Technical Report 70-112 April 1970
NGL-21-002-008 and CST-821-5-69

ON THE NUMBER OF TREES WITH n TERMINAL NODES

by

Jack Minker
Daniel H. Fishman

UNIVERSITY OF MARYLAND
COMPUTER SCIENCE CENTER
COLLEGE PARK, MARYLAND

Technical Report 70-112 April 1970
NGL-21-002-008 and CST-821-5-69

ON THE NUMBER OF TREES WITH n TERMINAL NODES

by

Jack Minker
Daniel H. Fishman

This research was supported by the National Bureau of Standards under contract number CST-821-5-69. The computer time for this project was supported by National Aeronautics and Space Administration Grant NGL-21-002-008 to the Computer Science Center of the University of Maryland.

TABLE OF CONTENTS

Abstract

The enumeration of trees with an arbitrary number, n, of terminal nodes is explored. Three types of trees are considered; trees with at least two branches from all but terminal nodes, trees with at least two but no more than m branches from all but terminal nodes, and trees with exactly m branches from all but terminal nodes. The last type of tree in this list, which we call a "complete m-ary tree", receives the major emphasis of the paper. Two approaches are pursued in the development of explicit formulae which give the number of complete m-ary trees with n terminal nodes, for arbitrary m and n. The first approach is along the more conventional line of generating functions. We obtain the generating function for m-ary trees with n terminal nodes. The generating function is a simple generalization of the result given in Knuth [2], for binary trees.

The second approach takes a different tack. A tree grammar is introduced which generates the complete m-ary trees as linear strings of 1's and 0's. The tree grammar is a simple generalization of the tree grammar specified by Scoins [4], for binary trees. Using the notion of a syntax recognizer, theorems are proved which give the number of strings (in the language defined by the tree grammar) of length n+k for arbitrary n and k which have at least k terminal nodes. This is done for binary trees, and then for the general case of m-ary trees. The explicit formulae sought are then derived as a special case of the theorems. We also specify another formula for complete m-ary trees which we believe to be valid, but for which we have not yet found a proof. The result is provided since it is several orders of magnitude

faster in computing the number of trees than the other derived formulae.

An appendix lists several of the values of the number of complete m-ary trees for alternative values of m. The listings of computer programs used to compute these values are also included.

In considering the number of trees with at least two branches from all but terminal nodes, the method of generating functions was used to derive a formula for the number of such trees with n terminal nodes.

The results obtained for the complete m-ary trees provide upper bounds on the number of parse trees which may be constructed for strings from certain classes of context-free languages.

1. Introduction

Tree structures play an important role in many areas of computer science. For example, in the area of artificial intelligence one often deals with search trees or proof trees, while in the area of language processing, one often deals with parse trees. It is useful to obtain bounds on the number of trees of a given size which may be generated during a process, for this often lends some insight into the computational complexity of the process or the associated memory requirements. Parsing algorithms normally have associated with them the maximum time and space which may be required to recognize a string of given length from some language. These bounds are usually proportional to the length of the input string raised to some power. For a context-free language, the time required to recognize a string is proportional to n^3, where n is the number of symbols in the string [1]. The recognition process implicity contains all parse trees. It is sometimes of interest to consider the number of parse trees that are implicitly contained for strings of a given length.

In this paper, we consider the enumeration of trees based upon the number of terminal nodes that the trees may contain. Three types of trees are considered. In Section 3 we consider the enumeration of m-ary trees with n terminal nodes which have exactly m branches at all but the terminal nodes. This type of tree, which we call a complete m-ary tree, is in one-to-one correspondence with the representation of an n-ary function or relation as the composition of m-ary functions or relations. For example, there are three distinct 3-ary trees with five terminal nodes. The correspondence of these trees with the three

possible representations of R(a,b,c,d,e), a five-ary relation, as the composition of ternary relations R_1 and R_2 is shown in Figure 1.

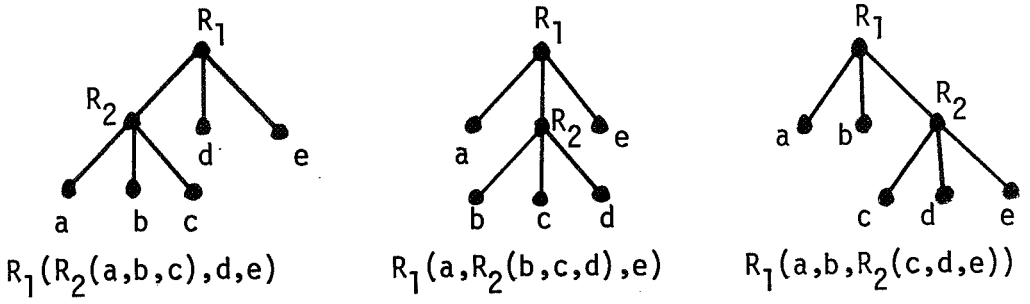


Figure 1

Relation-Tree Correspondence

The linear strings shown in Figure 1 are simply the preorder [2] representation of the tree, if we remove the parentheses and commas. Furthermore, since the scope of each relation is three, all parentheses and commas may be deleted from the forms without causing any ambiguity. It is therefore evident that the problem of determining the number of ways an n-ary relation can be represented as the composition of m-ary relations is equivalent to the problem of determining the number of complete m-ary trees with n terminal nodes. This observation motivates the approach to tree enumeration taken in Section 3.2.

In Section 4, we consider the enumeration of trees with n terminal nodes which have at least two branches from all but the terminal nodes. This is the most general type of tree we consider. Finally, in Section 5, we consider the enumeration of trees with n terminal nodes which have at least two branches from all but the terminal nodes but which have at most m branches from any node.

The results obtained for the complete m-ary trees provide upper bounds on the number of parse trees which may be obtained for certain classes of context-free languages.

2. Definitions

By a tree is meant a graph consisting of a set of points (nodes) and lines (edges) connecting these points such that between every pair of nodes, a and b, there is exactly one sequence of edges (called a path) which may be traversed in going from a to b. One restriction is placed on such a path, namely, that no two consecutive edges in the sequence may be the same. One particular node of each tree is called the root. It is distinguished from the other nodes pictorially by being the upper most node in the tree. In Figure 2 are given examples of trees in which the root of each is labeled r.

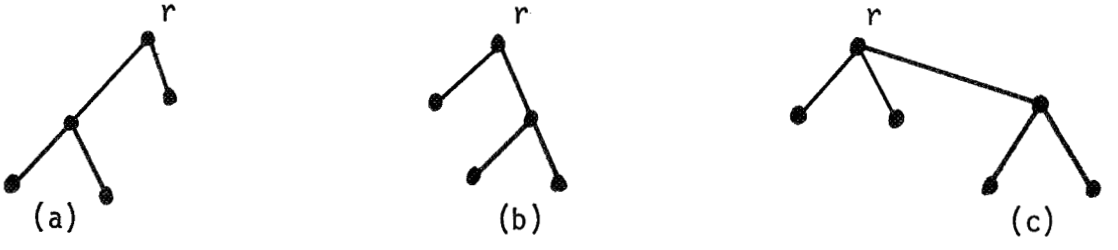


Figure 2

Examples of Trees

The way in which we draw trees is to have every node except the root entered from above by exactly one edge. The root is not entered from above by any edge.

An m-ary tree is a tree in which each node is connected by an edge to at most m nodes at a lower level (its descendents) in the tree. In Figure 2, (a) and (b) are 2-ary (binary) trees, while (c) is an 3-ary (ternary) tree.

An ordered tree is a tree in which the horizontal ordering of the

descendents of each node help to characterize the tree. Thus, in Figure 2, (a) and (b) are distinct ordered trees, but if ordering were not considered, they would be indistinguishable.

A multiple descendent tree is an ordered tree in which every node has either zero descendents or more than one descendent. A node with zero descendents is called a terminal node.

A multiple descendent m-ary tree is a multiple descendent tree in which the number of descendents of any node is at most m.

A complete m-ary tree is an ordered m-ary tree in which every node has zero or m descendents. In Figure 2, (a) and (b) are complete binary trees, each with three terminal nodes, (c) is a ternary tree, but it is not complete since it also contains a node with fewer than three descendents, but more than zero descendents. In the remainder of the paper, when no ambiguity will arise, we shall often write "m-ary tree" or simple "tree" when we mean "complete m-ary tree".

We shall denote by $t_n^{(m)}$, the number of (complete) m-ary trees with n terminal nodes. When it is clear from the context which m-ary trees are being considered, we shall often write t_n instead of $t_n^{(m)}$.

A context-free grammar (CFG) is an ordered four-tuple $G = (V_N, V_T, P, S)$. where

(a) V_N is a finite, non-empty set of symbols, called non-terminal symbols, or syntactic variables;

(b) V_T is a finite, non-empty set of symbols, called terminal symbols;

(c) P is a finite, nonempty set of syntactic rules, called productions, of the form $U \rightarrow x$, where $U \in V_N$, and x is a nonempty string of symbols from $V_N \cup V_T$. For a production $U \rightarrow x$, the number of symbols in the string x is called the length of the production;

(d) S is a special symbol in V_N called the start symbol. It designates the highest syntactic category in the grammar.

The string x directly produces the string y ($x \Rightarrow y$) if there exists strings u and v such that $x = uAv$ and $y = uwv$ and $A \rightarrow w$ is in P.

The string x produces the string y ($x \overset{*}{\Rightarrow} y$) if $x = y$ or there exists a sequence of nonempty strings $w_0, w_1, \cdots, w_n$ such that $x = w_0$, $y = w_n$, and $w_0 \Rightarrow w_1 \Rightarrow \cdots \Rightarrow w_n$.

A context-free language (CFL) is the set of strings x consisting entirely of terminal symbols, which can be produced by a context-free grammar, G, starting from the symbol S. The language defined by G, denoted L(G), is therefore the set $L(G) = \{x \mid S \overset{*}{\Rightarrow} x \text{ and } x \in V_T^+\}$ where V_T^+ denotes the set of all possible strings of terminal symbols containing at least one symbol.

Let $G = (V_N, V_T, P, S)$ be a context-free grammar. A parse tree for the string $x = a_1 \cdots a_n$ in L(G) may be defined as follows:

(a) Every node of the tree has a label which is a symbol either from V_N or from V_T;

(b) The tree will have n terminal nodes whose labels, ordered from the left, are $a_1, \cdots, a_n$, respectively;

(c) All non-terminal nodes will be labeled with symbols from V_N;

(d) The label of the root is S;

(e) If nodes $P_1, \cdots, P_k$ are the direct descendents of node P, in

order from the left, with labels $A_1,\cdots,A_k$, respectively, then $A \to A_1 \cdots A_k$ must be a production in P.

A CFG, G, is unambiguous if there exists a unique parse tree for each element of L(G); otherwise, G is said to be ambiguous. If G is an unambiguous CFG, then L(G) is an unambiguous CFL; otherwise, L(G) is an ambiguous CFL.

3. Enumeration of Complete m-ary Trees with n Terminal Nodes

3.1 The Generating Function Approach

3.1.1 Binary Trees

We present here the development of an expression which gives the number of complete binary trees with n terminal nodes. When one considers the way in which binary trees may be combined to form larger ones, certain relationships become apparent. In Figures 3(a) - 3(d) are shown all the binary trees with 1, 2, 3, and 4 terminal nodes, respectively. Dashed lines are used in this Figure to indicate the way in which the trees were combined, and expressions are given which describe the relationships induced by these combinations. In Figure 3(a) is depicted the binary tree with one terminal node, and we indicate that $t_1 = 1$. In Figure 3(b) we see that two binary trees with one node each are joined via a root node to produce a binary tree with two terminal nodes. The relationship $t_2 = t_1 t_1$ is given to describe the situation. The juxtoposition of the symbols $t_1 t_1$ lexically describes the graphical situation while the numerical value of their product, $t_1 \cdot t_1$ indicates the number of ways two trees with one terminal node each may be combined to form a binary tree with two terminal nodes. In Figure 3(c) the two possible trees with three terminal nodes are shown. One of these is constructed by joining a tree with two terminal nodes with a tree with one terminal node by means of a root node. This is indicated by $t_1 t_2$. Thus, we have the relationship $t_3 = t_2 t_1 + t_1 t_2$, to indicate the number of ways that such trees may be constructed from trees of one and two terminal nodes, respectively. Finally, Figure 3(d)

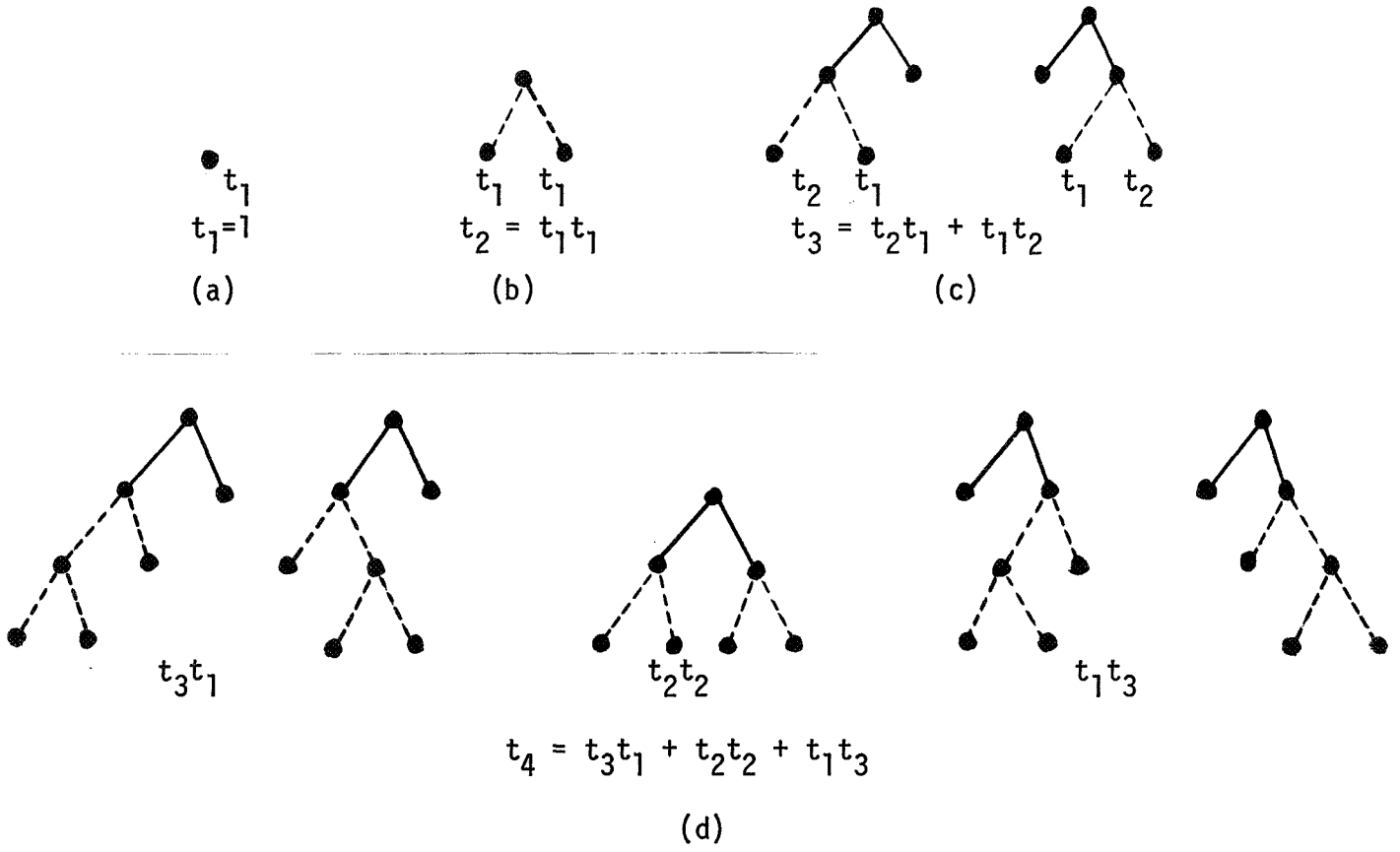


Figure 3

The Complete Binary Trees with 1,2,3, and 4 Terminal Nodes

depicts the five ways that trees with one, two, and three terminal nodes may be combined to form trees with four terminal nodes.

The manner of constructing arbitrarily large trees may be generalized easily. The following expression represents the general case:

$$(1)\quad t_n = \sum_{i=1}^{n-1} t_{n-i}\, t_i \quad , \; n \geq 2.$$

If we let $T_2(x)$ be the generating function for binary trees, then we have

$$(2)\quad T_2(x) = t_1 x + t_2 x^2 + t_3 x^3 + \cdots .$$

Squaring both sides of this equation, we obtain

$$T_2^2(x) = (t_1 t_1)x^2 + (t_1 t_2 + t_2 t_1)x^3 + (t_1 t_3 + t_2 t_2 + t_3 t_1)x^4 \cdots .$$

But, by Equation (1), the coefficients of x^k in this equation, for each k, is just t_k, so that we have,

$$T_2^2(x) = T_2 x^2 + t_3 x^3 + t_4 x^4 + \cdots .$$

Hence,

$$T_2^2(x) = T_2(x) - t_1 x.$$

Making use of the fact that $t_1 = 1$, we have

$$(3)\quad T_2(x) = T_2^2(x) + x.$$

Solving the quadratic equation for $T_2(x)$ we obtain

$$(4)\quad T_2(x) = 1/2 - 1/2 \sum_{j=0}^{\infty} \binom{1/2}{j} \left(-4x\right)^j \quad ,$$

where we have merely expanded the radical in the solution of the quadratic equation. Thus,

$$T_2(x) = x + x^2 + 2\,x^3 + 5\,x^4 + 14\,x^5 + 42\,x^6 + \cdots .$$

It is clear from (2) and (4) that,

$$(5)\begin{cases} t_n = -1/2\binom{1/2}{n}(-4)^n, \text{ or} \\ t_n = \dfrac{2^{n-1}\sum_{i=0}^{n-2}(2i+1)}{n!}, \; n \geq 2 \text{ or} \\ t_n = \dfrac{2(2n-3)!}{(n-2)!\,n!} \end{cases}$$

The development presented above is essentially similar to that given in Knuth [2] with minor differences. One difference worth noting is that Knuth develops an equation for the number of ordered (not necessarily complete) binary trees with n nodes while we have derived an equation for the number of complete binary trees with n terminal nodes. It turns out that the number of the former type of trees with n nodes is equal to the number of the latter type of trees with n+1 terminal nodes, and hence with 2n+1 nodes. Riordan [5] also considers the problem of obtaining the number of trees with n nodes, and has developed the generating function for such trees.

In his 1967 paper, Scoins [3] also developed Equations (1) and (3) for ordered rooted trees. While he indicates that Equation (3) can be solved to obtain explicit formulae for t_n, he states that these would take longer to evaluate than the recurrence relation given by Equation (1). However, in view of the relative simplicity of Equation (5), this statement would appear to be unjustified.

We may find an asymptotic approximation for t_n by using Stirling's Approximation,

$$n! \cong \sqrt{2\pi}\, n^{n+1/2}\, e^{-n} ,$$

and the definition of the exponential function,

$$\lim_{n\to\infty} \left(1 - \frac{a}{n}\right)^n = e^{-a} .$$

It is easy to see that, asymptotically,

$$(6) \quad t_n \cong \frac{2^{2n-2}}{n\sqrt{\pi n}}$$

This result is also stated in Knuth [2]. Furthermore, using (5), it follows that

$$(7) \begin{cases} \dfrac{t_{n+1}}{t_n} = 4 - \dfrac{6}{n+1} , \text{ and hence, since } \lim\limits_{n\to\infty} \dfrac{t_n}{n+1} \to 0, \\ t_{n+1} \cong 4\, t_n \end{cases}$$

for large values of n.

The results obtained in this Section may be applied to obtain an upper bound on the number of parse trees that may be constructed for strings of a given length from context-free languages of the appropriate type. In particular, if the productions of a CFG are all of length two, then the parse trees will be complete binary trees. If the grammar is unambiguous, then, by definition, any string of the corresponding language will admit only a unique parse tree. But, regardless of the degree of ambiguity of the grammar, a string of length n from the language will admit at most t_n parse trees if the grammar contains no two productions with the same right-hand side. This latter restriction

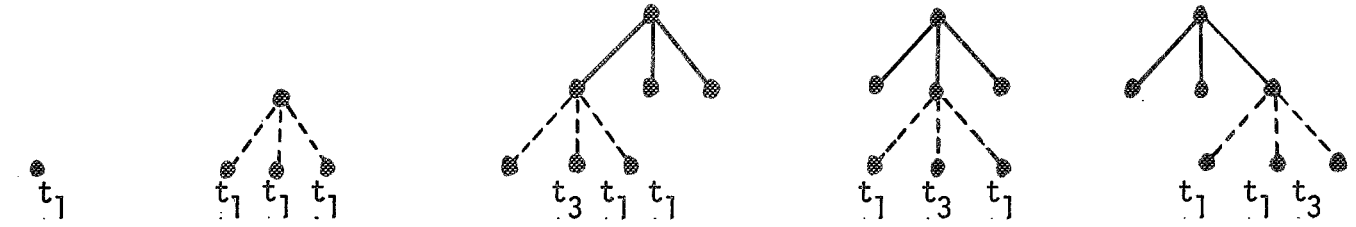
is required to prevent the possibility of parse trees with the same graphical form but with different labels. How closely the actual bound for a given grammar comes to the theoretical upper bound will of course depend upon the grammar.

A language in which all strings of length n have t_n distinct parse trees is defined by the grammar $G = (V_N, V_T, P, S)$, where $V_N = \{S\}$, $V_T = \{a,b\}$, and P is the set consisting of the four productions $S \rightarrow ab|aS|Sb|SS$. Notice that each of the productions is of length two. This leads to the property that in the parse tree, every S node will have two descendents. By definition, the terminal symbols with respect to the grammar will be terminal nodes in the parse tree. Furthermore, since the productions display non-recursiveness, right-, left-, and double-recursiveness, respectively, all possible complete binary trees will be represented by the parse trees, with one trivial exception; there is no such thing as a single node parse tree.

An additional application of the results of this Section is to context-free grammars in Chomsky Normal Form; that is, to grammars whose productions are of the form $A \rightarrow B\ C$, and $D \rightarrow e$, where A, B, C, and D designate non-terminals and e designates a terminal symbol. While a grammar of this type contains productions of length one, no problems result since each such production involves a terminal symbol. Therefore, in a parse tree, the only nodes which have a single descendent are those which lead directly to a terminal symbol, and hence, a terminal node. If we think of these paired nodes as single terminal nodes, then the parse trees may be regarded as complete binary trees, and hence the bounds on the number of parse trees may be obtained.

3.1.2 Extension to m-ary Trees

The results obtained for binary trees will be extended here for arbitrary m-ary trees. To motivate the extensions to the general case, we shall consider the special case of ternary trees. Analogous relationships of those shown for binary trees in Figure 3 are given for ternary trees in Figure 4.



$t_1 = 1$ $\qquad$ $t_3 = t_1t_1t_1$ $\qquad$ $t_5 = t_3t_1t_1 + t_1t_3t_1 + t_1t_1t_3$

(a) $\qquad$ (b) $\qquad$ (c)

Figure 4

The Complete Ternary Trees with 1,3, and 5 Terminal Nodes

Several observations may be made from Figure 4. First, the summands for t_5 are each the product of three terms. This will obviously generalize to m-ary trees where summands will each be the product of m terms. Second, the sum of the subscripts of each of the summands is equal to the number of terminal nodes of the tree they describe, and for a given tree size, all distinct permutations of the subscripts in a summand will appear as other summands in the same sum. This again will generalize to m-ary trees. Third, while Figure 4 does not illustrate this case, it is evident that t_7 for example, is equal to

$$t_5t_1t_1 + t_1t_5t_1 + t_1t_1t_5 + t_3t_3t_1 + t_3t_1t_3 + t_1t_3t_3 ,$$

and that in general, the summands for t_i will include all possible smaller tree sizes such that if i_1, i_2, i_3 are legitimate tree sizes such that $i_1 + i_2 + i_3 = i$, then $t_{i_1} \cdot t_{i_2} \cdot t_{i_3}$ will appear as a summand. Finally, we notice that for complete ternary trees, the number of terminal nodes that a tree can have is 1, 3, 5, ... , 2n-1, ...

Thus, the relationships shown in Figure 4 may be generalized and expressed as follows:

$$t^{(3)}_{2n+3} = \sum_{i_1,i_2,i_3} t_{i_1} t_{i_2} t_{i_3} \quad , \qquad n \geq 0 \ ,$$

summed over all indices i_1, i_2, i_3 for which

$$\left\{ \begin{array}{l} 1 \leq i_1, i_2, i_3 \leq 2n+1 \\ i_1 + i_2 + i_3 = 2n+3 \\ i_j = 2n_j + 1 \ , \ 1 \leq j \leq 3, \ n_j \geq 0 \end{array} \right\} \ ,$$

where $t^{(3)}_1 = 1$. This may be extended further to handle arbitrary m-ary trees to be:

$$(8) \quad t^{(m)}_{n(m-1)+m} = \sum_{i_1,\ldots,i_m} t_{i_1} t_{i_2} \cdots t_{i_m} \quad , \quad n \geq 0 \ ,$$

summed over all indices $i_1,\ldots,i_m$ for which

$$\left\{ \begin{array}{l} 1 \leq i_1, \ \ldots \ , \ i_m \leq n(m-1)+1 \\ i_1 + i_2 + \ \ldots \ + i_m = n(m-1)+m \\ i_j = n_j(m-1)+1, \ 1 \leq j \leq m, \ n_j \geq 0 \end{array} \right\} \ ,$$

where $t^{(m)}_1 = 1$.

The generating function for m-ary trees may now be specified. This is just

$$(9) \quad T_m(x) = t^{(m)}_1 x + t^{(m)}_m x^m + t^{(m)}_{2m-1} x^{2m-1} + \cdots + t^{(m)}_{j(m-1)+1} x^{j(m-1)+1} + \cdots \ ,$$

where $t^{(m)}_{j(m-1)+1}$ are the number of m-ary trees with $j(m-1)+1$ terminal nodes. Now, if we raise $T_m(x)$ to the m^{th} power, and apply the multinomial theorem, the coefficients of x^m are just those given by (8). Hence, the generating function for m-ary trees satisfies

$$(10)\quad T_m(x) = T_m^m(x) + x .$$

This equation may be solved for m=2, 3, and 4 in closed form. Indeed, for m=2, Equation (4) results. For m=3 and 4, the formulae for cubic and quartic equations may be used. However, the expansion of the equations in powers of x is, at best, complex.

We note that, from (10),

$$\frac{d^n}{dx^n}[T_m(x)] = \frac{d^n}{dx^n}[T_m^m(x)] , \quad \text{for } n>1$$

and

$$\frac{d}{dx}[T_m(x)] = \frac{d}{dx}[T_m^m(x)] + 1.$$

Taking the limit as $x\to 0$, and noting that $T_m(0) = 0$, we may find the values of $T_m^{(n)}(0)$. But, we know that

$$\frac{T_m^{(n)}(0)}{n!} = t_n^{(m)}.$$

Hence, it would be easy to find the values of $t_n^{(m)}$ using, for example, a programming language that permits formal manipulation of symbols, such as in FORMAC, or LISP.

The material in this Section may be summarized by the following theorem.

Theorem 1: Given a complete m-ary tree, where $t^{(m)}_{n(m-1)+m}$ represents the number of m-ary trees with $n(m-1)+m$ terminal nodes, then the following holds.

The generating function is given by

$$(9)\quad T_m(x) = \sum_{j=0}^{\infty} t^{(m)}_{j(m-1)+1} x^{j(m-1)+1} \quad ,$$

and the generating function satisfies the functional equation

$$(10)\quad T_m(x) = T^m_m(x) + x \quad .$$

The values of $t^{(m)}_{n(m-1)+m}$ are given by the relationships

$$(8)\quad t^{(m)}_{n(m-1)+m} = \sum_{i_1,\ldots,i_m} t_{i_1} t_{i_2} \cdots t_{i_m} \quad , \quad n \geq 0.$$

summed over all indices $i_1, i_2, \ldots, i_m$ for which

$$\left\{ \begin{array}{l} 1 \leq i_1, \ldots, i_m \leq n(m-1)+1 \\ i_1 + i_2 + \cdots + i_m = n(m-1)+m \\ i_j = n_j(m-1)+1,\ i \leq j \leq m,\ n_j \geq 0 \end{array} \right\}$$

where $t^{(m)}_1 = 1$.

For m=2,

$$(1)\quad t^{(2)}_n = \sum_{i=1}^{n-1} t_{n-i} t_i \quad , \quad n \geq 2$$

and $$(5)\quad t^{(2)}_n = \frac{2(2n-3)!}{(n-2)!n!} \quad , \quad n \geq 2$$

and $$(6)\quad t_n \cong \frac{2^{2n-2}}{n\sqrt{\Pi n}} \quad , \text{ for large } n,$$

and $$(7)\quad t_{n+1} \cong 4t_n \quad , \text{ for large } n.$$

In this Section, we have developed a recurrence relationship for the number of m-ary trees with n terminal nodes. In Section 3.2.2, we develop an explicit formula for the number of such trees. These

results may be applied to obtain the upper bound on the number of parse trees that may be constructed for strings from special types of context-free languages. In particular, $t_n^{(m)}$ is the upper bound on the number of parse trees of a string of length n from a language generated by a grammar whose productions are all of length m, and in which no two productions have the same right-hand side. As we did at the end of Section 3.1.1, we may relax the restriction on the length of the productions to allow, in addition to productions of length m, productions of length one where such productions involve a terminal symbol. For this type of grammar, $t_n^{(m)}$ is also the upper bound on the number of parse trees for a string of length n.

For m equal to 2, we have given in Section 3.1.1 a grammar which will realize all $t_n^{(2)}$ parse trees for a given n. We may generalize that grammar to produce a grammar with the same property for arbitrary m and n. Such a grammar may be specified as $G = (V_N, V_T, P, S)$ where $V_N = \{S\}$, $V_T = \{a\}$, and the productions are all of length m having the property that all the permutations of $0, 1, \ldots,$ up to m S's with $m, m-1, \ldots, 0$ a's, respectively, appear as the right part, x, of a rule of the form $S \rightarrow x$.

3.2 New Proof Using a Recognizer

3.2.1 Binary Trees

An alternative derivation of the expression for computing the number of complete binary trees with n terminal nodes is presented here. The approach has been motivated by considering the parsing of strings in the language defined by the grammar, $G = (V_N, V_T, P, S)$ where V_N contains the sole non-terminal symbol S, which is also, therefore, the start symbol of the grammar; V_T contains two terminal symbols, 0 and 1; and P contains the following two productions, $S \rightarrow 0|1SS$.

The strings in L(G), the language defined by G, correspond to the complete binary trees, as is shown in Figure 5. The correspondence shown in Figure 5 is derived from the fact that every string of L(G), other than the string 0, is composed of a 1 followed by two substrings, each of which is also a member of L(G). Therefore, by corresponding the left-most 1 of a string in L(G) to the root of a tree, and the left and right well-formed substrings to the left and right subtrees of the root, respectively, the correspondence shown in Figure 5 is obtained.

Notice that the trees in Figure 5 are not the parse trees for elements of L(G), since the latter trees will be ternary trees in which every S node has either one or three descendents. A parse tree for the string 1011000 is given in Figure 6 to illustrate this fact. The grammar given at the end of Section 3.1.1 defines a language for which the parse trees correspond to all the complete binary trees.

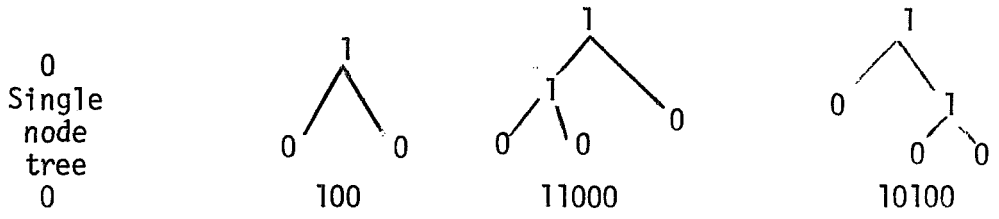


Figure 5

Correspondence Between L(G) and Complete Binary Trees

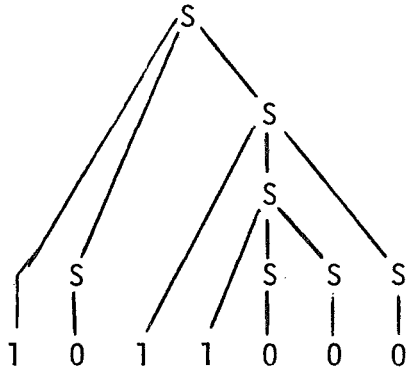


Figure 6

Parse Tree for an Element of L(G)

By identifying certain classes of strings of L(G), and counting the members in these classes, one gains information about the corresponding classes of trees. For example, if one had an expression which gave for arbitrary n the number of strings in L(G) which are of the form

$$\underbrace{X\ X\ .\ .\ .\ X}_{n-2}\ 0\ 0$$

where an X indicates that the symbol is a zero or a one, then the same expression gives the number of complete binary trees with n terminal nodes. Such an expression is a special case of one which gives the number of strings of the form

$$\underbrace{X\ X\ .\ .\ .\ X}_{n}\ \underbrace{0\ 0\ .\ .\ .\ 0}_{k}$$

which are in L(G). The more general formula will be developed first, and then an expression which gives the number of complete binary trees with n terminal nodes will be derived as a special case.

Let S(n,k) denote the set of all well-formed strings in L(G) which have at least k terminal zeros, and are of length n+k. That is, the strings of S(n,k) are of the form

$$\underbrace{X\ X\ .\ .\ .\ X}_{n}\ \underbrace{0\ 0\ .\ .\ .\ 0}_{k}\ ,$$

where an X indicates that the symbol is a zero or a one (as permitted by well-formedness considerations), and n and k are integers such that $0 \leq k \leq n+1$, and $n>0$.

Let F(n,k) denote the number of elements in S(n,k). To determine the number of well-formed strings of the form F(n,n-(k-1)), we require the following Lemma:

Lemma 1.

Let F(n,k) denote the number of well-formed strings in L(G) with at least k zeros in its rightmost configuration. Then, the following relations hold:

(11) $F(2,0) = 0$;

(12) $F(n,0) = F(n-1,1) = F(n-2,2)$ for $n \geq 3$;

(13) $F(n,n+1) = 1$ for $n \geq 0$;

(14) $F(n,k) = 0$ for $k>n+1$;

(15) $F(n,k) = F(n-1,k+1) + F(n-1,k-1)$ for $n \geq 2$, $2 \leq k \leq n+1$.

Proof.

Equation (11) is an obvious consequence of the grammar since all

well-formed strings must have an odd number of symbols. Equation (12) is a consequence of the fact that every string in L(G) of length greater than 1 must have two zeros at the right-hand end, for otherwise it is not well-formed. To prove (13), let y be a string in S(n,n+1). If n=0, then y=0 is the only possible string. If n>0, then the length of the string y, denoted by $|y|$, is 2n+1. Any string of this length must have n ones and n+1 zeros. But, since y has n+1 terminal zeros, the first n symbols must all be ones, and there is only one such string y for each n. To prove (14), we observe once again that any well-formed string with n ones has exactly n+1 zeros. Therefore, if k>n+1, then there will not be enough ones in the string to satisfy the well-formedness criterion.

We must now prove (15) to complete the proof of Lemma 1. Let $y \in S(n,k)$. Then, $y = X_1X_2 \ldots X_n0_10_2 \ldots 0_k$, and $|y| = n+k$, where $X_i = 0$ or 1 for each i, and $0_j = 0$ for each j, under the assumption that the string is well-formed according to the grammar. To test whether or not a string is well-formed, we may work from right to left with the string, and place each element on a stack. Whenever we come to a case where $X_i = 1$, we pop-up the last two entries on the stack. If they are not zero's or S's, we place an S on the stack. Now consider the set S(n,k) of well-formed strings. This set may be subdivided in an obvious manner into the class of well-formed strings with $X_n = 0$, and the set of well-formed strings with $X_n = 1$. Now consider the set of well-formed strings with $X_n = 1$. If we are testing the well-formedness, the k zero's go on a stack. When we reach X_n, it is a 1, and for the string to be well-formed, clearly $k \geq 2$. Therefore, we replace the last two entries on the stack by an S. If we now return the entries on the

stack to the modified string, we have, $y' = X_1 \ldots X_{n-1}\ S\ 0_3 \ldots 0_k$, or a well-formed string with (n-1) X's and k-1 zero's (where S counts as a zero) in L(G). Hence, we have

$$F(n,k) = F(n-1,k+1) + F(n-1,k-1) \text{ for } k \geq 2.$$

Also, since we must have $k \leq n+1$, by Equation (14), the proof of the Lemma is complete.

Lemma 2.

$F(n,n-1) = n-1$ for $n \geq 2$.

Proof.

The proof is by induction on n. Let n=2, then we have, from Equation (12) and (13),

$$F(2,1) = F(1,2) = 1$$

Now suppose the Lemma is valid for some arbitrary value of $N \geq 2$, and that $n = N + 1$. We have from Equation (15), from Equation (13), and the induction hypothesis

$F(N+1,N) = F(N,N+1) + F(N,N-1)$, and

$F(N+1,N) = N$.

This concludes the proof of Lemma 2.

We may now prove the following theorem required to develop the number of complete binary trees.

Theorem 2.

Suppose n and k are integers such that $2 \leq k \leq \frac{n}{2}$.

Then

$$(16)\quad F(n,n-(2k-1)) = (n-(2k-1)) \cdot \frac{\prod_{i=0}^{k-2} (n-i)}{k!} .$$

Proof.

The theorem will be proved by induction on both n and k.

Let k=2. Then we want to show that for all $n \geq 4$,

$$(17)\quad F(n,n-3) = (n-3) \cdot \frac{\prod_{i=0}^{0} (n-i)}{2!} .$$

This will be shown by induction on n. Let n=4. Then we have, by Equation (12),

$$F(4,1) = F(3,2);$$

and by Equation (15),

$$F(4,1) = F(2,3) + F(2,1);$$

and by Equation (12),

$$F(4,1) = F(2,3) + F(1,2);$$

and by Equation (13),

$$F(4,1) = 2.$$

This last equation may be written as,

$$F(4,1) = 1 \cdot \frac{\prod_{i=0}^{0} (n-i)}{2!} ,$$

thereby proving the base of the induction on n. Now assume Equation (17) holds for arbitrary $N \geq 4$ and consider the case where n = N + 1. We have, by Equation (15),

$$F(N+1,N-2) = F(N,N-1) + F(N,N-3) .$$

By Lemma 2, and the induction hypothesis,

$$F(N+1,N-2) = (N-1) + (N-3) \cdot \frac{\prod_{i=0}^{0} (N-i)}{2!} .$$

This last expression may be rewritten as

$$F(N+1,N-2) = ((N+1)-3) \cdot \frac{\prod_{i=0}^{0} ((N+1)-i)}{2!}$$

thereby concluding the induction on n and hence the base of the induction on k.

Now assume that Equation (16) is valid for some arbitrary value of $K \geq 2$ and for all $n \geq 2K$. Notice, in the equations that follow, that for some fixed value of k, the difference between the first and second subscripts of F will remain fixed at 2k-1 for all values of n. We now consider F(2K+2,2K+2-(2(K+1)-1)); that is, we shall prove the case for k=K+1 by induction on n, so we begin by letting n=2(K+1). Simplifying the second subscript, we obtain

$$F(2K+2,2K+2-(2(K+1)-1)) = F(2K+2,1);$$

and from Equation (12),

$$F(2K+2,2K+2-(2(K+1)-1)) = F(2K+1,2).$$

But in this last expression, the difference between the subscripts of F is $2K-1$. This, therefore satisfies the induction hypothesis of Equation (16). Therefore, for n=2K+2, Equation (16) is valid. This concludes the base of the induction on n.

Assume now that Equation (16) is valid for some arbitrary value of $N \geq 2(K+1)$, and consider n=N+1. Consider the identity:

$$F(N+1,(N+1)-(2(K+1)-1)) = F(N,N-(2(K+1)-1)).$$

We have, by Equation (15) of Lemma 1,

$$(18)\quad F(N+1,(N+1)-(2(K+1)-1)) = F(N,N-(2K-1)) + F(N,N-(2(K+1)-1)).$$

But, by the induction hypothesis for n,

$$F(N,N-(2(K+1)-1)) = (N-(2(K+1)-1)) \cdot \frac{\prod_{i=0}^{K-1} (N-i)}{(K+1)!}$$

$$= \frac{(N-2K-1)(N-K+1)}{(N+1)} \cdot \frac{\prod_{i=0}^{K-1}(N+1-i)}{(K+1)!}$$

In addition, by the induction hypothesis for k.

$$F(N,N-(2K-1)) = (N-(2K-1)) \cdot \frac{\prod_{i=0}^{K-2}(N-i)}{K!}$$

$$= \frac{(N-2K+1)(K+1)}{(N+1)} \cdot \frac{\prod_{i=0}^{K-1}(N+1-i)}{(K+1)!} .$$

Thus, Equation (18) may be rewritten as

$$F(N+1,(N+1)-(2(K+1)-1))$$

$$= \left[\frac{(N-2K+1)(K+1)}{(N+1)} + \frac{(N-2K-1)(N-K+1)}{(N+1)}\right] \cdot \frac{\prod_{i=0}^{K-1}(N+1-i)}{(K+1)!}$$

$$= (N-2K) \cdot \frac{\prod_{i=0}^{K-1}(N+1-i)}{(K+1)!}$$

$$= \left[(N+1) - (2(K+1)-1)\right] \cdot \frac{\prod_{i=0}^{(K+1)-2}((N+1)-i)}{(K+1)!} .$$

This concludes the induction on n and on k, and hence, the proof of Theorem 2.

Now that Equation (16) has been established, the number of complete binary trees with n terminal nodes can be derived. In order for a binary tree to have n terminal nodes, it must have n-1 non-terminal nodes, or a total of 2n-1 nodes. As noted previously, F(n,k) represents

the number of well-formed strings with at least k zeros at the right. Now, F(n,2) further represents the number of complete binary trees with n+2 nodes. F(2n-3,2) is the number of complete binary trees with n terminal nodes. Therefore, if we evaluate F(2n-3,2), we obtain t_n. We may use Equation (16) to find F(2n-3,2), since

$$F(2n-3,2) = F(2n-3,2n-3-(2k-1)),$$

where obviously, k=n-2. Therefore,

$$t_n = F(2n-3,2) = \frac{2\prod_{i=0}^{n-4}(2n-3-i)}{(n-2)!}, \quad n \geq 3, \text{ and}$$

$$t_n = \frac{2(2n-3)!}{n!(n-2)!}, \quad n \geq 3.$$

We note that this is the same formula as given in Equation (5), Section 3.1.1.

3.2.2 Extension to m-ary Trees

In the previous Section, we developed, for binary trees, an expression for computing F(n,n-(2k-1)) in general, and applied this to obtain the values of $t_n^{(2)}$ = F(2n-3,2). In this Section we treat the general case of m-ary trees by deriving an expression for $F_m(n,(m-1)n-(mk-1))$, the number of strings $L(G_m)$, $m \geq 2$, of length mn-(mk-1) which have at least (m-1)n-(mk-1) terminal zeros. By $L(G_m)$, we mean the language defined by the grammar $G_m=(V_n,V_T,P,S)$ where $V_n=\{S\}$, $V_T=\{0,1\}$, and P consists of the productions:

$$S \rightarrow 0 \mid 1\underbrace{S\ S \cdots S}_{m\ S's}.$$

It is clear that the elements of $L(G_m)$ correspond to the complete m-ary trees. Once we derive $F_m(n,(m-1)n-(mk-1))$, we apply the results to produce, as a special case, the number of complete m-ary trees with n terminal nodes, $t_n^{(m)}$. To support the development to be pursued, we need a generalization of Lemma 1 given in the previous Section. Since it is an obvious generalization of Lemma 1, it will be stated without proof.

Lemma 3.

Let $F_m(n,k)$ denote the number of well-formed strings in $L(G_m)$ with at least k-zeros at the right-hand end of the string. Then,

(19) $F_m(2,0) = F_m(3,0) = \cdots = F_m(m,0) = 0$, for $m \geq 2$;

(20) $F_m(n,0) = F_m(n-1,1) = \cdots = F_m(n-m,m)$, for $n \geq m+1$;

(21) $F_m(n,n(m-1)+1) = 1$, for $n \geq 0$;

(22) $F_m(n,k) = 0$, for $k > n(m-1)+1$;

(23) $F_m(n,k) = F_m(n-1,k+1) + F_m(n-1,k-m+1)$, for $n \geq 2$ and $m \leq k \leq n(m-1)+1$.

We now derive a result similar to Theorem 2, but for arbitrary m.

Theorem 3.

Let m, k, and n be integers such that $m \geq 2$, $k \geq 1$, and $n \geq \frac{mk}{m-1}$. Then,

$$(24) \quad F_m(n,(m-1)n-(mk-1)) = \sum_{i=N_k}^{n-1} F_m(i,(m-1)i-(mk-(m+1)))$$

$$\text{where } N_k = \begin{cases} \frac{mk}{m-1} - 1 & \text{if } (m-1) \text{ divides } k \\ \left[\frac{mk}{m-1}\right]^* & \text{otherwise.} \end{cases}$$

Proof.

The theorem will be proved by induction on n. Let m and k be arbitrary integers such that $k \geq 1$, and $m \geq 2$.

$$\text{Let } n = \begin{cases} \frac{mk}{m-1} & \text{if } (m-1) \text{ divides } k \\ \left[\frac{mk}{m-1}\right] + 1 & \text{otherwise.} \end{cases}$$

Case 1.

In this case we suppose that m-1 divides k. We therefore, consider the case where $n = \frac{mk}{m-1}$. Then,

$$F_m\left(\frac{mk}{m-1}, \frac{mk}{m-1}(m-1) - (mk-1)\right) = F_m\left(\frac{mk}{m-1}, 1\right), \text{ and by Equation (20),}$$

$$= F_m\left(\frac{mk}{m-1} - 1, 2\right).$$

* We use $\left[\frac{p}{q}\right]$ to denote the greatest integer in $\frac{p}{q}$.

This last equation can be rewritten, obviously, as,

$$= \sum_{i = \frac{mk}{m-1} - 1}^{\frac{mk}{m-1} - 1} F_m(i,2) \ .$$

This equation may be rewritten as,

$$= \sum_{i = \frac{mk}{m-1} - 1}^{\frac{mk}{m-1} - 1} F_m(i,(m-1)i-(mk-(m+1))),$$

since when $i = \frac{mk}{m-1} - 1$, evaluation of the second subscript yields the value of 2, as it should. But, this last sum is precisely of the form of Equation (24), where $N_k = \frac{mk}{m-1} - 1$, and $n = \frac{mk}{m-1}$, which is what we wanted to show.

Case 2.

We now suppose that m-1 does not divide k. In this case therefore, we must consider $n = \left[\frac{mk}{m-1}\right] + 1$. Then we have

$$(25) \quad F_m\left(\left[\frac{mk}{m-1}\right] + 1, \left(\left[\frac{mk}{m-1}\right] + 1\right)(m-1) - (mk-1)\right)$$

$$= F_m\left(\left[\frac{mk}{m-1}\right] + 1, \left[\frac{mk}{m-1}\right](m-1) + m - mk\right).$$

Let $\frac{mk}{m-1} = p + \frac{r}{m-1}$, where $0<r<m-1$, $p \geq 1$.

Then $\left[\frac{mk}{m-1}\right] = p = \frac{mk}{m-1} - \frac{r}{m-1}$, so that

$$\left[\frac{mk}{m-1}\right](m-1) = mk-r.$$

Substituting this value into Equation (25), we obtain

$$F_m\left(\left[\frac{mk}{m-1}\right] + 1,\ m-r\right) \quad , \text{ where } 2 \leq m-r \leq m-1.$$

Now, by Lemma 3, Equation (20),

$$F_m\left(\left[\frac{mk}{m-1}\right] + 1,\ m-r\right) = F_m\left(\left[\frac{mk}{m-1}\right],\ m-r+1\right) \quad .$$

This latter equation may be rewritten, obviously, as

$$(26)\quad F_m\left(\left[\frac{mk}{m-1}\right] + 1, m-r\right) = \sum_{i=\left[\frac{mk}{m-1}\right]}^{\left[\frac{mk}{m-1}\right]} F_m(i, m-r+1) = \sum_{i=\left[\frac{mk}{m-1}\right]}^{\left[\frac{mk}{m-1}\right]} F_m(i, (m-1)i-(mk-(m+1))).$$

But, this last sum is precisely of the form of Equation (24), where $N_k = \left[\frac{mk}{m-1}\right]$, and $n = \left[\frac{mk}{m-1}\right]$. This concludes the base of the induction.

Now assume the Theorem holds for some integer N such that $N \geq \frac{mk}{m-1}$, where $m \geq 2$ and $k \geq 1$ are fixed integers. Suppose now that $n = N+1$. Then we have, from Lemma 3, Equation (23),

$$F_m(N+1, (N+1)(m-1)-(mk-1))$$

$$= F_m(N, (N+1)(m-1)-(mk-1)+1) + F_m(N, (N+1)(m-1)-(mk-1) + 1-m)$$

Rewriting this equation, we have,

$$= F_m(N, N(m-1)-(mk-(m+1))) + F_m(N, N(m-1)-(mk-1));$$

and, by the induction hypothesis on the second term, we have,

$$= F_m(N, N(m-1)-(mk-(m+1))) + \sum_{i=N_k}^{N-1} F_m(i, i(m-1)-(mk-(m+1))).$$

This obviously becomes,

$$= \sum_{i=N_k}^{N} F_m(i, i(m-1)-(mk-(m+1))), \text{ concluding the proof.}$$

Theorem 4.

Let m, k, and n be integers such that $m \geq 2$, $k \geq 1$, and $n \geq \frac{mk}{m-1}$. Then

$$(27)\quad \sum_{i=N_k}^{n-1} F_m(i, i(m-1)-(mk-(m+1))) =$$

$$\sum_{i_k=N_k}^{n-1} \sum_{i_{k-1}=N_{k-1}}^{i_k-1} \cdots \sum_{i_2=N_2}^{i_3-1} \sum_{i_1=1}^{i_2-1} 1,$$

where $N_k = \begin{cases} \frac{mk}{m-1} - 1 & \text{if } (m-1) \text{ divides } k \\ \left[\frac{mk}{m-1}\right] & \text{otherwise.} \end{cases}$

Proof.

The Theorem will be proved by induction on k. Let k=1, and let m and n be arbitrary integers such that $m \geq 2$ and $n \geq \frac{m}{m-1}$. Then, by Lemma 3, Equation (21), it follows immediately that

$$\sum_{i_1=N_1}^{n-1} F_m(i_1, i_1(m-1)+1) = \sum_{i_1=N_1}^{n-1} 1.$$

Now assume that Equation (27) holds for some $K \geq 1$, and $m \geq 2$ and $n \geq \frac{mK}{m-1}$, where K, m, and n are integers. Then we have, from Theorem 3,

$$\sum_{i_{K+1}=N_{K+1}}^{n-1} F_m(i_{K+1}, i_{K+1}(m-1)-((m(K+1)-1)) =$$

$$\sum_{i_{K+1}=N_{K+1}}^{n-1} \sum_{i_K=N_K}^{i_{K+1}-1} F_m(i_K, i_K(m-1)-(mk-1))).$$

By the induction hypothesis, we have,

$$\sum_{i_{K+1}=N_{K+1}}^{n-1} \sum_{i_K=N_K}^{i_{K+1}-1} \cdots \sum_{i_2=N_2}^{i_3-1} \sum_{i_1=1}^{i_2-1} 1$$

Thus, concluding the proof of Theorem 4.

We now derive as a special case of Theorem 4, $t_n^{(m)}$, the number of m-ary trees with n terminal nodes. We observe that if a complete m-ary tree has n terminal nodes, then n is an integer of the form

$$n = m+(m-1)j,\ j = 0, 1, 2, \cdots .$$

In addition, if N is the total number of nodes in such a tree, then

$$N = (m+1) + mj,\ j = 0, 1, 2, \cdots .$$

Now,

$$t_n^{(m)} = F(N-m,n)$$

and from the above relations that define the form of n in terms of m and j, we have,

$$(28)\quad t^{(m)}_{m+(m-1)j} = F_m(1+mj,m).$$

Applying Theorem 3 to Equation (28), we obtain

$$t^{(m)}_{m+(m-1)j} = \sum_{i=N_{(m-1)j}}^{mj} F_m(i,i(m-1)-(mj(m-1)-(m+1))),$$

where in Equation (24), we have let $n=1+mj$, and $k=j(m-1)$. From Theorem 4, we obtain,

$$(29)\quad t_{m+(m-1)j} = \sum_{i_{(m-1)j}=N_{(m-1)j}}^{mj} \sum_{i_{(m-1)j-1}=N_{(m-1)j-1}}^{i_{(m-1)j}-1} \cdots \sum_{i_1=1}^{i_2-1}$$

for $j \geq 1$, $m \geq 2$,

$$\text{where } N_t = \begin{cases} \frac{mt}{m-1} - 1 & \text{if } m-1 \text{ divides } t \\ \left[\frac{mt}{m-1}\right] & \text{otherwise.} \end{cases}$$

Equation (29) gives an explicit formula for the number of m-ary trees with n terminal nodes, where n is of the form $m+(m-1)j$, $j=0, 1, \ldots$.

3.3 A Conjectured Formula

In the previous sections we have derived explicit formulae for the number of complete m-ary trees. We will, in this section, provide an alternative formula for the number of complete m-ary trees. The formula was developed by observation of how the complete trees are parsed. However, we have not derived a rigorous proof that the formula is correct. We have verified that the conjectured formula given below works in a large number of cases. We provide the formula here since the computer implementation of the conjectured formula is significantly faster than computing the number of complete m-ary trees for either of the formulae given in Section 3.1 or Section 3.2.

Conjecture

The number of complete m-ary trees with n terminal nodes, where n is of the form $m+(m-1)r$, $r=0, 1, 2, \ldots$, and $m=2, 3, \ldots$, is given as follows:

$$t_1^{(m)} = 1$$

$$t_n^{(m)} = \sum_{i=1}^{n-1} a_{i,n}^{(m)} ,$$

where,

$$a_{i,n}^{(m)} = 0 \text{ for } i>n-m+1,$$

$$a_{1,m}^{(m)} = 1 \text{ for all } m ,$$

and $$a_{k,n} = \sum_{i=1}^{k} a_{i,n-m+1}.$$

The reason for the ease in computation may be seen by observing the following triangles for cases, m=2 and m=3.

$$a^{(2)}_{1,n} \quad a^{(2)}_{2,n} \quad a^{(2)}_{3,n} \quad a^{(2)}_{4,n} \quad a^{(2)}_{5,n}$$

$t^{(2)}_2 = 1 = 1$

$t^{(2)}_3 = 2 = 1 + 1$

$t^{(2)}_4 = 5 = 1 + 2 + 2$

$t^{(2)}_5 = 14 = 1 + 3 + 5 + 5$

$t^{(2)}_6 = 42 = 1 + 4 + 9 + 14 + 14$

...

$$a^{(3)}_{1,n} \quad a^{(3)}_{2,n} \quad a^{(3)}_{3,n} \quad a^{(3)}_{4,n} \quad a^{(3)}_{5,n} \quad a^{(3)}_{6,n} \quad a^{(3)}_{7,n}$$

$t^{(3)}_3 = 1 = 1$

$t^{(3)}_5 = 3 = 1 + 1 + 1$

$t^{(3)}_7 = 12 = 1 + 2 + 3 + 3 + 3$

$t^{(3)}_9 = 55 = 1 + 3 + 6 + 9 + 12 + 12 + 12$

...

To indicate the speed in computing the conjectured formula, in contrast to that of the formula in Section 3.2.2, to find the 3-ary trees up to t_{21}, it took in the order of minutes for the latter, while the former was obtained in the order of milliseconds.

4. Enumeration of Multiple Descendent Trees with n Terminal Nodes

We now consider trees with an arbitrary number of branches emanating from each node. Specifically, we consider the enumeration of multiple descendent trees which were defined in Section 2. We shall derive an explicit formula for the generating function and, an explicit formula for t_n, the number of multiple descendent trees with exactly n terminal nodes. Specifically, the result that we shall now prove is stated as follows:

Theorem 5.

Let t_n denote the number of multiple descendent trees with n terminal nodes. Then, the generating function,

$$(30)\quad T(x) = \sum_{n=1}^{\infty} t_n x^n \text{ is given by the functional equation}$$

$$(31)\quad 2T^2(x) - (x-1)\,T(x) + x = 0,$$

which may be solved to yield

$$(32)\quad T(x) = \frac{x+1}{4} - \frac{1}{4}\sum_{i=0}^{\infty} \binom{\frac{1}{2}}{i} (x^2 - 6x)^i.$$

Furthermore, $t_1 = 1$, and for $n>1$, t_n satisfies the relationship

$$(33)\quad t_n = \sum_{\substack{i_1,i_2 \\ i_1+i_2=n}} t_{i_1} t_{i_2} + \sum_{\substack{i_1,i_2,i_3 \\ i_1+i_2+i_3=n}} t_{i_1} t_{i_2} t_{i_3} + \cdots + \sum_{\substack{i_1,\ldots,i_n \\ i_1+\cdots+i_n=n}} t_{i_1} \cdots t_{i_n},$$

for $n>1$.

An explicit formula for t_n is given by

$$(34)\quad \begin{cases} t_1 = 1 \\ t_{2n-1} = -\frac{1}{4}\sum_{i=n}^{2n-1} \binom{\frac{1}{2}}{i}\binom{i}{2n-i-1}\left(-6\right)^{2(i-n)+1} & ,n>1 \\ t_{2n} = -\frac{1}{4}\sum_{i=n}^{2n} \binom{\frac{1}{2}}{i}\binom{i}{2n-i}\left(-6\right)^{2(i-n)} & ,n\geq 1. \end{cases}$$

Proof.

The assumption that the trees are multiple descendent trees prevents long strings of the forms shown in Figure 7.

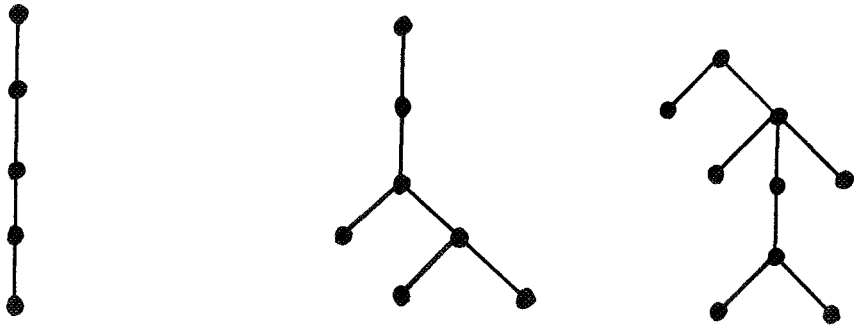


Figure 7

Trees Excluded from Consideration

The trees shown would be considered as mapping into the trees shown in Figure 8.

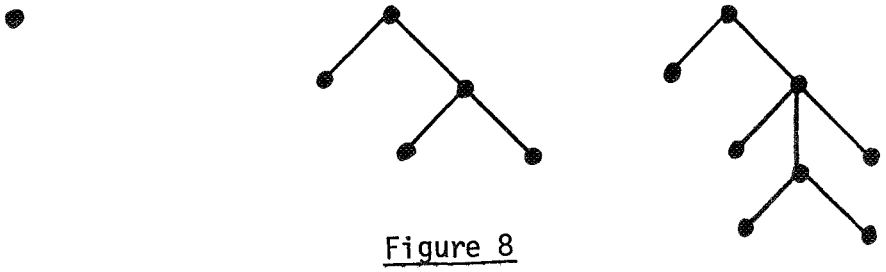


Figure 8

Multiple Descendent Trees

Now consider an arbitrary tree of the appropriate form. We observe the way in which we can generate trees with n terminal nodes. We can generate such trees by starting with a tree that has at its topmost node exactly two branches, one with exactly three branches, proceeding in sequence to a tree with exactly n branches from the root. Now, select an arbitrary tree from this set, say it has m branches emanating from the root. Then, consider the nodes at the next level; they consist of trees with a certain number of terminal nodes. The sum total of all terminal nodes in each of the m trees must be equal to n. Hence, we must have the relation

$$(33)\quad t_n = \sum_{\substack{i_1, i_2 \\ i_1+i_2=n}} t_{i_1} t_{i_2} + \sum_{\substack{i_1, i_2, i_3 \\ i_1+i_2+i_3=n}} t_{i_1} t_{i_2} t_{i_3} + \cdots + \sum_{\substack{i_1, \cdots, i_n \\ i_1+\cdots+i_n=n}} t_{i_1} \cdots t_{i_n},$$

as was to be shown.

Now consider the generating function for such trees,

$$(30)\quad T(x) = \sum_{n=1}^{\infty} t_n x^n ,$$

where the t_n are given by Equation (33), it is easy to see that the m^{th} power of the generating function has the form

$$(35)\quad T^m(x) = \sum_{n=1}^{\infty} t_n^{(m)} x^n ,$$

where

$$(36)\quad t_n^{(m)} = \sum_{\substack{i_1, \cdots i_m \\ i_1+i_2+\cdots+i_m=n}} t_{i_1} \cdots t_{i_m} .$$

Then, from Equations (33), (30), (35), and (36), it may be seen that

$$T(x) - \sum_{i=2}^{\infty} T^i(x) = x \ ,$$

and hence,

$$T(x) - T^2(x) \sum_{i=0}^{\infty} T^i(x) = x \ ,$$

which is readily seen to be

$$T(x) - \frac{T^2(x)}{1-T(x)} = x \ .$$

Hence, T(x) satisfies

(31) $2T^2(x) - (x+1)T(x) + x = 0.$

Solving this quadratic equation, we readily obtain (32). Expanding the sum to obtain powers of x, we obtain (34). This completes the proof of Theorem 5.

By computing the first few values of t_n, one finds that the generating function may be rewritten as

$$T(x) = x + x^2 + 3x^3 + 11x^4 + 45x^5 + 197x^6 + \cdots \ .$$

5. Enumeration of Multiple Descendent m-ary Trees with n Terminal Nodes

We consider now the number of multiple descendent m-ary trees with n terminal nodes. Let the generating function for these trees be given by

$$(37)\quad U_m(x) = u_1^{(m)}\, x + u_2^{(m)}\, x^2 + u_3^{(m)}\, x^3 + \cdots$$

We seek a relationship which gives $u_n^{(m)}$, the number of multiple descendent m-ary trees with n terminal nodes. If $n \leq m$ then it is obvious that the number of such trees is equal to the number of multiple descendent trees with n terminal nodes. An explicit formula for the latter has been derived in the previous Section. For the present purposes, it suffices to give the recurrence relationship for $u_n^{(m)}$:

$$(38)\quad u_n^{(m)} = \sum_{\substack{i_1, i_2 \\ i_1+i_2=n}} u_{i_1}^{(m)}\, u_{i_2}^{(m)} + \cdots + \sum_{\substack{i_1, \cdots, i_n \\ i_1+\cdots+i_n=n}} u_{i_1}^{(m)} \cdots u_{i_n}^{(m)},$$

for $n \leq m$.

If $n>m$, then by considering the possible subtrees of the root as we have done previously, one obtains the relationship

$$(39)\quad u_n^{(m)} = \sum_{\substack{i_1, i_2 \\ i_1+i_2=n}} u_{i_1}^{(m)}\, u_{i_2}^{(m)} = \sum_{\substack{i_1, i_2, i_3 \\ i_1+i_2+i_3=n}} u_{i_1}^{(m)}\, u_{i_2}^{(m)}\, u_{i_3}^{(m)} +$$

$$\cdots + \sum_{\substack{i_1, \cdots i_m \\ i_1+\cdots+i_m=n}} u_{i_1} \cdots u_{i_m}, \text{ for } n>m.$$

Equation (38) and Equation (39), therefore, define the coefficients of the generating function of Equation (37).

6. Conclusion

In this paper we have considered the enumeration of trees based upon the number of terminal nodes that the trees may contain. The types of trees considered were the multiple descendent trees, the multiple descendent m-ary trees, and the complete m-ary trees. The motivation for these explorations stemmed from the authors' consideration of the composing of binary relations. Since the resulting strings correspond to the complete m-ary trees, these trees have been treated most extensively. Explicit formulae have been developed for the complete m-ary trees both from the generating function and the syntax recognition approaches. Certain ways of analyzing the linear strings representing the trees have also led us to a conjectured formula.

To complete the study of the enumeration of the trees based on the number of terminal nodes, the authors have developed the explicit formula for the number of multiple descendent trees and a recurrence relationship for the number of multiple descendent m-ary trees. The latter result lends itself to further extension in that it is likely that an explicit formula may be obtained.

The results obtained on the number of complete m-ary trees were used to obtain upper bounds on the number of parse trees which may be constructed for strings of given length from context-free languages of specified types.

References

[1] Hopcroft, J. E., and J. D. Ullman. Formal Languages and Their Relation to Automata. Addison-Wesley, Reading, Mass., 1969.

[2] Knuth, D. E. The Art of Computer Programming, Volume 1/Fundamental Algorithms. Addison-Wesley, Reading, Mass., 1968, pp. 388-389.

[3] Scoins, H. I. "Linear Graphs and Trees." In: Michie, D. (Ed.). Machine Intelligence 1, American Elsevier, New York, 1967, pp. 3-15.

[4] Scoins, H. I. "Placing Trees in Lexicographic Order." In: Michie, D. (Ed.). Machine Intelligence 3, American Elsevier, New York, 1968, pp. 43-60.

[5] Riordan, J. Introduction to Combinatorial Analysis. John Wiley and Sons, New York, 1958.

Appendix

This Appendix includes the program written in FORTRAN V to implement the formula given in Section 3.2.2, and the program written in MAD to implement the formula in Section 3.3. Both programs were written for and executed on the UNIVAC 1108.

We also include the computer printouts of the MAD program as Table A1 through Table A19. In these Tables, we have, in the column labeled SUM, given the values of $t_j^{(k)}$ for each value of k in the range $2 \leq k \leq 20$, and values of $j = k+(k-1)(i-1)$, where $1 \leq i \leq n$ for the value of n indicated in each Table. In Tables A1 through A4 and Table A9, several of the values are asterisked to indicate that these were verified by the FORTRAN program. Complete verification of the values was not attempted because to do so would have required excessive computer time.

```
      IMPLICIT INTEGER(A-Z)
      DIMENSION BASE(200),LB(200),UB(200)
    1 READ(5,100)M,N
      WRITE(6,101)M,N,M
      J=1
      M1=M-1
    2 SUM=0
      IMAX=M1*J
      LB(IMAX)=IMAX+J-1
      UB(IMAX)=IMAX+J
      IF(IMAX.EQ.1)GO TO 11
      I=IMAX-1
    3 K=I/M1
      KK=M1*K
      IF(I.EQ.KK)GO TO 4
      BASE(I)=M*I/M1
      GO TO 5
    4 BASE(I)=M*K-1
    5 IF(I.LE.1)GO TO 6
      I=I-1
      GO TO 3
    6 I=IMAX-1
    7 LB(I)=BASE(I)
      UB(I)=LB(I+1)-1
      IF(I.EQ.1)GO TO 8
      I=I-1
      GO TO 7
    8 SUM=SUM+UB(1)
      I=2
    9 IF(LB(I).EQ.UB(I))GO TO 10
      LB(I)=LB(I)+1
      I=I-1
      GO TO 7
   10 IF(I.EQ.IMAX)GO TO 12
      I=I+1
      GO TO 9
   11 SUM=UB(1)
   12 K=M+IMAX
      WRITE(6,102)K,SUM
      IF(J.EQ.N)GO TO 1
      J=J+1
      GO TO 2
  100 FORMAT(I2,1X,I2)
  101 FORMAT(1H1,9X,'THE FOLLOWING TABLE GIVES THE NUMBER OF ',I2,
     1   '-ARY TREES',/,10X,'WITH K TERMINAL NODES (1ST ',I2,
     2   ' TERMS IN THE SEQUENCE)',///,25X,'K',4X,'NUMBER OF TREES',
     3   //,25X,'1',15X,'1',//,24X,I2,15X,'1')
  102 FORMAT(1H0,23X,I2,4X,I12)
      END
```

Figure A1.

FORTRAN V Program for the Number of m-ary Trees

```
            NORMAL MODE IS INTEGER
            DIMENSION A(100*100),SUM(100)
BEGIN
            READ DATA N,K,SKIP
            K1=K-1
            K2=2*K-3
            K3=K1-K2
            A(1,1)=1
            THROUGH L1,FOR I=1,1,I.G.K
L1          A(2,I)=1
            THROUGH L3,FOR I=3,1,I.G.N
            JJ=I*K1-K2
           *** K1 AND K2 WERE COMPUTED TO SPEED UP THE
           *** COMPUTATION OF THE UPPER LIMIT, K+(I-3)*(K-1).
            THROUGH L2,FOR J=1,1,J.G.JJ
            A(I,J)=0
            THROUGH L2,FOR L=1,1,L.G.J
L2          A(I,J)=A(I,J)+A(I-1,L)
            M1=JJ+1
            M2=JJ+K1
            THROUGH L3,FOR M=M1,1,M.G.M2
L3          A(I,M)=A(I,M-1)
            THROUGH L4,FOR I=1,1,I.G.N
            SUM(I)=0
            JJ=I*K1+K3
            THROUGH L4,FOR J=1,1,J.G.JJ
L4          SUM(I)=SUM(I)+A(I,J)
            PRINT FORMAT$1H1,S10,16HTABLE OF VALUES  *$
            PRINT FORMAT$S14,2HK= ,I2,3H,N=,I2,/*$,K,N
            PRINT FORMAT$S12,1HI,S9,3HSUM,/*$
            THROUGH L5,FOR I=1,1,I.G.N
            WHENEVER SKIP.E.1
            JJ=I*K1+K3
            PRINT FORMAT$ /,I2,I12,3H = ,10I11*$,I,SUM(I),
          1   A(I,1),...A(I,JJ)
            OTHERWISE
            PRINT FORMAT$S11,I2,I12*$,I,SUM(I)
            END OF CONDITIONAL
L5
            TRANSFER TO BEGIN
            END OF PROGRAM
```

Figure A2.

MAD Program for the Number of m-ary Trees

I	SUM
1	1 *
2	2 *
3	5 *
4	14 *
5	42 *
6	132 *
7	429 *
8	1430 *
9	4862 *
10	16796 *
11	58786 *
12	208012
13	742900
14	2674440
15	9694845
16	35357670
17	129644790
18	477638700
19	1767263190
20	6564120420

TABLE A1. K=2, N=20

I	SUM
1	1 *
2	3 *
3	12 *
4	55 *
5	273 *
6	1428 *
7	7752 *
8	43263 *
9	246675 *
10	1430715 *
11	8414640
12	50067108
13	300830572
14	1822766520
15	11124755664

TABLE A2. K=3, N=15

I	SUM
1	1 *
2	4 *
3	22 *
4	140 *
5	969 *
6	7084 *
7	53820
8	420732
9	3362260
10	27343888

TABLE A3. K=4, N=10

I	SUM
1	1 *
2	5 *
3	35 *
4	285 *
5	2530 *
6	23751 *
7	231880
8	2330445
9	23950355
10	250543370

TABLE A4. K=5, N=10

* Results verified with FORTRAN V program.

I	SUM
1	1
2	6
3	51
4	506
5	5481
6	62832
7	749398
8	9203634
9	115607310
10	1478314266

TABLE A5. K=6, N=10

I	SUM
1	1
2	7
3	70
4	819
5	10472
6	141778
7	1997688
8	28989675
9	430321633
10	6503352856

TABLE A6. K=7, N=10

I	SUM
1	1
2	8
3	92
4	1240
5	18278
6	285384
7	4638348
8	77652024
9	1329890705
10	23190029720

TABLE A7. K=8, N=10

I	SUM
1	1
2	9
3	117
4	1785
5	29799
6	527085
7	9706503
8	184138713
9	3573805950

TABLE A8. K=9, N=9

I	SUM
1	1*
2	10*
3	145*
4	2470*
5	46060*
6	910252
7	18730855
8	397089550
9	8612835715

TABLE A9. K=10, N=9

I	SUM
1	1
2	11
3	176
4	3311
5	68211
6	1489488
7	33870540
8	793542167
9	19022518084

TABLE A10. K=11, N=9

I	SUM
1	1
2	12
3	210
4	4324
5	97527
6	2331924
7	58068792
8	1489899060

TABLE A11. K=12, N=8

I	SUM
1	1
2	13
3	247
4	5525
5	135408
6	3518515
7	95223414
8	2655417765

TABLE A12. K=13, N=8

* Results verified with FORTRAN V program.

I	SUM
1	1
2	14
3	287
4	6930
5	183379
6	5145336
7	150374056
8	4528486314

TABLE A13. K=14, N=8

I	SUM
1	1
2	15
3	330
4	8555
5	243090
6	7324878
7	229906300
8	7435946115

TABLE A14. K=15, N=8

I	SUM
1	1
2	16
3	376
4	10416
5	316316
6	10187344
7	341772552
8	11815724400

TABLE A15. K=16, N=8

I	SUM
1	1
2	17
3	425
4	12529
5	404957
6	13881945
7	495729741
8	18243038385

TABLE A16. K=17, N=8

I	SUM
1	1
2	18
3	477
4	14910
5	511038
6	18578196
7	647859273
8	26401476771

TABLE A17. K=18, N=8

I	SUM
1	1
2	19
3	532
4	17575
5	636709
6	24467212
7	759312631

TABLE A18. K=19, N=7

I	SUM
1	1
2	20
3	590
4	20540
5	784245
6	31763004
7	863917560

TABLE A19. K=20, N=7