{dx^3}\sin(x^4 e^{\sqrt{1-x^4}}).$$

There, a large number of duplicate computations take place unless some sharing is used. The Macsyma system also allows the user to save expressions in a shared format: this is the "Fassave" command [23, Chap. 10].

All these applications demonstrate the interest of sharing subexpressions or subtrees in a diversity of contexts. It thus seems of interest to be able to quantify under well defined statistical models of tree usage the gains to be expected from tree compaction.

Each tree has a maximally factored representation as a dag which is unique up to isomorphism. The size of that dag representation, measured by the number of its nodes, will be referred to here as the *compacted size* of the tree. The extremal cases for the cost of this structure are easy to characterize. In this paper, we consider trees with a fixed finite number of node types so that node degrees are bounded by some constant. Then, *the compacted size of a tree with n nodes lies between $O(\log n)$ and $O(n)$*. Our objective is to prove under a large

class of statistical models, some simple and others closer to real–life applications, the following fact.

The expectation of the compacted size of a tree with n nodes is asymptotically

$$\bar{K}_n = C_{\mathcal{M}} \frac{n}{\sqrt{\log n}} (1 + O(\frac{1}{\log n})) \tag{1}$$

where the constant $C_{\mathcal{M}}$ is explicitly computable from a specification of the class of trees and the associated statistical model $\mathcal{M}$ which is used.

In the particular case of binary trees under the uniform model (all trees with n internal nodes being taken equally likely), the value of the constant is

$$C = 2\sqrt{\frac{\log 4}{\pi}} \approx 1.\,32856\,49405.$$

For instance approximation (1) is within 2.5% of the exact expected value when $n = 500$. The sublinear estimate (1) shows that the compaction saving asymptotically approaches 100% for large trees, when "full sharing" is used, a fact first noticed by Casas *et al.* [5]. This situation is to be contrasted with the fact that sharing limited to atoms (i.e., terminals or leaves) only leads to an average gain factor that remains bounded away from 100%. For practical considerations, a closer examination is called for since expression (1), though being $o(n)$, is only slightly sublinear. It is precisely the object of this work to prove the basic claim (1) with its curious growth of $n/\sqrt{\log n}$, and also to provide ways of estimating the constants involved.

Plan of the paper. Section 1 introduces a brief discussion of algorithmic aspects of tree compaction, and it also serves to introduce some of the necessary concepts. Our objective is of course to justify the very general claim presented above. To do so, we first discuss in some detail the analysis of binary trees under the random uniform model (Sect. 2, 3). The analysis itself decomposes into an "algebraic part" providing exact enumerations through generating functions (Sect. 2) followed by asymptotic analysis (Sect. 3). The unusual form of the result is obtained through singularity analysis of a generating function, and the key lemmas are also given in Section 3.

Section 4 discusses a general class of models called "branching models" which include standard combinatorial models, probabilistic models from the theory of branching processes, and a formalisation of some empirical models suggested by statistical evidence. The exact enumeration results form the contents of Section 5, and they are established by an extension of the corresponding algebraic methods employed for binary trees. Section 6 contains some brief indications that justify our basic asymptotic claim (1) for such general models. Final remarks on this type of analysis are given in Section 7.

1 Algorithmic Aspects of Tree Compaction

Constructing the factored representation of a tree is often considered to be a computationally inefficient process. However, procedures which dynamically maintain a global "unique identifier" table allow the compacted form to be determined in time ranging from $O(n^2)$ to $O(n)$. Such procedures have been part of the folklore since the 1970's (see the "value–number" method in compiling, [1, Sec. 5.2], and especially the general discussion of [13]).

Proposition 1 *The compacted form of a tree can be computed in expected time $O(n)$ using a top–down recursive procedure in conjunction with hashing, and in worst–case time $O(n)$ using a bottom–up iterative procedure based on pseudo–sorts.*

```
function UID(T : tree) : integer;
begin
  if T = nil
      then return(0)
      else
         triple := <root(T),UID(left(T)),UID(right(T))>;
         if Found(triple,Table)
            then return(value_found)
            else counter := counter+1;
                 Insert pair (triple,counter) in Table;
                 return(counter)
         fi
  fi
end;
```

Figure 2. The main procedure UID computes "unique identifiers" for all subtrees of a given binary tree T. It is assumed that the global counter is initially set to 0; Table is a global list that maintains associations between triples and UID's that have been already computed (Table is initially empty); root(T) denotes the symbol that labels the root of tree T.

PROOF. Two labelled trees are isomorphic iff their planar representations coincide and nodes at corresponding places have identical labels. Given a tree t, nodes of t can be partitioned into equivalence classes: Two nodes s_1 and s_2 in t are equivalent iff the subtrees of t rooted at s_1 and s_2 are isomorphic. The compacted (dag) representation of t is a graph G, such that: a node $\bar{s}$ of G is an equivalence class of nodes of t; there is an edge $\langle \bar{s}_1 \bar{s}_2 \rangle$ in G iff there are nodes of t, $s_1 \in \bar{s}_1$ and $s_2 \in \bar{s}_2$, that are connected by an edge in t.

Computing the partition of nodes of t into equivalence classes reduces to associating with nodes (and their dangling subtrees) "unique identifiers" (*uid*'s), two nodes being equivalent iff they have the same uid. Figure 2 describes a top-down procedure, UID that determines uid's. It is specialised to labelled binary trees which we may take in a standard Lisp format, an example being (- (* (+ x y) (+ x y)) (* (- x y) (- x y))).

The procedure is essentially a postorder traversal that maintains a global association table. In order to determine the uid of (f u v), determine the uid's i and j of u and v. If the triple <f i j> already exists in the global table, then the corresponding (already allocated) unique identifier is returned; otherwise, a new unique identifier is allocated, and the table is updated. Atomic symbols are treated in a similar fashion.

Optimising that procedure is a data structure exercise. If the global Table is maintained sequentially, then the basic procedure is of complexity $O(n^2)$, and the complexity would reduce to $O(n \log n)$ if some balanced search tree structure were used. In practice, hashing will reduce the complexity to an *average* of $O(n)$. Finally, a more complex bottom up procedure with *worst-case* running time $O(n)$ can be found, based on pseudo radix sorting techniques (see e.g. [13]) but it seems to be mainly of theoretical interest due to larger implied constants and an intrinsically non recursive structure. ■

2 Exact Enumerations and Generating Functions

In this section, we analyze the *expectation* of the gain brought by tree compaction applied to the class $\mathcal{B}$ of (unlabelled) binary trees. All such trees with n binary nodes are taken equally likely. This constitutes one of the standard models in the analysis of random tree algorithms [21]. It is well known that there are $B_n = \frac{1}{n+1}\binom{2n}{n}$ binary trees with n internal binary nodes.

(We refer to n as the *size* of the tree, and generally we use $|t|$ to denote the size of tree t.) Each of these trees is thus taken in the model under consideration with probability $1/B_n$. The analysis that follows aims at *exact* results, the corresponding asymptotic estimates being the subject of the next section.

The major technique used here is that of *generating functions*. If f_n is a sequence of numbers, then its (ordinary) generating function, gf, is by definition

$$f(z) = \sum_{n\geq 0} f_n z^n,$$

and $[z^n]f(z)$ denotes the coefficient, f_n, of z^n in $f(z)$. As is well known the gf of the B_n is

$$B(z) = \sum_{n\geq 0} B_n z^n = \frac{1-\sqrt{1-4z}}{2z}.$$

Let $K[t]$ denote the compacted size of tree t. We consider the cumulated quantity $K_n = \sum K[t]$, with the sum extending to all binary trees of size n, and seek an expression for the expected value $\bar{K}_n = K_n/B_n$. Observe that $K[t]$ is also equal to the number of *distinct* subtrees of tree t. Thus, letting $A_{u,n}$ denote the number of trees of size n that contain u as a subtree, we get $K_n = \sum_{u\in\mathcal{B}} A_{u,n}$; accordingly for generating functions, we have $K(z) = \sum_{u\in\mathcal{B}} A_u(z)$.

Theorem 1 *The expected compacted size of a binary tree with n nodes has the explicit expression*

$$\bar{K}_n = \frac{1}{B_n}\sum_{p\geq 1}\sum_{k\geq 1}(-1)^{k-1}\binom{n-kp+1}{k}B_p B_{n-kp} \qquad \textit{with} \qquad B_n = \frac{1}{n+1}\binom{2n}{n}. \tag{2}$$

PROOF. We in fact determine the generating function of the K_n as

$$K(z) = \frac{1}{2z}\sum_{p\geq 0} B_p[\sqrt{1-4z+4z^{p+1}} - \sqrt{1-4z}]. \tag{3}$$

This form of $K(z)$ derives immediately from the determination of $A_u(z)$, the gf of trees containing the subtree u,

$$A_u(z) = \frac{1}{2z}[\sqrt{1-4z+4z^{|u|+1}} - \sqrt{1-4z}], \tag{4}$$

which is itself based on a curious *inclusion–exclusion* principle.

We employ a two step argument: (i) by standard generating function techniques, it is easy to "overcount" trees containing k occurrences of a fixed tree u; (ii) from this, the exact count of trees containing u at least once can be recovered using an inclusion–exclusion argument. That line of reasoning has the advantage of carrying over almost *verbatim* to much more general random tree models.

Let $B_{n,l}$ be the number of trees of size n having l external nodes. Clearly $B_{n,l} = 0$ if $l \neq n+1$ and $B_{n,n+1} = B_n$. The bivariate generating function of the $B_{n,l}$ is thus

$$B(z,v) \equiv \sum_{n,l\geq 0} B_{n,l} v^l z^n = vB(zv). \tag{5}$$

Consider now combinatorial configurations called k–marked trees which are trees with k distinct leaves marked. The number of such configurations of size n has generating function

$$C^{(k)}(z) = \frac{1}{k!}\frac{\partial^k}{\partial v^k}B(z,v)\Big|_{v=1}.$$

By grafting, on marked leaves, occurrences of a fixed tree u (with size p), one obtains a second type of combinatorial configurations called u^k–marked trees: These are trees with k nodes marked, each of the subtrees dangling from the marked nodes being isomorphic to u. The u^k–marked trees have gf

$$D_u^{(k)}(z) = \frac{1}{k!}\frac{\partial^k}{\partial v^k} B(z,v)\Big|_{v=1} (z^p)^k. \tag{6}$$

We claim—and delay the proof—that the gf of trees containing pattern u as a subtree at least once satisfies

$$A_u(z) = \sum_{k\geq 1} (-1)^{k-1} D_u^{(k)}(z). \tag{7}$$

If this is granted, then, by (6) and (7), we have

$$A_u(z) = \sum_{k\geq 1} \frac{(-1)^{k-1}}{k!}\frac{\partial^k}{\partial v^k} B(z,v)\Big|_{v=1} z^{pk},$$

so that, by the Taylor expansion applied around v=1 to $B(z,v)$, we get

$$A_u(z) = B(z,1) - B(z,1-z^p),$$

a form which leads to (3) using relation (5) and the fact that $K(z) = \sum_{u\in\mathcal{B}} A_u(z)$.

Now comes the inclusion–exclusion argument that justifies (7). It relates the enumeration of u^k–marked trees to the valuation $\alpha_u[t]$. Let $\omega_u[t]$ be the number of times u occurs as a subtree of t. Then, whether $\omega = 0$ or $\omega \geq 1$,

$$\alpha_u[t] = \sum_{k\geq 1} (-1)^{k-1} \binom{\omega}{k} \qquad \text{with } \omega \equiv \omega_u[t], \tag{8}$$

since in the first case the sum is empty and has value 0 and in the second case it is 1 by virtue of the binomial theorem applied to $(1-1)^\omega$. By linearity of generating functions, we find

$$A_u(z) = \sum_{k\geq 1} (-1)^{k-1} \bar{D}_u^{(k)}(z) \qquad \text{with} \qquad \bar{D}_u^{(k)}(z) = \sum_{t\in\mathcal{B}} \binom{\omega_u[t]}{k} z^{|t|}. \tag{9}$$

Finally $D_u^{(k)}(z) \equiv \bar{D}_u^{(k)}(z)$, since a tree that has ω occurrences of u gives rise to $\binom{\omega}{k}$ different u^k–marked trees. (The binomial coefficient counts the number of ways of choosing k marked occurrences out of a total of ω.) Therefore Eq. (9) entails (7) and $A_u(z)$ has been found. ∎

Variations. A simple extension of the inclusion–exclusion argument that we employed also provides the gf, written $A_{u^r}^{=}(z)$, for trees containing *exactly* r occurrences of a fixed pattern u,

$$A_{u^r}^{=} = \frac{(2r-2)!}{(r-1)!} \frac{z^{|u|r+r-1}}{(1-4z+4z^{|u|+1})^{r-1/2}}.$$

Also, the gf A_u can be determined by top down recurrences through the relation

$$A_u(z) = z^{|u|} + zA_u(z)B(z) + z(1-A_u(z))A_u(z).$$

This quadratic equation simply expresses the fact that if u occurs within t, then—boundary conditions apart—it is either in the left or in the right root subtree of t.

3 Asymptotic Analysis from Generating Functions

It is known that the asymptotic nature of coefficients f_n of a function $f(z)$ is intimately related to the asymptotic behaviour of $f(z)$ around its dominant *singularities*. (Dominant singularities are singularities of smallest modulus.) There are several methods that can be used to accomplish the "transfer" of properties of the function to the coefficients are: (i) Tauberian theorems, (ii) Darboux's method, (iii) the singularity analysis approach of [16]. The first two methods do not appear to be simply applicable, Darboux's method being even excluded by the very nature of the singularity.

Theorem 2 *The expectation of the compacted size of a binary tree of size n satisfies*

$$\bar{K}_n = \frac{K_n}{B_n} = 2\sqrt{\frac{\log 4}{\pi}}\,\frac{n}{\sqrt{\log n}}\left(1+O(\frac{1}{\log n})\right).$$

The function $K(z)$ is analytic in a larger domain than its definition implies. Analytic continuation is a crucial ingredient in applying singularity analysis. The method which we use is summarized by Lemma 2. It eventually relies on Cauchy's integral formula,

$$K_n = \frac{1}{2i\pi}\oint K(z)\,\frac{dz}{z^{n+1}}, \tag{10}$$

the contour of integration coming close enough to the singularity in order to "extract" the necessary information from the function's behaviour.

Lemma 1 *The generating function $K(z) = \sum_n K_n z^n$ is analytic in the domain D defined by $|z| < \frac{1}{2}$ and $z \notin [\frac{1}{4}, \frac{1}{2}]$. For z in the intersection of D and a sufficiently small neighbourhood of $\frac{1}{4}$, as $z \to \frac{1}{4}$, one has*

$$K(z) = \frac{C_0}{\sqrt{(1-4z)\log(1-4z)^{-1}}} + O\left(\frac{1}{\sqrt{(1-4z)\log^3(1-4z)^{-1}}}\right) \tag{11}$$

with $C_0 = 2\sqrt{\log 4/\pi}$.

PROOF. The proof starts from equation (3) in Theorem 1 by factoring out $\sqrt{1-4z}$:

$$K(z) = \frac{\sqrt{1-4z}}{2z}\sum_{p\geq 0} B_p[\sqrt{1+u_p}-1] \qquad \text{where} \qquad u_p = u_p(z) = \frac{4z^{p+1}}{1-4z}. \tag{12}$$

Then, problems on $K(z)$ are reduced to studying the function,

$$f(z) = \sum_{p\geq 0} B_p[\sqrt{1+u_p}-1-\frac{u_p}{2}]. \tag{13}$$

A simple idea consists in observing that for fixed z near the singularity $1/4$, two different approximations apply, namely

$$\sqrt{1+u} = 1+\frac{u}{2}+O(u^2) \qquad \text{and} \qquad \sqrt{1+u} = O(\sqrt{u}),$$

depending whether $|u| = O(1)$ or $1/|u| = O(1)$. In other words, there is a *threshold* function

$$p_0 \equiv p_0(z) = \log\frac{1}{|1-4z|} \tag{14}$$

and two different regimes need to be taken into account depending on the conditions $p < p_0$ or $p \geq p_0$.

The rough analysis sketched above is sufficient to establish the analytic continuation result for $f(z)$, whence for $K(z)$. A more detailed analysis reveals that the main contribution to $K(z)$ is due to the terms with $p \geq p_0$ in Eq. (12,13). It leads to the stated approximation of $f(z)$ and $K(z)$ inside their extended existence domain. ■

The last lemma of this section is borrowed from [16]. The proof is based on Cauchy's formula and it provides us with a means of translating each of the terms appearing in the singular expansion (11).

Lemma 2 (*i*). *Let α and β be real numbers with $\alpha \notin \{0, +1, +2, \ldots\}$, and*

$$g^{\langle\alpha,\beta\rangle}(z) = (1-z)^{\alpha} \log^{\beta} \frac{1}{1-z}.$$

Then, the n-th Taylor coefficient of $g^{\langle\alpha,\beta\rangle}(z)$ satisfies

$$g_n^{\langle\alpha,\beta\rangle} \equiv [z^n] g^{\langle\alpha,\beta\rangle}(z) = \frac{n^{-1-\alpha}}{\Gamma(-\alpha)} \log^{\beta} n (1 + O(\frac{1}{\log n})).$$

(*ii*). *Let $h^{\langle\alpha,\beta\rangle}(z)$ be a function analytic in the domain $\Delta_{\eta,\theta} = \{z \;/\; |z| \leq 1+\eta$ and $|\mathrm{Arg}(z-1)| \geq \theta\}$ for some $\eta > 0$ and $\theta < \frac{\pi}{2}$. Assume that, as $z \to 1$ in $\Delta_{\eta,\theta}$, for some reals α, β,*

$$h^{\langle\alpha,\beta\rangle}(z) = O((1-z)^{\alpha} \log^{\beta} \frac{1}{1-z}).$$

Then, we have

$$h_n^{\langle\alpha,\beta\rangle} \equiv [z^n] h^{\langle\alpha,\beta\rangle}(z) = O(n^{-1-\alpha} \log^{\beta} n).$$

From Lemma 1 the normalised function $K(\frac{z}{4})$ is singular at 1. The first part of Lemma 2 applies to the main term of its asymptotic expansion (11) with $\alpha = -\frac{1}{2}$ and $\beta = -\frac{1}{2}$. The error term $O(.)$ of (11) is similarly subjected to the lemma with $\alpha = -\frac{1}{2}$ and $\beta = -\frac{3}{2}$. Thus,

$$\frac{K_n}{4^n} = \frac{C_0}{\Gamma(\frac{1}{2})} n^{-1/2} (\log n)^{-1/2} (1 + O(\frac{1}{\log n})).$$

Theorem 2 is finally proved using the value of $C_0 = 2\pi^{-1/2}\sqrt{\log 4}$, $\Gamma(\frac{1}{2}) = \sqrt{\pi}$, and the classical asymptotic form $B_n \sim 4^n/\sqrt{\pi n^3}$. ■

The analytic treatment can also be applied to the function $A_u(z)$; in this way, we find that the probability that a tree of size $n \to \infty$ does not contain a fixed pattern of size p decreases exponentially like ζ^n where ζ is the root close to 1 of $\zeta^p - \zeta^{p-1} + 4^{1-p} = 0$. Several results on the occurrences of patterns in trees can be derived in this way. (The corresponding results for strings are due to Guibas and Odlyzko, cf, e.g., [27].)

Finally, we tabulate below the exact value of $\bar{K}_n$ and its asymptotic approximation $\bar{K}_n^*$ given by the main term of Theorem 2. The agreement is quite good. The compaction ratio $\bar{K}_n/n$ crosses the value 50% for n somewhere near 800.

n	10	20	50	100	200	500	1000
$\bar{K}_n$	7.745	14.232	32.080	59.781	112.098	259.757	493.495
$\bar{K}_n^*$	8.755	15.351	33.585	61.909	115.436	266.468	505.492

4 General Random Tree Models

The previous sections have demonstrated in a simple case the techniques employed to analyse the expected gain brought by compacting random binary trees. The model considered was that of "pure" (*i.e.* unlabelled) binary trees under the uniform distribution over all trees of the same size n.

Trees in actual programming practice are not always binary and unlabelled, and constraints implied by our previous model can be relaxed in several ways:

1. A natural extension consists in allowing a fixed set of node degrees. (Binary trees are restricted to node degrees $\{0, 2\}$.)
2. One may further allow nodes to contain labels (symbols): the framework becomes then that of the algebra of terms built over a fixed set $\mathcal{F}$ of functional symbols.
3. One may finally attempt to model tree usage under which certain symbols are more frequent than others: for many applications in symbolic manipulations, an exponential or a sine function will occur more frequently than a Bessel or hypergeometric function, say.

All three extensions can be covered by the following probabilistic model, called the *weighted model* or the *branching model*:

> Let $\mathcal{F}$ be a set of functional symbols, each with an *arity*. The corresponding set of term trees $\mathcal{T}$ is defined in the usual way. Let w be a *weight function* that assigns to each symbol $f \in \mathcal{F}$ a non negative real number $w[f]$. Then that weight is extended multiplicatively to trees: If t is a tree, its weight $w[t]$ is defined as
>
> $$w[t] = \prod w[f],$$
>
> where the product is extended to all node labels of t.

In the model induced by the pair $\langle \mathcal{F}; w \rangle$, trees are taken with a probability proportional to their weight. Formally, under this model, a tree of size n will be taken with probability

$$\pi[t] = \frac{w[t]}{W_n} \qquad \text{where} \qquad W_n = \sum_{|t|=n} w[t]. \tag{15}$$

Thus if we assign to symbols *sin* and *log* weights that are respectively 1 and 3, and to symbols $+$ and $\times$ weights 5 and 7, the probabilities of the two terms $(x + \sin(x)) \times (y + \log(y))$ and $(x \times \log(x)) + (y \times \log(y))$ are in the ratio $5/(7 \cdot 3)$. Furthermore, by adjusting the coefficients between symbols of different arities, one can model asymptotically any predetermined probability distribution over symbols that respects the obvious "conservation law" of trees. In this way, we can thus design a model in which ternary symbols are, e.g., twice less likely than binary symbols *etc.*

The weighted models are of interest for several reasons. By construction, they are likely to represent real-life situations better than the uniform models (that only combine points 1 and 2 above). In effect, Clark [10, 11] has shown, from statistics on actual large data structures created by a number of Lisp programmes that, rather independently of particular applications:

(i) There is a fairly constant probability, in the interval $[\frac{2}{3}, \frac{3}{4}]$, that a left son of a *cons* node be an atom, and similarly for right nodes, the range being then $[\frac{1}{4}, \frac{1}{3}]$.

(ii) The (non-nil) atom symbols tend to have a frequency distribution obeying the Zipf Law (where the i–th item has probability $\frac{C}{i}$).

Those models are also natural from a mathematical point of view since they are equivalent to assuming that the trees are generated by a *branching process* with a conditioning by the size n of the result.

Our objective is now to extend our previous approach to this whole class of weighted models. First, we shall show that expected values of the compaction ratio are still accessible via generating functions. This leads to exact expressions that generalise Theorem 1 above, and our approach follows rather directly the lines of Eq. (5-9). Next, we shall discuss how the analytic approach can be progressively extended to cover the case of general branching models. In all cases, a tree of size n is found to have an expected compacted size (1) asymptotic to

$$C\frac{n}{\sqrt{\log n}}.$$

5 General Models: Algebraic Enumerations

Let us fix from now on a set $\mathcal{F}$ of functional symbols, where each $f \in \mathcal{F}$ has an arity (or degree) $\deg(f)$ and a weight $w[f]$. We first define the *structure polynomial* associated with $\langle \mathcal{F}; w \rangle$ by

$$\phi(y) = \sum_{f \in \mathcal{F}} w[f] y^{\deg(f)}. \tag{16}$$

For instance if $\mathcal{F} = \{x, z, \sqrt{}, +, \times\}$ with corresponding set of weights $\{1, 2, 4, 8, 16\}$, then $\phi(y) = 3 + 4y + 24y^2$. We let $T(z)$ denote the generating function of the cumulated weights (15):

$$T(z) = \sum_{n \geq 0} W_n z^n = \sum_{t \in \mathcal{T}} w[t] z^{|t|}.$$

It is convenient to think of the weighted set of trees $\mathcal{T}$ as a multiset each tree being counted with a multiplicity equal to its weight, and refer to W_n as the "number" of trees in the multiset. Our tools relie on standard techniques of tree enumerations via generating functions [25, 31, 20].

Lemma 3 *(i). The generating function $T(z)$ for the number of trees is defined implicitly by*

$$T(z) = z\phi(T(z)) \tag{17}$$

where $\phi(y)$ is defined in Eq (16).

(ii). The generating function $A_u(z)$ for the number (i.e., total weight) of trees containing a fixed tree u is

$$A_u(z) = T(z, \phi(0)) - T(z, \phi(0) - w[u] z^{|u|-1})$$

where $T(z, v)$ is defined implicitly by

$$T(z, v) = zv + z\phi^+(T(z, v)) \qquad \text{and} \qquad \phi^+(y) = \phi(y) - \phi(0).$$

The first result (17) is classical in combinatorial theory (and also in branching processes) and it relates $T(z)$ to the structure polynomial ϕ. The second result regarding $A_u(z)$ also involves ϕ in a more intricate manner. The argument used is then a generalisation of the inclusion–exclusion proof of Theorem 1.

Proceeding further requires expanding $T(z, v)$, a task made relatively easy by the *Lagrange inversion theorem* (see *e.g.* [20]): *If function $h(z)$ is defined implicitly from function $\psi(y)$ by the equation*

$$h(z) = z\psi(h(z))$$

then, the Taylor coefficients of $h(z)$ and its powers are expressible in terms of coefficients of powers of ψ by

$$[z^n]h^k(z) = \frac{k}{n}[y^{n-k}]\psi^n(y).$$

This theorem provides a means of expanding the implicitly defined function $T(z,v)$ as well as determining the (unweighted) number[1] $U_{m_1,m_2,\ldots,m_r}$ of trees with m_j nodes of type j,

$$U_{m_1,m_2,\ldots,m_r} = \frac{1}{m}\binom{m}{\mathbf{m}} = \frac{1}{m}\binom{m}{m_1,m_2,\ldots,m_r} \qquad \text{with } m = \sum m_i.$$

This last result serves to count the possible "patterns" u of Lemma 3. Proceeding similarly with the generating functions of Lemma 3 leads to the main result for algebraic enumerations in the general branching model.

Theorem 3 *The expected compacted size of a tree with N nodes under a general branching model is*

$$\bar{K}_N = \frac{K_N}{W_N},$$

where W_N is the total weight of trees of size N (Eq (15)) given by

$$W_N = \frac{1}{N}\sum_{\mathbf{n}}\binom{N}{\mathbf{n}}\phi^{\mathbf{n}} = \frac{1}{N}\sum_{n_0+\cdots+n_d=N}\binom{N}{n_0,\ldots,n_d}\phi_0^{n_0}\ldots\phi_d^{n_d}, \tag{18}$$

and

$$K_N = \sum_{\mathbf{m},\mathbf{n},k}\frac{(-1)^{k-1}}{mn}\binom{n_0}{k}\binom{n}{\mathbf{n}}\binom{m}{\mathbf{m}}\phi^{\mathbf{n}}\left[\frac{\mathbf{w}^{\mathbf{m}}}{\phi(0)}\right]^k,$$

the sums being extended to all vectors[2] $\mathbf{m}$, $\mathbf{n}$, and scalar $k \geq 1$ such that

$$n + k(m-1) = N \qquad \sum_{j=0}^{d} n_j(j-1) = -1 \qquad \sum_{i=1}^{r} m_i(\alpha_i - 1) = -1,$$

with $n = n_0 + \cdots + n_d$, $m = m_1 + \cdots + m_r$, $\phi = (\phi_0,\ldots,\phi_d)$, and $\mathbf{w} = (w_1,\ldots,w_r)$.

This theorem vastly generalises Theorem 1 by providing combinatorial expressions for *any* branching model: If $\mathcal{F}$ has r symbols with δ different degrees, quantity K_N is expressed by a multiple sum of order $r+\delta-2$. In the case of "pure" binary trees, we obtain, in agreement with Theorem 1, a double sum ($\delta = r = 2$). Double sums also arise if one considers "pure" ternary trees. Binary trees with two types of leaves ($\delta = 2$, $r = 3$) lead to triple sums; unlabelled unary-binary trees or (unbalanced!) 2–3–trees lead to fourfold summations ($\delta = r = 3$) etc.

6 General Models: Asymptotic Analysis

Our objective here is to give some background to support the claim made in the introduction, Eq. (1). In Section 2, we considered the special model $\mathcal{M}$ of uniform binary trees, and we showed how singularity analysis could lead to the result. The same strategy, based on singularity analysis of generating functions, is employed here. However, two serious types of difficulties await us in the more general case.

[1] We use here standard multi–index conventions for multinomial coefficients. For vectors $\mathbf{a}$ and $\mathbf{b}$, we also let $\mathbf{a}^{\mathbf{b}}$ denote $a_1^{b_1}a_2^{b_2}a_3^{b_3}\cdots$.

[2] The maximum node degree under the model is d, there are r different node types, and α_i is the arity associated with node type i; ϕ_j represents the coefficient of y^j in the structure polynomial $\phi(y)$.

1. For families with high node degrees, generating functions of interest are algebraic functions that need not have closed form solutions. In contrast, for binary trees, all generating functions were expressible with square roots.

2. If several types of nodes with the same degree (*e.g.* $\times$ and $\div$) are present with different weights, we know from Section 3.1 that both generating functions and combinatorial enumerations involve multinomial coefficients. The presence of these multinomials rather complicates the singularity analysis of function $K(z)$.

Each of the corresponding problems can be overcome as follows.

1. Singularities of algebraic functions arising in enumeration problems are still *analytically* of a square–root type [25, 26]. This is well illustrated by combinatorial families of trees simply defined by restrictions on node degrees.

2. Multinomial (discrete) distributions are well approximated in the asymptotic limit by multivariate (continuous) Gaussian distributions. The multinomial expressions involved in the gf $K(z)$ can be analyzed in this way, and a singularity analysis can be performed. This technique is needed already when considering binary trees with several types of leaves.

In the most general case, we shall find that the model–dependent constant $C_{\mathcal{M}}$ is expressible in terms of certain numbers (characteristic of model $\mathcal{M}$) as well as the *entropy* of the asymptotic distribution of node labels in the second case. We first need to describe the "composition" of a large random tree.

Proposition 2 *Under a branching model with structure polynomial $\phi(y)$, a node type i with weight w_i and arity α_i occurs in a large random tree with asymptotic frequency given by*

$$\pi_i = w_i \frac{\tau^{\alpha_i}}{\phi(\tau)}, \tag{19}$$

where τ is the smallest positive root of the equation $\phi(\tau) - \tau\phi'(\tau) = 0$.

The dominant singularity of $T(z)$ and $K(z)$ is at $\rho = \tau/\phi(\tau)$. By standard arguments, $T(z)$ has an algebraic singularity at ρ that is of a $\sqrt{\ }$ type, and many important parameters are expressible as functions of τ [25, 31].

Theorem 4 *Given any branching model $\mathcal{M}$, the expectation of the compacted size of a tree with n nodes*[3] *satisfies asymptotically, as $n \to \infty$,*

$$\bar{K}_n = C_{\mathcal{M}} \frac{n}{\sqrt{\log n}} (1 + O(\frac{1}{\log n})) \qquad \textit{with} \qquad C_{\mathcal{M}} = \sqrt{2 \frac{H}{\pi} \frac{F}{\tau}} \tag{20}$$

There F is a fine structure constant *defined below; H is the* entropy function

$$H = \sum_i \pi_i \log \frac{1}{\pi_i}, \tag{21}$$

where π_i is the asymptotic frequency of node type i.

[3] This statement is to be understood for all n such that the number of trees of size n in the model is non zero.

We briefly explain how the fine structure constant is computed. We have r node types, with α_i being the arity of node type i. Introduce $\omega_i = \log(1/\pi_i)$. Define the Jacobian of the family of trees in terms of a determinant as

$$J = H \begin{vmatrix} \alpha_r - 1 & \omega_r \\ \alpha_{r-1} - 1 & \omega_{r-1} \end{vmatrix}^{-1}.$$

Then F is given by

$$F = \frac{|J|}{(\Delta \prod_{i=1}^{r} \pi_i)^{1/2}},$$

where Δ is the determinant of the quadratic form,

$$\sum_{i=1}^{r} \frac{1}{\pi_i} \lambda_i^2 - \left(\sum_{i=1}^{r} \lambda_i \right)^2,$$

this form being taken in the $r-2$ variables $(\lambda_1, \ldots, \lambda_{r-2})$, with λ_{r-1} and λ_r defined from the other variables by the two linear relations,

$$\sum_{i=1}^{r} \omega_i \lambda_i = 0 \qquad \text{and} \qquad \sum_{i=1}^{r} (\alpha_i - 1)\lambda_i = 0.$$

As a check, Theorem 4 gives back the estimates for binary trees, using either $\phi(y) = 1 + y^2$ or $\phi(y) = (1 + 2y + y^2)$. For unary–binary trees, it provides

$$\bar{K}_n \sim \sqrt{\frac{3 \log 3}{\pi}} \frac{n}{\sqrt{\log n}}.$$

For unbalanced 2–3 trees ($\phi(y) = 1 + y^2 + y^3$), the constant $c_{\mathcal{M}} \approx 0.97206$ has the exact value

$$\sqrt{\frac{6\tau^2 + 53\tau + 43}{26\pi} \log \frac{2\tau}{3 + \tau^2}} \qquad \text{where} \qquad \tau = \left(\frac{53}{216} + \frac{\sqrt{13}}{6\sqrt{6}}\right)^{1/3} + \left(\frac{53}{216} - \frac{\sqrt{13}}{6\sqrt{6}}\right)^{1/3} - 1/6.$$

In general these constants are best estimated using computer algebra systems!

7 Conclusions

Analysis of algorithms classically concerns itself with search trees (related to *order* properties) or digital trees (based on *digital properties* of records) [22]. The corresponding tools are recurrence equations, and differential or difference equations over generating functions for more advanced applications [14]. There is by now a well–established tradition on these models.

The branching models considered here are in comparison new–comers in the area of analysis of algorithms. They arise naturally as combinatorial models (see [21, Sect 2.3.4] for introductory material), but also as a specialisation of branching processes or as a way of making precise empirical observations on large structures created by symbolic systems [11, 10], as we have discussed in Section 4.

Meir and Moon's seminal paper [25], itself inspired by Pólya's works [29, 30], showed the possibility of counting trees in models that are essentially equivalent to our general branching model. What is interesting is a situation not unlike what we witness in statistical physics: many characteristic parameters exhibit a *qualitative* behaviour which is model–independent. For instance, it was first shown by Meir and Moon that random trees in model $\mathcal{M}$ have a profile

which, *after normalisation*, exhibits a "rain drop" shape (Rayleigh distribution) and appears to be independent of $\mathcal{M}$.

We now know more. Under any branching model, trees tend to have depth $O(\sqrt{n})$ [15] rather than $O(\log n)$ in other models. The frequency of occurrences of patterns is discussed in [31], where it is shown to be geometrically decreasing in the size of the pattern. (By analogy, in a random binary string, a pattern of size k occurs at a given place with probability 2^{-k}). Such a result entails that even naïve pattern matching algorithms have linear, $O(n)$, rather than quadratic, $O(n^2)$, complexity. A valuable analysis of multiple pattern matching along these lines has been carried out by Albert and Fages [3]. Pedersen [28] has several interesting results on the counting of trees containing patterns of various types. In particular, he has obtained independently the form of our $A_u(z)$ for binary trees, Eq. (4).

In the realm of symbolic manipulation algorithms, we have mentioned already that symbolic differentiation is greatly improved by sharing which causes its complexity to decrease from $O(n^{3/2})$ to $O(n)$, on average. Casas *et al.* [6] present difficult analyses of bottom up simplification algorithms for algebraic and logical expressions. The effect of mixed expanding–contracting rules in term rewriting systems is precisely quantified in [9]. Very recently, symbolic combinatorial methods and singularity analysis have been combined in order to analyze in detail unification algorithms [4, 2]. More generally, a whole class of algorithms in the area of symbolic manipulations is now known to be amenable to automated analysis [18, 17].

The present paper has similar objectives. However, contrary to pattern matching problems, the parameter under study is not simply recursively defined, so that the mathematical problems involved are somewhat different. Perhaps what distinguishes it is the combinatorial and analytic techniques needed to solve the original problem. With an unusual compaction ratio that involves $1/\sqrt{\log n}$, with multiplicative constants whose expression mixes the entropy of asymptotic probability distributions as well as certain algebraic numbers, it is unlikely that elementary derivations could easily be found for our most general results.

Acknowledgement. This research was supported in part by the ESPRIT II Basic Research Actions Program of the EC under contract No. 3075 (Project ALCOM).

References

[1] AHO, A. V., SETHI, R., AND ULLMAN, J. D. *Compilers: Principles, Techniques and Tools.* Addison–Wesley, 1986.

[2] ALBERT, L., CASAS, R., FAGES, F., AND ZIMMERMANN, P. Average case analysis of unification algorithms, 1990. Technical report, INRIA, in preparation.

[3] ALBERT, L., AND FAGES, F. Average case analysis of the Rete pattern–matching algorithm. In *Automata, Languages and Programming* (1988), T. Lepistö and A. Salomaa, Eds., vol. 317 of *Lecture Notes in Computer Science*, Springer Verlag. Proceedings of 15th ICALP Colloquium, Tempere, Finland, July 1988.

[4] CASAS, R., DIAZ, J., AND STEYAERT, J.-M. Average case analysis of Robinson's unification algorithm with two different variables. *Inf. Process. Lett. 31* (June 1989), 227–232.

[5] CASAS, R., DIAZ, J., STEYAERT, J.-M., AND VERGES, M. On compact representation of trees. In *Proceedings of the Colloquium on Algebra, Combinatorics and Logic for Computer Science* (1984), Janos Bolyai Mathematical Society, North Holland Publishing Company.

[6] CASAS, R., FERNANDEZ CAMACHO, M.-I., AND STEYAERT, J.-M. Algebraic simplification in computer algebra: an analysis of bottom-up algorithms. Tech. Rep. LIX–RR–89.04, Ecole Polytechnique, Palaiseau, France, 1989. To appear in *Theoretical Computer Science*, 1990.

[7] CHAR, B., GEDDES, K., GONNET, G., MONAGAN, M., AND WATT, S. *MAPLE: Reference Manual.* University of Waterloo, 1988. 5th edition.

[8] CHAR, B. W., FEE, G. J., GEDDES, K. O., GONNET, G. H., AND MONAGAN, M. M. A tutorial introduction to Maple. *Journal of Symbolic Computation 2*, 2 (1986), 179–200.

[9] CHOPPY, C., KAPLAN, S., AND SORIA, M. Complexity analysis of term rewriting systems. *Theoretical Computer Science 67* (1989), 261–282.

[10] CLARK, D. W. Measurements of dynamic list structure use in Lisp. *IEEE Trans. Software Eng. SE-5*, 1 (1979), 51–59.

[11] CLARK, D. W., AND GREEN, C. C. An empirical study of list structure in Lisp. *Commun. ACM 20*, 2 (1977), 78–87.

[12] DONZEAU-GOUGE, V., HUET, G., KAHN, G., AND LANG, B. Programming environments based on structured editors: the MENTOR experience. In *Interactive Programming Environments* (1984), D. Barstow, E. Sandewall, and H. Shrobe, Eds., McGraw–Hill, pp. 128–140.

[13] DOWNEY, P. J., SETHI, R., AND TARJAN, R. E. Variations on the common subexpression problem. *J. A.C.M. 27* (1980), 758–771.

[14] FLAJOLET, P. Mathematical methods in the analysis of algorithms and data structures. In *Trends in Theoretical Computer Science*, E. Börger, Ed. Computer Science Press, Rockville, Maryland, 1988, ch. 6, pp. 225–304. (Lecture Notes for *A Graduate Course in Computation Theory*, Udine, 1984).

[15] FLAJOLET, P., AND ODLYZKO, A. The average height of binary trees and other simple trees. *J. Comput. Syst. Sci. 25* (1982), 171–213.

[16] FLAJOLET, P., AND ODLYZKO, A. M. Singularity analysis of generating functions. *SIAM Journal on Discrete Mathematics 3*, 1 (February 1990). To appear. (Also available as INRIA Research Report 826, 1987, 25 pages).

[17] FLAJOLET, P., SALVY, B., AND ZIMMERMANN, P. Lambda–Upsilon–Omega: The 1989 Cookbook. Research Report 1073, Institut National de Recherche en Informatique et en Automatique, August 1989. 116 pages.

[18] FLAJOLET, P., AND STEYAERT, J.-M. A complexity calculus for recursive tree algorithms. *Mathematical Systems Theory 19* (1987), 301–331.

[19] GOTO, E. Monocopy and associative algorithms in an extended LISP. Tech. Rep. 74-03, Information Sciences Lab., University of Tokyo, April 1974.

[20] GOULDEN, I. P., AND JACKSON, D. M. *Combinatorial Enumeration.* John Wiley, New York, 1983.

[21] KNUTH, D. E. *The Art of Computer Programming*, vol. 1: Fundamental Algorithms. Addison-Wesley, 1968.

[22] KNUTH, D. E. *The Art of Computer Programming*, vol. 3: Sorting and Searching. Addison-Wesley, 1973.

[23] MACSYMA. *VAX UNIX MACSYMA Reference manual*, 1985.

[24] MCCARTHY, J. *LISP 1.5 Programmer's Manual.* M.I.T. Press, Cambridge, Mass., 1962.

[25] MEIR, A., AND MOON, J. W. On the altitude of nodes in random trees. *Canadian Journal of Mathematics 30* (1978), 997–1015.

[26] MEIR, A., AND MOON, J. W. On an asymptotic method in enumeration. *Journal of Combinatorial Theory,* Series A *51* (1989), 77–89.

[27] ODLYZKO, A. M. Enumeration of strings. In *Combinatorial Algorithms on Words* (1985), A. Apostolico and Z. Galil, Eds., vol. 12 of *NATO Advance Science Institute Series.* Series F: Computer and Systems Sciences, Springer Verlag, pp. 205–228.

[28] PEDERSEN, J. Enumeration of trees containing variable patterns, 1988. Manuscript.

[29] PÓLYA, G. Kombinatorische Anzahlbestimmungen für Gruppen, Graphen und chemische Verbindungen. *Acta Mathematica 68* (1937), 145–254.

[30] PÓLYA, G., AND READ, R. C. *Combinatorial Enumeration of Groups, Graphs and Chemical Componds.* Springer Verlag, New York, 1987.

[31] STEYAERT, J.-M., AND FLAJOLET, P. Patterns and pattern-matching in trees: an analysis. *Information and Control 58*, 1–3 (July 1983), 19–58.

[32] TERASHIMA, M. Algorithms used in an implementation of HLISP. Tech. Rep. 75-03, Information Sciences Lab., University of Tokyo, January 1975.