Note

Multidimensional trees

William A. Baldwin
Park College, Parkville, MO, USA

George O. Strawn
Iowa State University, Ames, IA, USA

Communicated by M. Nivat
Received November 1989
Revised November 1990

Abstract


A new data structure is presented which may be used to specify programming languages. It is called a multidimensional tree. It is an extension of the normal concept of a tree, which is a two-dimensional concept, into higher dimensions. It is shown that a string may be considered a one-dimensional tree, and a node by itself, a zero-dimensional tree. Forests are defined, analogous to tree forests, except that they may have higher dimensions. A frontier operation is defined which reduces the dimension of its argument by one, enabling strings to be generated from higher dimensional objects. Grammars and automata are defined which allow the definition of languages of multidimensional trees, and therefore string languages. It is shown that this concept is useful in computations — there are useful internal representations. The language hierarchy is related to the language classes of regular, context free, and macro languages.

1. Background

There have been many schemes proposed to define formal languages. For example, context free grammars and context sensitive grammars are well known. Also, macro grammars [6] and indexed grammars [1] have been defined but are not as well known. A method based on algebras has been defined by Damm [5] and others. Other lesser known methods have also been defined.

Each of these methods is a generative scheme for defining the languages. That is, given the scheme, one can generate the elements of the resulting language. Although parsing methods exist, no clue is given as to how this is done by the generative method. Finite automata and push down automata form another class
of language definition models. Also, nested stack automata [2] have been introduced. These form a second set of models where the scheme does not generate the elements of the language, but rather recognizes whether a particular string is in the language.

This paper will consider a generalization of both techniques which use a new data structure called a multidimensional tree.

A third method has also been used. This is called regular expressions. Here, an expression is written such that it uniquely determines a language. Such expressions are well known for regular languages— that is where the language class got its name. They have also been defined for tree languages, and therefore context free sets [10]. Although such a system has been defined for multidimensional trees [4], it will not be presented here.

2. Introduction

The definition of trees that is used in most literature on the topic begins with the definition of a ranked set. A ranked set is a set with a function mapping the elements of the set to the set of positive integers. This function is known as the rank. The definition of trees, then, require that any node, which must be an element of the ranked set, can only have the number of subtrees specified by the rank.

This definition has several problems. It does not correspond to the idea of “tree” that is presented in most freshman or sophomore computer science courses. Usually, the family tree is given as an example. Any one family may have any number of children, so requiring trees to have a fixed number of children is not reasonable. Another more relevant example is the syntax trees used to define a language. If an expression is made up of terms, it is reasonable that all terms should be on the same level of the tree, that is, no one term is more “special” than any other. It makes no difference whether the expression is evaluated from starting from the right, the left, or if it is evaluated in random order. On a multiprocessor system, it may be possible that all of the terms will be evaluated at the same time! Other examples could be given, from natural language processing to operating systems, the idea that a node in a tree can only have a fixed number of children is not acceptable.

Ordered trees can be defined in a fashion which does not have this restriction. Let the nodes be from any set. Then there is one node in the tree which is special in that it does not have any arc going into it called the root node. The root node has zero or more subtrees under it arranged in some order, that is, there is a first subtree, a second subtree, etc. These may be regarded as a string of subtrees. This forms a basis for the extension of trees to more dimensions.

There is a connection between strings and trees. In both cases, there is a special node, called the root of the tree, or front of the string. If the string is represented using arcs as in a linked list, then the head is the node without an arc going into it. In general if the head is deleted from the string, what follows is not a string of
strings, but rather another single string. That is a node of strings! Therefore, a string can be defined recursively as a head node followed by a node of strings.

To summarize, a tree is a node followed by a string of trees and a string is a node followed by a node of strings. Going in the other direction, a “three-dimensional” tree is a node followed by a tree of sub-“three-dimensional” trees.

If these are represented in a graphical form, it can be seen that these objects have dimensions. In particular, a node has no dimension at all, just as a geometric point has no dimension. A string has one dimension, called the length, as a line has length. A string could be drawn on a one-dimensional field if the nodes really had no dimension. A tree, on the other hand, has two dimensions, the depth and the breadth. It requires a two-dimensional space to draw the tree. The object that has just been defined has the depth and breadth of the tree, but in addition there is one more dimension as yet unnamed. It would require three dimensions to draw this object. Having made the step once in the generation of three-dimensional trees, it should be obvious that the same step can be made again to a four-dimensional tree, to a five, and, in fact, to any positive integer dimension that is desired.

3. Formal definitions

To formally define trees, given the discussion above, it would be desirable to first formally define “string”. Before that can be defined, however, “node” must be defined. All will be based on a set, denoted as Σ, which will be the alphabet. This set will not be further defined, and may be either finite, or infinite. Thus, a “string of trees” can have a formal meaning even though the set of trees is, in fact, infinite.

Since the hierarchy discussed above is to be extended into new dimensions, and since the notation used for strings and trees is not consistent, a new notation needs to be developed. The set of nodes over Σ is a zero-dimensional “tree” and will, therefore, be denoted as \( T_0(\Sigma) \). The set of strings over Σ is a one-dimensional tree and will be denoted as \( T_1(\Sigma) \). Trees themselves, similarly, will be denoted as \( T_d(\Sigma) \).

**Definition 3.1.** The set of nodes over set Σ, denoted \( T_0(\Sigma) \), is the smallest set such that Σ is a subset of \( T_0(\Sigma) \) and \( \emptyset \) is a subset of \( T_0(\Sigma) \).

**Definition 3.2.** The set of strings over set Σ, denoted \( T_1(\Sigma) \), is the smallest set such that Σ is a subset of \( T_1(\Sigma) \) and Σ\{\emptyset\} \( T_0(T_1(\Sigma)) \) is a subset of \( T_1(\Sigma) \).

In this case, the subscripted brackets \{∅ and \} have been introduced. These are needed because as the number of dimensions increase, the number of types of brackets also increase without bound. The one level brackets refer to strings, the two level brackets will refer to trees, the three level to “three-dimensional trees”, etc.
Definition 3.2. The set of trees over set \( \Sigma \), denoted \( T_2(\Sigma) \), is the smallest set such that \( \Sigma \) is a subset of \( T_2(\Sigma) \) and \( \Sigma \{ T_1(T_2(\Sigma)) \} \) is a subset of \( T_2(\Sigma) \).

At this point, a pattern is obvious. The only changes between the definition of "tree" and the definition of "string" is the name, and all of the numbers were increased by one. This indicates that the general definition should have each of these numbers incremented by the appropriate amount depending on the dimension.

Definition 3.4. The set of multidimensional trees over set \( \Sigma \) of dimension \( n \), denoted \( T_n(\Sigma) \), where \( n \) is greater than or equal to zero, is the smallest set such that \( \Sigma \) is a subset of \( T_n(\Sigma) \) for all \( n \geq 0 \) and \( \Sigma \{ n, T_n(\Sigma) \} \) is a subset of \( T_n(\Sigma) \) if \( n > 0 \).

The "(\( \Sigma \))" in the above discussion will be dropped if it is not needed to indicate what set the trees are over. This will result in the more compact notation \( T_n \) being used to mean \( T_n(\Sigma) \) if the set \( \Sigma \) is obvious. Thus, the first set in the second point of the definition above may be written as \( \Sigma \{ n, T_{n-1}(T_n) \} \).

In order to simplify the definition somewhat, an alternative definition is given in the following theorem.

Theorem 3.5. \( T_n(\Sigma) \) is the smallest set such that \( T_0(\Sigma) = \Sigma \) and \( T_n(\Sigma) = \Sigma \cup \Sigma \{ n, T_{n-1}(T_n) \} \) for all \( n \geq 0 \).

4. Multidimensional forests

In proving theorems about trees it is often convenient to prove similar theorems about forests of trees, and then restrict the result to trees. For multi-dimensional trees, a similar concept exists. It is called a multidimensional forest. The idea can be illustrated by removing the root nodes from the set of multidimensional trees as given in the definition above. That leaves \( T_{n-1}(T_n) \). If this expression is expanded using the definition, it can be seen that

\[
T_{n-1}(T_n) = T_n \cup T_n \{ n, T_{n-2}(T_n) \} \}
\]

If the root node is removed from this set the expression \( T_{n-2}(T_n) \) remains. In general, any expression, \( T_k(T_{k+1}(\ldots T_{n-1}(T_n)) \ldots)) \) can appear if the proper number of recursions are unwound in the fashion given above. This forms the informal definition of multidimensional forests and will be denoted as \( T_n^k(\Sigma) \). As was the case with multidimensional trees, the \( (\Sigma) \) will be dropped if the set is obvious. A more formal definition is given below.

Definition 4.1. The set of multidimensional forests over set \( \Sigma \) of dimension \( n \) and degree \( k \), denoted \( T_n^k(\Sigma) \), where \( n \) is greater than or equal to zero and \( k \) is less than or equal to \( n \) and greater than zero is defined such that

\[
T_n^k(\Sigma) = \begin{cases} 
\Sigma & \text{if } k = n + 1, \\
T_k(T_n^{k+1}(\Sigma)) & \text{otherwise.}
\end{cases}
\]
It can be easily seen that \( T_n^a = T_n \) for all \( n \).

The definition given above is difficult to use in proving theorems because properties which need to be proved would need to be known for trees before they could be shown to hold for forests. This situation is changed by the introduction of an alternative definition in the following theorem which does not depend on the definition of multidimensional trees.

**Theorem 4.2.** \( T_n^k(\Sigma) \) is the smallest set such that

\[
T_n^k(\Sigma) = \begin{cases} 
\Sigma & \text{if } k = n + 1, \\
T_n^{k+1}(\Sigma) \cup T_n^{k+1}(\Sigma)_{k} \cup T_n^{k-1}(\Sigma)_{k} & \text{if } n \geq k > 0, \\
T_n^1(\Sigma) & \text{if } k = 0,
\end{cases}
\]

for all \( n \geq 0 \).

Later in this paper grammars will be defined. This will require that non-terminals be replaced with something in the manner of a grammar. Therefore, a new type of multidimensional forest must be defined which allows some of the elements on the “frontier” to be replaced by multidimensional trees or forests in such a way that the resulting objects are still legal multidimensional forests. To allow the maximum flexibility, the “arguments” are elements of a ranked set where each rank will correspond to a given dimensional forest which can replace the element.

**Definition 4.3.** A multidimensional forest with variables \( V \) over set \( \Sigma \) of dimension \( n \) and degree \( k \), denoted \( T_n^k(\Sigma, V) \), where \( n \) is greater than or equal to zero, \( k \) is less than or equal to \( n \) and greater than zero, and \( V \) is a ranked set, is defined as the smallest set such that

1. \( T_n^k(\Sigma, V) \subseteq T_n^k(\Sigma, V) \),
2. if \( \alpha \in T_n^k(\Sigma, V) \), \( \beta \in T_n^k(\Sigma, V) \), \( \exists \alpha_1, \alpha_2, \alpha = \alpha_1 \beta \alpha_2 \), and \( v \in V_i \), then \( \alpha_1 \alpha_2 v \in T_n^k(\Sigma, V) \).

If \( \alpha \in T_n^k(\Sigma, V) \) and a variable \( v \in V_i \) which is included in \( \alpha \) is replaced by an element \( \beta \) of \( T_n^k(\Sigma, V) \), then the resulting expression is an element of \( T_n^k(\Sigma, V) \) as well.

As has been the case, if the sets \( \Sigma \) and \( V \) are known, then the set \( T_n^k(\Sigma, V) \) will be written \( T_n^k \).

**Theorem 4.4.** \( T_n^k(\Sigma, V) \) is the smallest set such that

\[
T_n^k(\Sigma, V) = \begin{cases} 
\Sigma \cup V_{n+1} & \text{if } k = n + 1, \\
T_n^{k+1}(\Sigma, V) \cup V_k \cup T_n^{k+1}(\Sigma, V)_{k} \cup T_n^{k-1}(\Sigma, V)_{k} & \text{if } n \geq k > 0, \\
T_n^1(\Sigma, V) \cup V_0 & \text{if } k = 0,
\end{cases}
\]

for all \( n \geq 0 \).
5. The frontier operation

Grammars will be defined for multidimensional forests, and therefore multidimensional trees, given that $T^n_1$ and $T^n_n$ are the same set. These grammars will, of course, yield multidimensional trees and forests, which is not directly of any interest as yet. A method of obtaining languages in a simpler data structure such as strings is desirable. In the classical case, the frontier of a tree language can be taken to arrive at a language over strings [9, 8]. This is also the case here, except the frontier operation needs to be extended to cover the case where the trees have more than two dimensions. Since each frontier operation will reduce the dimension of the object tree by one, it may have to be applied several times to generate strings.

To take the frontier of a tree for classical trees, if the tree is a single node, then the frontier is equal to the tree (except that the single node is taken as a string instead of a tree). If the tree has subtrees, then the frontier is taken of each of the subtrees recursively, forming a string of strings. These are then concatenated in order to form the frontier.

In the two-dimensional case, the frontier operation should proceed in a similar fashion since these are trees. First, if the tree is a single node, then it is changed to a string which is the frontier. If the tree has more than one node, then the root node is dropped and the frontier is taken of each of the subtrees forming a string of strings and the results are "concatenated" in order.

Some thought has to go into what is meant by "concatenated." If more than one dimension is allowed, this becomes more difficult than just writing the trees side by side — the point at which the trees need to be attached needs to be specified. This has been defined for trees [10] similar to the way that the substitution for variables was defined above. Therefore, a special type of variable has been defined called an "$X$". These are allowed to exist in the generated trees by extending the definition of the language associated with a grammar to allow them to exist. In fact, the notation $T^n_1(\Sigma)$ will be relaxed to mean $T^n_1(\Sigma, X)$ where $X$ is the ranked set of all $X$'s unless specified otherwise.

In addition to indicating where the concatenated tree is to be attached, the $X$'s must indicate which tree to attach. This is accomplished by use of a "path expression" added to the notation for the $X$'s as a superscript. Therefore, $X$ will refer to the ranked set of all $X$'s, $X_i$ will refer to the $i$th rank of that set, and $X_i^p$ will refer to an element of the $i$th rank with associated path $p$.

**Definition 5.1.** If $\alpha$ is an element of $T_2(\Sigma)$, then the frontier of $\alpha$, denoted $\text{frontier}_2(\alpha)$ is defined as

1. $\text{frontier}_2(\alpha) = \alpha$ if $\alpha$ is a tree with only one node.
2. If $\alpha$ is not a single node then $\text{frontier}_2(\alpha)$ is arrived at by removing the root node, and recursively applying $\text{frontier}_2$ to each of the subtrees forming a string of strings. These strings are then concatenated in order to form the frontier.
3. $\text{frontier}_2(X_i^p) = X_i^p$. 
If any string in the string of frontiers is not followed by an \( X_i^* \) then all of the following strings are dropped from the frontier. In classical trees, this does not happen since it is assumed that any string can have another string concatenated onto the end of it. However, in the case of concatenating trees [10], this does happen.

The concatenation of the frontiers of subtrees is different in the case where the original subtree was a singleton. In this case, the concatenation is formed by taking the singleton as a root node and following it by the string to be concatenated. This only occurs if the original subtree was a singleton, not if it results in a singleton frontier.

**Theorem 5.2.** \( \text{frontier}^2_{\Sigma}(\alpha) \) is defined as

\[
\text{frontier}^2_{\Sigma}(\alpha) = \begin{cases} 
\alpha & \text{if } \alpha \in \Sigma, \\
\text{concatenate}_1(\text{frontier}^2_{\Sigma}(\beta_0), \text{frontier}^2_{\Sigma}(\beta_1), \ldots) & \text{if } \alpha = a\{ \beta_0, \beta_1, \ldots \}, a \in \Sigma, \text{ and } \\
\beta_i \in T^2_{\Sigma}(\Sigma) \text{ for all } i, & \\
X_i^* & \text{if } \alpha = X_i^*.
\end{cases}
\]

The function \( \text{concatenate}_1 \) will concatenate its arguments by replacing all elements of \( X_i \) with the next string in the sequence unless the string being concatenated was the frontier of an element of \( \Sigma \) as is indicated above.

The function \( \text{frontier}^2_{\Sigma} \) maps \( T^2_{\Sigma} \) to \( T^1_{\Sigma} \), that is, from the set of trees over \( \Sigma \) to the set of strings over \( \Sigma \). This can be easily seen since the right hand side of each of the cases above is an element of \( T^1_{\Sigma} \) assuming that this is true of shorter elements of \( T^2_{\Sigma} \).

**Definition 5.3.** If \( \beta \) is an element of \( T^1_{\Sigma}(\Sigma) \), then the frontier of \( \beta \), denoted \( \text{frontier}^1_{\Sigma}(\beta) \), is defined as

\[
\text{frontier}^1_{\Sigma}(\beta) = \begin{cases} 
\text{frontier}^2_{\Sigma}(\beta) & \text{if } \alpha \in T^2_{\Sigma}(\Sigma), \\
\text{concatenate}_1(\text{frontier}^2_{\Sigma}(\alpha), \text{frontier}^1_{\Sigma}(\beta_1)) & \text{if } \beta = a\{ \beta_1, \ldots \}, a \notin \Sigma, \text{ and } \beta_1 \in T^1_{\Sigma}(\Sigma), \\
a\{ \text{frontier}^1_{\Sigma}(\beta) \} & \text{if } \beta = a\{ \beta_1 \}, a \in \Sigma, \text{ and } \beta_1 \in T^1_{\Sigma}(\Sigma).
\end{cases}
\]

The function \( \text{function}_1^\Sigma \) will map \( T^1_{\Sigma} \) to the set \( T^1_{\Sigma} \), that is, from the set of forest over \( \Sigma \) to the set of strings over \( \Sigma \). The right hand side of each of the lines above is an element of \( T^1_{\Sigma} \) assuming the \( \text{frontier}^2_{\Sigma} \) has the specified range and that the frontier of shorter forest are in this set.

The definition of \( \text{concatenate}_1 \) can be cleaned up by the addition of the third point above. It will take two arguments and concatenate the second with the first by replacing the \( X_i^* \) if it exists in the first argument with the second argument. The rather obscure (but important) case where the original tree was an element of \( \Sigma \) is now entirely handled by the third case above.
Definition 5.4. If $\alpha$ is an element of $T_n(\Sigma)$, then the frontier of $\alpha$, denoted $\text{frontier}_n^\alpha(\alpha)$, is defined as

1. $\text{frontier}_n^\alpha(\alpha) = \alpha$ if $\alpha$ is a single node.
2. If $\alpha$ is not a single node then $\text{frontier}_n^\alpha(\alpha)$ may be arrived at by removing the root node of $\alpha$, recursively applying the frontier$_n^\alpha$ function to all of the sub-$n$-dimensional trees, then concatenating the results.
3. $\text{frontier}_n^\alpha(X_k^p) = X_k^p$ if $k$ is less than $n$.

The concatenate function handles the case where the original tree was a singleton by the same special case given for two-dimensional trees above.

Theorem 5.5. $\text{frontier}_n^\alpha(\alpha)$ is

$$\text{frontier}_n^\alpha(\alpha) = \begin{cases} \alpha & \text{if } \alpha \in \Sigma, \\ \text{concatenate}_{n-1}(\text{frontier}_n^{\beta_0}(\beta_0), \text{frontier}_n^{\beta_1}(\beta_1), \ldots, \text{frontier}_n^{\beta_{n-1}}(\beta_{n-1})) & \text{if } \alpha = a\{n \beta_0 \beta_1 \beta_2 \cdots \beta_{n-1}\}, \\ X_k^p & \text{if } \beta = X_k^p \text{ and } k < n. \end{cases}$$

This definition is a more formal statement of the definition given above. The function concatenate$_{n-1}$, as above, will replace all of the $X_{n-1}$'s with $(n-1)$-dimensional trees. The trees will each be selected from the $(n-2)$-dimensional tree of $n-1$ objects which follow the node containing the $X$. The path in $X_{n-1}^p$ will be used to decide which tree to select. An exception will be made for the case when the original subtree was a single element of $\Sigma$ as above.

Definition 5.6. If $\alpha$ is an element of $T_n^\alpha(\Sigma)$, where $n \geq 0$ and $k = n + 1$, then the frontier of $\alpha$, denoted $\text{frontier}_n^\alpha(\alpha)$, is defined as

1. $\text{frontier}_n^\alpha(\alpha) = \begin{cases} \alpha & \text{if } \alpha \in \Sigma, \\ \text{frontier}_{n-1}^{\beta}(\beta) & \text{if } \alpha = a\{n \beta\}. \end{cases}$
2. $\text{frontier}_{n-1}^{\alpha}(\alpha) = \begin{cases} \alpha & \text{if } \alpha \in X_{n-1}, \\ \text{frontier}_n^\alpha(\alpha) & \text{if } \alpha \in T_n^\alpha(\Sigma), \\ \text{concatenate}_{n-1}(\text{frontier}_n^{\beta}(\beta), \text{frontier}_n^{\gamma}(\gamma)) & \text{if } \alpha = b\{n \gamma - 1\} \text{ and } \beta \in \Sigma, \\ a\{n-1 \text{ frontier}_n^{\gamma}(\gamma)_{n-1}\} & \text{if } \alpha = a\{n, \gamma_{n-1}\} \text{ and } \beta \in \Sigma. \end{cases}$
6. Encodings of multidimensional trees

So far each time this paper has referred to a multidimensional tree, it has used the formal encoding using \( \{k, \}_{k} \). While this notation is good for formal proofs it is not good for visualizing what is happening, or for working with multidimensional trees on a computer. This section will deal with encodings to help the reader understand what is happening as well as use this data structure on a computer.

In the case of a classical node, there is no organization within the node, primarily because no organization is possible.

In classical strings, however, this is not the case. Consider the relationship between the nodes. There is a special node called the head of the string which is special because it has no other nodes in front of it. There is at most one node that can follow this one. Quite often this is indicated by just writing the nodes side by side. This does state a relationship, however, in that the string is considered a different string if two of the nodes are written in a different order, that is, there is an null operator. Therefore, in this paper the relationship “next to” will be specifically indicated by a line being drawn from a node to the node which follows it. This is similar to a linked list in a computer implementation of strings (see Fig. 1 for an example).

In classical trees, a similar situation exists, except that a tree is a two-dimensional object. Usually, lines are drawn from each node to head nodes of all of the subtrees. The relationship between the subtrees is usually assumed by the placement of the subtrees. As was the case above, the order between the nodes of the string of trees will be indicated by a line. However, this line is different than the line used for indicating the relationship between a head node and its subtrees and will, therefore, be represented differently. The line used to indicate the relationship between a node and its subtrees will be a solid line (---), whereas the relationship between the nodes of a string will be represented by a dashed line (---). This is illustrated in Fig. 2.

\[
\text{frontier}^k_n(\alpha) = \begin{cases} 
\alpha & \text{if } \alpha \in X_k, \\
\text{frontier}^{k+1}_n(\alpha) & \text{if } \alpha \in T^{k+1}_n(\Sigma), \\
\text{frontier}^{k+1}_n(\beta) \text{ frontier}^{k-1}_n(\gamma) & \text{if } \alpha = \beta \gamma. 
\end{cases}
\]

Fig. 1. One-dimensional tree.

Fig. 2. Two-dimensional tree.
Extending this idea further, there is a similar method of illustrating three-dimensional trees. The problem is that the three-dimensional tree requires three dimensions to give the corresponding illustration. However, a two-dimensional projection of the three-dimensional object is given in Fig. 3. There are three kinds of lines floating around — those that correspond to strings are illustrated by dashed lines ( - - - ), those corresponding to the two-dimensional object are represented by solid lines ( — ), and the new three-dimensional lines are represented by dotted lines ( ·····).

Fig. 3. Three-dimensional tree.

Four and higher dimensions could also be illustrated by taking the objects and projecting onto the next lower dimension repeatedly until the dimension is two. The problem is each dimension will add a new type of line which would rapidly cause confusion. For that reason, no illustrations are included of this point.

There is an intermediate representation between the formal representation used in the introductory material and the representation used here. Different types of brackets are used for each dimension. In this case, strings may be represented as in the classical — if two nodes are adjacent, then they will be written next to each other with the head node at the left. This will be accomplished by simply deleting the { and }. Thus the string in Fig. 1 may be represented as “abc.” The second dimension could be represented by parentheses. That is, replace { by ( and } by ). Figure 2, therefore, becomes “a(bc(d)e).” In a similar fashion, { and } could be used for the third dimension so that Fig. 3 becomes “a{b(cd{g(hi)}(f)e)}.” Also, [ and ] could be used for the fourth dimension, ⟨ and ⟩ for the fifth dimension, / and \ could be used for the sixth dimension, etc. The types of brackets used is limited only by the imagination of the author and the keys on the keyboard.

These encodings are good for looking at, but are not very usable on a computer. Therefore, another encoding scheme must be used. One possible encoding is called binary encoding. In Theorem 4.4 a single object may be used to make up the multidimensional forest, as in \( T_n^{k+1} \), \( V_k \), and \( \Sigma \), or else two objects make up the tree, as in \( T_n^{k+1} \{k T_n^{k+1} \} \). The multidimensional tree could be encoded by having a binary node encode each multidimensional tree where the two children of any node are the encodings of the multidimensional subtrees which make up the original tree. No value is stored at any node which is not at a leaf. In the case of \( V_k \) or \( \Sigma \) the first pointer in the binary tree could be left at nil indicating that the terminal
object is pointed at by the second pointer. In the case $T_n^{k+1}$ the second pointer could be nil indicating that the only sub-multidimensional forest is pointed to by the first pointer.

The dimension and degree of the encoded forest do not need to be kept except at the top level as this information can be inferred from the position of the node. If the dimension is known, the degree can be derived from the encoding by counting how many nodes need to be traversed through the first node before a leaf node is encountered.

**Definition 6.1.** A binary encoding of multidimensional forest $\alpha$ of dimension $n$ and degree $k$, denoted $\text{binary}_n^\alpha(\alpha)$, is defined recursively as

1. $\text{binary}_n^{n+1}(\alpha) = [\, \alpha \,].$

2. $\text{binary}_n^k(\alpha) = \begin{cases} \text{binary}_n^{k+1}(\alpha), & \text{if } \alpha \in T_n^{k+1}(\Sigma), \\ [\, \alpha \,], & \text{if } \alpha \in V_k, \\ [\text{binary}_n^{k+1}(\beta), \text{binary}_n^{k+1}(\gamma)] & \text{if } \alpha = \beta^k \gamma^k, \end{cases}$

and $n \geq k > 0$.

3. $\text{binary}_n^0(\alpha) = \begin{cases} \text{binary}_n^k(\alpha), & \text{if } \alpha \in T_n^l(\Sigma), \\ [\, \alpha \,], & \text{if } \alpha \in V_0. \end{cases}$

Here the binary tree is encoded by a $[\,$, followed by the first subtree, a comma, the second subtree, and a final closing $\,]$. Null subtrees are indicated by a null string.

So far the encoding of the path expressions has been ignored. This last encoding suggests a workable encoding for paths using bit strings. The suggested encoding is obtained by looking at the nodes used to encode the selecting multidimensional forest as above. Start with a null bit string. Beginning at the root if the selected node is down the first pointer, append a “1” to the encoding and continue by encoding the sub-multidimensional tree. If the selected node is down the second pointer, append a “0” to the encoding and encode the sub-multidimensional tree. By knowing the dimension and degree at the beginning of the path, the dimension and degree at any point down the path can be determined using the same logic as was used for the binary encoding above. The end of the path does not need to be marked since the path will be used to select an element of $T_n^{n+1}$ when the tree that is being traversed is of dimension $n$. As soon as it is detected that the path had led to a degree $n+1$ multidimensional forest, the end of the path is reached. Every path that can appear on an $X$ must have exactly one more “1” in the encoding than “0”’s, since the “0” represent nodes which decrease the degree of the path by one, and “1”’ represents nodes which increase the degree of the path by one.
Definition 6.2. A binary path encoding of the path \( \alpha \) of dimension \( n \) and degree \( k \), denoted \( \text{path}_n^k(\alpha) \), is defined recursively as

\[
\begin{align*}
(1) \quad \text{path}_n^{k+1}(\alpha) &= 0. \\
(2) \quad \text{path}_n^k(\alpha) &= \begin{cases}
1\text{path}_n^{k+1}(\alpha) & \text{if } \alpha \in T_n^{k+1}(\Sigma), \\
1\text{path}_n^{k+1}(\beta) & \text{if } \alpha = \beta\{k \gamma k\}, \beta \in T_n^{k+1}(\Sigma), \gamma \in T_n^{k-1}(\Sigma) \text{ and the mark is in } \beta, \\
0\text{path}_n^{k-1}(\gamma) & \text{if } \alpha = \beta\{k \gamma k\}, \beta \in T_n^{k+1}(\Sigma), \gamma \in T_n^{k-1}(\Sigma) \text{ and the mark is in } \gamma,
\end{cases}
\end{align*}
\]

and \( n \geq k > 0 \).

(3) \quad \text{path}_n^0(\alpha) = 1\text{path}_n^1(\alpha).

Not all binary strings can form legal paths. The theorem below defines a set which contains exactly those strings which may be legal in some trees.

Theorem 6.3. The set of legal paths of dimension \( n \) and degree \( k \), denoted \( \text{Path}_n^k \), is the smallest set such that

\[
\begin{align*}
(1) \quad \text{Path}_n^{k+1} &= 0. \\
(2) \quad \text{Path}_n^k &= 1\text{Path}_n^{k+1} \cup 0\text{Path}_n^{k-1}(\gamma) \text{ where } n \geq k > 0. \\
(3) \quad \text{Path}_n^0 &= 1\text{Path}_n^1.
\end{align*}
\]

Theorem 6.4. Any legal path encoding on an \( X \) must have exactly one more "1" than "0."

Although this is a binary encoding of the multidimensional forest, it does not correspond to the usual binary encoding of trees. A second method, which does correspond to binary encoded trees, is \( n \)-ary trees, where \( n \) is the dimension of the tree. If a degree zero multidimensional tree, \( \alpha \), is considered, in the most complex case this would be \( \alpha = \beta_1 \) where \( \beta_1 \in T_n^1 \). If \( \beta_1 \) is the most complex case possible, it will equal \( \beta_2\{1 \gamma_0 1\} \) where \( \beta_2 \in T_n^2 \) and \( \gamma_0 \in T_n^0 \). Similarly, \( \beta_2 = \beta_3\{2 \gamma_1 2\} \) and \( \beta_3 = \beta_4\{3 \gamma_2 3\} \). Eventually, the degree must equal the dimension and the construction stops.

\[
\alpha = \beta_{k+1}\{k \gamma_{k-1} k\} \{k-1 \gamma_{k-2} k-1\} \ldots \{2 \gamma_1 2\} \{1 \gamma_0 1\}.
\]

This leads to an encoding in an \( n \)-ary tree. Each node will contain an element of \( \Sigma \) (or an \( X \) or a variable) and \( n \) pointers. The first pointer will point the encoding of the sub forest between the \{\( n \) and the \( n \)}, the second will contain a pointer to the encoding of the sub forest between the \{\( n-1 \) and \( n-1 \)}, etc. This is known as the \( n \)-ary encoding of a multidimensional forest.
Definition 6.5. The \( n \)-ary encoding of a multidimensional forest \( \alpha \) of dimension \( n \) and degree \( k \), denoted \( n\text{-ary}_n^k(\alpha) \), is defined recursively as

\[
\begin{align*}
(1) \quad n\text{-ary}_n^{n+1}(\alpha) &= [\alpha, \ldots, ,] \\
(2) \quad n\text{-ary}_n^k(\alpha) &= \begin{cases} 
n\text{-ary}_n^{k+1}(\alpha) & \text{if } \alpha \in T_n^{k+1}(\Sigma), \\
[\alpha, \ldots, ,] & \text{if } \alpha \in V_k, \\
E(\beta, \gamma) & \text{if } \alpha = \beta\{k \gamma_k\}, \beta \in T_n^{k+1}(\Sigma), \text{ and } \gamma \in T_n^{k-1}(\Sigma),
\end{cases}
\end{align*}
\]

where \( E(\beta, \gamma) \) is the same as \( n\text{-ary}_n^{k+1}(\beta) \) except the \( k \)th subtree is replaced by \( n\text{-ary}_n^{k-1}(\gamma) \) and \( n \geq k > 0 \).

\[
(3) \quad n\text{-ary}_n^0(\alpha) = \begin{cases} 
[\alpha, \ldots, ,] & \text{if } \alpha \in V_0, \\
n\text{-ary}_n^k(\alpha) & \text{if } \alpha \in T_n^k(\Sigma).
\end{cases}
\]

Theorem 6.6. In the \( n \)-ary encoding of any \( n \)-ary forest of dimension \( n \) and degree \( k \), the low \( k - 1 \) subtrees of the root must be nil.

Just as the binary encoding had its suggested path encoding, so does this encoding. A path may be encoded by listing the level of the nodes used for the selection in the \( n \)-ary encoding above. To make a selection, at each node of the encoding go down to the sub \( n \)-ary tree corresponding to the top number on the path encoding and delete the number.

7. Multidimensional tree grammars

At the beginning of this paper it was stated that a generative method for producing languages would be presented. The best known classical generative methods for specifying languages are regular grammars, context free grammars, and, to a lesser degree, context sensitive grammars and unrestricted grammars. In each case, there is a set called the alphabet, a set called the nonterminals, a set of “productions,” and a start symbol. Analogous grammars can be defined over multidimensional trees.

Definition 7.1. A multidimensional tree grammar of dimension \( n \) and rank \( k \) is a four-tuple \((\Sigma, V, P, S)\)_n, where \( \Sigma \) is a set called the alphabet, \( V \) is a ranked set of nonterminals, \( P \) is a ranked set of productions, and \( S \) is a start symbol such that

\[
P_i \subseteq V_i \otimes T_n^i(\Sigma, V)
\]

and \( P_i \) is finite for all \( i \) and \( S \) is an element of \( V_k \).

As is usually done, if \((v, \alpha)\) is in \( P_i \), it will be written as \( v \rightarrow \alpha \) for simplicity.

In generating strings from the grammar, any time that a string has a nonterminal in it, the nonterminal may be replaced by the right side of any element of the set of productions which has the original nonterminal as its right side. The final language
specified will be the set of all multidimensional trees which may be derived from the start symbol after finitely many steps, and which do not contain any nonterminals.

**Definition 7.2.** Given a grammar \((\Sigma, V, P, S)_h\) then \(\alpha\) derives \(\beta\), denoted \(\alpha \Rightarrow \beta\), is defined such that \(\exists \alpha_1, \alpha_2, v \in V, v \rightarrow \gamma\) in \(P_i\), \(\alpha = \alpha_1 \gamma \alpha_2\), and \(\beta = \alpha \gamma \alpha_2\).

This allows any variable to be replaced by an element from the appropriate rank of multidimensional forests as the previous discussion suggests.

The reflexive and transitive closure of \(\Rightarrow\) will be represented as \(\Rightarrow\). If it becomes necessary to distinguish the grammar that the derivation refers to, it will be added under the \(\Rightarrow\) as in \(\Gamma\) where \(G\) is the grammar.

**Definition 7.3.** The language associated with grammar \(G = (\Sigma, V, P, S)_h\), denoted \(\text{Language}^h_G\), is the set \(\{\alpha \mid S \Rightarrow^* \alpha\text{ and }\alpha \in T^h(\Sigma)\}\).

This will allow the definition of multidimensional tree languages. The language function given above will be extended to define lower level languages from higher level grammars.

**Definition 7.4.** If \(G = (\Sigma, V, P, S)_h\) is a multidimensional tree grammar, then the \(m\)-dimensional tree language associated with \(G\), denoted \(\text{language}^{h}_{n,m}(G)\), is defined as

\[
\text{language}^{h}_{n,m}(G) = \begin{cases} 
\text{language}^{h}_n(G) & \text{if } m = n, \\
\{\alpha \mid \alpha = \text{frontier}^{m-1}_n(\beta) \text{ and } \beta \in \text{language}^{h}_{n,m+1}(G)\} & \text{if } k < m + 1, \\
\{\alpha \mid \alpha = \text{frontier}^{m+1}_m(\beta) \text{ and } \beta \in \text{language}^{h}_{n,m+1}(G)\} & \text{if } k \geq m + 1.
\end{cases}
\]

The language hierarchy can be obtained by defining the \(n\)th level as the set of languages that can be arrived at by giving an \(n\)-dimensional tree grammar, and taking the associated string language.

**Definition 7.5.** If \(G = (\Sigma, V, P, S)_h\) then the string language associated with \(G\), denoted \(\text{string}(G)\), is the language \(\text{language}^{h}_{n,1}(G)\).

**8. Multidimensional tree automata**

Just as grammars were defined for multidimensional trees, automata can also be defined. The generalization that will be used is a generalization of the two-dimensional tree case [10]. In order to make the connection with grammars, first a normal form will be defined for grammars.
Definition 8.1. A normal form grammar is a grammar $G = (\Sigma, V, P, S)$ such that

$$P_i \subseteq V \times \begin{cases} X_i \cup V_{i+1} \cup V_{i+1}, V_{i-1} & \text{if } n \geq i > 0, \\ \Sigma & \text{if } i = n + 1, \\ X_0 \cup V_1 & \text{if } i = 0. \end{cases}$$

It is very easy to see that any grammar can be decomposed into a normal form grammar which has the same language. This will become important when it comes time to prove that the set of languages recognizable by automata is the same as the set of languages generateable from grammars.

Definition 8.2. A nondeterministic finite automata of dimension $n$ and rank $k$ is a four-tuple $(\Sigma, V, \delta, F)$ such that

$$\delta_n: \begin{cases} X_i \cup V_{i+1} \cup V_{i+1}, V_{i-1} & \text{if } n \geq i > 0, \\ \Sigma & \text{if } i = n + 1, \\ X_0 \cup V_1 & \text{if } i = 0 \end{cases} \rightarrow P(V_i).$$

$\delta_n(\alpha)$ is empty in all but a finite number of cases and $F$ is an element of $V_\alpha$, where $P(V_i)$ is the power set of $V_i$.

The similarity between the finite automata and the normalized grammar given above should be obvious.

Definition 8.3. Given a nondeterministic finite automata $A_n^k = (\Sigma, V, \delta, F)$, the complete transition function of degree $i$, denoted $A_n^{k,i}$, is defined such that

$$A_n^{k,i}(a) = \delta_n^{k,i}(a).$$

The language accepted by the nondeterministic finite automata above is the set of those elements of $T_n^k$ which have the final state as a state in the complete transition function of degree $k$. That is, the language is precisely those elements $\alpha$ of $T_n^k$ such that $F$ is an element of $A_n^{k,0}(\alpha)$. If $F$ is in $A_n^{k,0}(\alpha)$, then there must be a sequence of elements of $T_n^k$ such that the beginning of the sequence is $\alpha$ and the end of the sequence is $F$ and adjacent elements of the sequence differ only in that a piece of the multidimensional tree has been replaced by a state. The pieces that are replaced...
must be related to the state in that one of the transition functions, when applied to the piece of the multidimensional tree which has been replaced, must result in \( V \). This is called an accepting sequence.

If the power sets, \( P(V_i) \) is replaced by \( V_i \) in the above definitions, then the result is a definition of deterministic finite automata. Using the same logic as in the string case, it can be shown that any nondeterministic finite automata can be made into a deterministic finite automata which recognizes the same language.

**Theorem 8.4.** For every regular grammar \( G^k_n \) there is a finite automata \( A^k_n \) such that the language recognized by the finite automata \( A^k_n \) is the same as the language generated by the grammar \( G^k_n \). For every finite automata \( A^k_n \) there is a regular grammar \( G^k_n \) such that the language generated by the grammar \( G^k_n \) is the same as the language recognized by the finite automata \( A^k_n \).

This may be established by generating a normalized grammar for the grammar given. The finite automata is the same as this grammar, except for the changes in syntax required; the ordered pairs which form the productions need to be turned around.

### 9. Properties of multidimensional trees and forests

The discussion so far has led to a hierarchy of languages defined by using progressively more complex data structures in the grammars, then using the frontier function to extract the string language. The first level of this hierarchy are those languages which may be defined by applying the map function to the language generated by a one-dimensional tree grammar.

**Theorem 9.1.** If \( L \) is the string language associated with \( G = (\Sigma, V, P, S) \), then there exists a regular grammar \( G' \) which has the same associated language \( L \).

The next level of the hierarchy are those languages definable by the frontier of the language generated by a two-dimensional tree grammar.

**Theorem 9.2.** If \( L \) is the string language associated with \( G = (\Sigma, V, P, S) \), then there exists a context free grammar \( G' \) which has the same associated language \( L \).

The grammar is obtained by applying the frontier operation to each of the two-dimensional trees used in the grammar \( G \) to generate the grammar \( G' \). Each nonterminal in \( G \) must generate two nonterminals in \( G' \), one that will generate all strings which will end with \( X^* \) and one that does not end in an \( X \).
Theorem 9.3. If $L$ is the string language associated with $G = (\Sigma, V, P, S)_3^i$, then there exists an IO macro grammar $G'$ which has the same associated language $L$.

The grammar is obtained by applying a similar construction to that used in establishing level 2 of the hierarchy. First, a simpler form of the grammar is constructed such that the $X$'s at level two form a simple series. These, then, are used to generate that arguments, the first in the series being the first argument, the second the second argument, and so forth. The nonterminals are ordered triplets which keep track of whether the nonterminal produces a singleton after the first frontier operation, or whether the string produced ends in $X^+_f$ or not. The complete proof is given in the dissertation on the topic [4].

Theorem 9.4. The set of languages generated by grammars of dimension $n$, where $n$ is greater than zero, is closed under union and intersection.

Theorem 9.5. The string languages generated by multidimensional grammars of any dimension is closed under union and under intersection with regular sets.

The set of string languages cannot be closed under intersection because context free languages, which form one level of the hierarchy, are not closed.

Theorem 9.6. If $L$ is a language which is the string language associated with multidimensional tree grammar $G = (\Sigma, V, P, S)_n^i$, then there is a grammar $G' = (\Sigma', V', P', S')_n$ a "pure" multidimensional tree grammar, such that the string language associated with $G'$ is $L$.

This is easily established by adding a new start symbol to the old grammar, and one production

$$S' \rightarrow \star \{n \star \{n-1 \ldots \{k+1 S_{k+1} \ldots n-1\}_n\}.$$

The infinite size of the hierarchy is established in the next two theorems.

Theorem 9.7. If $\alpha$ is an element of $T^\alpha_n(\Sigma)$, and $n$ is the number of nodes in $\alpha$ then the number of nodes in $\text{frontier}_n^b(\alpha)$ must be strictly less than $2^n$.

Theorem 9.8. For any $n$ the set of languages at level $n$ of the hierarchy is included in the set of languages at level $n + 1$, but is not equal to it.

It can be shown by induction that a sequence of grammars may be defined for the sequence of languages $\{a^{2^n}\}$, where the number of 2's in the exponent is equal to the sequence number. The first language is the string language associated with the two-dimensional grammar:

$$S \rightarrow A,$$
$$A \rightarrow \star \{2 \star \{a \star \{A \}_{2}\},$$
$$A \rightarrow a.$$
The subsequent elements of the series of languages are generated by
(1) Incrementing each number in the grammar at the previous level by one.
(2) For each $a$ in the original grammar, substitute
$\ast\{n\ast\{n-1\ldots\ast\{3\ast\{2X^+\{1X^+\{1X^+\{1\{2\{3\{3\ldots\{n-1\{\}n\}_{n}\}}$.
(3) Add a new start symbol $S'$ and the production
$S' \rightarrow \ast\{n\ast\{n-1\ldots\ast\{3S\}2a3\}3\ldots\{n-1\}n\}$,
where $n$ is the dimension of the new grammar.

**Theorem 9.9.** Not all context sensitive languages have a corresponding language in
the language hierarchy.

The proof of this theorem is a diagonalization of the above proof. A context
sensitive grammar for the language
$\{a, a^2, a^{22}, a^{222}, \ldots\}$
is easy to write, however, this language cannot be in any level of the hierarchy.

**Theorem 9.10.** Any language contained in the hierarchy is contained in the set of
context sensitive languages.

This proof is very long and is included in the dissertation on the topic [4].

In the theorem dealing with level 3 of the language hierarchy above, it was noted
that the hierarchy follows the IO type of macro grammars. The theorem below
shows how to generate OI macro grammars, if that is desirable.

**Theorem 9.11.** If a language is generated from a level 3 multidimensional grammar
by alternating production application and the frontier operation, then the resulting
language has an OI macro grammar.

10. Conclusions

This work has many possible extensions. OI multidimensional trees were men-
tioned briefly in the author’s dissertation [4]. This will result in a new hierarchy of
languages which may be studied. The author has shown in another paper that the
concept of IO-OI is itself extendable to a hierarchy of hierarchies [3]. Work needs
to be done to determine the properties of this hierarchy of hierarchies — a two-
dimensional hierarchy.

It is suspected that this hierarchy is related to the hierarchy proposed by Maibaum
[7] and Damm [5]; perhaps it is the same hierarchy. This needs to be explored.
The author has constructed a top down backtracking parser for multidimensional tree grammars, both IO and OI. This work may be extended to, perhaps lead to an LR parser for these languages. If these can be parsed directly, then it is felt that macro grammars may offer a reasonable alternative to context free grammars for specifying languages.

References