Hypertrees: a study in language specification

William Allen Baldwin
Iowa State University

Follow this and additional works at: http://lib.dr.iastate.edu/rtd
Part of the Computer Sciences Commons

Recommended Citation
INFORMATION TO USERS

This reproduction was made from a copy of a document sent to us for microfilming. While the most advanced technology has been used to photograph and reproduce this document, the quality of the reproduction is heavily dependent upon the quality of the material submitted.

The following explanation of techniques is provided to help clarify markings or notations which may appear on this reproduction.

1. The sign or “target” for pages apparently lacking from the document photographed is “Missing Page(s)”. If it was possible to obtain the missing page(s) or section, they are spliced into the film along with adjacent pages. This may have necessitated cutting through an image and duplicating adjacent pages to assure complete continuity.

2. When an image on the film is obliterated with a round black mark, it is an indication of either blurred copy because of movement during exposure, duplicate copy, or copyrighted materials that should not have been filmed. For blurred pages, a good image of the page can be found in the adjacent frame. If copyrighted materials were deleted, a target note will appear listing the pages in the adjacent frame.

3. When a map, drawing or chart, etc., is part of the material being photographed, a definite method of “sectioning” the material has been followed. It is customary to begin filming at the upper left hand corner of a large sheet and to continue from left to right in equal sections with small overlaps. If necessary, sectioning is continued again—beginning below the first row and continuing on until complete.

4. For illustrations that cannot be satisfactorily reproduced by xerographic means, photographic prints can be purchased at additional cost and inserted into your xerographic copy. These prints are available upon request from the Dissertations Customer Services Department.

5. Some pages in any document may have indistinct print. In all cases the best available copy has been filmed.
Baldwin, William Allen

HYPERTREES -- A STUDY IN LANGUAGE SPECIFICATION

Iowa State University

Ph.D. 1983

University Microfilms International 300 N. Zeeb Road, Ann Arbor, MI 48106
Hypertrees -- A study in language specification

by

William Allen Baldwin

A Dissertation Submitted to the
Graduate Faculty in Partial Fulfillment of the
Requirements for the Degree of
DOCTOR OF PHILOSOPHY

Major: Computer Science

Approved:

Signature was redacted for privacy.

In Charge of Major Work

Signature was redacted for privacy.

For the Major Department

Signature was redacted for privacy.

For the Graduate College

Iowa State University
Ames, Iowa

1983
<table>
<thead>
<tr>
<th>TABLE OF CONTENTS</th>
<th>PAGE</th>
</tr>
</thead>
<tbody>
<tr>
<td>ABSTRACT</td>
<td>1</td>
</tr>
<tr>
<td>INTRODUCTION</td>
<td>2</td>
</tr>
<tr>
<td>BASIC DEFINITIONS</td>
<td>15</td>
</tr>
<tr>
<td>REPRESENTATION OF LEVEL N HYPERTREES</td>
<td>77</td>
</tr>
<tr>
<td>REGULAR HYPERTREE GRAMMARS</td>
<td>105</td>
</tr>
<tr>
<td>FINITE AUTOMATA</td>
<td>132</td>
</tr>
<tr>
<td>REGULAR EXPRESSIONS</td>
<td>152</td>
</tr>
<tr>
<td>CHARACTERIZATION OF THE ALGEBRAIC LANGUAGE HIERARCHY</td>
<td>183</td>
</tr>
<tr>
<td>CONTAINMENT IN CONTEXT SENSITIVE</td>
<td>236</td>
</tr>
<tr>
<td>CONCLUSION</td>
<td>285</td>
</tr>
<tr>
<td>BIBLIOGRAPHY</td>
<td>288</td>
</tr>
<tr>
<td>ACKNOWLEDGEMENTS</td>
<td>292</td>
</tr>
<tr>
<td>APPENDIX: LINEARLY BOUNDED GRAMMARS</td>
<td>294</td>
</tr>
</tbody>
</table>
iii

LIST OF TABLES

TABLE 1. Some Examples of Hypertrees over \{a,b,c\} . . . . 20
TABLE 2. Equation c of Definition 9 . . . . . . . . . . . . . . . . . 46
TABLE 3. Some Examples of n-ary Encoded Hypertrees . . . . 101
TABLE 4. Regular Expressions over \{a\} . . . . . . . . . . . . . . 160
TABLE 5. Miscellaneous Routines . . . . . . . . . . . . . . . . . . 239
TABLE 6. The Over Routine . . . . . . . . . . . . . . . . . . . . . . 240
TABLE 7. The Search Routine . . . . . . . . . . . . . . . . . . . . . 241
TABLE 8. The Apply Routine . . . . . . . . . . . . . . . . . . . . . 243
TABLE 9. The Frontier Routine . . . . . . . . . . . . . . . . . . . . 245
TABLE 10. The String Routine . . . . . . . . . . . . . . . . . . . . . 246
TABLE 11. Over Derivation for \( \gamma \in F_{n,0}^k \) . . . . . . . . . . 249
TABLE 12. Over Derivation for \( \gamma \in \varnothing \) . . . . . . . . . . . 249
TABLE 13. Over Derivation for \( \gamma \in F_{n,m-1}^{k'+1} \) . . . . . . . 250
TABLE 14. Over for \( \gamma \in F_{n,m-1}^{k'+1} \) . . . . . . . . . . . . 251
TABLE 15. Over for \( \gamma \in <\text{frontier}_{n}^{k}>_{F_{n+1,m-1}^{k}} \) . . . . . 252
TABLE 16. Over Derivation for \( \gamma \in F_{n,m}^{0} \) . . . . . . . . . . 252
TABLE 17. \( S_{n}^{k} \) Derivation for \( \gamma \in H_{n+1,0}^{k} \) . . . . . . . . . 254
TABLE 18. \( S_{n+1}^{k} \) Derivation for \( \gamma \in \varnothing \) . . . . . . . . . . . 254
TABLE 19. \( S_{n+1}^{k} \) for \( \gamma \in H_{n+1,m-1}^{k+1} \) . . . . . . . . . . 255
TABLE 20. \( S_{n}^{k} \) Derivation for \( \gamma \in H_{n,m-1}^{k-1} \) . . . . . . . . 256
TABLE 21. \( S_{n}^{k} \) for \( \gamma \in H_{n+1,m-1}^{k} \) . . . . . . . . . . . . 257
TABLE 22. $S^0_n$ Derivation for $\gamma \in H^0_{n,m}$ ........................................ 258
TABLE 23. $<S1^0_n>$ Derivation for $\gamma_2 \in F^0_{n,m}$ .................................... 259
TABLE 24. $<S1^k_n>$ for $\gamma_2 \in F^k_{n}[k F^{k-1}_{n} k]$ and $j = ki$ .......... 260
TABLE 25. $<S1^k_n>$ Derivation for $\gamma_2 \in F^{k+1}_{n}$ and $j \in F^{k+1}_{n}$ .... 261
TABLE 26. $<S1^k_n>$ for $\gamma_2 \in F^{k+1}_{n}[k F^{k-1}_{n} k]$ and $j \in F^{k+1}_{n}$ .... 262
TABLE 27. $<S1^0_n>$ Derivation ................................................................. 263
TABLE 28. $<A1^k_n>$ Derivation for $\gamma_1 \in X_k$ and $k < n$ ...................... 263
TABLE 29. $<A1^0_n>$ Derivation for $\gamma_1 = X^j_n$ ...................................... 264
TABLE 30. $<A1^{n+1}_n>$ Derivation ............................................................ 265
TABLE 31. $<A1^k_n>$ Derivation for $\gamma_1 \in H^{k+1}_{n,m-1}$ ......................... 265
TABLE 32. $<A1^k_n>$ for $\gamma_1 \in H^{k+1}_{n,m-1}[k H^{k-1}_{n,m-1} k]$ .......... 266
TABLE 33. $<A1^0_n>$ Derivation ................................................................. 267
TABLE 34. Frontier Derivation for Basis ..................................................... 267
TABLE 35. Frontier Derivation for $\gamma = \alpha[\beta_{n+1}]$ ........................... 268
TABLE 36. Frontier Derivation for $k = n$, $\gamma \in E^0_{n+1}$ ......................... 269
TABLE 37. Frontier for $k = n$, $\gamma = \alpha[\beta_{n}]$ and $\alpha \in \Sigma$ .......... 270
TABLE 38. Frontier for $k = n$, $\gamma = \alpha[\beta_{n}]$ and $\alpha \notin \Sigma$ .......... 273
TABLE 39. Frontier Derivation for $\gamma \in H^{k+1}_{n+1}$ ............................... 275
TABLE 40. Frontier Derivation for $\gamma = \alpha[k \beta_k]$ ............................ 276
TABLE 41. Frontier Derivation for $k = 0$ .................................................. 277
TABLE 42. Construction for Lemma 103 .................................................... 325
v

LIST OF FIGURES

FIGURE 1. Sample Tree Grammar .......................... 10
FIGURE 2. Language Defined by the Sample Grammar .... 11
FIGURE 3. Illustration of a Context Free Tree Language . 12
FIGURE 4. Known Language Types .......................... 13
FIGURE 5. A Projection of a Level 3 Hypertree .......... 21
FIGURE 6. A Tree ............................................ 36
FIGURE 7. A Hypertree of level 4 and degree 2 ......... 43
**LIST OF DEFINITIONS**

<table>
<thead>
<tr>
<th>Definition</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>DEFINITION 1. Syntax</td>
<td>2</td>
</tr>
<tr>
<td>DEFINITION 2. Semantics</td>
<td>2</td>
</tr>
<tr>
<td>DEFINITION 3. Regular Grammars</td>
<td>4</td>
</tr>
<tr>
<td>DEFINITION 4. Context Free Grammars</td>
<td>6</td>
</tr>
<tr>
<td>DEFINITION 5. Context Sensitive Grammar</td>
<td>7</td>
</tr>
<tr>
<td>DEFINITION 6. Unrestricted Grammars</td>
<td>8</td>
</tr>
<tr>
<td>DEFINITION 7. Classical Strings</td>
<td>16</td>
</tr>
<tr>
<td>DEFINITION 8. Classical Trees</td>
<td>16</td>
</tr>
<tr>
<td>DEFINITION 9. Modified Classical Trees</td>
<td>16</td>
</tr>
<tr>
<td>DEFINITION 10. Hypertrees</td>
<td>20</td>
</tr>
<tr>
<td>DEFINITION 11. Hyperforests</td>
<td>23</td>
</tr>
<tr>
<td>DEFINITION 12. Hyperforests of Depth m</td>
<td>26</td>
</tr>
<tr>
<td>DEFINITION 13. n-paths of Degree k</td>
<td>37</td>
</tr>
<tr>
<td>DEFINITION 14. Frontierable Hypertrees</td>
<td>38</td>
</tr>
<tr>
<td>DEFINITION 15. Frontierable Hyperforests</td>
<td>39</td>
</tr>
<tr>
<td>DEFINITION 16. Frontierable Hyperforests of Depth m</td>
<td>39</td>
</tr>
<tr>
<td>DEFINITION 17. Search</td>
<td>45</td>
</tr>
<tr>
<td>DEFINITION 18. Apply</td>
<td>52</td>
</tr>
<tr>
<td>DEFINITION 19. Frontier</td>
<td>57</td>
</tr>
<tr>
<td>DEFINITION 20. Strings</td>
<td>65</td>
</tr>
<tr>
<td>DEFINITION 21. Trees</td>
<td>66</td>
</tr>
<tr>
<td>Theorem</td>
<td>Statement</td>
</tr>
<tr>
<td>---------</td>
<td>---------------------------------------------------------------------------</td>
</tr>
<tr>
<td>1</td>
<td>$h_n^k$ and $h_n$ Relationship</td>
</tr>
<tr>
<td>2</td>
<td>$h_n = h_n^k$</td>
</tr>
<tr>
<td>3</td>
<td>$h_n^j \subset h_n^k$</td>
</tr>
<tr>
<td>4</td>
<td>Alternative Line 3 in Hyperforest Definition</td>
</tr>
<tr>
<td>5</td>
<td>$h_{n,m}^k \subset h_n^k$</td>
</tr>
<tr>
<td>6</td>
<td>$h_{n,m}^k$ is Nested in $m$</td>
</tr>
<tr>
<td>7</td>
<td>$h_{n,m}^k$ and $h_n^k$ Relationship</td>
</tr>
<tr>
<td>8</td>
<td>$H_n^k$ and $H_n$ Relationship</td>
</tr>
<tr>
<td>9</td>
<td>$H_n = H_n^k$</td>
</tr>
<tr>
<td>10</td>
<td>$H_n^j \subset H_n^k$</td>
</tr>
<tr>
<td>11</td>
<td>$H_{n,m}^k \subset H_n^k$</td>
</tr>
<tr>
<td>12</td>
<td>$H_{n,m}^k$ is Nested in $m$</td>
</tr>
<tr>
<td>13</td>
<td>Alternative Line 3 in Frontierable Hyperforest Definition</td>
</tr>
<tr>
<td>14</td>
<td>$H_{n,m}^k$ and $H_n^k$ Relationship</td>
</tr>
<tr>
<td>15</td>
<td>Codomain for search Using $H_{n,m}^k$</td>
</tr>
<tr>
<td>16</td>
<td>Search Function Codomain</td>
</tr>
<tr>
<td>17</td>
<td>Codomain for apply Using $H_{n,m}^k$</td>
</tr>
<tr>
<td>18</td>
<td>Apply Function Codomain</td>
</tr>
<tr>
<td>19</td>
<td>Codomain for frontier Using $H_{n,m}^k$</td>
</tr>
<tr>
<td>20</td>
<td>Frontier Function Codomain</td>
</tr>
</tbody>
</table>
LEMMA 21. \( h_{n,m}^k \subseteq H_{n,m}^k \) ........................................ 66
THEOREM 22. \( h_n \subseteq H_n \) ........................................ 68
THEOREM 23. \( H_n^k(\xi,\nu) \) and \( H_n(\xi,\nu) \) Relationship .... 73
COROLLARY 24. \( H_n(\xi,\nu) = H_n^k(\xi,\nu) \) ..................... 73
LEMMA 25. \( H_n^j(\xi,\nu) \subseteq H_n^k(\xi,\nu) \) ..................... 73
LEMMA 26. \( H_n^k(\xi,\nu) \subseteq H_n^k(\xi,\nu) \) ..................... 74
LEMMA 27. \( H_n^k(\xi,\nu) \) is Nested in \( m \) ..................... 74
LEMMA 28. Alternative Line 3 in Frontierable
Hyperforests with Arguments Definition .................. 74
THEOREM 29. \( H_n^k(\xi,\nu) \) and \( H_n^k(\xi,\nu) \) Relationship .... 74
LEMMA 30. Nesting of Hypertree Types ..................... 76
THEOREM 31. \( h_n \subseteq H_n(\xi,\nu) \) ..................... 76
LEMMA 32. \( T_n^j \subseteq T_n^k \) ................................. 80
LEMMA 33. \( T_n^k \subseteq T_n^k \) ................................. 82
LEMMA 34. \( T_n^k \) is Nested in \( m \) ..................... 85
THEOREM 35. \( T_n^k \) and \( T_n^k \) Relationship ............. 87
LEMMA 36. \( T_n^k \) is a Set of \( n \)-ary Trees ............. 89
LEMMA 37. \( T_n^k \) is a Set of \( n \)-ary Trees ............. 94
THEOREM 38. \( T_n \) is an \( N \)-ary Tree ............. 94
THEOREM 39. Encode Function Domain -- \( N \)-ary Encoded
Hypertrees ........................................ 97
LEMMA 40. \( G_n^k \Rightarrow Relation Codomain Lemma ........ 107
THEOREM 41. \( G_n^k \Rightarrow Relation Codomain ........ 111
THEOREM 42. \( G_n^{k*} \Rightarrow Relation Codomain ........ 112
THEOREM 43. \( m \)-normal Grammar Exists ........... 119
LEMMA 44. \( m-l \)-normal Grammar Exists ........... 119
THEOREM 45. Normal Grammar Exists .................. 131
THEOREM 46. Existence of a Deterministic Finite
Automaton ...................................... 138
LEMMA 47. Grammar -- Automata Construction .... 146
LEMMA 48. Normal Grammar -- Automata Equivalence . 150
THEOREM 49. Finite Automata -- Regular Grammar
Equivalence ..................................... 150
LEMMA 50. Concatination Codomain for Hypertrees of
Depth m .......................................... 155
THEOREM 51. Concatination Codomain .................. 157
THEOREM 52. Closure of a Set Codomain ............... 158
THEOREM 53. $R^k_{n,m}$ and $R^k_n$ Relationship .......... 161
THEOREM 54. Regular Set Language Codomain .......... 163
THEOREM 55. Equivalence of Partial and Full Finite
Automata ........................................ 170
THEOREM 56. Regular Expression and Automata
Equivalence ..................................... 172
THEOREM 57. Grammar and Regular Expression
Equivalence ..................................... 174
COROLLARY 58. $ALH_n^k$ Using Automata and Regular
Expressions ..................................... 185
THEOREM 59. Hypertree Closure Under Union .......... 186
THEOREM 60. Hypertree Closure Under Intersection .. 186
COROLLARY 61. $ALH$ Closed Under Union ............. 190
THEOREM 62. Closure Under Intersection with Regular
Sets ............................................. 191
THEOREM 63. Completed Grammar Exists ............... 197
COROLLARY 64. Existence of Non-dropping Grammar ... 201
THEOREM 65. Only $V_i \ k > i > 0$ are needed .......... 202
COROLLARY 66. Only $V_n$ needed for $ALH_n^1$ ............ 203
COROLLARY 67. $G_n^k$ exists for $L \in ALH_n^k$ ............ 204
THEOREM 68. $G_n^k$ for $L \in ALH_n^k$ modified ............ 204
THEOREM 69. Linear Grammar for $L \in ALH_n^k$ ............ 206
COROLLARY 70. Linear Grammars for Strings ............ 207
COROLLARY 71. Linear Grammars for trees ............ 208
COROLLARY 72. Linear Grammars for Hypertrees of Level 3 208
THEOREM 73. ALH Level 1 Corresponds to Regular Languages ............ 209
THEOREM 74. ALH Level 2 Corresponds to Context Free Languages ............ 210
THEOREM 75. ALH Level 3 Corresponds to Macro Languages ............ 213
THEOREM 76. Pumping Lemma ............ 222
LEMMA 77. Limit on Count of the Frontier ............ 223
LEMMA 78. Language to Establish Hierarchy Introduced ............ 227
LEMMA 79. Language in Lemma 78 is in the Hierarchy 230
THEOREM 80. Proof of Hierarchy ............ 232
COROLLARY 81. Proof of Hierarchy at Level $k$ ............ 233
LEMMA 82. Language not in the Hierarchy Introduced 234
LEMMA 83. Language in Lemma 82 is Context Sensitive 234
THEOREM 84. Non-Equivalence with Context Sensitive 235
LEMMA 85. Over Derivation Width ............ 248
COROLLARY 86. Completed Over Derivation ............ 251
LEMMA 87. Frontier Derivation to frontier$^n$ ............ 253
LEMMA 88. Search Derivation Width ............ 256
LEMMA 89. Apply Derivation Width .............. 261
LEMMA 90. Frontier Derivation Width .............. 267
LEMMA 91. Frontier Function Works .............. 277
LEMMA 92. String Function Works .............. 277
LEMMA 93. Linearly Bounded Sequence of Frontiers is Sufficient .............. 278
LEMMA 94. K Bounded Grammar Exists .............. 279
THEOREM 95. Containment in Context Sensitive .............. 283
LEMMA 96. P c (V^* V) U (V -> E) Exists .............. 296
LEMMA 97. Reduce Width of Productions by One .............. 302
LEMMA 98. Width of Productions Reduced to 2 .............. 306
LEMMA 99. P c (VV V) U (V -> VV) U (V -> ) U (V -> ) U (V -> E) .............. 307
LEMMA 100. P c (VV V) U (V -> VV) U (V -> ) U (V -> ) .............. 309
THEOREM 101. Normal Form Grammar Exists .............. 312
LEMMA 102. Length Bounded Grammar Corresponds to Context Sensitive .............. 317
LEMMA 103. Width can be Halved .............. 323
LEMMA 104. Arbitrary is/is not in a Bounded Language .............. 327
COROLLARY 105. is/is not in a Bounded Language .............. 329
LEMMA 106. Context Sensitive Grammar for K Bounded Grammars .............. 329
THEOREM 107. Linearly Bounded Languages are Context Sensitive .............. 331
ABSTRACT

A hierarchy of data structures, called the hypertree hierarchy, is presented which has strings and trees as its smallest two elements. A generalized frontiering operation is presented. Grammars, automata and regular expressions are extended to this hierarchy. These are shown to be equivalent and the resulting languages are called regular. This leads to a hierarchy of regular languages on the hypertree hierarchy. These are projected onto the set of strings by the frontier operation resulting in a true hierarchy of string languages. This hierarchy is called the (IO) algebraic language hierarchy. It has regular languages, context free languages and macro languages as its first three levels. It is contained in the set of context sensitive languages but is not equal to it. Other characterizations are presented.
INTRODUCTION

In the area of programming languages, two words are used quite extensively and with some variation as to the meaning. They are "syntax" and "semantics". In order to be certain that there is no confusion in the use of these terms, a definition will be given here.

Definition 1:

The syntax of a language is the set of all legal programs in the language.

Definition 2:

The semantics of a language is function which maps the syntax of the language to a set of meanings.

The term "meanings" will remain undefined in this context.

This paper concerns itself with the specification of a syntax a formal language.

Typically, the syntax of a formal language is specified by giving a context free grammar for the language. However, this has its drawbacks because some of the aspects of common programming languages cannot be specified using a context free grammar. For example, in Pascal a variable must be declared before it is used, yet this simple fact cannot be
expressed via a context free grammar. Therefore, in addition to the grammar of most programming languages there has to be a piece of English prose which describes these sets of facts. English, being a natural language, has a problem in that it is subject to different interpretations by different people. Even more important, the English prose is too complex and ill-defined to allow automatic generation of a parser given these specifications. Therefore, typically a parser is written which recognizes the context free language specified, and the errors that are defined by the English prose are postponed and allowed to show up in the code generation (semantic analysis) phase of a compiler. For this reason, they are typically called semantic errors, although, in fact, they are not errors in the semantics of the program at all but rather are syntax errors. All these consideration have led some people to explore other means of specifying syntax.

There are five methods of specifying syntax that are of interest at this point. The first four were collected into a hierarchy by Chomski in the 1950s, and are therefore known as the Chomski hierarchy. They are: regular grammars, context free grammars, context sensitive grammars and unrestricted grammars. The fifth method, macro grammars, is due to Fischer (Fischer 1968). In 1974, it was shown by
Wand (Wand 1975) and Maibaum (Maibaum 1974) that of these, regular grammars, context free grammars and macro grammars form a natural hierarchy. This hierarchy will be called the algebraic language hierarchy in this paper since Wand and Maibaum showed that this hierarchy was algebraic in nature.

Regular grammars are defined as follows:

Definition 3:
A regular grammar is a four-tuple, $$(\Sigma, N, P, S)$$, where
1. $\Sigma$ is a set called the terminals.
2. $N$ is a set called the nonterminals.
3. $P$ is a finite set called the productions.
4. $S$ is a symbol called the start symbol.

In addition, $S \in N$ and $P \subseteq N \times (\Sigma^* N \cup \Sigma^*)$.

An example of a regular grammar is

$$
({a,b}, \{S,B\},\{(S,aB),(B,bS),(S,a)\},S)
$$

which is typically written as

$$
S \rightarrow aB \\
B \rightarrow bS \\
S \rightarrow a
$$

where each of the lines is called a production. Note that $A \rightarrow \alpha$ is a production is exactly equivalent to $(A,\alpha) \in P$. 
Also, typically, the start symbol is written as the first symbol, that is, the left hand side of the first production. This notation will be used here. These may be used to generate a language as follows: If $A \rightarrow \beta$ is a production, then,

$$aA \Rightarrow a\beta$$

for all $a$ and $\gamma$. Let this (informally) define $\Rightarrow$. Then, let $\Rightarrow^*$ be the reflexive and transitive closure of $\Rightarrow$. The language defined by the grammar above is, therefore,

$$\{a \mid S \Rightarrow^* a \text{ and } a \in \Sigma^*\}$$

Therefore, the grammar above defines the language

$$\{(ab)^n a\}_{n \geq 0}$$

It is a well-known fact that regular grammars are very limited in the languages they can define. They are used for defining only very simple languages such as are used in lexical analyses. Therefore, more powerful grammars are typically used to define real programming languages. The most common is context free grammars. Context free grammars are defined as follows:
Definition 4: A context free grammar is a four-tuple \((Z, N, P, S)\) where

1. \(Z\) is a set called the terminals.
2. \(N\) is a set called the nonterminals.
3. \(P\) is a finite set called the productions.
4. \(S\) is a symbol called the start symbol.

Further, \(S \in N\) and \(P \subseteq N \times (E \cup N)^*\).

The definition of \(\Rightarrow\) and the language generated is exactly analogous to the corresponding notation used in regular grammars above. An example of a context free grammar is

\[
S \rightarrow E \\
E \rightarrow (E) \\
E \rightarrow a
\]

Note, this grammar defines the language

\[
\{(a^n)^n\}_{n \geq 0}
\]

It is easy to show this language cannot be represented by a regular grammar; therefore, it is known that context free grammars can define a larger set of languages than regular grammars can define. For this reason, context free grammars are used to define a number of programming languages as was
mentioned above. It should be obvious also that any regular grammar is also context free.

The third type of grammar is context sensitive grammars. It is defined as follows:

Definition 5:

A context sensitive grammar is a four-tuple $(\Sigma, N, P, S)$ where

1. $\Sigma$ is a set of terminals.
2. $N$ is a set of nonterminals.
3. $P$ is a finite set of productions.
4. $S$ is a symbol called the start symbol.

Further, $S \in N$ and $P \subseteq (\Sigma \cup N)^*N(\Sigma \cup N)^* \times (\Sigma \cup N)^*$ such that if $(A, B) \in P$ then the length of $A$ is less than or equal to the length of $B$.

The definition of $\Rightarrow$ requires now that the $A$ may be a string of symbols. An example of a context sensitive grammar is

\[
S \Rightarrow aSBc \\
S \Rightarrow abc \\
cB \Rightarrow Bc \\
bB \Rightarrow bb
\]

It can be shown that this grammar defines the language

\[
\{a^n b^n c^n \}_{n>0}
\]
which cannot be expressed with a context free grammar.

The fourth type of grammar in the Chomsky hierarchy is called unrestricted.

Definition 6:

An unrestricted grammar is a four-tuples $(\Sigma, N, P, S)$ where

1. $\Sigma$ is a set of terminals.
2. $N$ is a set of nonterminals.
3. $P$ is a finite set of productions.
4. $S$ is a symbol called the start symbol.

In addition, $S \in N$ and $P \subseteq (\Sigma \cup N)^*N(\Sigma \cup N)^* \times (\Sigma \cup N)^*$.

The language defined by this grammar is analogous to the previous definitions.

The fifth type of grammar was first formally defined by Fischer (Fischer 1968). It is called macro grammars and will not be formally defined here. In general, macro grammars allow productions in which each nonterminal is allowed zero or more arguments. For example,

$$S \rightarrow T(a, b, c)$$
$$T(X_1, X_2, X_3) \rightarrow T(X_1 a, X_2 b, X_3 c)$$
$$T(X_1, X_2, X_3) \rightarrow X_1 X_2 X_3$$
Here, the "(a,b,c)" following the first nonterminal T is a listing of the three arguments, "a", "b" and "c".

Similarly, => must be defined differently. If A(X₁...Xₙ) → β is a production, then

αA(β₁...βₙ)γ => αβ'γ

where β' is β with all occurrences of X₁ replaced by β₁, X₂ replaced by β₂ etc. *=> is the reflexive and transitive closure as before. Using these rules, it can easily be seen that the grammar given produces the language

\{a^n b^n c^n \}_{n \geq 0}

so that macro grammars are more powerful than context free grammars. Fischer also showed that macro grammars are not as powerful as context sensitive grammars. The same paper also shows that there are two types of macro grammars, IO and OI. IO macro grammars insist that in order to apply the production given above, β₁...βₙ must be elements of I*. OI, on the other hand, insists that the A not be an argument to any other nonterminal at the point where the production is applied. This gives the same language as is obtained if there is no restrictions at all placed on the order of evaluation.
At the same time, some of the grammars that have just been described were extended to operate on trees by Rounds (Rounds 1969). These included regular grammars and context free grammars. An example of a regular grammar over trees is given in figure 1 where each nonterminal is required to be on the frontier of the tree. This will define the language given in figure 2.

![Sample Tree Grammar Diagram](image)

FIGURE 1. Sample Tree Grammar

Note that an alternative method of representing trees is to put parentheses around the subtrees at each node. Therefore, the above grammar can be written
such that \( n \geq 0 \).

FIGURE 2. Language Defined by the Sample Grammar

\[
S \rightarrow 1(E) \\
E \rightarrow 2([E]) \\
E \rightarrow 3(a)
\]

and the language can be represented as

\[
\left\{ 1(2(\text{[}^{\text{n}}\text{a}\text{]}\text{]})^{\text{n}}) \right\}_{\text{n} \geq 0}
\]

Note that the frontier of the elements of this language form the language

\[
\left\{ \text{[}^{\text{n}}\text{a}\text{]}^{\text{n}} \right\}_{\text{n} \geq 0}
\]

which is a context free language. In general, this is the case. The frontier of any regular tree language is a context free string language.
FIGURE 3. Illustration of a Context Free Tree Language

Rounds also talks about context free grammars over trees (Rounds 1969). For example,

\[ S \rightarrow T(abc) \]
\[ T(X_1 X_2 X_3) \rightarrow T(\ast(X_1 a) \ast(X_2 b) \ast(X_3 c)) \]
\[ T(X_1 X_2 X_3) \rightarrow \ast(X_1 X_2 X_3) \]

Some examples of the trees resulting from this grammar are given in figure 3. Note that the frontier of this language is the language
which is a macro language. Rounds showed that in general this is the case. Also, for each context free tree grammar, there is a macro grammar which describes the frontier language. Note that just as there were two distinct classes of macro grammars, IO and OI, there are two classes of context free tree grammars on trees -- top-down and bottom-up.

<table>
<thead>
<tr>
<th></th>
<th>Regular</th>
<th>Context Free</th>
<th>Indexed</th>
</tr>
</thead>
<tbody>
<tr>
<td>String</td>
<td>*</td>
<td>➔*</td>
<td>➔*</td>
</tr>
<tr>
<td>Tree</td>
<td>*</td>
<td>➔*</td>
<td>➔*</td>
</tr>
<tr>
<td>?</td>
<td>-</td>
<td>➔*</td>
<td>➔*</td>
</tr>
</tbody>
</table>

FIGURE 4. Known Language Types

These last few remarks lead to the figure given in figure 4. Here each of the known grammars are represented by an asterisk, with an arrow representing the frontier function. This is similar to the table given by Maibaum (Maibaum 1974), with his hierarchy of algebras replaced by a hierarchy (of two) data structures. As Maibaun noted, this
suggests a hierarchy of data structures, and it is this hierarchy of data structures that is the subject of this paper. This hierarchy is called the hypertree hierarchy.
BASIC DEFINITIONS

The set of hypertrees over \( I \) has two well-known special cases. These are the set of strings over \( I \) and the set of trees over \( I \), which are designated as level 1 hypertrees and level 2 hypertrees respectively.

Historically, there have been a few discrepancies in the treatment of strings and trees. For instance, a null string is provided, but a null tree is not. Also, any element of \( I \) can appear at any location in a string, but only a subset can appear at any particular location on a tree, such as at the frontier, due to the ranking relation (Rounds 1970). In each case, a decision had to be made as to which structure to follow; a similar development could be made using the other choices. In the above examples, it was decided that eliminating the null string and the ranking relation would make the smoothest development. Therefore, \( I^+ \) is used as the set of strings instead of \( I^* \). Also, each of the ranks \( I_i \)'s in the ranked set \( I \) is assumed to be equal to \( I \) itself. Normally, only a finite number of the ranks may be nonempty so that tree automata may be developed, however, it will be shown in chapter 5 that this is not needed.

A recursive definition of the set of strings over \( I \), \( I^+ \), is given by
Definition 7:
The set of classical strings over $\Sigma$, denoted $\Sigma^+$, is the smallest set such that
$$\Sigma^+ = \Sigma \cup \Sigma \Sigma^+$$

Similarly, a recursive definition of the set of trees over $\Sigma$, $T^\Sigma$, is given by

Definition 8:
The set of classical trees over the ranked set $\Sigma$, denoted $T^\Sigma$, is the smallest set such that
$$T^\Sigma = \Sigma_0 \cup \bigcup_{i \geq 1} \Sigma_i \{ (T^\Sigma)^i \}$$
where $\Sigma_i$ is the $i^{th}$ rank of $\Sigma$.

Keeping in mind the above discussion, this may be rewritten as

Definition 9:
The set of modified classical trees over $\Sigma$, denoted $MT^\Sigma$, is the smallest set such that
$$MT^\Sigma = \Sigma \cup \bigcup_{i \geq 1} \Sigma \{ (MT^\Sigma)^i \}$$

since each of the $\Sigma_i$'s is equal to $\Sigma$.

In developing the definition of hypertrees, first a common notation is needed for the set of strings over $\Sigma$ and
the set of trees over $I$. Since a level number is also needed, an entirely new notation was developed rather than expand either the $T_I$ notation or the $I^+$ notation. Therefore, the set of strings over $I$, which is the set of hypertrees of level 1 over $I$, is denoted by $H_1(I)$, and the hypertrees of level 2 over $I$, the set of trees over $I$, is denoted as $H_2(I)$. In either case, if the hypertrees are over $I$ for the remainder of this paper, the "$(I)$" will be elided unless confusion would result. Therefore, the set of strings over $I$ is denoted as $H_1$, the set of trees over $I$ is denoted $H_2$, and in general, the set of hypertrees of level $n$ over $I$ will be denoted by $H_n$.

In order to develop the general definition for hypertrees, some additional changes need to be made to the definition of strings given in definition 7 and the definition of trees given in definition 9. Using the new notation, the equation in definition 7 may be written as

$$H_1 = \Sigma U \Sigma H_1$$

At this level, there is no ambiguity about what follows the initial element of $I$ in the string, but in general this is not the case. Therefore, the part of the string which follows $I$ is indicated by the use of brackets.

$$H_1 = \Sigma U \Sigma [ H_1 ]$$
In order that these brackets not be confused with brackets at other levels of the hierarchy, a subscript is introduced. Therefore, the equation becomes

\[ H_1 = \Sigma \cup \Sigma[H_1] \]

Similarly, the equation in definition 9 may be written as

\[ H_2 = \Sigma \cup \Sigma[H_2]^i \]

As above, subscripts need to be added to avoid confusion. The equation now becomes

\[ H_2 = \Sigma \cup \Sigma[H_2]^i \]

A reasonable interpretation for the \( U_{i \geq 1}(H_2)^i \) is the set of strings over \( H_2 \), that is the set of strings of trees, which is \( H_1(H_2) \). Therefore, the above equation becomes

\[ H_2 = \Sigma \cup \Sigma[H_1(H_2)] \]

Note that since \( H_2 \) is an infinite set, the set of strings must be defined over infinite sets. This is also extended to all hypertrees.
A logical extension of the hypertree hierarchy in the downward direction is to the set of elements of \( \mathcal{E} \), that is, to \( \mathcal{E} \) itself. This is reasonable if it is noted that trees have two dimensions, height and width, whereas strings have only one dimension, length. Continuing on down, the elements of \( \mathcal{E} \) have neither height, width nor length and are, therefore, zero dimensional objects. Therefore, it is reasonable to expect that the set of level 0 hypertrees over \( \mathcal{E} \) is simply \( \mathcal{E} \) itself. That is

\[ H_0 = \mathcal{E} \]

Using this fact, equation 1 may be rewritten as

\[ H_1 = \mathcal{E} U \mathcal{E}[_1 H_0(H_1) _1] \]

since \( H_0 \) is the identity function.

From equations 2 and 4, it should be evident that in general

\[ H_n = \mathcal{E} U \mathcal{E}[_n H_{n-1}(H_n) _n] \]

and indeed this is the case. The one exception has been noted above in equation 3. It will be seen later that this definition is somewhat restrictive, therefore, in the formal definition of hypertrees a lower case \( h \) is used in place of the upper case \( H \).
Definition 10:

The set of hypertrees over $\Sigma$ of level $n$, denoted $h_n$, is the smallest set such that

(a) $h_0 = \Sigma$

(b) $h_n = \Sigma \cup \Sigma \{ n \ h_{n-1}(h_n) \ n \}$

if $n > 0$.

---

TABLE 1. Some Examples of Hypertrees over \{a,b,c\}

<table>
<thead>
<tr>
<th>level examples</th>
<th>line level examples</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>0 a</td>
</tr>
<tr>
<td>2</td>
<td>1 a</td>
</tr>
<tr>
<td>3</td>
<td>1 a[1b[1c1]1]</td>
</tr>
<tr>
<td>4</td>
<td>2 a</td>
</tr>
<tr>
<td>5</td>
<td>2 a[2a[1b[1c1]1]2]</td>
</tr>
<tr>
<td>7</td>
<td>3 a</td>
</tr>
<tr>
<td>8</td>
<td>3 a[3a[2a[1b[1c1]1]2]3]</td>
</tr>
</tbody>
</table>

If $\Sigma = \{a,b,c\}$, then some examples of hypertrees are given in table 1. In each case, at level 2 and below, in view of the above discussion, the usual parenthesized
notation for the string or tree can be recovered by deleting the one level brackets and deleting the subscripts on the two level brackets. Therefore, line 6 of table 1 is usually written as $a[a[ab]bc[a[ab]]]$.

![A Projection of a Level 3 Hypertree](image)

**FIGURE 5. A Projection of a Level 3 Hypertree**

Throughout this paper, hypertrees are defined and used as sets of strings rather than graphs. The reason for this is that strings and trees, being one and two dimensional objects, have simple two dimensional projections. However, this is not the case in general. A level $n$ hypertree is an $n$ dimensional object. For example, the hypertree given in line 9 of table 1 is of level 3, and, therefore, is a three dimensional object. This object may be represented by a 3 dimensional graph. A projection of this graph into 2 dimensions is given in figure 5. Hypertrees of levels higher than 3 are not uncommon, so even models cannot be built of these. For this reason, no further attempts will
be made at using graphs to illustrate hypertrees in this paper.

In definition 10, line b, the expression "h_{n-1}(h_n)" occurs. This expression occurs so often in the following pages that a special notation is given for it

\[ h_n^k = h_k(h_{k+1}(\ldots (h_{n-1}(h_n))\ldots )) \]

In the case of trees, this reduces to

\[ h_2^1 = h_1(h_2) \]

which is the set of strings of trees, or in other words the set of forests over \( \Sigma \). For this reason, \( h_n^k \) is referred to as the set of hyperforests of degree \( k \) and level \( n \) over \( \Sigma \). In order to reduce the number of special cases in the formal definition, the hyperforest \( h_n^{n+1} \) is taken as being equal to \( \Sigma \). Note also that

\[ h_n^0 = h_0(h_n^1) \]

\[ = h_n^1 \]

In general,

\[ h_n^k = h_k(h_n^{k+1}) \]

\[ = h_n^{k+1} U h_n^{k+1}[k h_{k-1}(h_n^k) k] \]

\[ = h_n^{k+1} U h_n^{k+1}[k h_n^{k-1} k] \]
Since it is often useful to prove things about hyperforests and then note that a hypertree is a special case of a hyperforest, a formal definition of hyperforests is in order.

Definition 11:
The set of hyperforests of degree $k$ and level $n$ over $I$, denoted $h_n^k$, is the smallest set such that

(a) $h_n^{n+1} = I$

if $n \geq 0$.

(b) $h_n^k = h_n^{k+1} \cup h_n^{k+1} [ h_n^{k-1} \backslash h_n^k ]$

if $n \geq k > 0$.

(c) $h_n^0 = h_n^1$

if $n \geq 0$.

This definition of hyperforest does not obviously have the connection to hypertrees that is mentioned in the above discussion. This connection is proved in the following theorem.

Theorem 1:
For all $n \geq k \geq 0$,

$h_n^k = h_k(h_{k+1}( \ldots (h_n) \ldots ))$

Proof:
Define

$q_n^k(\xi) = h_k(h_{k+1}( \ldots (h_n) \ldots ))$
for all \( n \geq k \geq 0 \). Also, define

\[
\alpha_{n+1}^{k}(\xi) = \xi
\]

for all \( n \geq 0 \). If \( n > k \geq 0 \), then

\[
\alpha_{n}^{k}(\xi) = h_{k}(\alpha_{n+1}^{k}(\xi))
\]

If \( n = k \), then

\[
\alpha_{n}^{k}(\xi) = h_{n} = h_{n}(\alpha_{n+1}^{k}(\xi)) = h_{k}(\alpha_{n}^{k+1}(\xi))
\]

By definition 10, therefore, \( \alpha_{n}^{k}(\xi) \) is the smallest set such that

\[
\alpha_{n}^{0}(\xi) = h_{0}(\alpha_{n}^{1}(\xi)) = \alpha_{n}^{1}(\xi)
\]

if \( n \geq 0 \).

\[
\alpha_{n}^{k}(\xi) = h_{k}(\alpha_{n}^{k+1}(\xi)) = \alpha_{n}^{k+1}(\xi) \cup \alpha_{n}^{k+1}(\xi)[h_{k-1}(h_{k}(\alpha_{n}^{k+1}(\xi))) k]
\]

\[
= \alpha_{n}^{k+1}(\xi) \cup \alpha_{n}^{k+1}(\xi)[\alpha_{n}^{k-1}(\xi) k]
\]
if \( n \geq k > 0 \). But equations 5, 6 and 7 are precisely the same as equations a, c and b respectively in definition 11 except that \( a_n^k(\xi) \) replaces \( h_n^k \). Therefore, since \( h_n^k \) is the smallest set that meets these conditions, and \( a_n^k(\xi) \) is also the smallest set, it must be true that

\[
\begin{align*}
  h_n^k &= a_n^k(\xi) \\
       &= h_k(h_{k+1}(\ldots(h_n\ldots)))
\end{align*}
\]

and the theorem is proved.

---

Corollary 2:

For all \( n \geq 0 \),

\[
h_n = h_n^n
\]

Proof:

Substitute \( n \) for \( k \) in theorem 1.

---

A lot of the proofs in this paper depend on the use of induction. However, definition 11 does not lend itself readily to the use of induction because of the occurrence of the "\( k+1 \)" as well as the "\( k-1 \)" in equation b. Therefore, a new variable is introduced to act as the induction variable. Since it corresponds loosely with the height of trees, this variable is called the depth of the hypertree. The definition corresponds rather closely to the definition of
hyperforests, except that "m" is used to keep track of how many times the definition has recurred.

**Definition 12:**

The set of hyperforests over \( \mathcal{I} \) of level \( n \), degree \( k \) and depth \( m \), denoted \( h_{n,m}^k \), is the smallest set such that

(a) \( h_{n,m}^{n+1} = \Sigma \) if \( n \geq 0 \) and \( m \geq 0 \).

(b) \( h_{n,0}^k = \phi \) if \( n \geq k \geq 0 \).

(c) \( h_{n,m+1}^k = h_{n,m}^k \cup h_{n,m}^{k+1} \cup h_{n,m}^{k+1} [ h_{n,m}^{k-1} ] \) if \( n \geq k > 0 \) and \( m \geq 0 \).

(d) \( h_{n,m}^0 = h_{n,m}^1 \) if \( n > 0 \) and \( m \geq 0 \).

The task now is to prove that this definition does in fact allow proofs about hyperforests as has been claimed. This will be accomplished by showing that

\[
h_n^k = \bigcup_{m \geq 0} h_{n,m}^k
\]

This, in turn, requires some lemmas.

**Lemma 3:**

For all \( n \geq 0 \) and \( n+1 \geq j \geq k \geq 0 \),

\[
h_j^k \subset h_n^k
\]
Proof by induction on $k$:

Basis: if $j = k$ then

$$h_n^j = h_n^k$$

Therefore,

$$h_n^j \leq h_n^k$$

Induction step: Assume that if $k' > k$ then

$$h_n^j \leq h_n^{k'}$$

Since $k \geq 0$, there are two cases to be considered:

1. If $k > 0$, then

$$h_n^k = h_n^{k+1} \cup h_n^{k+1} \cup h_n^{k-1}$$

and by induction hypothesis,

$$h_n^j \leq h_n^{k+1} \leq h_n^k$$

2. If $k = 0$, then

$$h_n^j \leq h_n^1$$

by induction hypothesis. Also,

$$h_n^1 = h_n^0 = h_n^k$$

Therefore,

$$h_n^j \leq h_n^k$$
In both 8 and 9,

\[ h_n^j \subset h_n^k \]

so the theorem is proved by induction.

---

Lemma 4:

If \( n \geq k > 0 \) and \( m \geq 0 \), then

\[ h_{n,m+1}^k = h_{n,m}^k U \ h_{n,0}^k U \ h_{n,m+1}[k \ h_{n,m}^{k-1}] \]

Proof by induction on \( m \):

Basis: If \( m = 0 \), then

\[ h_{n,1}^k = h_{n,0}^k U \ h_{n,0}^k U \ h_{n,0}^k[k \ h_{n,0}^{k-1}] \]

\[ = h_{n,m}^k U \ h_{n,0}^k U \ h_{n,m}[k \ h_{n,m}^{k-1}] \]

by definition 12.

Induction step: Assume that if \( m \geq 0 \) then

\[ h_{n,m+1}^k = h_{n,m}^k U \ h_{n,0}^k U \ h_{n,m}[k \ h_{n,m}^{k-1}] \]

Substituting this equation into the appropriate line in definition 12 will allow the expansion

\[ (10) \ h_{n,m+2}^k = h_{n,m+1}^k U \ h_{n,m}^k U \ h_{n,0}^k \]

\[ U \ h_{n,m+1}[k \ h_{n,m}^{k-1}] U \ h_{n,m+1}[k \ h_{n,m+1}^{k-1}] \]

\[ = h_{n,m+1}^k U \ h_{n,0}^k U \ h_{n,m+1}[k \ h_{n,m+1}^{k-1}] \]
since
\[ h_{n,m}^{k+1} < h_{n,m+1}^{k+1} \]
and
\[ h_{n,m}^{k-1} < h_{n,m+1}^{k-1} \]
Notice that equation 10 is the same as the induction hypothesis except each occurrence of \( m \) has been incremented by one. Therefore,
\[
\begin{align*}
\cdot & \cdot \\
\Rightarrow h_{n,m+1}^{k} = h_{n,m}^{k+1} U h_{n,0}^{k} U h_{n,m}^{k+1} U h_{n,m}^{k+1} U h_{n,m}^{k-1}
\end{align*}
\]
for all \( m \geq 0 \) by induction.

---

**Lemma 5:**

For all \( m \geq 0, n \geq 0 \) and \( n+1 \geq k \geq 0 \),
\[ h_{n,m}^{k} < h_{n}^{k} \]
Proof by induction on \( m \):

**Basis:** if \( m = 0 \) then there are three cases that need to be considered, \( k = n+1, n \geq k > 0 \) and \( k = 0 \).

1. If \( k = n+1 \), then
   \[ h_{n,m}^{k} = 2 \]
   \[ = h_{n}^{k} \]

2. If \( n \geq k > 0 \), then
   \[ h_{n,m}^{k} = \phi \]
   \[ = h_{n}^{k} \]

3. If \( k = 0 \), then
30

\[ h_{n,m}^0 = h_{n,m}^1 \]

By part 1 if \( n = 0 \) or part 2 above if \( n > 0 \)

\[ h_{n,m}^1 < h_{n}^1 \]

But since \( h_{n}^1 = h_{n}^0 \) it must be that

\[ h_{n,m}^0 < h_{n}^0 \]

Therefore, in all three cases if \( m = 0 \) then

\[ h_{n,m}^k < h_{n}^k \]

Induction step: Assume that if \( m' < m \) then

\[ h_{n,m'}^k < h_{n}^k \]

There are three cases that need to be considered.

1. If \( k = n+1 \), then

\[ h_{n,m}^{n+1} = \emptyset \]

Therefore,

\[ h_{n,m}^k < h_{n}^k \]

(11)

2. If \( n \geq k > 0 \), then

\[ h_{n,m}^k = h_{n,m-1}^{k+1} \cup h_{n}^k \cup h_{n,m-1}^{k+1} \cup h_{n}^{k-1} \]

by lemma 4. Therefore, since \( h_{n,0}^k = \emptyset \) it must be that
31

\[ h_{n,m}^k = h_{n,m-1}^{k+1} \cup h_{n,m-1}^{k+1}[k h_{n,m}^{k-1} k] \]

By induction hypothesis

\[ h_{n,m-1}^{k+1} \subseteq h_{n,n}^{k+1} \]

\[ h_{n,m-1}^{k-1} \subseteq h_{n,n}^{k-1} \]

therefore,

\[ (12) \quad h_{n,m}^k \subseteq h_{n,n}^{k+1} \cup h_{n,n}^{k+1}[k h_{n,n}^{k-1} k] \]

3. If \( k = 0 \), then

\[ h_{n,m}^0 = h_{n,n}^1 \]

But

\[ h_{n,m}^1 \subseteq h_{n,n}^1 \]

as a special case of equation 11 or 12 depending on the value of \( n \). Therefore,

\[ h_{n,m}^0 \subseteq h_{n,n}^1 \]

\[ \subseteq h_{n,n}^0 \]

So, in this case also

\[ (13) \quad h_{n,m}^k \subseteq h_{n,n}^k \]

Therefore, by equations 11, 12 and 13

\[ h_{n,m}^k \subseteq h_{n,n}^k \]
if $m > 0$ and the induction hypothesis is assumed. Therefore, the lemma is proved by induction.

---

Lemma 6:

If $n \geq 0$, $n+1 \geq k \geq 0$ and $m \geq 0$, then

$$h^k_{n,m} = \bigcup_{m \geq m' \geq 0} h^k_{n,m'}$$

Proof by induction on $m$:

Define

$$\alpha^k_{n,m}(\Sigma) = \bigcup_{m \geq m' \geq 0} h^k_{n,m'}$$

Basis: if $m = 0$ then

$$\alpha^k_{n,0}(\Sigma) = \bigcup_{m'=0} h^k_{n,m'} = h^k_{n,0}$$

Induction step: Assume that if $m > 0$ then

$$\alpha^k_{n,m}(\Sigma) = h^k_{n,m}$$

Then by the definition of union,

$$\alpha^k_{n,m+1}(\Sigma) = h^k_{n,m+1} \bigcup \alpha^k_{n,m}(\Sigma)$$

Therefore, by induction hypothesis
\[ a_{n,m+1}^k(x) = h_{n,m+1}^k U h_{n,m}^k \]

Since \( h_{n,m}^k \subseteq h_{n,m+1}^k \), it must be true that

\[ a_{n,m+1}^k(x) = h_{n,m+1}^k \]

if \( m \geq 0 \). Therefore, by induction

\[ a_{n,m}^k(x) = h_{n,m}^k \]

Which is the same as saying

\[ h_{n,m}^k = U_{m \geq m' \geq 0} h_{n,m'}^k \]

Therefore, the lemma is proved.

---

**Theorem 7:**

For all \( n \geq 0 \) and \( n+1 \geq k \geq 0 \),

\[ h_{n}^k = U_{m \geq 0} h_{n,m}^k \]

**Proof:**

To prove that

\[ h_{n}^k = U_{m \geq 0} h_{n,m}^k \]

it will be shown that \( U_{m \geq 0} h_{n,m}^k \) has all the properties that define \( h_{n}^k \). Define

\[ a_{n,m}^k(x) = U_{m \geq m' \geq 0} h_{n,m'}^k \]
By lemma 6

\[ a_{n,m}^k(\xi) = h_{n,m}^k \]

Therefore, \( a_{n,m}^k(\xi) \) is the smallest set such that the following three equations are true:

1. If \( n \geq 0, k = n+1 \) and \( m \geq 0 \), then

\[ a_{n,m}^{n+1}(\xi) = \xi \]

2. If \( n \geq k > 0 \) and \( m \geq 0 \), then by lemma 4

\[ a_{n,m}^k(\xi) = a_{n,m}^{k+1}(\xi) \cup h_{n,0}^k(\xi) \cup a_{n,m}^{k+1}(\xi)[k a_{n,m}^{k-1}(\xi)]_k \]

3. If \( n \geq 0 \) and \( m \geq 0 \), then

\[ a_{n,m}^0(\xi) = a_{n,m}^1(\xi) \]

By the definition of infinite union

\[ \lim_{m \to \infty} a_{n,m}^k(\xi) = U_{m \geq 0} h_{n,m}^k \]

Therefore, if \( a_{n}^k(\xi) \) is defined as

\[ a_{n}^k(\xi) = U_{m \geq 0} h_{n,m}^k \]

then \( a_{n}^k(\xi) \) is the smallest set such that
1. If \( n \geq 0 \) and \( k = n+1 \), then

\[
\alpha_{n+1}^{n}(\xi) = \xi
\]

2. If \( n \geq k > 0 \), then

\[
a_{n}^{k}(\xi) = a_{n}^{k+1}(\xi) \cup h_{n,0}^{k} \cup a_{n}^{k+1}(\xi) [k a_{n}^{k-1}(\xi) k]
\]

3. If \( n \geq 0 \) and \( k = 0 \), then

\[
\alpha_{n}^{0}(\xi) = \alpha_{n}^{1}(\xi)
\]

But since \( h_{n,0}^{k} = \emptyset \) the second equation becomes

\[
a_{n}^{k}(\xi) = a_{n}^{k+1}(\xi) \cup a_{n}^{k+1}(\xi) [k a_{n}^{k-1}(\xi) k]
\]

Now since \( h_{n}^{n} \) is the smallest set with properties 14, 15 and 16 it must also be true that

\[
h_{n}^{k} = a_{n}^{k}(\xi)
\]

\[
= U_{m \geq 0} h_{n,m}^{k}
\]

and the theorem is proved.

- • -
As was stated earlier, the definition of hypertrees given in definition 10 is somewhat restrictive. Consider the case of the frontier. For a tree, this is simple. All that needs to be done is that the leaves of the tree -- those nodes which don't have any subtrees -- are listed in the order that they occur. For example, the tree in figure 6 has the frontier "abbab". However, consider the hypertree in figure 5. Since this is a level 3 hypertree, the frontier should be a level 2 hypertree, a tree. It is easy to see that the frontier starts out as a[ba[bc]a], but the bottommost "a" doesn't have any subhypertrees, and, therefore, by the logic given above, should be on the frontier. But where does the "a" get attached? This particular problem leads to the redefinition of hypertree.
In this new definition, it has to be explicitly stated where the subtrees are to be attached during the frontiering operation, as will be seen.

First, some way of referring to a particular subhypertree in a hypertree must be found. This can be done by finding a way to refer to the head of the hypertree only since once the location of the head is known it is quite easy to find a complete hypertree of any given level. In the case of level 2 hypertrees (trees), this is trivial, and even at level 3 it is not a difficult problem but at higher levels it becomes very much more difficult. Therefore, a set of "paths" are defined for locating a particular subhypertree in a hypertree. A path describes how to follow the pointers in a hypertree to get to the head of the desired subhypertree.

Definition 13:

The set of legal n-paths of degree k, denoted $P^k_n$, is the smallest set such that

(a) $P^0_n = \{1\}$

if $n \geq 0$.

(b) $P^k_n = P^{k+1}_n \cup kP^{k-1}_n$

if $n > k > 0$.

(c) $P^0_n = P^1_n$

if $n > 0$. 

Having defined n-paths, a new definition for hypertrees can be given. Since this definition allows for frontiering, these are called frontierable hypertrees. In frontierable hypertrees, a new type of symbol is introduced. It is called "X" and is used to hold one of the n-paths just defined. Since in the definition of frontiering, it is important to know a level also, there is a level encoded in the X. The use of the X will be obvious when frontiering is introduced.

Definition 14:

The set of frontierable hypertrees over $\Sigma$ of level $n$, denoted $H^*_n$, is the smallest set such that

(a) $H^*_0 = \Sigma$

(b) $H^*_n = \Sigma \cup X_n \cup \Sigma[H_n(H^*_n)]$

if $n > 0$ and $X_n = \{X^j_n \mid j \in \mathbb{P}^n\}$.

The same discussion which lead to the definition of hyperforests of degree $k$, definition 11, leads to the definition of frontierable hyperforests of degree $k$ which correspond directly to hyperforests. Similarly, hyperforests of degree $k$ and depth $m$ have a counterpart called frontierable hyperforests of degree $k$ and depth $m$. These are defined as follows:
Definition 15:

The set of frontierable hyperforests over $\mathcal{I}$ of level $n$ and degree $k$, denoted $H_n^k$, is the smallest set such that

(a) $H_n^{n+1} = \mathcal{I}$ if $n \geq 0$.

(b) $H_n^0 = H_n^1$ if $n \geq 0$.

(c) $H_n^k = H_n^{k+1} \cup X_k \cup H_n^{k+1}[k H_n^{k-1} k]$ if $n \geq k > 0$ and $X_k = \{X_k^j \mid j \in P_k^{k-1}\}$.

Definition 16:

The set of frontierable hyperforests over $\mathcal{I}$ of level $n$, degree $k$, and depth $m$, denoted $H_{n,m}^k$, is the smallest set such that

(a) $H_{n,m}^{n+1} = \mathcal{I}$ if $n \geq 0$ and $m \geq 0$.

(b) $H_{n,0}^k = X_k$ if $n \geq k > 0$ and $X_k = \{X_k^j \mid j \in P_k^{k-1}\}$.

(c) $H_{n,m+1}^k = H_{n,m}^{k+1} \cup H_{n,m}^k \cup H_{n,m}^{k+1}[k H_{n,m}^{k-1} k]$ if $n \geq k > 0$ and $m \geq 0$.

(d) $H_{n,m}^0 = H_{n,m}^1$ if $n \geq 0$ and $m \geq 0$. 

There is also a set of theorems corresponding to theorems 1 to 7. In each case, the proof follows the corresponding proof in the original theorem so closely that it need not be given here.

Theorem 8:
For all \( n \geq k \geq 0 \),

\[
H_n^k = H_k(H_{k+1}( \ldots (H_n) \ldots ))
\]

Proof:
This proof is the same as the proof of theorem 1.

- - -

Corollary 9:
For all \( n \geq 0 \),

\[
H_n = H_n^n
\]

Proof:
This proof is the same as the proof of corollary 2.

- - -

Lemma 10:
For all \( n \geq 0 \) and \( n+1 \geq j \geq k \geq 0 \),

\[
H_n^j \subset H_n^k
\]

Proof:
This proof is the same as the proof for lemma 3.

- - -
Lemma 11:
For all $n \geq 0$, $n+1 \geq k \geq 0$ and $m \geq 0$,
\[
H_{n,m}^k \subseteq H_n^k
\]
Proof:
This proof is the same as the proof of lemma 5.

Lemma 12:
If $n \geq 0$, $n+1 \geq k \geq 0$ and $m \geq 0$, then
\[
H_{n,m}^k = \bigcup_{m' \geq 0} H_{n,m'}^k
\]
Proof:
This proof is the same as lemma 6.

Lemma 13:
For all $n \geq k > 0$ and $m \geq 0$,
\[
H_{n,m+1}^k = H_{n,m}^k \cup H_{n,0}^k \cup H_{n,m}^{k+1} \cup H_{n,m}^{k-1}
\]
Proof:
This proof is the same as the proof for lemma 4.

Theorem 14:
For all $n \geq 0$ and $n+1 \geq k \geq 0$,
\[
H_n^k = \bigcup_{m \geq 0} H_{n,m}^k
\]
Proof:
This proof is the same as the proof for theorem 7.

- - -

Having defined a path as a means of selecting a subhypertree, and having defined frontierable hypertrees, the selection function can now be defined. This function, called the search function, takes as its arguments a path and a hyperforest and returns the selected hypertree. The method of doing this is to consider the top level of the hyperforest -- that is the root nodes of all the sub-hypertrees -- and proceed down the path in the following manner. If there is a subhyperforest in the top level hyperforest described above with a degree corresponding to the first number in the path, then that hyperforest is selected and the top number is removed from the path. If there is no hyperforest corresponding to that number, then the function is not defined. This continues recursively until the end of the path is reached at which time the subhypertree corresponding to the first node is selected.

For example, if the hyperforest of level 4 and degree 2 is the hyperforest given in figure 7 and the path is "21321" then, after the first recursion the path becomes "1321" and the hyperforest is

\[ b[1c[3a[2b[1c12]\3][2a2][1a11}\]
After recursion 2, the path is "321" and the hyperforest is

\[ c[3a[2b[1c_1]_2]_3][2a_2][1a_1] \]

After the third recursion, the path is "21" and the hyperforest is

\[ a[2b[1c_1]_2] \]
Then, after this pass, the path becomes "1" and the hyperforest is

\[ b[1c_1] \]

And lastly, the path becomes null and the hyperforest is

\[ c \]

so the selected hypertree is "c". Note that although this is a level 4 hypertree there are no level four brackets so that any selected subhypertrees will be single nodes. If the path had been "221", which is illegal, then recursion 2 would have had path "21" and hyperforest

\[ b[1c_3a_2b[1c_1]a_3][2a_2][1a_1]1 \]

which has no subhypertree corresponding to number 2, so the select function fails and is not defined for these arguments.

Notice that the level of the hypertree has to be known at the time that the hypertree is finally selected so that the end of the selected hypertree can be determined. Also, it is convenient to know the level of the hyperforest that
is being input. Therefore, instead of a single selection function there is a family of selection functions, each corresponding to hyperforests of a given level and degree.

Definition 17:
Search is a family of functions, denoted $\text{search}^k_n$, such that

$$\text{search}^k_n : P^k_n \times H^k_n \rightarrow H_n$$

if $n \geq k \geq 0$. The value of the functions are defined as follows:

(a) $\text{search}^n_n(j, \gamma) = \gamma$
if $n > 0$.

(b) $\text{search}^0_n(j, \gamma) = \text{search}^1_n(j, \gamma)$
if $n > 0$.

(c) $\text{search}^k_n(j, \gamma) = \text{search}^{k+1}_n(j, \gamma)$
if $\gamma \in H^{k+1}_n$ and $j = k \cdot i$.

$\text{search}^{k+1}_n(j, \alpha)$
if $\gamma \in X_k$.

$\text{search}^{k+1}_n(j, \alpha)$
if $\gamma = \alpha[j \beta \kappa] \text{ and } j \in P^{k+1}_n$.

$\text{search}^{k-1}_n(i, \beta)$
if $\gamma = \alpha[j \beta \kappa] \text{ and } j = k \cdot i$.

if $n \geq k > 0$. 


Notice in equation a of definition 17, j must equal \( \lambda \) due to domain constraints.

### TABLE 2. Equation c of Definition 9

<table>
<thead>
<tr>
<th>j ( \text{other} )</th>
<th>ki</th>
</tr>
</thead>
<tbody>
<tr>
<td>other</td>
<td>search( n^{k+1}(j,\gamma) )</td>
</tr>
<tr>
<td>( \alpha[\kappa \beta \kappa] )</td>
<td>search( n^{k+1}(j,\alpha) )</td>
</tr>
</tbody>
</table>

if \( n > k > 0 \).

Table 2 gives a diagram to aid in the understanding of definition 17. Notice that since \( n > k > 0 \)

\[
j \in p_n^{k+1} U k p_n^{k-1}
\]

by line c of definition 13. Therefore,

\[
j \in p_n^{k+1}
\]

or else j is a string that begins with k. Similarly,

\[
\gamma \in H_n^{k+1} U X_k U H_n^{k+1} \{k H_n^{k-1} k\}.
\]
In general, if the $n$-path $j$ does not start with a number corresponding to the level of this search function then the next higher level search function is called on that part of the hypertree that forms the next higher level. This will continue to search the hyperforest until either the level which corresponds with the first number in $j$ is called or $\text{search}_n^n$ is called. In the latter case,

$$j \in \mathcal{P}_n^n = \{1\}$$

so that a hypertree is selected via line 1 of the definition. In the former case, the part of the hypertree in the brackets that correspond to the level selected is searched using the remainder of the path. If nothing is bracketed at that level, then the search is undefined.

It has been stated that the codomain of the $\text{search}_n^k$ function is $H_n^n$. However, there is nothing in the definition which guarantees that this is the case. Therefore, a theorem is presented which proves that the elements defined by the search function are in fact hypertrees, or, in other words, that the search function is well-formed.

Lemma 15:

If $n \geq k \geq 0$, $m \geq 0$, $\gamma \in H_{n,m}^k$, $j \in \mathcal{P}_n^n$ and $\text{search}_n^k(j, \gamma)$ is defined, then

$$\text{search}_n^k(\gamma) \in H_n^n$$
Proof by induction on m:

Basis: if $m = 0$ then either $n = k$, $n > k > 0$ or $k = 0$.

1. If $n = k$, then

$$\exists \in H^0_{n, 0}$$

$$\in H^n_n$$

$$\in H_n$$

by theorem 14 and corollary 9. Therefore,

$$search^k_n(j, \exists) = \exists$$

$$\in H_n$$

by definition 17.

2. If $n > k > 0$, then

$$\exists \in H^k_{n, 0}$$

$$\in X_k$$

by definition 16. Therefore, $search^k_n(j, \exists)$ is undefined.

3. If $k = 0$, then

$$search^0_n(j, \exists) = search^1_n(j, \exists)$$

and is undefined if $n > 0$. If $n = 0$, then

$$search^0_n(j, \exists) \in H_n$$

by part 1 of this proof.
In any event, if \( \text{search}^k_n(j, \gamma) \) is defined then

\[
\text{search}^k_n(j, \gamma) \in H_n
\]

Induction step: Assume that if \( m > m' \geq 0 \) and \( \gamma \in H^k_{n,m'} \), then

\[
\text{search}^k_n(j, \gamma) \in H_n
\]

if it is defined. If \( \gamma \in H^k_{n,m} \), then one of the following cases must be true:

1. If \( k = n \), then

\[
\text{(18)} \quad \text{search}^k_n(j, \gamma) = \gamma
\]

by theorem 14 and corollary 9.

2. If \( n > k > 0 \), \( \gamma \in H^{k+1}_{n,m} \) and \( j \in \mathbb{P}^{k+1}_n \), then

\[
\text{search}^k_n(j, \gamma) = \text{search}^{k+1}_n(j, \gamma)
\]

\[
\in H_n
\]

or is not defined by induction hypothesis.

3. If \( n > k > 0 \), \( \gamma \in H^{k+1}_{n,m} \) and \( j \in \mathbb{P}^{k-1}_n \), then \( \text{search}^k_n(j, \gamma) \) is not defined.

4. If \( n > k > 0 \), \( \gamma \in H^k_{n,m} \) and \( \text{search}^k_n(j, \gamma) \) is defined, then

\[
\text{search}^k_n(j, \gamma) \in H_n
\]
by induction hypothesis.

5. If \( n > k > 0 \), \( \gamma = a_k \beta_k \) and \( j \in \mathbb{P}_{n+1} \), then

\[
\gamma \in H_{n,m},
\]

and

\[
\text{search}^k_n(j, \gamma) = \text{search}^{k+1}_n(j, \alpha)
\]

\[
\in H_n
\]

6. If \( n > k > 0 \), \( \gamma = a_k \beta_k \) and \( j = ki \), then

\[
\beta \in H_{n,m-1}
\]

and

\[
\text{search}^k_n(j, \gamma) = \text{search}^{k-1}_n(i, \beta)
\]

\[
\in H_n
\]

by induction hypothesis.

7. If \( k = 0 \), then

\[
\text{search}^k_n(j, \gamma) = \text{search}^1_n(j, \gamma)
\]

\[
\in H_n
\]

as a special case of one of the previous cases of this proof.

In all cases, assuming the induction hypothesis allows a proof that

\[
\text{search}^k_n(j, \gamma) \in H_n
\]
or else it is not defined, so that the lemma is proved by induction.

---

Theorem 15:
If \( n \geq k \geq 0, \ y \in H^k_n, \ j \in P^k_n \) and \( \text{search}_n^k(j, y) \) is defined, then
\[
\text{search}_n^k(j, y) \in H_n
\]

Proof:
If \( y \in H^k_n \), then there must be an \( m \) such that
\[
y \in H^k_{n,m}
\]
by theorem 14. Therefore,
\[
\text{search}_n^k(j, y) \in H_n
\]
if it is defined and the theorem is proved.

---

Another function which is needed for the definition of the frontier function is the apply function. The apply function will take a hyperforest and replace all of the \( X^j_n \)'s with corresponding hypertrees. This is done by searching each of the nodes of the input hyperforest for nodes which are of the form \( X^j_n \). All other nodes are ignored. At these nodes, the appropriate search function is called to locate the subhypertree in the hyperforest that is to replace the
Like search, apply is a family of functions, denoted $\text{apply}_n^k$, each returning a different level or degree of hyperforest.

**Definition 18:**

Apply is a family of partial functions, denoted $\text{apply}_n^k$, such that

$$\text{apply}_n^k : H_n^k \times H_{n-1}^k \rightarrow H_n^k$$

for all $n > 0$ and $n+1 \geq k \geq 0$. The value of the functions are defined as follows:

(a) $\text{apply}_n^{n+1}(\gamma, \psi) = \gamma$

if $n > 0$.

(b) $\text{apply}_n^0(\gamma, \psi) = \text{apply}_n^1(\gamma, \psi)$

if $n > 0$.

(c) $\text{apply}_n^k(\gamma, \psi) = \text{search}_n^{n-1}(j, \psi)$

if $\gamma \in X_n^j$ and $n = k$.

$\text{apply}_n^k(\gamma, \psi) = \text{apply}_n^k(\gamma, \psi)$

if $\gamma \in H_n^{k+1}$.

$\text{apply}_n^k(\gamma, \psi) = \text{apply}_n^{k+1}(\alpha, \psi)[\text{apply}_n^{k-1}(\beta, \psi)_k]$

if $\gamma = \alpha \{ \beta \}_k$. 

if $n \geq k > 0$. 

*
Just as was the case with the search function, the statement of the codomain of the apply function was made with no proof that the definition in fact maps the domain into the codomain. This proof will now be supplied in the following theorem.

Lemma 17:

Proof by induction on \( m \):

Basis: If \( m = 0 \), then there are four possible cases:

1. If \( k = n+1 \), then by the definition of apply

\[
\text{apply}_n^k(\gamma, \psi) = \gamma
\]

\( \in H_{n+1}^k \)

2. If \( k = n \) and \( \text{search}_{n-1}^n(j, \psi) \) is defined, then

\[
\text{apply}_n^k(\gamma, \psi) = \text{search}_{n-1}^n(j, \psi)
\]

\( \in H_n^k \)

\( \in H_n^k \)

by theorem 16 and corollary 9 where \( \gamma = X_{n,j}^d \).

3. If \( n > k > 0 \), then
(19) \[ \text{apply}^k_n(\gamma, \psi) = \gamma \in H^k_n \]

since \( \gamma \in X_k \) by definition 16.

4. If \( k = 0 \), then

\[ \text{apply}^0_n(\gamma, \psi) = \text{apply}^1_n(\gamma, \psi) \in H^1_n \]

as a special case of equation 19. Notice that \( n \neq 0 \).

Therefore, since \( H^0_n = H^1_n \),

\[ \text{apply}^k_n(\gamma, \psi) \in H^0_n \]

Since \( \text{apply}^k_n(\gamma, \psi) \in H^k_n \) in each case, the basis is proved.

Induction step: Assume that if \( m > m' \geq 0 \), \( \gamma \in H^k_{n,m'} \), and \( \psi \in H^{n-1}_n \) then

\[ \text{apply}^k_n(\gamma, \psi) \in H^k_n \]

if it is defined. If \( \gamma \in H^k_{n,m'} \) then one of the cases given below must be true:

1. If \( k = n+1 \), then

\[ \text{apply}^{n+1}_n(\gamma, \psi) = \gamma \in H^{n+1}_n \]

due to the domain constraint.
2. If $n \geq k > 0$, $\xi \varepsilon H_{n,m-1}^{k+1}$ and $\text{apply}_n^{k+1}(\xi,\psi)$ is defined, then

$$\text{apply}_n^k(\xi,\psi) = \text{apply}_n^{k+1}(\xi,\psi)$$

by induction hypothesis. Also, since $H_{n}^{k+1} \subseteq H_{n}^{k}$,

$$\text{apply}_n^k(\xi,\psi) \varepsilon H_{n}^{k}$$

3. If $n \geq k > 0$, $\xi \varepsilon H_{n,m-1}^{k}$ and $\text{apply}_n^{k}(\xi,\psi)$ is defined, then

$$\text{apply}_n^k(\xi,\psi) \varepsilon H_{n}^{k}$$

by induction hypothesis.

4. If $n \geq k > 0$ and $\xi = \alpha [_{k} \beta ]_{k}$, then $\alpha \varepsilon H_{n,m-1}^{k+1}$ and $\beta \varepsilon H_{n,m-1}^{k-1}$. Therefore, if $\text{apply}_n^{k+1}(\alpha,\psi)$ and $\text{apply}_n^{k-1}(\beta,\psi)$ are both defined then

$$\text{apply}_n^k(\xi,\psi) = \text{apply}_n^{k+1}(\alpha,\psi)[_{k} \text{apply}_n^{k-1}(\beta,\psi)_{k}]$$

$$\varepsilon H_{n}^{k+1}[_{k} H_{n}^{k-1} ]$$

by induction hypothesis. But since

$$H_{n}^{k+1}[_{k} H_{n}^{k-1} ] \subseteq H_{n}^{k}$$

(20) $$\text{apply}_n^k(\xi,\psi) \varepsilon H_{n}^{k}$$

5. If $k = 0$, then
apply^k(\gamma, \psi) = apply^1(\gamma, \psi)
\in H^1_n

if it is defined. This is a special case of equation 20. But since \(H^0_n = H^1_n\),
apply^k(\gamma, \psi) \in H^0_n
\in H^k_n

Therefore, since
apply^k(\gamma, \psi) \in H^k_n

in each case, the lemma is proved by induction.

- • -

Theorem 18:
If \(n > 0, n+1 \geq k \geq 0, \gamma \in H^k_{n+1}, \psi \in H^k_n\), and
apply^k(\gamma, \psi) is defined, then
apply^k(\gamma, \psi) \in H^k_n

Proof:
By theorem 14 there exists an \(m\) such that
\(\gamma \in H^k_{n,m}\)
Therefore, by lemma 17,
apply^k(\gamma, \psi) \in H^k_n

if it is defined at that point.

- • -

Using the apply function, the frontier may now be defined
Definition 19:

Frontier is a family of partial functions, denoted \( \text{frontier}_n^k \), such that

\[
\text{frontier}_n^{n+1} : H_n^{n+1} \to H_n^n
\]

if \( n \geq 0 \).

\[
\text{frontier}_n^k : H_n^{n+1} \to H_n^k
\]

if \( n \geq k \geq 0 \). The values are defined as follows:

(a) \( \text{frontier}_n^k(\gamma) = \gamma \) if \( \gamma \in H_n^{k+1} \).

\( \text{frontier}_n^k(\gamma) = \text{undefined} \) if \( \gamma \in X_k \).

\( \text{frontier}_n^k(\gamma) = \text{frontier}_n^n(\gamma) \) if \( \gamma = \alpha[k^\beta_k] \).

if \( k = n+1 \) and \( n \geq 0 \).

(b) \( \text{frontier}_n^0(\gamma) = \text{frontier}_n^1(\gamma) \)

if \( n \geq 0 \).

(c) \( \text{frontier}_n^k(\gamma) = \text{frontier}_n^{k+1}(\gamma) \)

if \( \gamma \in X_k \) and \( n \geq k > 0 \).

\( \text{frontier}_n^k(\gamma) = \text{frontier}_n^{k+1}(\alpha[k^\beta_k], \text{frontier}_n^{k-1}(\beta) \text{ frontier}_n^k(\gamma)) \)

if \( \gamma = \alpha[k^\beta_k] \) and \( n > k \).

\( \text{frontier}_n^k(\gamma) = \text{apply}_n(\text{frontier}_n^{n+1}(\alpha), \text{frontier}_n^{n-1}(\beta)) \)

if \( \gamma = \alpha[k^\beta_k], \alpha \notin \Sigma \) and \( n = k \).

\( \text{frontier}_n^k(\gamma) = \alpha[k^\beta_k] \)

if \( \gamma = \alpha[k^\beta_k], \alpha \in \Sigma \) and \( n = k \).
As has been done previously, a theorem will now be proved establishing that the codomain stated in the definition of the frontier is correct.

Lemma 19:
If \( m \geq 0 \), \( y \in H_{n+1,m}^k \) and \( \text{frontier}^k_n(y) \) is defined, then

\[ \text{frontier}^k_n(y) \in H_n^k \]

if \( n \geq k \geq 0 \).

\[ \text{frontier}^k_n(y) \in H_n^n \]

if \( n \geq 0 \) and \( k = n+1 \).

Proof by induction on \( m \):

Basis: if \( m = 0 \) then one of the following cases is true:

1. If \( k = n+1 \), then

\[ y \in X_{n+1}^k \]

and \( \text{frontier}^k_n(y) \) is not defined.

2. If \( n \geq k > 0 \), then

\[ y \in X_k \]

and

\[ \text{frontier}^k_n(y) = y \]

But since \( H_{n,0}^k = X_k \) it must be true that

\[ y \in H_{n,0}^k \]

and
3. If $k = 0$ and $n = 0$, then
\[ \text{frontier}_{n}^{k}(\gamma) = \text{frontier}_{0}^{1}(\gamma) \]
But by case 1 above, $\text{frontier}_{0}^{1}(\gamma)$ is not defined. Therefore, $\text{frontier}_{n}^{k}(\gamma)$ is not defined.

4. If $k = 0$ and $n \geq 0$, then
\[ \text{frontier}_{n}^{k}(\gamma) = \text{frontier}_{n}^{1}(\gamma) \]
By case 2 above,
\[ \text{frontier}_{n}^{k} \in H_{n}^{1} \]
Since $H_{n}^{0} = H_{n}^{1}$, it follows that
\[ \text{frontier}_{n}^{k}(\gamma) \in H_{n}^{0} \]
Since in each case, either the frontier was not defined or was an element of the appropriate codomain, the basis is established.

Induction step: Assume that if $m > m' \geq 0$ and $\gamma \in H_{n+1,m'}^{k}$ then $\text{frontier}_{n}^{k}(\gamma)$ is an element of the appropriate codomain or else is undefined. If $\gamma \in H_{n+1,m'}^{k}$ then one of the following conditions is true:

1. If $k = n+1$ and $\gamma \in H_{n+1,m-1}^{n+2}$, then $\gamma \in \Sigma$ and
\[ \text{frontier}_{n}^{k}(\gamma) = \gamma \]
Since $\Sigma \subset H_{n,m'}^{n}$
2. If \( k = n+1 \) and \( \gamma \in \mathcal{H}^{n+1}_{n+1,m-1} \), then

\[
\text{frontier}^{k}_{n}(\gamma) = \mathcal{H}^{n}_{n}
\]

or is not defined by the induction hypothesis.

3. If \( k = n+1 \) and \( \gamma = \alpha[k \beta \gamma] \), then

\[
\text{frontier}^{k}_{n}(\gamma) = \text{frontier}^{n}_{n}(\beta)
\]

But \( \beta \in \mathcal{H}^{n}_{n+1,m-1} \) and by induction hypothesis

\[
\text{frontier}^{n}_{n}(\beta) = \mathcal{H}^{n}_{n}
\]

if \( \text{frontier}^{n}_{n}(\beta) \) is defined. Therefore,

\[
\text{frontier}^{k}_{n}(\gamma) = \mathcal{H}^{n}_{n}
\]

if \( \text{frontier}^{k}_{n}(\gamma) \) is defined.

4. If \( n \geq k > 0 \) and \( \gamma \in \mathcal{H}^{k+1}_{n+1,m-1} \), then

\[
\text{frontier}^{k}_{n}(\gamma) = \text{frontier}^{k+1}_{n}(\gamma)
\]

But since \( \text{frontier}^{k+1}_{n}(\gamma) \in \mathcal{H}^{k+1}_{n} \) by induction hypothesis, and \( \mathcal{H}^{k+1}_{n} \subset \mathcal{H}^{k}_{n} \)

\[
\text{frontier}^{k}_{n}(\gamma) \in \mathcal{H}^{k}_{n}
\]

if it is defined.

5. If \( n \geq k > 0 \) and \( \gamma \in \mathcal{H}^{k}_{n+1,m-1} \), then

\[
\text{frontier}^{k}_{n}(\gamma) \in \mathcal{H}^{k}_{n}
\]
by induction hypothesis provided that \( \text{frontier}^k_n(\gamma) \) is defined.

6. If \( k = n \) and \( \gamma = a[k \beta k] \), then

\[
\text{frontier}^k_n(\gamma) = \text{apply}^n_n(\text{frontier}^{n+1}_n(\gamma), \text{frontier}^{n-1}_n(\beta))
\]

But by theorem 18, if the frontier is defined

\[
\text{frontier}^k_n(\gamma) \in H^n_n
\]

(Note that the arguments to \( \text{apply} \) are in domain for \( \text{apply} \) by induction hypothesis.)

7. If \( n > k > 0 \) and \( \gamma = a[k \beta k] \), then

\[
\text{frontier}^k_n(\gamma) = \text{frontier}^{k+1}_n(a)[k \text{ frontier}^{k-1}_n(\beta) k]
\]

Notice that since \( n > k \), neither \( k+1 \) nor \( k-1 \) can equal \( n+1 \). Therefore,

\[
\text{frontier}^{k+1}_n(a) \in H^{k+1}_n
\]

and

\[
\text{frontier}^{k-1}_n(\beta) \in H^{k-1}_n
\]

if they are defined, and

\[
\text{frontier}^k_n(\gamma) \in H^{k+1}_n[k H^{k-1}_n k]
\]

\[\in H^k_n\]

if it is defined.

8. If \( k = 0 \), then
frontier^k_n(\gamma) = frontier^1_n(\gamma)

Therefore, by a previous step of this proof,

frontier^1_n(\gamma) \in H^1_n

and finally,

frontier^0_n(\gamma) \in H^0_n

In any event, if frontier^k_n(\gamma) is defined it is in the appropriate codomain, so the lemma is proved by induction.

---

Theorem 20:
If \gamma \in H^k_{n+1} and frontier^k_n(\gamma) is defined, then if

n \geq k \geq 0 then

frontier^k_n(\gamma) \in H^k_n

If n \geq 0 and k = n+1, then

frontier^k_n(\gamma) \in H^n_n

Proof:
This follows directly from lemma 19.

---

For example, the frontier of the tree given on line 5 of table 1 is given by

frontier^2_1(a_2[a_1b_1c_1]_2_1) = frontier^1_1(a_1b_1c_1)_1

= a_1frontier^0_1(b_1c_1)_1
which is the string "abc". The tree given may also be
written a(abc), so the frontier function obtained the
traditional frontier. If the tree given in line 6 of table
1 is chosen, then the frontier will be

$$\text{frontier}_1^2(a[2a[2a[1b_1]_2]_1b[1c[2a[2a[1b_1]_2]_1]_2]_1]_2]_1]_2])$$
$$= \text{frontier}_1^1(a[2a[1b_1]_2]_1b[1c[2a[2a[1b_1]_2]_1]_2]_1]_2]_1]_2])$$
$$= \text{apply}_1^1(\text{frontier}_1^2(a[2a[1b_1]_2]_1],$$
$$\text{frontier}_1^0(b[1c[2a[2a[1b_1]_2]_1]_2]_1]_2]_1]_2])$$
$$= \text{apply}_1^1(\text{frontier}_1^0(a[1b_1]_2],$$
$$\text{frontier}_1^1(b[1c[2a[2a[1b_1]_2]_1]_2]_1]_2]_1]_2])$$
$$= \text{apply}_1^1(a[1\text{frontier}_1^0(b)]_1],$$
$$b[1\text{frontier}_1^0(c[2a[2a[1b_1]_2]_2]_1]_2]_1]_2])$$
$$= \text{apply}_1^1(a[1\text{frontier}_1^1(b)]_1],$$
$$b[1\text{frontier}_1^1(c[2a[2a[1b_1]_2]_2]_1]_2]_1]_2])$$
$$= \text{apply}_1^1(a[1\text{frontier}_1^2(b)]_1],$$
$$b[1\text{frontier}_1^2(c[2a[2a[1b_1]_2]_2]_1]_2]_1]_2])$$
Note that the frontier corresponds to the string "ab". If the tree \( a(a(ab)bc(a(ab))) \) is drawn out as a graph, it can be seen that the frontier should be "ababb". This discrepancy is caused by the fact that with level 2 hypertrees it must be indicated how the frontier is hooked up using the "X"'s. If a portion of the nodes with no subtrees is not referenced by an "X", then those nodes are dropped. This is exactly analogous to dropping an argument in a macrogrammar. Therefore, the hypertree corresponding to \( a(a(ab)bc(a(ab))) \) is actually

\[
a_{2}a_{2}a_{1}b_{1}X_{11}^{1}1_{1}1_{2}1_{1}b_{1}c_{1}a_{2}a_{2}a_{1}b_{1}2_{2}1_{1}1_{2}
\]

The frontier of this hypertree is
\[
\text{frontier}^2_1(a_2a_1b_1x_\lambda^1_1)_2[1_1b_1c_1a_2a_1b_1)_2,l_1]
\]

\[
= \text{frontier}^1_1(a_2a_1b_1x_\lambda^1_1)_2[1_1b_1c_1a_2a_1b_1)_2,l_1]
\]

\[
= \text{apply}^1_1(\text{frontier}^2_1(a_2a_1b_1x_\lambda^1_1)_2),
\]

\[
\text{frontier}^0_1(b_1c_2a_2a_1b_1)_2,l_1)
\]

\[
= \text{apply}^1_1(a_1b_1x_\lambda^1_1), b_1a_1b_1)_1)
\]

\[
= \text{apply}^2_1(a, b_1a_1b_1)_1)
\]

\[
[i\text{apply}^0_1(b_1x_\lambda^1_1), b_1a_1b_1)_1]
\]

\[
= a_1[i\text{apply}^1_1(b_1x_\lambda^1_1), b_1a_1b_1)_1]
\]

\[
= a_1b_1[i\text{apply}^1_1(x_\lambda^1_1, b_1a_1b_1)_1]
\]

\[
= a_1b_1[i\text{search}^0_1(\lambda, b_1a_1b_1)_1]
\]

\[
= a_1b_1[i\text{search}^1_1(a_1b_1)_1]
\]

which is the "abbab" that was expected.

At this point, two definitions can be made which will formalize the premise of this paper.

Definition 20:

A string is a frontierable hypertree of level 1.

*
Definition 21:
A tree is a frontierable hypertree of level 2.

Previously in this paper it has been stated that hypertrees over \( \mathbb{Z} \), \( h_n \), is a restricted case of frontierable hypertrees over \( \mathbb{I} \), \( H_n \). This can now be proven. First, a lemma must be proved to establish that hyperforests are a restricted case of frontierable hyperforests. This is done by induction on the depth of the hyperforest.

Lemma 21:
For all \( n \geq 0 \), \( m \geq 0 \) and \( n+1 \geq k \geq 0 \),
\[
H_n^k \subseteq H_{n,m}^k
\]
Proof by induction on \( m \):
Basis: If \( m = 0 \), then one of the following is the case
1. If \( k = n+1 \), then
\[
h_{n,0}^{n+1} = \mathbb{I}
\]
\[
= H_{n,0}^{n+1}
\]
therefore,
\[
h_{n,0}^{n+1} \subseteq H_{n,0}^{n+1}
\]
2. If \( n \geq k > 0 \), then
\[
h_{n,0}^k = \emptyset
\]
\[
\subset H_{n,0}^k
\]
3. If \( k = 0 \), then
\[ h_{n,0}^0 = h_{n,0}^1 = h_{n,0}^1 = h_{n,0}^0 \]

Since in each of these cases,
\[ h_{n,0}^k = h_{n,0}^k \]
the basis is proved.

Induction step: Assume that if \( m > m' \geq 0 \) then
\[ h_{n,m}^k = h_{n,m}^k \]

If \( m > 0 \), then there are three cases to consider

1. if \( k = n+1 \) then
\[ h_{n,m}^{n+1} = \Sigma \]
\[ = h_{n,m}^{n+1} \]

Therefore,
\[ (21) \quad h_{n,m}^{n+1} < h_{n,m}^{n+1} \]

2. If \( n \geq k > 0 \), then
\[ h_{n,m}^k = h_{n,m-1}^{k+1} U h_{n,m-1}^k U h_{n,m-1}^{k+1} \]

but since by induction hypothesis
\[ h_{n,m-1}^{k+1} < h_{n,m-1} \]
\[ h_{n,m-1}^k < h_{n,m-1}^k \]
\[ h_{n,m-1}^{k+1} < h_{n,m-1}^k \]
\[ h_{n,m-1}^{k+1} < h_{n,m-1}^k \]
Therefore, the previous equation becomes

\[(22) \quad h_{n,m}^k \subseteq H_{n,m-1}^{k+1} \cup H_{n,m-1}^k \cup H_{n,m-1}^{k+1} \cdot H_{n,m-1}^{k-1} \cdot H_{n,m} \]

so that the theorem is true in this case also.

3. Lastly, if \( k = 0 \) then

\[h_{n,m}^0 = h_{n,m}^1\]

But by equation 21 or 22 depending on the value of \( n \)

\[h_{n,m}^1 \subseteq H_{n,m} \cap H_{n,m}^0\]

Therefore, in this case also

\[h_{n,m}^k \subseteq H_{n,m}^k\]

Since in each of the above cases, \( h_{n,m}^k \subseteq H_{n,m}^k \), the lemma is proved by induction.

---

**Theorem 22:**

For all \( n \geq 0 \),

\[h_n \subseteq H_n\]

**Proof:**
\[ h_n = h_n^m \]
\[ = \bigcup_{m \geq 0} h_{n,m}^m \]
\[ \subseteq \bigcup_{m \geq 0} H_{n,m}^n \]

by lemmas 2, 7 and 21. Therefore,

\[ h_n \subseteq H_n^n \]
\[ \subseteq H_n \]

Therefore,

\[ h_n \subseteq H_n \]

for all \( n \geq 0 \) and the theorem is proved.

---

The original justification for studying hypertrees was the characterization of the algebraic language hierarchy. This requires that each level of the hypertree hierarchy have at least a regular grammar defined over it. The frontier function will be used then to extract the elements of the algebraic language hierarchy from the regular hypertree languages. Since a regular language requires that nonterminals appear only on the "frontier" of each of the right hand sides of the productions in the hypertree grammar, a new definition must be given for the hierarchy which will allow nonterminals to appear on the "frontier". With the definitions given above, requiring the nonterminals to be on the frontier can only be done informally since the
phrase "on the frontier" has not been defined. This new
definition of hypertrees, hypertrees with arguments,
arguments are provided which are only allowed on the
frontier of the hypertrees.

Before a formal definition for hypertrees with
arguments can be given, a formal definition for ranked sets
must be given since these will be used in the definition of
hypertrees with arguments.

Definition 22:
A ranked set $V$ is an ordered pair $(V',r)$ where $V'$
is a set and $r$ is a function such that

$$r: V' \to N$$

where $N$ is the set of positive integers.

Definition 23:
A rank of the ranked set $V$, denoted $V_k$, is defined
by

$$V_k = r^{-1}(k) = \{a \mid r(a) = k\}$$

Note that since a function is used in defining ranked
sets the ranks are disjoint. The definition of hypertrees
and hyperforests with arguments can now be given. As should be expected, these definitions correspond rather closely to the definition of hypertrees, hyperforests, frontierable hypertrees and frontierable hyperforests already given.

**Definition 24:**

The set of hypertrees over $\mathcal{I}$ of degree $n$ with arguments from the ranked set $V$, denoted $H_n(\mathcal{I},V)$, is the smallest set such that

(a) $H_0(\mathcal{I},V) = \emptyset$

(b) $H_n(\mathcal{I},V) = \Sigma U V_n U X_n U \sum_{n-1} H_n-1(\mathcal{I},V), V)$

if $n > 0$.

**Definition 25:**

The set of hyperforests over $\mathcal{I}$ of level $n$ and degree $k$ with arguments from the ranked set $V$, denoted $H_n^k(\mathcal{I},V)$, is the smallest set such that

(a) $H_n^{n+1}(\mathcal{I},V) = \emptyset$

if $n \geq 0$.

(b) $H_n^0(\mathcal{I},V) = H_n^1(\mathcal{I},V)$

if $n \geq 0$.

(c) $H_n^k(\mathcal{I},V) =$

$H_n^{k+1}(\mathcal{I},V) U V_k U X_k U H_n^{k+1}(\mathcal{I},V)[k H_n^{k-1}(\mathcal{I},V)]$

if $n \geq k > 0$. 
Definition 26:

The set of hyperforests over $\Sigma$ of level $n$, degree $k$ and depth $m$ with arguments from the ranked set $V$, denoted $H^k_{n,m}(\Sigma,V)$, is the smallest set such that

(a) $H^{n+1}_{n,m}(\Sigma,V) = \Sigma$
if $n \geq 0$ and $m \geq 0$.

(b) $H^k_{n,0}(\Sigma,V) = V_k \cup X_k$
if $n \geq k > 0$.

(c) $H^k_{n,m+1}(\Sigma,V) = H^k_{n,m}(\Sigma,V) \cup H^k_{n,m}(\Sigma,V) \cup H^{k+1}_{n,m}(\Sigma,V) \cup H^{k-1}_{n,m}(\Sigma,V) \cup H^k_{n,m}(\Sigma,V)$
if $n \geq k > 0$ and $m \geq 0$.

(d) $H^0_{n,m}(\Sigma,V) = H^1_{n,m}(\Sigma,V)$
if $n \geq 0$ and $m \geq 0$.

Notice that throughout these definitions $V_k$ always appears beside the $X_k$. From this fact, it is apparent that there is a relationship between the $V_k$ and the $X_k$. In each case, the argument or the "$X"$ may be replaced with an element of $H^k_n(\Sigma,V)$ and a well-formed hypertree will result.

In the case of the "$X"$'s, this fact was used to substitute the appropriate subhypertree into the frontier. In the case of the argument, this will be used to develop a grammar.

Just as in the case of a hypertree over $\Sigma$, the $V$ will be elided throughout the rest of this paper if no confusion
would result. This will result in \( H_n^k(\Sigma) \) and \( H_n^j(\Sigma, V) \) both being represented by \( H_n^k \), but in most instances this will not cause a confusion because it will be obvious which notation is being referenced.

Theorems 1 to 7 can also be extended to hypertrees with arguments.

Theorem 23:

For all \( n \geq k \geq 0 \),

\[
H_n^k = H_k(H_{k+1}( \ldots (H_n(\Sigma, V), V) \ldots ), V)
\]

Proof:

This proof is the same as theorem 1.

---

Corollary 24:

For all \( n \geq 0 \),

\[
H_n(\Sigma, V) = H_n^0(\Sigma, V)
\]

Proof:

This proof is the same as corollary 2.

---

Lemma 25:

For all \( n \geq 0 \) and \( n+1 \geq j \geq k \geq 0 \),

\[
H_n^j(\Sigma, V) \subseteq H_n^k(\Sigma, V)
\]

Proof:

This proof is the same as the proof for lemma 3.

---
Lemma 26:

For all \( m \geq 0, n \geq 0 \) and \( n+1 \geq k \geq 0 \),

\[
H_{n,m}^k(\Sigma,V) \subseteq H_n^k(\Sigma,V)
\]

Proof:

This lemma is proved along the same lines as lemma 5.

Lemma 27:

If \( n \geq 0, n+1 \geq k \geq 0 \) and \( m \geq 0 \), then

\[
H_{n,m}^k(\Sigma,V) = \bigcup_{m \geq m^* \geq 0} H_{n,m}^k(\Sigma,V)
\]

Proof:

This proof is the same as the proof for lemma 6.

Lemma 28:

For all \( n \geq k \geq 0 \) and \( m \geq 0 \),

\[
H_{n,m+1}^k(\Sigma,V) = H_{n,m}^{k+1} U H_{n,m}^k U H_{n,m+1}^{k+1} \bigcup H_{n,m}^{k+1}
\]

Proof:

This proof follows the lines of lemma 4.

Theorem 29:

For all \( n \geq 0 \) and \( n+1 \geq k \geq 0 \),

\[
H_n^k = \bigcup_{m \geq 0} H_n^k
\]

Proof:

This proof is the same as theorem 7.
Without formally stating them, it will be assumed that the search, apply and frontier functions may be extended to hypertrees with arguments. For the search function, this extension will only require the appropriate change in the domain and codomain. For the apply function, besides the domain change, the line
\[ \text{apply}^k_n(y, \psi) = y \]
needs to be added to handle the case where \( y \in V_k \). Besides the domain changes, the frontier function needs the line
\[ \text{frontier}^k_n(y, \psi) = y \]
added to handle the case where \( y \in V_k \) and \( n > k \geq 0 \), and the line
\[ \text{frontier}^k_n(y, \psi) = \text{undefined} \]
needs to be added to handle the \( n = k > 0 \) case. Similarly, a string with arguments and a tree with arguments will be taken as the obvious extension of the definition of trees and strings.

Theorems 21 and 22 will now be extended.
Lemma 30:
For all $n \geq 0$, $m \geq 0$ and $n+1 \geq k \geq 0$,

$$h_{n,m}^k(\xi) \subset H_{n,m}^k(\xi) \subset H_{n,m}^k(\xi,V)$$

Proof:
This proof is the same as the proof for lemma 21.

- • -

Theorem 31:
For all $n \geq 0$,

$$h_n(\xi) \subset H_n(\xi) \subset H_n(\xi,V)$$

Proof:
This is proved in the same fashion as theorem 22.

- • -

This concludes the basic definitions that are needed for this development of hypertrees. Subsequent chapters will be involved in showing additional relationships between trees, strings and hypertrees. Also, grammars, automata and regular expressions will be developed to describe regular languages on hypertrees, and these languages will be characterized. This characterization then will be used to characterize the algebraic language hierarchy.
In order to be useful, there must be some reasonable way of representing hypertrees in a computer. Perhaps the most reasonable representation is obtained by extending the concept of binary encoded trees to hypertrees. It will be shown that, in fact, linked list representation of strings is related to the binary representation of trees. In general, it will be shown that a level n hypertree may be conveniently represented as an n-ary tree. The two special cases presented are that of trees, which may be represented as binary trees, and strings, which may be represented as unary trees (linked lists).

In order to establish this relationship, it is first necessary to give a formal definition of what is meant by n-ary trees.

Definition 27:

The set of n-ary trees over Σ, denoted T'_n(Σ), is the smallest set such that

\[ T'_n(Σ) = \{ \lambda \} \cup Σ[(T'_n(Σ))^n] \]

for all n ≥ 0.

As in the previous sections, the linear representation of n-ary trees will be used instead of the graph theoretic
representation. Also, the set $\Xi$ will be elided for the remainder of this paper unless it is not clear what set the trees are over.

In order to define a simple mapping from hypertrees to n-ary trees, a subset of the n-ary trees will be defined which exactly matches the image of the hypertrees of level $n$. Then it will be proved that, in fact, this image is a subset of the n-ary trees over $\Xi$. Also, it will be shown that a corresponding image can be defined for $H_n^k(\Xi)$ and $H_{n,m}^k(\Xi)$.

Definition 28:
The set of n-ary encoded hyperforests of degree $k$, denoted $T_n^k$, is defined as

$$T_n^k = t_n^k(\Xi)\lambda^{\max(k-1,0)}$$

if $n \geq 0$ and $n+1 \geq k \geq 0$, where $t_n^k(\Xi)$ is the smallest set such that

(a) $t_n^{n+1}(\Xi) = \Xi$ if $n \geq 0$.

(b) $t_n^0(\Xi) = t_n^1(\Xi)$ if $n \geq 0$.

(c) $t_n^k(\Xi) = t_n^{k+1}(\Xi)\lambda \cup t_n^{k+1}(\Xi)T_n^{k-1}$ if $n \geq k > 0$. 

•
Definition 29:

The set of n-ary encoded hypertrees over $\mathcal{E}$, denoted $T_n$, is defined as

$$T_n = T_n$$

if $n \geq 0$.

Definition 30:

The set of n-ary encoded hypertrees over $\mathcal{E}$ of degree $k$ and depth $m$, denoted $T_{n,m}^k$, is defined as

$$T_{n,m}^k = t_{n,m}^k(\mathcal{E}) \cup \max(k-1,0)$$

if $n \geq 0$, $n+1 \geq k \geq 0$ and $m \geq 0$ where $t_{n,m}^k(\mathcal{E})$ is the smallest set such that

(a) $t_{n,m}^{n+1}(\mathcal{E}) = \mathcal{E}$

if $n \geq 0$ and $m \geq 0$.

(b) $t_{n,0}^k(\mathcal{E}) = \emptyset$

if $n \geq k > 0$.

(c) $t_{n,m+1}^k(\mathcal{E}) = t_{n,m}^k(\mathcal{E}) \cup t_{n,m}^{k+1}(\mathcal{E}) t_{n,m}^{k-1}$

if $n \geq k > 0$ and $m \geq 0$.

(d) $t_{0,m}^0(\mathcal{E}) = t_{n,m}^1(\mathcal{E})$

if $n \geq 0$ and $m \geq 0$.

These definitions are so similar to the definitions of hypertrees given in the previously in chapter 2 that a set
80

of theorems may be given which correspond to lemma 3 to theorem 7 in that chapter.

Lemma 32:

For all \( n \geq j \geq k \geq 0 \)

\[ T_n^j \subseteq T_n^k \]

Proof (patterned after the proof of lemma 3 in chapter 2):

Define \( t_n^k(\xi) \) as it is defined in definition 28. Prove the lemma by induction on \( k \).

Basis: If \( k = j \), then

\[ T_n^j = T_n^k \]

Therefore,

\[ T_n^j \subseteq T_n^k \]

Induction step: Assume that if \( j \geq k' > k \geq 0 \) then

\[ T_n^j \subseteq T_n^{k'} \]

Since \( k \geq 0 \) it must be true that either \( k > 0 \) or \( k = 0 \).

1. If \( k > 0 \), then

\[ T_n^k = t_n^k(\xi)\lambda^{\max(k-1,0)} \]
\[ = t_n^{k+1}(\xi)\lambda^{\max(k-1,0)} \]
\[ \cup t_n^{k+1}(\xi)T_n^{k-1}\lambda^{\max(k-1,0)} \]

Now, since \( k > 0 \),

\[ \max(k-1,0) = k-1 \]
equation 1 becomes

\[ T_n^k = t_n^{k+1}(\xi)\lambda^{\max((k+1)-1,0)} \cup t_n^{k+1}(\xi)T_n^{k-1}\lambda^{\max(k-1,0)} \]

But by induction hypothesis

\[ T_n^j \subseteq T_n^{k+1} \]

Now,

\[ T_n^{k+1} = t_n^{k+1}(\xi)\lambda^{\max((k+1)-1,0)} \]

\[ \subseteq T_n^k \]

therefore,

\[ T_n^j \subseteq T_n^k \]

2. If \( n \geq j > k = 0 \), then

\[ T_n^0 = t_n^0(\xi)\lambda^{\max(0-1,0)} \]

\[ = t_n^1(\xi)\lambda^{\max(0-1,0)} \]

\[ = t_n^1 \]

By induction hypothesis

\[ T_n^j \subseteq T_n^1 \]

therefore,

\[ T_n^j \subseteq T_n^0 \]

and the lemma is proved by induction.
Lemma 33:

For all \( m \geq 0, \ n \geq 0 \) and \( n+1 \geq k \geq 0 \) it is true that

\[
T^k_{n,m} \subseteq T^k_n
\]

Proof (patterned after the proof for lemma 5 in chapter 2):

Define \( t_n^k(\Sigma) \) as it is defined in definition 28 and \( t_{n,m}^k(\Sigma) \) as it is defined in definition 30. Prove the lemma by induction on \( m \).

Basis: if \( m = 0 \) then

\[
T^k_{n,0} = t^k_{n,0}(\Sigma)^{\max(k-1,0)}
\]

and one of the following is true:

1. If \( k = n+1 \), then

\[
t^k_{n,0}(\Sigma) = \Sigma
\]

therefore,

\[
t^k_{n,0}(\Sigma) = t^k_n(\Sigma)
\]

and

\[
T^k_{n,0} = t^k_n(\Sigma)^{\max(k-1,0)} = T^k_n
\]

2. If \( n \geq k > 0 \), then

\[
t^k_{n,0}(\Sigma) = \phi
\]

therefore,
\[ T_{n,0}^k = \emptyset \]

and

\[ T_{n,0}^k \subset T_n^k \]

since the null set is a subset of every set.

3. If \( k = 0 \), then

\[ T_{n,0}^k = T_{n,1}^k \subset T_n^1 \]

by either step 1 or step 2 above. Since \( T_n^0 = T_n^1 \),

\[ T_{n,0}^k \subset T_n^0 \]

and the basis is proven.

Induction step: Assume that if \( m > m' \geq 0 \) then

\[ T_{n,m'}^k \subset T_n^k \]

If \( m > 0 \), then

\[ T_{n,m}^k = T_{n,m}^k(\Sigma)^{\max(k-1,0)} \]

and one of the following conditions must be true:

1. If \( k = n+1 \), then

\[ T_{n,m}^{n+1} = T_{n,m}^{n+1}(\Sigma)^{\max((n+1)-1,0)} \]

\[ = T_n^n(\Sigma)^{\max((n+1)-1,0)} \]

\[ = T_n^{n+1} \]
2. If $n \geq k > 0$, then

$$T_{n,m}^k = t_{n,m-1}^{k+1}(\Sigma)_{\lambda_{\max(k-1,0)}}$$

$$U t_{n,m-1}^{k+1}(\Sigma)T_{n,m-1}^{k-1}_{\lambda_{\max(k-1,0)}}$$

Note that

$$T_{n,m-1}^{k+1} \subset T_{n}^{k+1}$$

by induction hypothesis, so that

$$t_{n,m-1}^{k+1}(\Sigma)_{\lambda_{\max(k-1,0)}} \subset t_{n}^{k+1}(\Sigma)_{\lambda_{\max(k-1,0)}}$$

and therefore,

$$t_{n,m-1}^{k+1}(\Sigma) \subset t_{n}^{k+1}(\Sigma)$$

Therefore, by this equation and induction hypothesis

$$T_{n,m}^k \subset t_{n}^{k+1}(\Sigma)_{\lambda_{\max(k-1,0)}}$$

$$U t_{n}^{k+1}(\Sigma)T_{n,m-1}^{k-1}_{\lambda_{\max(k-1,0)}}$$

$$\subset (t_{n}^{k+1}(\Sigma) U t_{n}^{k+1}(\Sigma)T_{n,m-1}^{k-1}_{\lambda_{\max(k-1,0)}}$$

$$\subset t_{n}^{k}(\Sigma)_{\lambda_{\max(k-1,0)}}$$

$$\subset t_{n}^{k}$$

and the lemma is true in this case also.

3. If $k = 0$, then

$$T_{n,m}^0 = t_{n,m}^{0}(\Sigma)_{\lambda_{\max(0-1,0)}}$$

$$= t_{n,m}^{1}(\Sigma)_{\lambda_{\max(0-1,0)}}$$
Lemma 34:

If \( n \geq 0 \), \( n+1 \geq k \geq 0 \) and \( m \geq 0 \), then

\[
T^k_{n,m} = U_{m \geq m' \geq 0} T^k_{n,m'}
\]

Proof (patterned after the proof of lemma 6):

Define \( t^k_{n,m}(\xi) \) as it is defined in definition 30. Prove the theorem by induction on \( m \):

Define

\[
\alpha^k_{n,m}(\xi) = U_{m \geq m' \geq 0} T^k_{n,m}
\]
Basis: if $m = 0$ then

$$a_{n,0}^k(\xi) = U_{m'=0} t_{n,m}^k = T_{n,0}^k$$

Induction step: Assume that if $m > 0$ then

$$a_{n,m}^k(\xi) = T_{n,m}^k$$

then by the definition of union,

$$a_{n,m+1}^k(\xi) = T_{n,m+1}^k \cup a_{n,m}^k(\xi)$$

by induction hypothesis. Therefore,

$$a_{n,m+1}^k(\xi) = T_{n,m+1}^k \cup a_{n,m}^k(\xi)_{\lambda \max(k-1,0)}$$

since $t_{n,m}^k(\xi) \subset t_{n,m+1}^k(\xi)$. Therefore,

$$a_{n,m}^k(\xi) = T_{n,m}^k$$

if $m \geq 0$ by induction. Therefore, the lemma is proved.
Theorem 35:

For all $n \geq 0$ and $n+1 \geq k \geq 0$

$$T_n^k = U_{m \geq 0} T_{n,m}^k$$

Proof:

To prove that

$$T_n^k = U_{m \geq 0} T_{n,m}^k$$

a similar result will be shown for $t_n^k(\Sigma)$ and $t_n^k(\Sigma)$. If $t_n^k(\Sigma)$ is defined as in definition 28 and $t_n^k(\Sigma)$ is defined as in definition 30, then it is sufficient to show that

$$t_n^k(\Sigma) = U_{m \geq 0} t_{n,m}^k(\Sigma)$$

since

(2) $$T_n^k = t_n^k(\Sigma) \lambda \max(k-1,0)$$

and

(3) $$U_{m \geq 0} T_{n,m}^k = U_{m \geq 0} t_{n,m}^k(\Sigma) \lambda \max(k-1,0)$$

Therefore, it will be shown that $U_{m \geq 0} t_n^k(\Sigma)$ has all the properties that define $t_n^k(\Sigma)$. Define

$$\alpha_{n,m}^k(\Sigma) = U_{m \geq m' \geq 0} t_{n,m'}^k(\Sigma)$$

By lemma 34

$$\alpha_{n,m}^k(\Sigma) = t_{n,m}^k(\Sigma)$$

due to equations 2 and 3 given above. Therefore, $\alpha_{n,m}^{n+1}(\Sigma)$ is the smallest set such that

1. If $n \geq 0$ and $m \geq 0$, then
\[ a_{n,m}^{n+1}(\Sigma) = \Sigma \]

2. If \( n \geq 0 \) and \( m \geq 0 \), then
\[ a_{n,m}^0(\Sigma) = a_{n,m}^1(\Sigma) \]

3. If \( n \geq k > 0 \) and \( m \geq 0 \), then
\[ a_{n,m}^k(\Sigma) = a_{n,m}^{k+1}(\Sigma) \lambda U a_{n,m}^{k+1}(\Sigma) T^{k-1} a_{n,m}^k(\Sigma) \lambda \max ((k-1)-1,0) \]

By the definition of infinite union
\[ \lim_{m \to \infty} a_{n,m}(\Sigma) = U_{m \geq 0} a_{n,m}(\Sigma) \]

Therefore, if \( a_{n}(\Sigma) \) is defined as
\[ a_{n}(\Sigma) = U_{m \geq 0} a_{n,m}(\Sigma) \]

then it must be true that
1. If \( n \geq 0 \), then
\[ a_{n}^{n+1}(\Sigma) = \Sigma \]

2. If \( n \geq 0 \), then
\[ a_{n}^0(\Sigma) = a_{n}^1(\Sigma) \]

3. If \( n \geq k > 0 \), then
(6) \[ a_n^k(\Sigma) = a_n^{k+1}(\Sigma) \cup a_n^{k+1}(\Sigma) a_n^{k-1}(\Sigma)^{\max((k-1)-1,0)} \]
\[ = a_n^{k+1}(\Sigma) \cup a_n^{k+1}(\Sigma) T_n^{k-1} \]

Now, since \( t_k^n(\Sigma) \) is the smallest set with properties 4, 5 and 6 it must be true that

\[ t_k^n(\Sigma) = a_n^k(\Sigma) \]
\[ = \bigcup_{m \geq 0} t_{n,m}^k(\Sigma) \]

and the theorem is proved.

---

A theorem can now be proved which will show that n-ary encoded hypertrees are, in fact, n-ary trees. After this theorem is proved, it will be shown that these definitions allow a quite natural encoding of level n hypertrees, and hyperforests as n-ary trees.

Lemma 36:

If \( n \geq 0, n+1 \geq k \geq 0 \) and \( m \geq 0 \), then

\[ T_{n,m}^k \subset T'_n \]

Proof by induction on \( m \):

Basis: if \( m = 0 \) then one of the following three cases is true:

1. If \( k = n+1 \), then

\[ t_{n,m}^k(\Sigma) = \Sigma \]
and
\[ T_{n,m}^k = \lambda^{\max((n+1)-1,0)} \]
\[ = \lambda^n \]

Since \( \lambda \in T_n' \) it must be also true that
\[ \Sigma[\lambda^n] \subseteq T_n' \]

by definition 27. Therefore,
\[ T_{n,m}^k \subseteq T_n' \]

2. If \( n \geq k > 0 \), then
\[ T_{n,0}^k = T_{n,0}^k(\lambda^{\max(0-1,0)}) \]
\[ = \phi \]

Therefore,
\[ T_{n,0}^k \subseteq T_n' \]

3. If \( k = 0 \), then
\[ T_{n,m}^k = T_{n,m}^1 \]

But by either part 1 or part 2 of this proof,
\[ T_{n,m}^1 \subseteq T_n' \]

Therefore,
\[ T_{n,m}^0 \subseteq T_n' \]
Induction step: Assume that if \( m > m' \geq 0 \) then

\[
T_{n,m}' = \sum T_{n,m}(\lambda)\max(k-1,0)
\]

If \( m > 0 \), then

\[
T_{n,m}^k = t_{n,m}^k(\xi)\max(k-1,0)
\]

Therefore, one of the following cases must be true:

1. If \( k = n+1 \), then

\[
T_{n,m}^{n+1} = \sum T_{n,m}(\lambda)\max((n+1)-1,0)
\]

\[
= \sum T_{n,m}(\lambda)\max(n,0)
\]

\[
= \sum T_{n,m}(\lambda^n)
\]

Notice that \( \lambda \in T'_n \), therefore,

\[
\sum T_{n,m}(\lambda)\in T'_n
\]

and

\[
T_{n,m}^{n+1} \subset T'_n
\]

2. If \( n \geq k > 0 \), then

\[
T_{n,m}^k = t_{n,m}^k(\xi)\lambda^{\max(k-1,0)}
\]

\[
= t_{n,m}^k(\xi)\lambda^{k-1}
\]

by definition 30. Also by the definition of \( t_{n,m}^k(\xi) \)
\[ T_{n,m}^k = t_{n,m-1}^{k+1}(\Sigma)\lambda^k ] U t_{n,m-1}^{k+1}(\Sigma)T_{n,m-1}^{k-1} \lambda^{k-1} \]

and by the definition of \( T_{n,m-1}^{k+1} \)

(7) \[ T_{n,m}^k = T_{n,m-1}^{k+1} U t_{n,m-1}^{k+1}(\Sigma)T_{n,m-1}^{k-1} \lambda^{k-1} \]

Notice that since

(8) \[ \lambda \notin T_{n,m-1}^{k+1} \]

by induction hypothesis, and since

\[ \lambda \notin T_{n,m-1}^{k+1} \]

it must be that

\[ T_{n,m-1}^{k+1} \subseteq \Sigma[(T_n')^n] \]

Also note that

\[ t_{n,m-1}^{k+1}(\Sigma)\lambda^k ] \subseteq \Sigma[(T_n')^n] \]

since \((k+1)-1 \geq 0\). Therefore,

(9) \[ t_{n,m-1}^{k+1}(\Sigma)\lambda^k ] \subseteq \Sigma[(T_n')^n] \]
By noting that \( \lambda \in T'_n \) and matching subtrees it can be seen that
\[
t^{k}_{n,m-1}(\lambda) \subseteq \Sigma[(T'_n)^{n-k}]
\]
Therefore,
\[
t^{k}_{n,m-1}(\lambda)T^{k-1}_{n,m-1} \subseteq \Sigma[(T'_n)^{n-k+1}]
\]
by induction hypothesis and
\[
t^{k}_{n,m-1}(\lambda)T^{k-1}_{n,m-1} \lambda^{k-1} \subseteq \Sigma[(T'_n)^{n-k+1} \lambda^{k-1}]
\]
Since \( \lambda \in T'_n \) it must be that
\[
t^{k}_{n,m-1}(\lambda)T^{k-1}_{n,m-1} \lambda^{k-1} \subseteq \Sigma[(T'_n)^{n-k+1} (T'_n)^{k-1}]
\]
\[
\subseteq \Sigma[(T'_n)^{n}]
\]
and that
\[
(10) \quad t^{k+1}_{n,m-1}(\lambda)T^{k-1}_{n,m-1} \lambda^{k-1} \subseteq T'_n
\]
by equations 7, 8 and 10 therefore,
\[
t^{k}_{n,m} \subseteq T'_n
\]
3. Finally, if \( n \geq k = 0 \) then
\[
T^0_{n,m} = t^0_{n,m}(\lambda)^{\max(0-1,0)}
\]
Since
\[ T_{n,m}^1 \subseteq T'_n \]

as a special case of step 1 or step 2 of this proof
\[ T_{n,m}^0 \subseteq T'_n \]

Therefore, the lemma is true in this case also.

Since the lemma is true in each of the above cases, the lemma is proved by induction.

---

Lemma 37:
If \( n \geq 0 \) and \( n+1 \geq k \geq 0 \), then
\[ T_n^k \subseteq T'_n \]

Proof:
This follows immediately from lemma 36 and theorem 35.

---

Theorem 38:
If \( n \geq 0 \), then
\[ T_n \subseteq T'_n \]

Proof:
By lemma 37

\[ T^n \subset T'_n \]

Therefore,

\[ T^n \subset T'_n \]

by definition 29 and the theorem is proved.

\[ \cdot \cdot \cdot \]

Note that this theorem does not imply that

\[ T_n = T'_n \]

In fact, this is not true, as can be demonstrated by noting that

\[ a[b[\lambda^2]c[\lambda^2]] \]

is a legal \( n \)-ary tree, but not a legal \( n \)-ary encoded hypertree. This is true since this example is a 2-ary tree, corresponding to 2-ary encoded hypertrees, but 2-ary encoded hypertrees must all have the postfix "\( \lambda \)". This example does not.

The encoding that may be used will now be demonstrated.

It will be shown recursively that \( H_{n,m}^n(\lambda) \) can be easily encoded in \( T_{n,m}^k \), and once this is shown, it will be assumed that this encoding can be extended to \( H_{n}^k(\lambda) \) and \( H_{n}(\lambda) \).
Definition 31:

Encode is a family of functions, denoted $\text{encode}_{n,m}^k$, such that

$$\text{encode}_{n,m}^k : h_{n,m}^k(\xi) \to t_{n,m}^k$$

where

(a) $\text{encode}_{n,m}^{n+1}(\xi) = \xi[\lambda^n]$ if $n \geq 0$ and $m \geq 0$.

(b) $\text{encode}_{n,0}^k$ is a null function if $n \geq k > 0$.

(c) $\text{encode}_{n,m}^k = \text{encode}_{n,m}^{k+1}(\xi)$

if $\xi \in h_{n,m}^{k+1}$.

$$\text{encode}_{n,m}^k(\xi) = \text{encode}_{n,m}^k(\xi)$$

if $\xi \in h_{n,m}^k$.

$$\text{encode}_{n,m+1}^k(\xi) = \text{encode}_{n,m}^k(\xi)$$

if $\xi \in h_{n,m}^k$.

$= \nu$

if $\xi = a[\lambda^k \beta]$. 

where $n \geq k > 0$, $m \geq 0$ and if $\text{encode}_{n,m}^{k+1}(\alpha) = \nu[\lambda^k]$ then $\nu = \nu[\text{encode}_{n,m}^{k-1}(\beta)[\lambda^k]]$.

(d) $\text{encode}_{n,m}^0(\xi) = \text{encode}_{n,m}^1(\xi)$

if $n \geq 0$ and $m \geq 0$. 

•
Since this is the definition of a function, a proof of the codomain is in order.

Theorem 39:
If \( n \geq 0, \ n+1 \geq k \geq 0, \ m \geq 0 \) and \( \bar{y} \in h_{n,m}^k \), then
\[
\text{encode}_{n,m}^k(\bar{y}) \in T_{n,m}^k
\]

Proof by induction on \( m \):
Define \( t_{n,m}^k(\bar{z}) \) as it is defined in definition 30.
Basis: If \( m = 0 \), then one of the following three cases is true:
1. If \( k = n+1 \), then
\[
\text{encode}_{n,m}^{n+1}(\bar{y}) = \bar{z}[\lambda^n]
\]
Now, \( \bar{y} \in \Lambda \) therefore,
\[
\text{encode}_{n,m}^{n+1}(\bar{y}) \in \Sigma[\lambda^n]
\]
\[
\varepsilon \ T_{n,0}^{k+1,\lambda^n}
\]
\[
\varepsilon \ T_{n,m}^{n+1}
\]
2. If \( n \geq k > 0 \), then \( \bar{y} \) does not exist since
\[
\bar{y} \in h_{n,0}^k
\]
\( \varepsilon \emptyset \)
3. If \( k = 0 \), then
\[
\text{encode}_{n,m}^k(\bar{y}) = \text{encode}_{n,m}^1(\bar{y})
\]
Now, if $n > 0$ then

$$\gamma \in h_{n,0}^1$$

$$\varepsilon \emptyset$$

so that $\gamma$ does not exist and this case cannot happen.

If $n = 0$, then

$$\gamma \in h_{0,0}^1$$

$$\varepsilon \Sigma$$

and by case 1 above

$$\text{encode}_{0,0}^1(\gamma) \in T_{0,0}^1$$

But $T_{0,0}^1 = T_{0,0}^0$ so that

$$\text{encode}_{0,0}^0(\gamma) \in T_{0,0}^0$$

Induction step: Assume that if $m > m'$ and $\gamma \in h_{n,m}^k$ then

$$\text{encode}_{n,m'}^k(\gamma) \in T_{n,m'}^k$$

If $\gamma \in h_{n,m}^k$, then one of the following cases is true:

1. If $k = n+1$, then

$$\gamma \in h_{n,m}^{n+1}$$

$$\varepsilon \Sigma$$

and

$$\text{encode}_{n,m}^k(\gamma) = \Sigma[\lambda^n]$$

Note that $t_{n,m}^k(\Sigma) = \Sigma[\lambda^n]$ so that this becomes
Since max((n+1)-1,0) = n, it must be true that

\[ \text{encode}^k_{n,m}(\gamma) \leq t^{k+1}_{n,m}(\gamma) \text{, } \gamma \leq t^{k+1}_{n,m}(\gamma) \text{, } \gamma \leq t^{k+1}_{n,m}(\gamma) \text{, } \gamma \leq t^{k+1}_{n,m}(\gamma) \]

2. If \( n \geq k > 0 \) and \( \gamma \in h_{n,m}^{k+1} \), then

\[ \text{encode}^k_{n,m}(\gamma) = \text{encode}^{k+1}_{n,m}(\gamma) \]

But by induction hypothesis

\[ \text{encode}^{k+1}_{n,m}(\gamma) \leq t^{k+1}_{n,m}(\gamma) \]

Therefore,

\[ \text{encode}^k_{n,m}(\gamma) \leq t^{k+1}_{n,m}(\gamma) \]

since \( t^{k+1}_{n,m} \subseteq t^k_{n,m} \).

3. If \( n \geq k > 0 \) and \( \gamma \in h_{n,m-1}^k \), then

\[ \text{encode}^k_{n,m}(\gamma) = \text{encode}^k_{n,m-1}(\gamma) \]

by induction hypothesis. Since \( t^k_{n,m-1} \subseteq t^k_{n,m} \),

\[ \text{encode}^k_{n,m}(\gamma) \leq t^k_{n,m} \]

4. If \( n \geq k > 0 \) and \( \gamma = \sigma[k \beta_k] \), then
where $V$ is defined in definition 30. Now,

$$\text{encode}_{n,m}^k(\gamma) = V$$

by induction hypothesis. Therefore,

$$\text{encode}_{n,m-1}^{k+1}(\alpha) \in T_{n,m-1}^{k+1}$$

where $V_1 \in T_{n,m-1}^{k+1}(\beta)$. Therefore,

$$\text{encode}_{n,m}^k(\gamma) = V_1 \text{encode}_{n,m-1}^{k-1}(\beta)$$

Since $\text{encode}_{n,m-1}^{k-1}(\beta) \in T_{n,m-1}^{k-1}$ by induction hypothesis,

$$\text{encode}_{n,m}^k(\gamma) \in T_{n,m-1}^{k+1}(\beta) T_{n,m-1}^{k-1}(\beta)^{k-1}$$

Since if $k > 0$ then $\max(k-1,0) = k-1$.

5. If $k = 0$, then

$$\text{encode}_{n,m}^0 = \text{encode}_{n,m}^1$$

and by a previous part of this proof

$$\text{encode}_{n,m}^0(\gamma) \in T_{n,m}^1$$

Since $T_{n,m}^1 = T_{n,m}^0$ it must be true that

$$\text{encode}_{n,m}^0(\gamma) \in T_{n,m}^0$$
Therefore,

\[ \text{encode}_{n,m}^k (x) \in T_{n,m}^k \]

in all cases and the theorem is proved by induction.

---

**TABLE 3. Some Examples of n-ary Encoded Hypertrees**

<table>
<thead>
<tr>
<th>line level</th>
<th>hypertree</th>
<th>encoding</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>0 a</td>
<td>a[ ]</td>
</tr>
<tr>
<td>2</td>
<td>1 a</td>
<td>a[λ]</td>
</tr>
<tr>
<td>3</td>
<td>1 a[1b[1c[1]]]</td>
<td>a[b[c[λ]]]</td>
</tr>
<tr>
<td>4</td>
<td>2 a</td>
<td>a[λλ]</td>
</tr>
<tr>
<td>5</td>
<td>2 a[2a[1b[1c[1]]]2]</td>
<td>a[a[λb[λc[λλ]]]λ]</td>
</tr>
</tbody>
</table>

line encodings with n-ary trees represented with pointers.

<table>
<thead>
<tr>
<th>line</th>
<th>encoding</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>a[ ]</td>
</tr>
<tr>
<td>2</td>
<td>a[λ]</td>
</tr>
<tr>
<td>3</td>
<td>a[ ] b[ ] c[λ]</td>
</tr>
<tr>
<td>4</td>
<td>a[λλ]</td>
</tr>
<tr>
<td>5</td>
<td>a[λ] a[λ] b[λ] c[λλ]</td>
</tr>
<tr>
<td>6</td>
<td>a[λλ] a[λ] a[λλ] c[λλλ]</td>
</tr>
</tbody>
</table>
Some examples of hypertrees and their n-ary representations are given in table 3. Note that, as had been expected earlier, line 3 corresponds to the linked list representation of the string "abc". A detailed look at this example will show that any encoded string will result in its linked list representation. Similarly, line 5 is the binary encoded tree for "a(abc)", and it should be obvious that any tree will be encoded as the corresponding binary tree.

While the n-ary tree encoding of hypertrees is the only encoding which will be discussed at length in this paper, it might be wise to note a few other encoding schemes. Of course, one other scheme has been used rather extensively in this paper. This is the "normal" encoding using brackets, subscripts and superscripts. It is hoped that this is the easiest encoding for the reader to understand. For example

(11) \[ a_2^b[1^c[1^d_1]_1]_2 \]

Another significant encoding scheme is the recursive encoding. This encoding follows directly from the definition of hypertree given in definition 12 of chapter 2.
Definition 32:

Recursive is a family of functions, denoted

\( \text{recursive}^k_n \), such that

(a) \( \text{recursive}^{n+1}_n(\gamma) = [1, \gamma] \)
if \( n \geq 0 \) and \( \gamma \in h^{n+1}_n(\Xi) \).

(b) \( \text{recursive}^0_n(\gamma) = [2, \text{recursive}^1_n(\gamma)] \)
if \( n \geq 0 \) and \( \gamma \in h^0_n(\Xi) \).

(c) \( \text{recursive}^k_n(\gamma) = [3, \text{recursive}^{k+1}_n(\gamma)] \)
if \( n \geq k > 0 \) and \( \gamma \in h^{k+1}_n(\Xi) \).

\[ = [4, \text{recursive}^{k+1}_n(\alpha), \text{recursive}^{k-1}_n(\beta)] \]
if \( n \geq k > 0 \) and \( \gamma = \alpha[k \beta k] \).

Notice that although \( k \) is not mentioned, its value can always be inferred from the structure. \( n \), on the other hand, must be stored somewhere independent of this encoding as its value cannot be inferred. Since each of the alternatives above have at most two possible recursions, this can be stored in a binary tree. It could be demonstrated that this encoding is directly related to the binary encoding of the n-ary tree encoding of hypertrees.

The example given in equation 11 can be encoded as

\[ 2[4,[1,a],[4,[3,[1,b]],[2,[4,[3,[1,c],
[2,[3,[1,d]]]]]]]] \]
At first glance, this encoding scheme seems worse than the n-ary tree encoding of hypertrees given earlier, but if the level of the hypertrees that are encoded could vary over a large range of values, this encoding could result in less storage being required, since every node is at most binary.

In this chapter, three different encodings for hypertrees are mentioned. In the next chapter, the concept of a hypertree grammar is discussed so that work can begin toward characterizing the algebraic language hierarchy, which was the original justification for the existence of hypertrees.
REGULAR HYPERTREE GRAMMARS

The original justification for hypertrees was that they could model the algebraic language hierarchy. This requires that at least regular grammars be defined on hypertrees at all levels. This chapter attempts to do this. Context free grammars are not defined as a separate entity, but rather that term is reserved for the frontier of a next higher level hypertree grammar. Similarly, macrogrammars are not explicitly defined.

In the case of macrogrammars, there are two different methods of deriving a language from a given grammar. These are referred to as the inside-out grammars and the outside-in grammar by Fischer. This leads to two different methods of deriving a language for a given hypertree grammar. This paper will concern itself with the inside-out (IO) approach due to its pleasing properties. The outside in (OI) approach will be defined but characterization will be left for further study.

Regular grammars are defined in a method that is analogous to the definition of regular grammars over strings.
Definition 33:

A regular grammar of level $n$ and degree $k$, denoted $G_n^k$, is a four-tuple $(\Sigma, V, P, S)_n^k$ where $\Sigma$ is the set of terminals, $V$ is a ranked set of nonterminals, $P$ is a finite set of productions and $S$ is the start symbol. In addition,

(a) $S \in V_k$

(b) $P \subseteq \bigcup_{i \geq 1} (V_i \rightarrow H_n^i(\Sigma, V))$

By using a ranked set of nonterminals it will be possible to generate all of the set $H_n^k(\Sigma)$. As is commonly done, the grammar $G_n^k$ will commonly be designated by specifying the elements of $P$. These will be written vertically with the first production specifying the start symbol. For example

$$G_2^2 = (\{(a), ((S, T), (S\rightarrow 2, T\rightarrow 1))\},$$

$$\{S \rightarrow a[2 T 2], T \rightarrow a[1 T 1], T \rightarrow a\}, S)_2^2$$

will be written as

$$S \rightarrow a[2 T 2]$$

$$T \rightarrow a[1 T 1]$$

$$T \rightarrow a$$

An additional shorthand will be to combine productions with the same left hand side by separating the right hand sides with a "|". Thus, the above grammar may be written as
S → a[T2]
T → a[T1] | a

In addition to defining what is meant by regular grammar, it is also necessary to define the derivation relation so that eventually the language described may be defined.

Definition 34:
The IO derivation relation over grammar $G^k_n$, denoted $G^k_n \Rightarrow$, is defined as

$$G^k_n \Rightarrow : H^k_n(\Sigma, V) \rightarrow H^k_n(\Sigma, V)$$

if $n \geq k > 0$ such that, if $G^k_n = (\Sigma, V, P, S)_n$ and

$\alpha \psi \beta \in H^k_n(\Sigma, V)$ where $V \in V_j$ and $V \rightarrow \gamma \in P$ then

$\alpha \gamma \beta \in G^k_n \Rightarrow (\alpha \psi \beta)$

As has been done previously, a proof will be offered to show that the codomain stated is correct. First a lemma needs to be established.

Lemma 40:
If $n \geq 0$, $n+1 \geq k \geq 0$, $m \geq 0$, $\gamma \in H^k_{n,m'}$, $\gamma = \alpha \psi \beta$,

$\nu \in V_i$ and $\psi \in H^i_n$, then

$\alpha \psi \beta \in H^k_n$

Proof by induction on $m$:
Basis: if $m = 0$ then

1. If $k = n+1$, then

$$\gamma \in \Sigma$$

so that $\gamma$ cannot equal $\alpha \nu \beta$, so this cannot be the case.

2. If $n \geq k > 0$, then

$$\gamma \in V_{k} \cup X_{k}$$

But since $\gamma = \alpha \nu \beta$, it must be that $\gamma \in V_{k}$, $\alpha = \lambda$ and $\beta = \lambda$. Also, $k = i$ since $\nu \in V_{k}$, $\nu \in V_{i}$ and the ranks must be disjoint. Therefore,

$$\alpha \psi \beta = \psi$$

But $\psi \in H_{n}^{k}$ so that

$$\alpha \psi \beta \in H_{n}^{k}$$

3. If $k = 0$, then

$$\gamma \in H_{n,0}^{k}$$

Therefore, by part 2 of this proof,

$$\alpha \psi \beta \in H_{n}^{k}$$

if $\gamma = \alpha \nu \beta$.

Therefore, the basis is proven.
Induction step: Assume that if $m > m' \geq 0$ and $\gamma = \alpha \psi \beta$ then

$$\alpha \psi \beta \in H_n^k$$

If $\gamma \in H_{n,m}^k$ and $\gamma = \alpha \psi \beta$, then

1. If $k = n+1$, then

$$\gamma \in \Sigma$$

so that this cannot be the case.

2. If $n \geq k > 0$ and $\gamma \in H_{n,m-1}^{k+1}$, then

$$\alpha \psi \beta \in H_n^{k+1}$$

by induction hypothesis. But $H_n^{k+1} \subset H_n^k$ so

$$\alpha \psi \beta \in H_n^k$$

3. If $n \geq k > 0$ and $\gamma \in H_{n,m-1}^k$, then

$$\alpha \psi \beta \in H_n^k$$

by induction hypothesis.

4. If $n \geq k > 0$, $\gamma = \gamma_1[k \gamma_2 k]$ and $\gamma_1 = \alpha_1 \psi_1 \beta_1$, then

$$\alpha_1 \psi_1 \beta_1 \in H_n^{k+1}$$

by induction hypothesis. Also, $\alpha = \alpha_1$ and $\beta = \beta_1[k \gamma_2 k]$ so that

$$\alpha \psi \beta = \alpha_1 \psi_1 \beta_1[k \gamma_2 k]$$

$$\in H_n^{k+1}[k H_n^{k-1} k]$$
5. If \( n \geq k > 0 \), \( \gamma = \gamma_1[k, \gamma_2, k] \) and \( \gamma_2 = \alpha_2 \psi \beta_2 \), then

\[
\alpha_2 \psi \beta_2 \in H_n^{k-1}
\]

by induction hypothesis. Also, \( \alpha = \gamma_1[k, \alpha_2 \) and \( \beta = \beta_2 k] \) so that

\[
\alpha \psi \beta = \gamma_1[k, \alpha_2 \psi \beta_2, k]
\]

\[
\in H_n^{k+1}[k, H_n^{k-1}]
\]

\[
\in H_n^{k}
\]

6. If \( k = 0 \), then

\[
\gamma \in H_n^{1}
\]

and by the previous parts of this induction step

\[
\alpha \psi \beta \in H_n^{1}
\]

But since \( H_n^{1} = H_n^{0} \),

\[
\alpha \psi \beta \in H_n^{k}
\]

and the lemma is proved by induction.
Theorem 41:

If \( n \geq k > 0, \gamma_1 \in H_n^k \) and \( \gamma_1 \Rightarrow \gamma_2 \), then

\[ \gamma_2 \in H_n^k \]

Proof:

If \( \gamma_1 \Rightarrow \gamma_2 \), then

\[ \gamma_1 = \alpha \nu \beta \]

and

\[ \gamma_2 = \alpha \psi \beta \]

for some \( \alpha, \beta \) and \( \psi \), where \( \nu \psi \in P \). But, if \( \nu \psi \in P \) then

\[ \nu \in V_i \]

and

\[ \psi \in H_n^i \]

for some \( i \). Therefore,

\[ \gamma_2 \in H_n^k \]

by lemma 40 and the theorem is proved.

---

As is commonly done in the literature on languages, \( \beta \in G_n^k(\alpha) \) will be written in infix notation as \( \alpha G_n^k \Rightarrow \beta \).

Also, if the grammar \( G_n^k \) is obvious, this will be elided so that \( \alpha G_n^k \Rightarrow \beta \) will be written as \( \alpha \Rightarrow \beta \). The transitive and reflexive closure of \( G_n^k \Rightarrow \) will be written as \( G_n^{k \Rightarrow} \), as
defined in the following definition. Also, the $G_n^k$ will be
elided whenever the grammar is obvious in this case also.

Definition 35:

The IO derivation relation closure over grammar $G_n^k$,
denoted $G_n^{k*=>}$, is defined as follows:

$G_n^{k*=>} : H_n^k(\Sigma, V) \rightarrow H_n^k(\Sigma, V)$

such that if $\alpha \in H_n^k(\Sigma, V)$ then

$\alpha G_n^{k*=>} \alpha$

Also, if $\alpha \in H_n^k(\Sigma, V)$ and $\alpha G_n^{k*=>} \beta$ and $\beta G_n^{k*=>} \gamma$
then

$\alpha G_n^{k*=>} \gamma$

As stated above, the $G_n^k$ will be elided, and the *=> will be
written in infix notation if no confusion would result.

Theorem 42:

If $n \geq k > 0$, $\gamma_1 \in H_n^k$ and $\gamma_1 G_n^{k*=>} \gamma_2$, then

$\gamma_2 \in H_n^k$

Proof:

This follows directly from theorem 41 by induction.

The IO language of $G_n^k$ may now be defined
Definition 35:

The IO language defined by grammar $G^k_n = (\Sigma, V, P, S)_n^k$, denoted language($G^k_n$), is defined as

$$language(G^k_n) = \{ \gamma | S \rightarrow^* \gamma \text{ and } \gamma \in H^k_n(\Sigma) \}$$

For example, given the hypertree grammar above, then

$$S \rightarrow a[2 T 2]$$
$$\rightarrow a[2a[1 T 1]2]$$
$$\rightarrow a[2a[1a 1]2]$$

Also, since the second production could have been substituted for the $T$ instead of the third production in step 2 above

$$S \rightarrow a[2a[1 T 1]2]$$
$$\rightarrow a[2a[1a[1 T 1]1]2]$$
$$\rightarrow a[2a[1a[1a 1]1]2]$$

From this short example it should be evident that this grammar will generate all trees of height one over the one element alphabet {a}.

The above example illustrates that hypertree grammars over trees are more powerful than the classical approach defined by Brainard (Brainard 1967). In the classical approach, it would be impossible to write a grammar giving all trees of height one since the root node could only be in
a finite number of the ranks of the ranked alphabet that the language is over.

As was stated earlier, there is another method of deriving a language for a grammar \( G_n^k \) called the OI language. It will be defined here, but analysis of this approach will be left for further research.

Definition 37:

The OI derivation relation over grammar \( G_n^k \) denoted \( G_n^k *=>' \), is defined as follows:

\[
G_n^k *=>': H_{n-1}^k(\Sigma, V) \rightarrow H_{n-1}^k(\Sigma, V)
\]

if \( n > k > 0 \).

\[
G_n^k *=>': H_{n-1}^{n-1}(\Sigma, V) \rightarrow H_{n-1}^{n-1}(\Sigma, V)
\]

if \( n > 0 \). If \( \alpha \) is an element of the domain and \( \alpha => \beta \), then

\[
\text{frontier}_{n-1}^k(\beta) \in G_n^k *=>'(\alpha)
\]

As in the IO case, \( \beta \in G_n^k *=>'(\alpha) \) may be written as \( \alpha =>' \beta \).

Also, the transitive closure may be written as \( \alpha *=>' \beta \).

Notice that this actually defines a language \( \text{language'}(G_n^k) \) which is in \( H_{n-1}^k(\Sigma) \) rather than \( H_n^k(\Sigma) \) as is the case with the IO derivation. For proper comparison, therefore, the frontier language of language \( (G_n^k) \) should be
compared with language'(G_k^k). Call this language, the frontier language of language(G_k^k), Frlanguage(G_k^k). The basic difference between Frlanguage(G_k^k) and language'(G_k^k) is that Frlanguage(G_k^k) first builds up a large level n hyperforest then takes the frontier, whereas language'(G_k^k) takes the frontier after each derivation. The former approach guarantees that each given nonterminal always develops into the same hypertree, even if more than one copy of the result of the derivation of that nonterminal is made in the resulting sentence. The latter approach, on the other hand, allows each copy of the nonterminal to develop independently of the other copies so that different copies can have different values. As expected, this is the same as the difference that exists between IO macro languages and OI macro languages.

If the example given above is expanded using the OI derivation method, then an example of the resulting language is

\[ S \Rightarrow T \\ 
\Rightarrow a[1 \ T \ 1] \\ 
\Rightarrow a[1a[1 \ T \ 1]_1] \\ 
\Rightarrow a[1a[1a[1]_1]_1] \]

Notice that since this grammar is of level 2 the derivation yields a string which is equal to the frontier of one of the
trees generated by the IO derivation method. The grammar generated all trees of height one using the IO derivation method, however, it should be obvious that the set of all strings over the alphabet \{a\} is generated by the OI derivations.

The languages of ultimate interest are string languages, therefore, a new function is defined for extracting the string language associated with a given hypertree language.

Definition 38:

Extract is a family of functions, denoted extract\(_n\), such that

\[
\text{extract}_n: U_{n \geq n} H'_0 \rightarrow H'_0
\]

if \( n \geq 0 \) where

(a) \( \text{extract}_n(y) = y \)
if \( y \in H'_0 \).

(b) \( \text{extract}_n(y) = \alpha \)
if \( y \in H'_0, \beta = \text{frontier}^0_{n-1}(y) \) and
\[ \text{extract}_n(\beta) = \alpha. \]

This function uses the frontier function to repeatedly extract the frontier of a hypertree until the hypertree level is reduced to the desired level. Since the only requirement for the degree of the domain is that it be zero,
any hyperforest of any higher level can be operated on since
$H_n^k \preceq H_n^0$ for any value of $k$. (See lemma 8 chapter 2.)

As mentioned above, the special case of the extract
function which returns strings is important enough to be
given a special name.

Definition 39:

String is another name for the function $\text{extract}_i$.

Also, since the string language associated with a given
grammar is important, the following definition is made.

Definition 40:

The IO string language defined by grammar $G_n^k$,
denoted $\text{string}(G_n^k)$, is defined as

$$\text{string}(G_n^k) = \{ \alpha \mid \alpha = \text{string}(\beta) \text{ and } \beta \in \text{language}(G_n^k) \}$$

Again using the example above, it should be evident
that the IO string language defined by the grammar is the
set of all strings over $\{a\}$.

In order to simplify the proofs involving hypertree
grammars, a special normal form of the grammar has been
developed.
Definition 41:

A grammar $G_n^k = (\Sigma, V, P, S)_n^k$ is said to be in $m$-normal form if

$$P \subseteq \bigcup_{n \geq k > 0} (V_k \rightarrow H_n^k, m)$$

If a grammar is simply said to be in normal form, then it is in 1-normal form.

Given the grammar above

$$S \rightarrow a[T_2]_2$$

$$T \rightarrow a[T_1]_1 | a$$

notice that this is a 2-normal grammar. $a[T_2]_2 \in H_{2,2}^2$ since $a \in \Sigma = H_{2,1}^3$ and $T \in H_{2,1}^1$. The latter is true since $T \in H_{2,0}^1$. Similarly, $a[T_1]_1 \in H_{2,2}^1$ and $a \in H_{2,2}^1$.

Therefore, the grammar is 2-normal. Notice that the grammar, by definition, is also 3-normal since, for instance, $a \in H_{2,3}^1$ in addition to $H_{2,2}^2$. By similar reasoning, the grammar is 125-normal. In general, any grammar which is $m$-normal is also $m'$-normal where $m' \geq m$.

It will be shown that to an extent the opposite of this is also true, the difference being that some changes to the grammar may be needed to decrease the normalcy $m$. But first, a theorem is presented which demonstrates that any grammar in $m$-normal for some $m$. 

Theorem 43:

For every grammar \( G_n^k = (\Sigma, V, P, S)_n^k \) there is an \( m \) such that \( G_n^k \) is \( m \)-normal.

Proof:

By definition 33,

\[
P \subseteq \bigcup_{n \geq k > 0} (V_k \rightarrow H_n^k)
\]

and \( P \) is finite. If \( v \rightarrow a \in P \), then \( a \) must be an element of some \( H_{n,m}^k \) since

\[
H_{n,m}^k = \bigcup_{n \geq 0} H_{n,m}^k
\]

Since \( P \) is finite, each element of \( P \) may determine a value in this fashion. Let the largest of these numbers be the desired \( m \).

---

Lemma 44:

If \( G_n^k = (\Sigma, V, P, S)_n^k \) is \( m \)-normal and \( m > 1 \), then there is a grammar \( G_n^{k'} = (\Sigma', V', P', S')_n^k \) which is \( m-1 \)-normal such that

\[
\text{language}(G_n^k) = \text{language}(G_n^{k'})
\]

Proof:

If grammar \( G_n^k \) is also \( m-1 \)-normal, then the original grammar is the desired grammar. If \( G_n^k \) is not \( m-1 \)-normal, then there
is a finite number of productions, \(v_1 \rightarrow \gamma_1, v_2 \rightarrow \gamma_2, \ldots v_j \rightarrow \gamma_j\) such that

\[ \gamma_r \in \mathcal{H}_{n,m}^i \]

for some \(i\) and

\[ \gamma_r \notin \mathcal{H}_{n,m-1}^i \]

Therefore, construct \(G_n^{k'} = (\Sigma', \mathcal{V}', \mathcal{P}', S')\) as follows:

1. \(\Sigma' = \Sigma\).
2. \(\mathcal{V}'\) consists of \(\mathcal{V}_i\) and elements required for the construction of \(\mathcal{P}'\).
3. The construction for \(\mathcal{P}'\) is given below.
4. \(S' = S\).

\(\mathcal{P}'\) is constructed as follows:

1. \(\mathcal{P}' = \{v_1 \rightarrow \gamma_1, \ldots v_j \rightarrow \gamma_j\} \subseteq \mathcal{P}'\).
2. If \(v \rightarrow \gamma\) is one of the productions given above, \(v \in \mathcal{V}_i\) and \(\gamma \in \mathcal{H}_{n,m-1}^{i+1}\) where \(i < n\) then

   \[ \{v \rightarrow X, X \rightarrow \gamma\} \in \mathcal{P}' \]

   where \(X \in \mathcal{V}_{i+1}\). Note that it is not possible for \(v \rightarrow \gamma\) to be one of the productions given above where \(\gamma \in \mathcal{H}_{n,m-1}^{i+1}\) and \(i = n\) since in this case \(\gamma \in \Sigma\) and, therefore, is in \(\mathcal{H}_{n,0}^{i+1}\).
3. If \(v \rightarrow \gamma\) is one of the equations given above, \(\gamma = a_i^\beta\) and \(n > i > 0\) then

   \[ \{v \rightarrow Y[i \rightarrow Z[i]]\}, Y \rightarrow a_r, Z \rightarrow \beta_r\} \subseteq \mathcal{P}' \]
where \( Y_r \in V_{i+1} \) and \( Z_r \in V_{i-1} \).

4. If \( v_r \rightarrow \gamma_r \) is one of the productions given and 
\( \gamma = \alpha_1 \beta_n \), then 

\[
\{ v_r \rightarrow \alpha_1 Z_r \beta_n \mid Z_r \rightarrow \beta \} \subseteq P'
\]

where \( Z_r \in V_{n-1} \).

5. If \( v_r \rightarrow \gamma_r \) is one of the productions given above 
and \( \gamma = \alpha_1 \beta_1 \), then 

\[
\{ v_r \rightarrow Y_r Z_r \beta_1 \mid Y_r \rightarrow \alpha_1, Z_r \rightarrow \beta_r \} \subseteq P'
\]

where \( Y_r \in V_2 \) and \( Z_r \in V_1 \).

6. These are all of the elements of \( P' \) and \( V' \).

\( G_n^k \) is \( m-1 \)-normal since each of the elements of 

\[
P' = \{ v_1 \rightarrow \gamma_1 \ldots v_j \rightarrow \gamma_j \}
\]

is in \( U_{n \geq k \geq 0} V_k \rightarrow H_{n,m-1}^k \) by definition of the set 
\( \{ v_1 \rightarrow \gamma_1 \ldots v_j \rightarrow \gamma_j \} \). Also, the production \( v_r \rightarrow X_r \) above must 
be in the required set since \( X_r \in V_{k'+1} \) which is in 

\[
H_{n,0}^{k'+1} \subseteq H_{n,1}^k \subseteq H_{n,m}^k
\]

Similarly, \( X_r \rightarrow \gamma_r \) is in the set since \( \gamma_r \in H_{n,m-1}^{k'+1} \) by 
definition. Similar considerations will show that 

\[
\{ v_r \rightarrow Y_r Z_r \beta_r \mid Y_r \rightarrow \gamma_r, Z_r \rightarrow \beta_r \}
\]

are all in the required set.
If $\gamma \in \text{language}(G^k_n)$, then

$$S \Rightarrow \gamma$$

Form a sequence $\gamma_0, \gamma_1, \ldots, \gamma_r$ such that $\gamma_0 = S$, $\gamma_r = \gamma$ and for all $r \geq i \geq 0$

$$\gamma_i \Rightarrow \gamma_{i+1}$$

By theorem 42

$$\gamma_i \in H^k_n$$

To prove that for all $i$

$$S \Rightarrow \gamma_i$$

an induction proof will be presented on $i$

Basis: if $i = 0$ then

$$S \Rightarrow \gamma_0$$

and since $\gamma_0 = S$

$$S \Rightarrow \gamma_0$$

Induction step: Assume that if $i < r$ then

$$S \Rightarrow \gamma_i$$

Then,

$$\gamma_i \Rightarrow \gamma_{i+1}$$

Therefore, there is a $v_i \in V_j$ such that
and
\[ y_{i+1} = \alpha_i \delta_i \beta_i \]
where
\[ v_i \rightarrow \delta_i \in P \]

Therefore, one of the following cases must be true:

1. If \( \delta_i \in \mathcal{H}_{n,m-1} \), then
\[ v_i \rightarrow \delta_i \in P' \]

since these productions were copied from \( P \).

Therefore,
\[ y_i \Rightarrow^* y_{i+1} \]

and by definition
\[ S \Rightarrow^* y_{i+1} \]

2. If \( \delta_i \in \mathcal{H}_{n,m-1} \) and \( \delta \notin \mathcal{H}_{n,m-1} \), then
\[ v_i \rightarrow \alpha_i \delta_i \beta_i \in P' \]

where \( X_i \) occurs only in two specified productions in \( P' \). Therefore,
\[ y_i = \alpha_i v_i \beta_i \]
\[ \Rightarrow^* \alpha_i X_i \beta_i \]

also, \( X_i \rightarrow \delta_i \in P' \) so that
\[
\alpha_i X_i \beta_i \Rightarrow \alpha_i \delta_i \beta_i \\
\Rightarrow \gamma_{i+1}
\]
Therefore, since \( S \Rightarrow^* \gamma_i \)
\[
S \Rightarrow^* \gamma_{i+1}
\]

3. If \( \delta_i^* = \delta_1[k \delta_1 k] \) and \( \delta \notin H_{n,m-1}^j \), then there are \( Y_i \) and \( Z_i \) such that
\[
\nu_i \Rightarrow Y_i[k \ Z_i k] \in P'
\]
Therefore,
\[
\gamma_i = \alpha_i \nu_i \beta_i \\
\Rightarrow \alpha_i Y_i[k \ Z_i k] \beta_i
\]
Also, the productions to expand the \( Y_i \) and \( Z_i \) are included in \( P' \). They are
\[
Y_i \Rightarrow \delta_1 \in P'
\]
\[
Z_i \Rightarrow \delta_2 \in P'
\]
Therefore,
\[
\alpha_i Y_i[k \ Z_i k] \beta_i \Rightarrow \alpha_i \delta_1[k \ \delta_2 \ k] \beta_i \\
\Rightarrow \alpha_i \delta_i \beta_i \\
\Rightarrow \gamma_{i+1}
\]
Therefore, by the definition of \( \Rightarrow^* \)
\[
S \Rightarrow^* \gamma_{i+1}
\]
4. If $\delta = \delta_1[\bar{n} \delta_2 \bar{n}]$ and $\delta_i \notin H_{n,m-1}$, then there is a production

$$v_i \rightarrow \delta[Z_i n]$$

therefore,

$$\gamma_i = a_i v_i \beta_i$$

$$\Rightarrow a_i \delta[Z_i n] \beta_i$$

Also,

$$Z_i \rightarrow \delta_2 \in P'$$

so that

$$\gamma_i \Rightarrow a_i \delta[Z_2 n] \beta_i$$

$$\Rightarrow a_i \delta \beta_i$$

$$\Rightarrow \gamma_{i+1}$$

and by induction hypothesis

$$S \Rightarrow \gamma_{i+1}$$

Therefore, by induction

$$S \Rightarrow \gamma_i$$

for all $i$. In particular,

$$S \Rightarrow \gamma_r$$

Therefore, by the definition of language($G_{n}^k$')

$$\text{language}(G_{n}^k) \subseteq \text{language}(G_{n}^k')$$
If \( y \in \text{language}(G_n^k), \) then there is a sequence
\[
\gamma_0, \gamma_1, \ldots, \gamma_r
\]
such that
\[
\gamma_0 = S \\
\gamma_r = y
\]
and for all \( r > i \geq 0 \)
\[
\gamma_i \Rightarrow \gamma_{i+1}
\]
Construct a new sequence
\[
\gamma_0', \gamma_1', \ldots, \gamma_r'
\]
such that \( \gamma_i' = \gamma_i \) with all of the new nonterminals introduced in step 2 of the construction of \( G_n^k \) replaced by the right-hand-sides of the corresponding productions.
To prove: For all \( r \geq i \geq 0 \)
\[
S \Rightarrow \gamma_i'
\]
This will be proved by induction on \( i \).
Basis: if \( i = 0 \) then
\[
\gamma_0' = \gamma_0 \\
= S
\]
and by definition of \( \Rightarrow \)
\[
S \Rightarrow S
\]
Therefore,

\[ S \Rightarrow \gamma_0 \]

Induction step: Assume that if \( i < r \) then

\[ S \Rightarrow \gamma_i' \]

And there are two possible cases for the relationship between \( \gamma_i' \) and \( \gamma_{i+1}' \)

1. If \( \gamma_i' = \gamma_{i+1}' \), then

\[ S \Rightarrow \gamma_{i+1}' \]

by induction hypothesis.

2. If \( \gamma_i' = a_i v_i \beta_i' \), \( \gamma_{i+1}' = a_i \delta_i \beta_i \) and \( v_i \gamma_i \delta_i \in P \), then

\[ \gamma_i' = a_i v_i \beta_i \]

\[ \Rightarrow a_i \delta_i \beta_i \]

\[ \Rightarrow \gamma_{i+1} \]

and by the definition of \( \Rightarrow \)

\[ S \Rightarrow \gamma_{i+1}' \]

These are the only possible cases because suppose

\[ \gamma_i = a_i v_i \beta_i \]

and

\[ \gamma_{i+1} = a_i \delta_i \beta_i \]
where

\[ v_i \rightarrow \delta_i \in P' \]

and

\[ v_i \rightarrow \delta_i \notin P \]

then either \( v_i \) is one of the new nonterminals added due to the construction, in which case

\[ \gamma_i' = \gamma_{i+1}' \]

by construction or \( \delta_i \) contains one or more of the new nonterminals. In this latter case, if \( \delta_i = \delta \) with the new nonterminals replaced by the corresponding right hand sides, then

\[ v_i \rightarrow \delta_i' \in P \]

by construction. Therefore,

\[ \gamma_i' = a_i' v_i \beta_i' \]

\[ \Rightarrow a_i' \delta_i' \beta_i' \]

\[ \Rightarrow \gamma_{i+1}' \]

where \( a_i' \) is \( a_i \) with all the occurrences of the new nonterminals replaced by the corresponding right hand sides, etc. Therefore, for all \( i \)

\[ S \Rightarrow \gamma_i \]

In particular,
Therefore, since $S *\Rightarrow Y$ and it must be that

$$Y \in \text{language}(G^n)$$

Therefore,

$$\text{language}(G^n) = \text{language}(G^n')$$

The example that is being dealt with above is 2-normal. By this theorem, there should be a 1-normal grammar which has the same language. The productions given are

(a) $S \rightarrow a[2, T_2]$
(b) $T \rightarrow a[1, T_1]$
(c) $T \rightarrow a$

with $S \in V_2$ and $T \in V_1$. In this example, production a is also already in 1-normal form and is, therefore, carried into the new set of productions unchanged. Production b will generate three new productions,

$$T \rightarrow X[1, Y_1]$$
$$X \rightarrow a$$
$$Y \rightarrow T$$
each of which is 1-normal. Similarly, production \( c \) generates two new productions.

\[
T \rightarrow Z \\
Z \rightarrow a
\]

Therefore, a new grammar which is 1-normal and has the same language as the 2-normal grammar presented is

\[
S \rightarrow a[T_2] \\
T \rightarrow X[Y_1] \\
X \rightarrow a \\
Y \rightarrow T \\
T \rightarrow Z \\
Z \rightarrow a
\]

where the ranked nonterminals are

\[
V_2 = \{S, X, Z\} \\
V_1 = \{T, Y\}
\]

Since any grammar has a corresponding grammar at the next lower level, and that grammar has another grammar at the level below that, and so forth until the grammar has a 1-normal grammar that has the same language, it seems reasonable that any grammar has to have a normal grammar with the same language. It could not be continued to a
0-normal grammar since lemma 44 requires that $m$ be greater than one in order to reduce the level. This fact is stated in the following theorem.

**Theorem 45:**

For every grammar $G^k_n$ there is a corresponding normal grammar $G^k_n'$.

**Proof:**

This follows directly from theorem 43 and lemma 44.

---

In this chapter, grammars have been discussed, and a normal grammar has been presented. In the following chapter, it will be shown that not only is the generative approach of grammars available for hypertrees, but the recognizing approach of automata is also available.
FINITE AUTOMATA

An alternative method of describing regular languages that is used in the literature is called finite automata. This is a method of describing a function which takes as an argument an element of $H_n^k$. This function will, then, return a value of a state. If this state is one of a predefined set of states called the final states, then the original element is said to be in the language described by the finite automaton. In place of the nonterminals from the previous chapter, an automaton has states, and in place of productions, an automaton has a finite transition function.

Definition 42:
A finite automata, denoted $A_n^k$, is a four-tuple $(\Sigma, V, \delta, F)_n^k$ where $n \geq k > 0$ and

1. $\Sigma$ is a set of terminal symbols.
2. $V$ is a ranked set of states.
3. $\delta$ is a set of functions called the transition functions

$$\delta^i_n: H_{n,1}^i \to P(V_i)$$

where $n \geq i > 0$ and $\delta^i_n(\alpha) = \emptyset$ except for a finite number of $\alpha$'s.
4. $F$ is a finite subset of $V_k$ called the set of final states.
The similarity between the definition of finite automata and regular grammars should be noted. As in the case of grammars, a finite automaton can be specified by giving only the transition functions and the final states. A result that will be assumed but not proven is that any automaton has a corresponding automaton— one describing the same language— which has finite sets for terminals and states.

An example of a finite automaton is as follows:

\[
\delta^2_2(a[T]) = \{S\}, \\
\delta^1_2(X[Y]) = \{Y,T\}, \\
\delta^2_2(a) = \{X,Z\}, \\
\delta^1_2(Z) = \{Y,T\},
\]

final states = \{S\}

It will be possible to show that this finite automaton describes the same language as the grammar given in the last chapter.

As in the case of grammars in the last chapter, a function needs to be defined which will allow the application of this automaton to the selection of the elements of a language.
Definition 43:

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

\[
A_n^{k,i}(\gamma) = \begin{cases} 
\mathcal{P}(\Sigma) & \text{if } i = n+1, \\
\mathcal{P}(\Sigma_i) & \text{if } i = 0, \\
\mathcal{P}(V_i) & \text{if } n \geq i > 0.
\end{cases}
\]

where $A_n^{k,i}(\gamma)$ is the smallest set such that

(a) $A_n^{k,n+1}(\gamma) = \{\gamma\}$
(b) $A_n^{k,0}(\gamma) = A_n^{k,1}(\gamma)$
(c) $A_n^{k,i}(\gamma) = \delta_n^i(\gamma)$

if $\gamma \in \Sigma_i$.

$A_n^{k,i}(\gamma) = U_n^i(A_n^{k,i+1}(\gamma))$

if $\gamma \in \Sigma_i$.

$A_n^{k,i}(\gamma) = U_n^i(A_n^{k,i+1}(\alpha) \cup A_n^{k,i-1}(\beta))$

if $\gamma = \alpha \cup \beta$.

where $n \geq k > 0$ and $n \geq i > 0$.

Definition 44:

The language identified by finite automaton $A_n^k$, denoted $\text{language}(A_n^k)$, is defined as

\[
\text{language}(A_n^k) = \{\gamma \mid \gamma \in \Sigma_n^k(\Sigma) \text{ and } F_n A_n^k(\gamma) \neq \emptyset\}
\]
An example of the complete transition function for the automaton given earlier is

\[ A_{2,2}^{2,2}(a_{2}a_{1}a_{1}a_{1}a_{1}) = U_{2}^{2}(A_{2,3}^{2,3}(a)a_{2}A_{2}^{2,1}(a_{1}a_{1}a_{1})) \]

But

\[ A_{2,3}^{2,3}(a) = a \]

and

\[ A_{2}^{2,1}(a_{1}a_{1}a_{1}) = U_{2}^{1}(A_{2,2}^{2,2}(a)a_{1}A_{2}^{2,0}(a_{1}a_{1})) \]

and

\[ A_{2}^{2,2}(a) = U_{2}^{2}(A_{2}^{2,3}(a)) = U_{2}^{2}((a)) = U\{(X,Z)\} = \{X,Z\} \]

also

\[ A_{2}^{2,0}(a_{1}a_{1}) = A_{2}^{2,1}(a_{1}a_{1}) \]

\[ A_{2}^{2,1}(a_{1}a_{1}) = U_{2}^{1}(A_{2,2}^{2,2}(a)a_{1}A_{2}^{2,0}(a_{1})) \]

and

\[ A_{2}^{2,0}(a) = A_{2}^{2,1}(a) = U_{2}^{1}(A_{2,2}^{2,2}(a)) = U_{2}^{1}((X,Z)) = \{Y,T\} \]

Therefore,
\[ A_{2}^{2,1}(a[1a_1]) = U\delta_{2}^{1}((X,Z)[1(Y,T)_{1}]) \]
\[ = \{Y,T\} \]

and

\[ A_{2}^{2,0}(a[1a_1]) = \{Y,T\} \]

and

\[ A_{2}^{2,1}(a[1a_1a_1]) = U\delta_{2}^{1}((X,Z)[1(Y,T)_{1}]) \]
\[ = \{Y,T\} \]

and

\[ A_{2}^{2,2}(a[2a[1a_1a_1]_{1}]) = U\delta_{2}^{2}(a[2\{Y,T\}_{2}]) \]
\[ = \{S\} \]

Since S is a final state, a[2a[1a_1a_1]_{1}]_{2} is in the language specified by the finite automaton. It was stated above that this automaton specified the same language as the grammar given in the last chapter, therefore, this result is expected.

It might also be instructive to consider what happens when the complete transition function is applied to an invalid hypertree. For example, the hypertree a[2a[2a_2]_{2}] is of depth 2 and is, therefore, not expected to be in the language.

\[ A_{2}^{2,2}(a[2a[2a_2]_{2}]) = U\delta_{2}^{2}(A_{2}^{2,3}(a)[2A_{2}^{2,1}(a[2a_2])_{2}]) \]
\[ = U\delta_{2}^{2}(a[2U\delta_{2}^{1}(A_{2}^{2,2}(a[2a_2]))_{2}]) \]
\[ = U\delta_{2}^{2}(a[2U\delta_{2}^{1}(\delta_{2}^{2}(A_{2}^{2,3}(a)[2A_{2}^{2,1}(a)]_{2}))_{2}]) \]
\[ \begin{align*}
&= U_2 \delta_2^2 (a[2] U_2 \delta_2^1 (a[2] (Y,T)_2 )_2 )_2 ) \\
&= U_2 \delta_2^2 (a[2] U_2 \delta_2^1 (S)_2 )_2 ) \\
&= U_2 \delta_2^2 (a[2] U_2 \delta_2^1 (\emptyset)_2 )_2 ) \\
&= \emptyset
\end{align*} \]

Notice that here there are no states corresponding to the input tree, therefore, there are no final states.

Another concept that is helpful in using automatons is that of deterministic finite automata. In a deterministic finite automaton, each element of the domain of the transition function has only one state to go to. This is equivalent to saying that the codomain of \( \delta_n^k \) is further restricted so that each of the sets in the codomain have at most one element of \( V_k \) in it. That is, the codomain is \( V_k \) instead of \( P(V_k) \).

**Definition 45:**

A finite automaton \( A_n^k = (\Sigma, V, \delta, F)_n^k \) is deterministic if for all \( n \geq k > 0 \)

\[ \delta_n^k : B_n^k, 1 \to V_k \]

An example of a deterministic finite automaton is as follows:
\[ \delta_2^2(a_{[2T_2]}) = S \]
\[ \delta_2^1(X_{[1T_1]}) = T \]
\[ \delta_2^2(a) = X \]
\[ \delta_2^1(X) = T \]

final state = S

It should be evident that this automaton recognizes the same language as the previous nondeterministic automaton. In general, this is the case.

Theorem 46:

Given a finite automaton \( A_n = (\Sigma, V, \delta, F)_n \) there exists a deterministic finite automaton \( A'_n = (\Sigma', V', \delta', F')_n \) such that

\[ \text{language}(A'_n) = \text{language}(A_n) \]

Proof:

Construct the new finite automaton such that
1. \( \Sigma' = \Sigma \).
2. \( V_i' = P(V_i) \) where \( n \geq i > 0 \).
3. The construction for the \( \delta_i' \) functions is given below.
4. \( F' = \{ V \mid V \subseteq V_k \text{ and } F \cap V \neq \emptyset \} \)

The construction of the \( \delta_i' \) functions is
1. If \( \gamma \in \Sigma \cup X_{i+1} \cup X_i \cup X_{i+1}[i \ X_{i-1} \ i] \), then
\[ \delta_i'(\gamma) = (\delta_i^1(\gamma)) \]
2. If 

\[ \gamma \in V_i \cup V_{i+1} \cup (P(\Sigma)V_{i+1}\cup \Sigma_{i+1})[i \cdot (V_{i-1}\cup \Sigma_{i-1}) i], \]

then

\[ \delta^i_n(\gamma) = \{U\delta^i_n(\gamma)\} \]

\(A^k_n\) is deterministic, since the value of \(\delta^k_n(\gamma)\) is a single element of \(V^k = P(V_k)\) in each case. If \(\gamma \in \mathcal{H}^i_{n,m}(\Sigma)\), then

\[ A^{k,i}_{n}(\gamma) = \{A^{k,i}_{n}(\gamma)\} \]

if \(m \geq 0\) and \(n \geq i \geq 0\), since by induction on \(i\)

Basis: if \(m = 0\) then if \(i > 0\) then

\[ \gamma = X_i \]

Therefore,

\[ A^{k,i}_{n}(\gamma) = \delta^i_n(\gamma) \]

\[ = \{\delta^i_n(\gamma)\} \]

\[ = \{A^{k,i}_{n}(\gamma)\} \]

If \(i = 0\), then

\[ \gamma \in X_1 \]

and

\[ A^{k,0}_{n}(\gamma) = A^{k,1}_{n}(\gamma) \]

\[ = \{A^{k,1}_{n}(\gamma)\} \]

\[ = \{A^{k,0}_{n}(\gamma)\} \]
Induction step: Assume that if \( m > m' \geq 0 \) and \( \gamma \in H_{n,m}^i \), then

\[
A_{n}^{k,i}(\gamma) = \{A_{n}^{k,i}(\gamma)\}
\]

If \( \gamma \in H_{n,m}^i \), then

1. If \( i = n \) and \( \gamma \in H_{n,m-1}^{n+1} \), then

\[
A_{n}^{k,n'}(\gamma) = U\delta_{n}^{n'}(A_{n}^{k,n+1}(\gamma))
= U\delta_{n}^{n'}(\{\gamma\})
= \delta_{n}^{n'}(\gamma)
= \{\delta_{n}^{n'}(\gamma)\}
= \{U\delta_{n}^{n'}(\gamma)\}
= \{A_{n}^{k,n}(\gamma)\}
\]

2. If \( i = n \) and \( \gamma \in H_{n,m-1}^{n} \), then

\[
A_{n}^{k,n'}(\gamma) = \{A_{n}^{k,n}(\gamma)\}
\]

by induction hypothesis.

3. If \( i = n \) and \( \gamma = \alpha[\beta_n] \), then

\[
A_{n}^{k,n'}(\gamma) = U\delta_{n}^{n'}(A_{n}^{k,n+1}(\alpha)[A_{n}^{k,n-1}(\beta)_n])
= U\delta_{n}^{n'}(\{\alpha\}[A_{n}^{k,n-1}(\beta)_n])
= U\delta_{n}^{n'}(\{\alpha[A_{n}^{k,n-1}(\beta)_n]\})
= \delta_{n}^{n'}(A_{n}^{k,n+1}(\alpha)[A_{n}^{k,n-1}(\beta)_n])
= \{U\delta_{n}^{n'}(A_{n}^{k,n+1}(\alpha)[A_{n}^{k,n-1}(\beta)_n])\}
\]
4. If \( n > i > 0 \) and \( y \in H^{i+1}_{n,m-1} \), then

\[
A^{k,i}_{n}(y) = \delta^{i}_{n}(A^{k,i+1}_{n}(y))
\]

5. If \( n > i > 0 \) and \( y \in H^{i}_{n,m-1} \), then

\[
A^{k,i}_{n}(y) = (A^{k,i}_{n}(y))
\]

by induction hypothesis.

6. If \( n > i > 0 \) and \( y = \alpha[i \beta i] \), then

\[
A^{k,i}_{n}(y) = \delta^{i}_{n}(A^{k,i+1}_{n}[i \alpha] A^{k,i-1}_{n}[i \beta])
\]

\[
= \delta^{i}_{n}(A^{k,i+1}_{n}[i \alpha] A^{k,i-1}_{n}[i \beta])
\]

\[
= (A^{k,i}_{n}(y))
\]
7. If \( i = 0 \), then

\[
A_n^{k,0}(\gamma) = A_n^{k,1}(\gamma) = \{A_n^{k,1}(\gamma)\} = \{A_n^{k,0}(\gamma)\}
\]

by one of the previous sections of this proof (depending on the value of \( n \)).

Therefore, since \( i = n, n > i > 0 \) or \( i = 0 \),

\[
A_n^{k,i}(\gamma) = \{A_n^{k,i}(\gamma)\}
\]

if \( \gamma \in H_{n,m}^{k} \). Therefore, the conjecture is proved by induction.

If \( \gamma \in \text{language}(A_n^k) \), then

\[
A_n^{k,k}(\gamma) \cap F' \neq \emptyset
\]

Therefore, \( A_n^{k,k}(\gamma) \in F' \) by point 4 of the construction and

\[
\{A_n^{k,k}(\gamma)\} \cap F' = \{A_n^{k,k}(\gamma)\} \neq \emptyset
\]

Therefore, by the conjecture,

\[
A_n^{k,k}(\gamma) \cap F' \neq \emptyset
\]

Therefore, \( \gamma \in \text{language}(A_n^k) \) and

\[
\text{language}(A_n^k) \subset \text{language}(A_n^k)
\]
If \( y \in \text{language}(A^k_n) \), then
\[
A^k_n, k'(y) \cap F' \neq \emptyset
\]

Therefore, by the conjecture,
\[
\{A^k_n, k'(y)\} \cap F' \neq \emptyset
\]

Since \( \{A^k_n, k'(y)\} \) has only one element, it must be true that
\[
A^k_n, k'(y) \in F'
\]

But by the construction this implies that
\[
F \cap A^k_n, k'(y) \neq \emptyset
\]

Since this is the same condition that is required of \( y \) for it to be an element of \( \text{language}(A^k_n) \). Therefore,
\[
y \in \text{language}(A^k_n) \quad \text{and} \quad \text{language}(A^k_n) \subseteq \text{language}(A^k_n')
\]

and \( A^k_n' \) is a deterministic finite automaton which describes the same language as the original automaton \( A^k_n \), and the theorem is proved.

---

At this point, it might be well to see how this definition of finite automata compares to the automata given in the literature. Considering the case of string automata
first, it will be noted that the transition function $\delta^1_1$ (the only one) makes the following mapping

$$\delta^1_1: H^1_{1,1} \rightarrow P(V_1)$$

But notice that $H^1_{1,1}$ is a very limited case. Therefore,

$$H^1_{1,1} = \mathcal{E} U V_1 U X_1 U \mathcal{E}_1 (V_1 U X_1) 1$$

It can be seen that this automata differs from the automata given in the literature, since null transitions can be specified. In particular, this happens when a value of $V_1$ is mapped by $\delta^1_1$. In the definition of the complete transition function, $\delta^1_1$ is arbitrarily restricted so that this doesn't happen so that

$$H^1_{1,1} = \mathcal{E} U X_1 U \mathcal{E}_1 (V_1 U X_1) 1$$

Since $X_1$ cannot appear in classical strings they can be similarly deleted.

$$H^1_{1,1} = \mathcal{E} U \mathcal{E}_1 V_1 1$$
If a new state is chosen, call it $q_0$, then a new function, $\delta$, can be defined

$$\delta: \Sigma \times V \rightarrow P(V)$$

where $V = V_1 \cup \{q_0\}$ and

$$\delta(a,v) = \delta^1_{1}(a[1 v 1]) \quad \text{if } v \neq q_0,$$

$$= \delta^1_{1}(a) \quad \text{if } v = q_0.$$

The function $\delta$ is precisely the transition function needed to generate a classical automaton $(V,\Sigma,\delta,q_0,F)$.

A similar development would show that classical tree automata as defined by Thatcher and Wright (Thatcher 1968) is a special limited case of hypertree automata also. The method used is similar to that used in chapter 4 for grammars over trees. As was the case with grammars, hypertree finite automata can describe a larger class of tree languages than classical tree automata as the example given earlier in this chapter indicates. It has been shown that the set of languages accepted by finite automata over trees is the same as the set of languages described by regular grammars over trees in the classical case, so this could be expected. This same property is true in the case of hypertrees also, as the following theorem proves.
Lemma 47:

For every normal regular grammar $G^k_n = (I, V, P, S)^k_n$

there exists a finite automaton $A^k_n = (\Sigma', V', \delta', F')^k_n$

such that if

$v \Rightarrow^* \gamma$

then

$v \in A^k_n(\gamma)$

for all $v \in V_i, n \geq k > 0$ and $n \geq i > 0$.

Proof:

Since any production of the form $A \rightarrow B$ where $A$ and $B$ are
both in $V_i$ for some $i$ can be easily eliminated by forward
substitution assume no productions of this type occur in $G^k_n$.

In a similar fashion, it is possible to alter $G^k_n$ so that $X$'s
only appear in productions of the form

$v \rightarrow X^j_i$

Therefore, assume this is true of $G^k_n$ also. Construct the
finite automaton such that

1. $\Sigma' = \Sigma$.
2. $V' = V$.
3. If $v \rightarrow y \in P$ and $v \in V_i$, then $v \in \delta^i_n(\gamma)$ and these
   are the only elements is $\delta^i_n(\gamma)$.
4. $F' = \{S\}$. 
Claim: $A_n^k$ is the required automaton. This will be proved by induction on the number of applications of $\Rightarrow$ used to produce $\gamma$.

Basis: if only one step is required to produce $\gamma$ then there must be a production

$$v \rightarrow \gamma$$

in $P$. Therefore, by construction

$$v \in \delta_n^i(\gamma)$$

Therefore,

1. If $\gamma \in X_i$, then

$$v \in \delta_n^i(\gamma)$$

$$\varepsilon_{A_n^{k,i}}$$

2. If $\gamma \in \Sigma$ and $i = n$, then

$$v \in \bigcup_{\delta_n^i}(\gamma)$$

$$\varepsilon_{\bigcup_{\delta_n^i}(A_n^{k,n+1}(\gamma))}$$

$$\varepsilon_{A_n^{k,n}(\gamma)}$$

Note that $\gamma \neq \alpha_{[i \beta_i]}$, since $\beta$ must be a state. Therefore,

$\gamma \in H_n^i(\Sigma)$ and $\gamma \notin V_i \cup V_{i+1}$. 
Induction step: Assume that if v *=> y in less than m steps then

\[ v \in A^k_i(y) \]

Suppose that v *=> y in m > 1 steps where v \( \in V_i \). Then it must be true that v ==> y_1 *=> y where v=y_1 \( \in P \). Therefore, v \( \in \delta^i_n(y_1) \) by construction. y_1 \( \notin I \cup X_i \), since this would have to be the last production if that were the case and it was assumed that there were more than one production.

1. If y_1 \( \in V_i+1 \), then y_i *=> y in fewer than m steps, so that

\[ y_i \in A^k_{i+1}(y) \]

by induction hypothesis. Therefore,

\[ \delta^i_n(y_i) \subset \cup \delta^i_n(A^k_{n,i+1}(y)) \]

\[ \subset A^k_{n,i}(y) \]

Therefore,

\[ v \in A^k_{n,i}(y) \]

2. If y_1 = a_1[n b_1 n], then

\[ y = a[n b n] \]

where a = a_1 and b_1 *=> b in fewer than m steps. Therefore,

\[ b_1 \in A^k_{n,i-1}(b) \]
also
\[
\delta_n^i(\gamma_1) = \delta_n^i(\alpha_1[\beta_1])
\]
\[
c \cup \delta_n^i((\alpha)[A_n^k, A_n^{k-1}(\beta)])
\]
\[
c \cup \delta_n^i(A_n^k, A_n^{k+1}(\alpha)[A_n^k, A_n^{k-1}(\beta)])
\]
\[
c A_n^k, A_n^k(\alpha)[A_n^k, A_n^n(\beta)]
\]
\[
c A_n^k, A_n^n(\gamma)
\]
and \(\gamma \in A_n^k, A_n^n(\gamma)\).

3. If \(\gamma_1 = \alpha_1[\beta_1]\) and \(n > 1\), then
\[
\gamma = \alpha_1[\beta_1]
\]
where \(\alpha_1 \Rightarrow \alpha\) and \(\beta_1 \Rightarrow \beta\) with fewer than \(m\) steps each. Therefore, \(\alpha_1 \in A_n^{k, i+1}(\alpha)\), \(\beta_1 \in A_n^{k, i-1}(\beta)\) and
\[
\delta_n^i(\gamma_1) = \delta_n^i(\alpha_1[\beta_1])
\]
\[
c \cup \delta_n^i(A_n^k, A_n^{k+1}(\alpha)[A_n^k, A_n^{k-1}(\beta)])
\]
\[
c A_n^k, A_n^k(\alpha)[A_n^k, A_n^n(\beta)]
\]
\[
c A_n^k, A_n^i(\gamma)
\]
and \(\gamma \in A_n^{k, i}(\gamma)\).

Since \(\gamma \in A_n^{k, i}(\gamma)\) in each case the lemma is proved by induction.

- - -
Lemma 48:

For all normal grammars $G_n^k = (\Sigma, V, P, S)_n^k$ there is a finite automaton $A_n^k$ such that

$$\text{language}(A_n^k) = \text{language}(G_n^k)$$

for all $n \geq k > 0$.

Proof:
Use the automaton constructed for lemma 47.

$$\text{language}(G_n^k) = \{ \gamma | S^* \Rightarrow \gamma \text{ and } \gamma \in H_n^k(\Sigma) \}$$

$$= \{ \gamma | S \in A_n^k, \gamma(\gamma) \text{ and } \gamma \in H_n^k(\Sigma) \}$$

$$= \{ \gamma | A_n^k(\gamma) \cap \{ S \} \neq \emptyset \text{ and } \gamma \in H_n^k(\Sigma) \}$$

$$= \text{language}(A_n^k)$$

Q.E.D.

- • -

Theorem 49:

For every regular grammar $G_n^k$ there exists a finite automaton $A_n^k$ such that

$$\text{language}(G_n^k) = \text{language}(A_n^k)$$

for all $n \geq k > 0$.

Proof:
This follows directly from lemma 48.

- • -
Therefore, any regular grammar has a finite automaton that accepts the same language as the grammar generates. It is also true that any finite automaton has a grammar which produces the same language as the automaton accepts, however, before this is proved a third method of describing regular languages will be discussed, namely regular expressions.
Yet a third method of describing regular sets is called regular expressions. In this chapter, it will be shown that regular expressions, which have been defined over strings and trees, may be extended to hypertrees of any level. Again, in the case of trees the hypertree regular expressions will be more powerful than those described by Thatcher and Wright (Thatcher and Wright 1968).

Regular expressions describe regular sets by using operations on sets. These are put together into an expression, called a regular expression, which then describes the regular set. Using the ideas presented by Thatcher and Wright, rather than concatenating sets as is done in regular string expressions, elements or the frontier of hypertrees will be replaced to effect concatenation. However, in this case the symbols to be replaced will be collected together into a ranked set called the set of auxiliary symbols. These will correspond roughly to the nonterminals used in regular grammars and the states used in finite automata.

Before regular expressions can be defined, two new operations need to be defined for hypertrees.
Definition 45:

Concatenation using \( v \) over \( H_n^k \) is an operation, denoted \( *_{v_n}^k \), such that

\[
*_{v_n}^k : H_n^k \times P(H_n^i) \to P(H_n^k)
\]

where \( v \in V_i \), \( n \geq 0 \), \( n+1 \geq k \geq 0 \), \( n \geq i > 0 \) and the value of \( *_{v_n}^k \) is the smallest set such that

(a) \( *_{v_n}^{n+1}(A,B) = \{A\} \) if \( n \geq 0 \).

(b) \( *_{v_n}^0(A,B) = *_{v_n}^1(A,B) \) if \( n \geq 0 \).

(c) \( *_{v_n}^{k+1}(A,B) \) if \( A \in H_n^{k+1} \).

\[
*_{v_n}^k(A,B) = \{A\} \quad \text{if} \ A \in X_k \cup V_k \text{ and } A \neq V.
\]

= \( B \)

if \( A = V \).

= \( *_{v_n}^{k+1}(\alpha, \beta) [_{k} *_{v_n}^{k-1}(\beta, B) k] \)

if \( A = \alpha [_{k} \beta k] \).

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

Even though concatenation has only been defined for individual elements of \( H_n^k \) in the first argument, it will be extended to subsets of \( H_n^k \) with no formal definition. Also, if \( A \) is a subset of \( H_n^k \), \( B \) is a subset if \( H_n^i \) and \( v \) is an
element of $V_i$ then $*v^k_n(A,B)$ will be written using infix notation as

$$A *_v B = U^*v^k_n(A,B)$$

Even though the $k$, $n$, $I$ and $V$ do not appear in this notation they will be assumed.

Notice that concatenation replaces all occurrences of the argument symbol $v$ with a string from the set $B$. For example, if

$$A = a_{[2} a_{[2} v_{2]}[1 v_{1}]_{2}$$

and

$$B = \{c,d\}$$

where $v \in V_1$ then

$$A *_v B = a_{[2} a_{[2} v_{2]}[1 v_{1}]_{2} *_v B$$

$$= (a *_v B)[2(a_{[2} v_{2]}[1 v_{1}] *_v B)_{2}]$$

$$= a_{[2}(a_{[2} v_{2} *_v B)[1 (v *_v B)_{1}]_{2}]$$

$$= a_{[2}(a *_v B)[2 (v *_v B)_{2}][1 B_{1}]_{2}]$$

$$= a_{[2} a_{[2} B_{2}][1 B_{1}]_{2}$$

$$= a_{[2} a_{[2} (c,d)_{2}][1 (c,d)_{1}]_{2}$$

$$= \{a_{[2} a_{[2} c_{2}][1 c_{1}]_{2}, a_{[2} a_{[2} c_{2}][1 d_{1}]_{2},$$

$$a_{[2} a_{[2} d_{2}][1 c_{1}]_{2}, a_{[2} a_{[2} d_{2}][1 d_{1}]_{2}\}$$
As has been previously the case, the codomain of the concatenation operation has been assumed. This codomain will now be proved to be correct.

Lemma 50:
If \( n+1 \geq k \geq 0, \ n \geq i > 0, \ m \geq 0, \ v \in V_i, \ A \in H_{n,m}^k \) and \( B \in P(H_i^n) \), then
\[
A \ast_v B \subseteq H_n^k
\]
Proof by induction on \( m \):
Basis: If \( m = 0 \), then one of the following cases is true:
1. If \( k = n+1 \), then
\[
A \ast_v B = \{A\}
\]
but \( A \in H_n^{n+1} \), therefore, \( \{A\} \subseteq H_n^{n+1} \) and
\[
A \ast_v b \subseteq H_n^{n+1}
\]
2. If \( n \geq k > 0 \) and \( A \neq v \), then \( A \in X_k \cup V_k \).
\[
A \ast_v B = A
\]
\[
\subseteq H_n^k
\]
3. If \( A = v \), then \( i = k \) so that \( B \subseteq H_n^k \). Therefore,
\[
A \ast_v B = B
\]
\[
\subseteq H_n^k
\]
4. If \( k = 0 \), then \( A \ast_v B \) is a special case of one of the previous sections of this proof since \( A \in H^1_n \).

Induction step: Assume that if \( m > m' > 0 \) and \( A \in H^k_{n,m'} \), then

\[
A \ast_v B \subseteq H^k_n
\]

If, therefore, \( A \in H^k_{n,m'} \), then one of the following is true.

1. If \( k = n+1 \), then

\[
A \ast_v B = \{A\}
\]

2. If \( n \geq k > 0 \) and \( A \in H^{k+1}_{n,m-1} \), then \( A \ast_v B \subseteq H^{k+1}_n \)
   by induction hypothesis. But \( H^{k+1}_n \subseteq H^{k}_n \). Therefore,

\[
A \ast_v B \subseteq H^{k}_n
\]

3. If \( n \geq k > 0 \) and \( A \in H^k_{n,m-1} \), then

\[
A \ast_v B \subseteq H^{k}_n
\]

by induction hypothesis.

4. If \( n \geq k > 0 \) and \( A = \alpha[\beta_k] \), then \( \alpha \ast_v B \subseteq H^{k+1}_n \)
   and \( \beta \ast_v B \subseteq H^{k-1}_n \) by induction hypothesis. Since

\[
A \ast_v B = \alpha \ast_v B[\beta \ast_v B_k]
\]

it must be true that

\[
A \ast_v B \subseteq H^{k}_n
\]
5. If $K = 0$, then $A \in H_{n,m}^1$ and $A \ast_v B \subset H_{n}^1$ by a previous section of this proof. But since $H_{n}^1 = H_{n}^0$,

$$A \ast_v B \subset H_{n}^0$$

Since in each of these cases $A \ast_v B \subset H_{n}^k$ the lemma is proved by induction.

---

Theorem 51:

If $n+1 \geq k \geq 0$, $n \geq i > 0$, $v \in V_1$, $A \in H_{n}^k$ and $B \in H_{n}^i$, then

$$A \ast_v B \subset H_{n}^k$$

Proof:

This follows directly from the lemma.

---

Just as is the case with strings, concatenation will be extended to form the closure of a set.

Definition 47:

The closure of a set using $v$ over $H_{n}^k$ is an operation, denoted $^{+v}$, such that

$$^{+v} : P(H_{n}^k) \rightarrow P(H_{n}^k)$$

where $v \in V_1$, $n \geq 0$, $n \geq k > 0$ and $^{+v}(A)$ is the smallest set such that

$$^{+v}(A) = A \cup ^{+v}(A) \ast_v A$$
In order to simplify the notation, $^+V(A)$ is written $A^+V$ in this paper. The following theorem establishes the domain of the operation.

**Theorem 52:**

If $n \geq k > 0$, $V \in V_k$ and $A \subset H^n_k$, then

$A^+V \subset H^n_k$

**Proof:**

Note that the smallest solution to the equation

$$A^+V = A U A^+V$$

is the set defined by the following inductive definition

1. $A^+_0 = A$.

where

$$A^+V = U_{j \geq 0} A^+_j$$

Therefore, proof by induction on $j$:

**Basis:** If $j = 0$, then

$$A^+_0 = A \subset H^n_k$$

**Induction step:** Assume that if $j \geq 0$ then

$$A^+_j \subset H^n_k$$

Note that
but $A^+ = A \cup A^* \cup A^+$ by theorem 51. Therefore,

$$A^+ = H_n$$

and

$$A_j = H_n$$

for all $j \geq 0$ by induction. Since $V^+ = \cup_{j \geq 0} A_j^+$ and all of the $A_j^+$ are subsets of $H_n$ the theorem is proved.

At this point, the definition for regular expressions may be given.

Definition 48:

The set of regular expressions over $\Sigma$ of level $n$ and degree $k$ with auxiliary symbols $V$, denoted $R_n^k(\Sigma, V)$, is the smallest set such that

(a) $R_n^{n+1} = \Sigma$

if $n \geq 0$.

(b) $R_n^0 = R_1^1$

if $n \geq 0$.

(c) $R_n^k = R_n^{k+1} \cup X_k U V_k U R_n^{k+1} \cup R_n^{k+1} R_n^k [k R_n^{k-1} \cup R_n^{k+1} R_n^k] U R_n^{k+1} R_n^k$

$$U R_n^{k+n} V_k U U_{n \geq i > 0} (R_n^{k+1} R_n^i)$$

if $n \geq k > 0$. 


Table 4 gives some examples of regular expressions.

As was the case with hyperforests, proofs using regular expressions are difficult because of the expression $R^{k+1}_n [k R^{k-1}]^k$. For this reason, a second definition is added with an induction variable similar to the depth in hypertrees.
Definition 49:

The set of regular expressions over $\mathcal{I}$ of level $n$, degree $k$ and depth $m$ with auxiliary symbol $V$, denoted $R^k_n(\mathcal{I}, V)$, is the smallest set such that

(a) $R^{n+1}_{n,m} = \mathcal{I}$
if $n \geq 0$ and $m \geq 0$.
(b) $R^k_{n,0} = X_k \cup V_k$
if $n \geq k > 0$.
(c) $R^k_{n,m+1} = R^{k+1}_{n,m} \cup R^k_{n,m} \cup R^{k+1}_{n,m+1} \cup R^{k-1}_{n,m} \cup R^k_{n,m+1} \cup R^k_{n,m} \cup R^k_{n,m+1} \cup (R^k_{n,m} \cup \Sigma^{<}_n(n,m))$
if $n \geq k > 0$ and $m \geq 0$.
(d) $R^0_{n,m} = R^1_{n,m}$
if $n \geq 0$ and $m \geq 0$.

Theorem 53:

If $n \geq 0$ and $n+1 \geq k \geq 0$, then

$R^k_n(\mathcal{I}, V) = \cup_{m \geq 0} R^k_{n,m}(\mathcal{I}, V)$

Proof:

This proof follows the corresponding proof of theorem 7 in chapter 2, and will not be given here.
Definition 50:

The language function of level \( n \) and degree \( k \), denoted \( \text{language}_n^k \), is defined such that

\[
\text{language}_n^k : R_n^k \rightarrow P(H_n^k)
\]

where

(a) \( \text{language}_{n+1}^n(A) = \{A\} \)
if \( A \in \mathcal{I} \) and \( n \geq 0 \).

(b) \( \text{language}_0^n(A) = \text{language}_n^1(A) \)
if \( n \geq 0 \).

(c) \( \text{language}_n^k(A) = \text{language}_n^{k+1}(A) \)
    \hspace{1cm} if \( A \in R_n^{k+1} \).
    \hspace{1cm} = (A)
    \hspace{1cm} if \( A \in X_k \cup V_k \).

\[
\text{language}_n^k(A) = \text{language}_n^{k+1}(\alpha)[k \text{ language}_n^{k-1}(\beta) k]
\]
    \hspace{1cm} if \( A = \alpha[k \beta k] \).

\[
= \text{language}_n^k(\alpha) \cup \text{language}_n^k(\beta)
\]
    \hspace{1cm} if \( A = \alpha + \beta \).

\[
= (\text{language}_n^k(\alpha))^{*v}
\]
    \hspace{1cm} if \( A = \alpha^{*v} \).

\[
= \text{language}_n^k(\alpha) \ast_v \text{language}_n^i(\beta)
\]
    \hspace{1cm} if \( A = \alpha \ast_v \beta \) and \( v \in V_i \).

if \( n \geq k > 0 \).
Definition 51:

A regular expression, $A \in R_n^k$, is a normal regular expression if

$$
\text{language}_n^k(A) \subseteq H_n^k(\Sigma)
$$


If no confusion will result, the $k$ and $n$ in $\text{language}_n^k$ will be elided.

As has been the case, a proof is now in order to show that the codomain described in Definition 50 is in fact correct.

Theorem 54:

If $n \geq 0$, $n+1 \geq k \geq 0$, $m \geq 0$ and $A \in R_n^k(\Sigma, \mathcal{V})$, then

$$
\text{language}(A) \subseteq H_n^k
$$

Proof by induction on the induction variable $m$:

Basis: If $m = 0$, then

1. If $k = n+1$, then $A \in \Sigma$ and

$$
\text{language}(A) = \{A\} \\
\subseteq H_n^{n+1}
$$

2. If $n \geq k > 0$, then $A \in X_k \cup V_k$ so that

$$
\text{language}(A) = \{A\} \\
\subseteq X_k \cup V_k \\
\subseteq H_n^k
$$

3. If $k = 0$, then
language(A) = language(A)

but by either case 1 or case 2,

language(A) \subseteq H_n^1

since H_n^1 = H_n^0,

language(A) \subseteq H_n^k

Therefore, the basis is proved.

Induction step: Assume that if m > m' \geq 0 and A \in R_{n,m}', then

language(A) \subseteq H_n^k

If A \in R_{n,m}', then one of the following cases holds:

1. If k = n+1, then A \in \Sigma and

   language(A) = \{A\}
   \subseteq H_n^k

2. If n \geq k > 0 and A \in R_{n,m-1}', then

   language(A) = language^k(A)
   but
   language^k(A) \subseteq H_n^{k+1}
   by induction hypothesis. Since H_n^{k+1} \subset H_n^k
   language(A) \subseteq H_n^k

3. If n \geq k > 0 and A \in R_{n,m-1}', then
language(A) \subseteq H_n^k

by induction hypothesis.

4. If \( n \geq k > 0 \) and \( A = \alpha[\beta] \), then

\[
\text{language}(A) = \text{language}^{k+1}_n(\alpha)[\text{language}^{k-1}_n(\beta)]
\]

but both \( \text{language}^{k+1}_n(\alpha) \subseteq H_n^{k+1} \) and
\( \text{language}^{k-1}_n(\beta) \subseteq H_n^{k-1} \) by induction hypothesis so

\[
\text{language}(A) \subseteq H_n^k
\]

5. If \( n \geq k > 0 \) and \( A = \alpha + \beta \), then

\[
\text{language}(A) = \text{language}^k_n(\alpha) + \text{language}^k_n(\beta)
\]

but, by induction hypothesis \( \text{language}^k_n(\alpha) \subseteq H_n^k \) and
\( \text{language}^k_n(\beta) \subseteq H_n^k \), therefore,

\[
\text{language}(A) \subseteq H_n^k
\]

6. If \( n \geq k > 0 \) and \( A = \alpha^* \), then

\[
\text{language}(A) = (\text{language}^k_n(\alpha))^*\n\]

but since \( \text{language}^k_n(\alpha) \subseteq H_n^k \) by induction hypothesis,

\[
\text{language}(A) \subseteq H_n^k
\]

by theorem 52.

7. If \( n \geq k > 0 \) and \( A = \alpha \star \beta \), then
language(A) = language^k_n(α) * _v language^i_n(β)

But language^k_n(α) c H^k_n and language^i_n(β) c H^i_n by
induction hypothesis. Therefore,

language(A) c H^k_n

by theorem 51.

8. If k = 0, then

language(A) = language^1_n(A)

by case 1 if n = 0 or cases 2 to 7 if n > 0,

language^1_n(A) c H^1_n

but

H^1_n = H^0_n

therefore,

language(A) c H^k_n

and the theorem is proved by induction.

If n = 1 and k = 1, then the definition of regular
expressions becomes

R^1_1 = R^2_1 U X_1 U V_1 U R^2_1[R^0_1] U R^1_1[R^1_1] U R^1_1[R^1*] U R^1_1[V_1]

notice that R^0_1 = R^1_1 so that
\[ R_1^1 = R_1^2 \cup X_1 \cup V_1 \cup R_1^2 \{ R_1^1 \} \cup R_1^1 + R_1^1 \cup R_1^{1*} V \cup R_1^{1*} R_1^1 \]

also, \( R_1^2 = \Sigma \) so

\[ R_1^1 = \Sigma \cup X_1 \cup V_1 \cup \Sigma \{ R_1^1 \} \cup R_1^1 + R_1^1 \cup R_1^{1*} V \cup R_1^{1*} R_1^1 \]

If \( V_1 = \{ \lambda \} \), then

\[ R_1^1 = \lambda \cup X_1 \cup \{ \lambda \} \cup \Sigma \{ R_1^1 \} \cup R_1^1 + R_1^1 \cup R_1^{1*} \lambda \cup R_1^{1*} R_1^1 \]

If it is decided arbitrarily that all elements of \( R_1^1 \) are to end in \( \lambda \), then (call this set "\( R_1^1 \)"

\[ R_1 = \{ \lambda \} \cup \Sigma \{ R_1^1 \} \cup R_1^1 + R_1^1 \cup R_1^{1*} \lambda \cup R_1^{1*} R_1^1 \]

Notice that \( \Sigma \{ R_1^1 \} \) describes the same set as \( \Sigma \{ R_1^1 \lambda \} \) \( R_1^1 \) so the later may be substituted for the former in a new set of regular expressions \( R_2 \)

\[ R_2 = \lambda \cup \Sigma \{ \lambda \} \cup R_2^1 + R_2^1 \cup R_2^{1*} \lambda \cup R_2^{1*} R_2 \]

Lastly, if a new set of regular expressions is defined, denoted \( R \), with the extraneous \( \lambda \)'s elided, and the \([1 \text{ and } 1]\) elided, then the equation becomes

\[ R = \lambda \cup \Sigma \cup R + R \cup R^{*} R \cup R^{*} R \]

which is exactly the set of regular expression over \( \Sigma \) in the classical string case.
If \( n = 2 \) and \( k = 2 \) (the case of trees over \( \Sigma \)), then the definition of regular expressions becomes

\[
R_2^2 = R_2^3 \cup X_2 \cup V_2 \cup R_2^3 \cup \left\{ R_2^1 \right\} \cup R_2^2+R_2^2 \cup R_2^{2^*v} \cup R_2^{2^*v} R_2^2
\]

But \( R_2^3 = \Sigma \) so this becomes

\[
R_2^2 = \Sigma \cup X_2 \cup V_2 \cup \left\{ R_2^1 \right\} \cup R_2^2+R_2^2 \cup R_2^{2^*v} \cup R_2^{2^*v} R_2^2
\]

If a new set of regular expressions are defined with only elements of \( \Sigma \) on the frontier, and \( V_2 \) is set equal to \( \Sigma \), then the resulting regular set, \( R_1 \), is

\[
R_1 = \Sigma \cup \Sigma \left\{ R_2^1 \right\} \cup R_1^+R_1 \cup R_1^{*v} \cup R_1^{*v} R_1
\]

where \( v \in \Sigma \). If it is assumed that \( R_2^1 \) can only take a limited form, in particular, any element of \( \Sigma \) can only have a string of a fixed number of sub-expressions after it, and if this number is designated by ranking \( \Sigma \), then the equation becomes

\[
R_2 = \Sigma \cup \Sigma \left\{ R_2^i \right\} \cup R_1^+R_1 \cup R_1^{*v} \cup R_1^{*v} R_1
\]

where \( v \in \Sigma \). The first two terms can be used to build up any tree. Both of these first two terms can be described by a finite set of trees and concatenation, these may be replaced with the set of trees over \( \Sigma \), \( T_{\Sigma} \).

\[
R = T_{\Sigma} \cup R+R \cup R^{*\Sigma} \cup R^{*\Sigma} R
\]
is the resulting regular expressions, which are the regular expressions described by the Thatcher and Wright.

In order to prove that a normal regular expression exists that corresponds to any given automaton, a new type of automaton will be defined.

Definition 52:

A partial finite automata, denoted $P_n^k$, is a five-tuple $(\Sigma, V, V, \delta, F)_n^k$ where $n \geq k > 0$ and

1. $\Sigma$ is a set of terminal symbols.
2. $V$ is a ranked set of argument symbols.
3. $V$ is a ranked set of states.
4. $\delta$ is a set of incomplete finite functions called the transition functions.

$$\delta_n^i: H_n^i(\Sigma, V U V) \rightarrow P(V_i)$$

where $n \geq i > 0$.

5. $F$ is a finite subset of $V_k$ called the set of final states.

Notice that if $V = \emptyset$ this corresponds directly to the automata defined in chapter 5. The language of $P_n^k$, denoted $\text{language}(P_n^k)$, is defined as in the standard automata given except that elements of $V$ are taken as terminal symbols in addition to $\Sigma$. An interesting theorem is
Theorem 55:
If \( n \geq k > 0 \) and \( P^k_n = (\Sigma, \Sigma', V, \delta, F)^k_n \) is a partial finite automata, then there exists a finite automata \( A^k_n = (\Sigma', \Sigma', \delta', F')^k_n \) such that
\[
\text{language}(A^k_n) = \text{language}(P^k_n)
\]

Proof:
Construct \( A^k_n \) as follows

1. \( \Sigma' = \Sigma \cup \Sigma \).
2. \( V_i' = V_i \cup V^i \) where \( V^i = \{ a_i \mid a \in \Sigma \} \).
3. The construction of \( \delta_i^0 \) is given below.
4. \( F' = F \).

The construction of the \( \delta_i^0 \) functions is as follows

1. If \( a \in H_{n+1,1}(\Sigma) \) and \( \delta_n^0(a) = P \), then
\[
\delta_n^0(a) = P
\]
2. If \( \delta_n^0(a_i [i, \beta_i]) = P \) and \( a \in V_{i+1} \), then
\[
\delta_n^0(a_i [i+1, \beta_i]) = P
\]
3. If \( \delta_n^0(a_i [i, \beta_i]) = P \) and \( \beta \in V_{i-1} \), then
\[
\delta_n^0(a_i [i, \beta_{i-1}]) = P
\]
4. If \( \delta_n^0(a_i [i, \beta_i]) = P \), \( a \in V_{i+1} \) and \( \beta \in V_{i-1} \), then
\[
\delta_n^0(a_i [i+1, \beta_{i-1}]) = P
\]
5. If $\delta^i_n(\alpha) = P$ and $\alpha \in \mathcal{V}_{i+1}$, then

$$\delta^i_n(\alpha_{i+1}) = P$$

6. If $\delta^i_n(\alpha) = P$ and $\alpha \in \mathcal{V}_i$, then

$$\delta^i_n(\alpha_i) = P$$

7. If $\alpha \in \mathcal{V}_i$, then

$$\delta^n_n(\alpha) = \{\alpha_n\}$$

and

$$\delta^j_n(\alpha_{i+1})' = \{\alpha_j\}$$

for all $n > j > 0$.

8. These are all the elements of $\delta$.

A simple inductive proof will show that

$$\text{language}(A_n^k) = \text{language}(F_n^k)$$

so that the theorem is proved.

---

It is now time to close the loop of equivalences. That is, to prove that regular sets, finite automata, and regular grammars all describe the same set of languages. The first theorem will establish that there is a normal regular expression for the language described by any finite automata.
Theorem 55:

If \( n > k > 0 \) and \( A_n = (\Sigma, \mathcal{V}, \delta, \mathcal{F})_n \) is a finite automaton, then there exists a normal regular expression \( B \in \mathbb{R}_n^k \) such that

\[
\text{language}(B) = \text{language}(A_n^k)
\]

Proof:

Note that if \( A_n^k = (\Sigma, \mathcal{V}', \delta, \mathcal{F})_n^k \) then

\[
\text{language}(A_n^k) = \text{language}(A_n^k)
\]

if \( \mathcal{V}' \) is the set of those elements of \( \mathcal{V} \) which occur in \( \delta \).

Therefore, there is no loss in generality by assuming the \( \mathcal{V} \) is finite. Since \( \mathcal{V} \) is finite, the elements of \( \mathcal{V} \) can be ordered. Therefore, \( \mathcal{V} \) may be represented as the set \( \{v_1, \ldots, v_m\} \). Define the partial finite automata

\[
Q_{j,T}^i = (\Sigma', \mathcal{V}', \mathcal{V}', \delta', \mathcal{F}')_n^k
\]

where \( m \geq i \geq 1, m \geq j \geq 1 \) and \( T = \{v_{i+1}, \ldots, v_m\} \) such that

1. \( \Sigma' = \Sigma \).
2. \( \mathcal{V}' = \mathcal{T} \).
3. \( \mathcal{V}' = \mathcal{V} - \mathcal{T} \).
4. \( \delta_n^i(a) = \delta_n^i(a) \cap \{v_1, \ldots, v_i\} \).
5. \( \mathcal{F}' = \{v_j\} \).

Define \( S_{j,T}^i = \text{language}(Q_{j,T}^i) \) if \( i > 0 \). If \( i = 0 \), then

\[
S_{j,T}^0 = \{a \mid v_j \in \delta_n^i(a) \text{ for some } i\}
\]

It can be easily seen that
To prove this theorem a slightly stronger proof will be presented.

Conjecture:

For all \( m \geq i \geq 0, m \geq j \geq 1 \) and \( T \subseteq \{v_{i+1}, \ldots, v_m\} \),

there is a regular expression for \( S_{j,T}^i \).

Proof by induction on \( i \):

Basis: If \( i = 0 \), then \( S_{j,T}^0 \) is a finite set for all \( j \) and \( T \)
so it has a regular expression.

Induction step: Assume that if \( i' < i \) then there is a
regular expression \( R_{j,T}^{i'} \) such that

\[
\text{language}(R_{j,T}^{i'}) = S_{j,T}^{i'}
\]

Let

\[
A = R_{j,T}^{i-1} \cup \{v_i\}
\]

\[
B = R_{i,T}^{i-1} \cup \{v_i\}
\]

and

\[
C = R_{i,T}^{i-1}
\]

A, B and C exist by induction hypothesis. It is claimed that

\[
S_{j,T}^i = A * v_i B^* v_i * v_i C
\]
This may be established by a simple but lengthy induction on
the induction variable for each element of \( S^i_{j,T} \) and again
for each element of the expression on the right of the equal
sign. Therefore, in particular, \( R^m_{j,\emptyset} \) exists were \( v_j \in F \). A
regular expression for the entire language is

\[
R^m_{j_1,\emptyset} + \ldots + R^m_{j_p,\emptyset}
\]

where

\[
F = \{ v_{j_1}, \ldots, v_{j_p} \}
\]

Now the last link in the loop of equivalences will be
proved.

**Theorem 57:**

If \( n \geq k > 0 \), \( m \geq 0 \) and \( B \in R^{k}_{n,m} \), then there is a
grammar \( G^k_n = (\Sigma', V', P', S') \) such that

\[
\text{language}(G^k_n) = \text{language}(B)
\]

**Proof by induction on** \( m \):

**Basis:** If \( m = 0 \), then \( B \in X_k \cup V_k \). Construct \( G^k_n \) as follows:

1. \( \Sigma' = \{ B \} \).
2. \( V_k = \{ S' \} \).
3. \( P' = \{ S' \rightarrow B \} \).
4. \( S' \) does not occur elsewhere.
Induction step: Assume that if \( m > m' > 0 \) and \( B \in R^k_{n,m} \), then there is a grammar \( G^k_n \) such that

\[
\text{language}_{n}^{k}(B) = \text{language}(G^k_n)
\]

If \( B \in R^k_{n,m} \), then one of the following cases is true:

1. If \( k = n \) and \( B \in R^k_{n,m-1} \), then \( B \in L \). Construct \( G^k_n \) as follows:
   1. \( R = \{B\} \).
   2. \( V' = \{S'\} \).
   3. \( P' = \{S' \rightarrow B\} \).
   4. \( S' \) does not occur elsewhere.

2. If \( n > k > 0 \) and \( B \in R^{k+1}_{n,m-1} \), then there is a grammar such that

\[
\text{language}((\Sigma_{\alpha}, \nu_{\alpha}, P_{\alpha}, S_{\alpha})_{n}^{k+1}) = \text{language}_{n}^{k+1}(B) = \text{language}_{n}^{k}(B)
\]

Construct the new grammar \( G^k_n \) such that

1. \( \Sigma' = \Sigma_{\alpha} \).
2. \( V_{i} = V_{i\alpha} \) if \( i \neq k \) and \( V_k = V_{k\alpha} \cup \{S\} \).
3. \( P' = P_{\alpha} \cup \{S' \rightarrow S_{\alpha}\} \).
4. \( S' \) does not occur elsewhere.

The language is given by

\[
\text{language}(G^k_n) = \{ \gamma | S^* \Rightarrow \gamma \}
\]

\[= \{ \gamma | S \Rightarrow S_{\alpha}^* \Rightarrow \gamma \} \]
3. If $n \geq k > 0$ and $B \in \mathbb{R}^k_{n,m-1}$, then there is a grammar $G^k_n$ such that

$$\text{language}^k_n(B) = \text{language}(G^k_n)$$

by induction hypothesis.

4. If $k = n$ and $B = \alpha[\beta \gamma]$, then there is a grammar $(\Sigma^*_{\beta}, V^*_{\beta}, P^*_{\beta}, S^*_{\beta})^n$ such that

$$\text{language}((\Sigma^*_{\beta}, V^*_{\beta}, P^*_{\beta}, S^*_{\beta})^n) = \text{language}^{k-1}_n(\beta)$$

by induction hypothesis. Construct the new grammar $G^k_n$ such that

1. $\Sigma' = \Sigma_{\beta} \cup \{\alpha\}$.
2. $V'_i = V_{i\beta}$ if $i \neq n$ and $V'_n = V_{n\beta} \cup \{S\}$.
3. $P' = P_{\beta} \cup \{S' \rightarrow \alpha[\beta S_{\gamma}]\}$.
4. $S'$ does not occur elsewhere.

In this case, the language of $G^k_n$ is given as

$$\text{language}(G^k_n) = \{y | S \Rightarrow^* y\}$$

$$= \{y | S \Rightarrow \alpha[\beta S_{\gamma}]\}$$

$$= \text{language}^{k+1}_n(\alpha)[\{\gamma_1 \mid S_{\beta} \Rightarrow^* \gamma_1\}]$$

$$= \text{language}^{k+1}_n(\alpha)[\text{language}^{k-1}_n(\beta)]$$

$$= \text{language}^k_n(\alpha[\beta])$$
Therefore, \( G_n^k \) is the required grammar.

5. If \( n > k > 1 \) and \( B = \alpha[k \beta k] \), then there are grammars such that

\[
language((I_a,V_a,P_a,S_a)_n^{k+1}) = language(\alpha)
\]

and

\[
language((I_\beta,V_\beta,P_\beta,S_\beta)_n^{k-1}) = language(\beta)
\]

where \( V_\alpha \cap V_\beta = \phi \). Construct the new grammar \( G_n^k \) such that

1. \( \Sigma' = \Sigma_\alpha \cup \Sigma_\beta \).
2. \( V_i' = V_{i\alpha} \cup V_{i\beta} \) if \( i \neq k \) and
   \( V_k' = V_{k\alpha} \cup V_{k\beta} \cup \{ S' \} \).
3. \( P' = P_\alpha \cup P_\beta \cup \{ S' \rightarrow S_\alpha[k \beta k] \} \).
4. \( S' \) does not occur elsewhere.

language(\( G_n^k \)) is the required language since

\[
language(G_n^k) = \{ \gamma | S^* \Rightarrow \gamma \}
\]

\[
= \{ \gamma | S \Rightarrow S_\alpha[k \beta k]^* \Rightarrow \gamma \}
\]

Since \( V_\alpha \) and \( V_\beta \) have no nonterminals in common, the derivation of \( S_\alpha \) must be independent of the derivation of \( S_\beta \). Therefore,

\[
language(G_n^k) = \{ \gamma_1 | S_\alpha \Rightarrow \gamma_1 \}[k \{ \gamma_2 | S_\beta \Rightarrow \gamma_2 \} k]
\]

\[
= language_n^k(\alpha)[k \text{ language}_n^k(\beta) k]
\]
If $B = \alpha \beta \gamma$, then there are grammars such that

$$\text{language}(\{Z, V, P, S\}) = \text{language}(\alpha)$$

and

$$\text{language}(\{Z, V, P, S\}) = \text{language}(\beta)$$

where $V \cap V' = \emptyset$. Construct the new grammar $G_1$ such that

1. $\Sigma' = \Sigma_a \cup \Sigma_b$.
2. $V_i' = V_{i\alpha} \cup V_{i\beta}$ if $i \neq 1$ and
   $V_1' = V_{1\alpha} \cup V_{1\beta} \cup \{S'\}$.
3. $P' = P_a \cup P_b \cup \{S' \rightarrow S_{\alpha}[1 \beta 1] \}$.
4. $S'$ does not occur elsewhere.

$\text{language}(G_1)$ is the required language since

$$\text{language}(G_1) = \{ \gamma | S^* \Rightarrow \gamma \}$$

$$= \{ \gamma | S \Rightarrow S_{\alpha}[k \beta k]^* \Rightarrow \gamma \}$$

Since $V_a$ and $V_b$ have no nonterminals in common, the derivation of $S_a$ must be independent of the derivation of $S_b$. Therefore,

$$\text{language}(G_1) = \{ \gamma_1 | S_{\alpha}^* \Rightarrow \gamma_1 \}[k \{ \gamma_2 | S_{\beta}^* \Rightarrow \gamma_2 \}]_k$$
7. If $n \geq k > 0$ and $B = \alpha + \beta$, then there are two grammars such that

$$\text{language}(\{\Sigma_\alpha', V_\alpha', P_\alpha, S_\alpha\})^k = \text{language}(\alpha)$$

and

$$\text{language}(\{\Sigma_\beta', V_\beta', P_\beta, S_\beta\})^k = \text{language}(\beta)$$

Choose these two grammars such that

$$V_\alpha \cap V_\beta = \emptyset$$

Construct the new grammar $G_n^k$ such that

1. $\Sigma' = \Sigma_\alpha \cup \Sigma_\beta$.
2. $V_i' = V_{i\alpha} \cup V_{i\beta}$ if $i \neq k$ and

$$V_k' = V_{k\alpha} \cup V_{k\beta} \cup S.$$  
3. $P' = P_\alpha \cup P_\beta \cup \{S' \rightarrow S_\alpha', S' \rightarrow S_\beta\}$
4. $S'$ does not occur elsewhere.

The language of $G_n^k$ is given by

$$\text{language}(G_n^k) = \{\gamma \mid S \Rightarrow^* \gamma\}$$

$$= \{\gamma_1 \mid S \Rightarrow \gamma_1 \} \cup \{\gamma_2 \mid S \Rightarrow \gamma_2 \}$$

$$= \text{language}_n^k(\alpha) \cup \text{language}_n^k(\beta)$$
If $n \geq k > 0$ and $B = \alpha^*\nu$, then there is a grammar such that

$$
\text{language}(G^k_n) = \text{language}(\alpha^*)^* \cup \text{language}(\nu^*)^* \cup \text{language}(B)
$$

Construct the new grammar $G^k_n$ such that

1. $\Sigma' = \Sigma$.
2. $V_i' = V_i$ if $i \neq k$ and $V_k' = V_k \cup \{\nu\}$.
3. $P' = P \cup \{\nu \rightarrow S, S' \rightarrow \nu\}$.
4. $S'$ does not occur elsewhere.

Notice that $\nu \in V_k'$, therefore, it must occur in the right spots to be legitimately a member of $V_k'$.

$$
\text{language}(G^k_n) = \{\gamma \mid S' \Rightarrow^* \gamma\}
$$

$$
= \{\gamma \mid S' \Rightarrow^* \nu^* \Rightarrow \gamma_1^* \Rightarrow \gamma \text{ where } \gamma_1 \in \text{language}^k_n(\alpha)\}
$$

$$
= \{\gamma_1^* \nu^* \Rightarrow \gamma_1^* \}^* \text{ language}(G^k_n) \cup \text{language}(\nu^*)^* \text{ language}(\alpha^*)^*
$$

$$
= \text{language}^k_n(\alpha)^* \nu \text{ language}(G^k_n) \cup \text{language}^k_n(\nu^*) \cup \text{language}^k_n(\alpha)^* \nu
$$

$$
= \text{language}(B)
$$
9. If \( n \geq k > 0 \) and \( B = \alpha \ast_v \beta \) where \( B \in \mathbb{N}_{n,m} \), then there are two grammars such that

\[
\text{language}\left(\left(\Sigma_\alpha, V_\alpha, \mathcal{P}_\alpha, S_\alpha\right)_n^k\right) = \text{language}_n^k(\alpha)
\]

and

\[
\text{language}\left(\left(\Sigma_\beta, V_\beta, \mathcal{P}_\beta, S_\beta\right)_n^j\right) = \text{language}_n^j(\beta)
\]

by induction hypothesis. Construct these such that

\[
V_\alpha \cap V_\beta = \emptyset
\]

Construct the new grammar \( G_n^k \) such that

1. \( \Sigma' = (\Sigma_\alpha - \{v\}) \cup \Sigma_\beta \).
2. \( V_i' = V_i \cup V_i \beta \) for all \( i \).
3. \( \mathcal{P}' = \{ \gamma \mid \gamma \in \mathcal{P}_\alpha \text{ and } \gamma = \gamma' \text{ with all } v's \}
\]
   replaced by \( S_\beta \} \cup \mathcal{P}_\alpha \).
4. \( S' = S_\alpha \).

Now

\[
\text{language}(G_n^k) = \{ \gamma \mid S_\alpha \ast \Rightarrow \gamma \}
\]

\[
= \{ \gamma \mid S_\alpha \ast \Rightarrow \gamma_1 \ast \Rightarrow \gamma \}
\]

where \( \gamma_1 \in \text{language}(\alpha) \) with all \( v's \) replaced by \( S_\beta \).

\[
\text{language}(G_n^k) = \{ \gamma_1 \mid S_\alpha \Rightarrow \gamma_1 \} \ast S_\beta \{ \gamma_2 \mid S_\beta \ast \Rightarrow \gamma_2 \}
\]

\[
= X \ast S_\beta \text{ language}_n^j(\beta)
\]
where $X$ is the set $\text{language}_n^{k}(\alpha)$ with all occurrences of $\nu$ on the frontier replaced by $S_{\beta}$. But in the concatenation given above, the resulting language would be the same if the $S_{\beta}$ were replaced by $\nu$ so

$$\text{language}(G_{n}^{k}) = \text{language}_n^{k}(\alpha) \ast_{\nu} \text{language}_n^{j}(\beta)$$

$$= \text{language}_n^{k}(\alpha \ast_{\nu} \beta)$$

$$= \text{language}(B)$$

Since assuming the induction hypothesis in each case allowed the proof that there is a $G_{n}^{k}$ such that

$$\text{language}(G_{n}^{k}) = \text{language}(B)$$

when $B \in R_{n,m}^{k}$, the lemma is proved by induction.

- - -

From the preceding chapters, it can be seen that there is a type of language called a regular language, and that these language may be described in any one of three ways, using grammars, automata or regular expressions. In the following chapters, these languages will be characterized, and also the related languages of the algebraic language hierarchy.
In this chapter, various characteristics of the hypertree hierarchy and their influence on the algebraic language hierarchy will be explored. These will include some closure properties for the hypertree hierarchy -- closure under union and intersection -- as well as a pumping lemma. These will imply closure under union for the algebraic language hierarchy, however it will be shown that this does not imply closure under intersection. Closure under intersection with regular sets is also shown. It will also be shown that not all context sensitive languages are in the algebraic language hierarchy. Lastly, it will be shown that the algebraic language hierarchy does, in fact, start out as designed: level 1 is the set of regular language, level 2 is the set of context free languages and level 3 is the set of macro languages.

Before this discussion can begin it is necessary to give some rather obvious formal definitions. These definitions will allow a formal definition of the algebraic language hierarchy just as was done for the hypertree hierarchy in chapter 2.
Definition 53:

If $G_n^k$ is a hypertree grammar, then the extracted language of level $k'$, denoted $\text{extract}_{k'}(G_n^k)$, is the set

$$\text{extract}_{k'}(G_n^k) = \{ y | y \in \text{language}(G_n^k) \text{ and } y = \text{extract}_{k'}(y) \}$$

also, if $A_n^k$ is a hypertree automaton the extracted language of level $k'$, denoted $\text{extract}_{k'}(A_n^k)$, is the set

$$\text{extract}_{k'}(A_n^k) = \{ y | y \in \text{language}(A_n^k) \text{ and } y = \text{extract}_{k'}(y) \}$$

and finally, if $R_n^k$ is a normal regular hypertree expression then the extracted language of level $k'$, denoted $\text{extract}_{k'}(R_n^k)$, is the set

$$\text{extract}_{k'}(R_n^k) = \{ y | y \in \text{language}(R_n^k) \text{ and } y = \text{extract}_{k'}(y) \}$$

Definition 54:

The string language for a hypertree grammar, automata or regular expression, denoted $\text{string}(A_n^k)$ or $\text{string}(R_n^k)$ respectively, is another name for the $\text{extract}_{1}$ function.
Definition 55:

If there exists a regular hypertree grammar \( G_n^{k'} \) such that

\[
\gamma = \text{extract}_k(G_n^{k'})
\]

then \( \gamma \) is a member of level \( n \) and degree \( k \) of the algebraic language hierarchy, denoted \( \text{ALH}_n^k \). These are all the elements of \( \text{ALH}_n^k \).

Corollary 58:

\[
\gamma \in \text{ALH}_n^k
\]

If and only if there exists a hypertree automata \( A_n^{k'}, \) a normal hypertree regular expressions \( R_n^{k'}, \) as well as a hypertree grammar \( G_n^{k'} \) such that

\[
\gamma = \text{extract}_k(A_n^{k'})
\]

\[
\gamma = \text{extract}_k(R_n^{k'})
\]

and

\[
\gamma = \text{extract}_k(G_n^{k'})
\]

Proof:

This follows directly from the definition.

---

The first set of theorems that will be presented are the closure theorems. A pair of closure theorems will be offered for level \( n \) of the hypertree hierarchy.
Theorem 59:

If $A_n^k$ and $B_n^k$ are regular hypertree languages, then there is a regular hypertree language $C_n^k$ such that

$$C_n^k = A_n^k \cup B_n^k$$

Proof:

This follows directly from the definition of regular sets.

---

Theorem 60:

If $A_n^k$ is a regular hypertree language and $B_n^k$ is another regular hypertree language, then there is a third regular hypertree language $C_n^k$ such that

$$C_n^k = A_n^k \cap B_n^k$$

Proof:

Construct a normal grammar for $A_n^k$ and $B_n^k$. Then, in each case, eliminate productions of the form $A \rightarrow B$ where $A \in V_i$ by forward substitution. Lastly, for each $X_i^j$, add a production of the form $A \rightarrow X_i^j$ where $A$ is a new nonterminal, and substitute $A$ in for $X_i^j$ wherever it occurs. Note that $A \in V_i$ in this case. Call the resulting grammars $G_n^k = (\Sigma, V, P, S)_n^k$ and $G_n^k' = (\Sigma', V', P', S')_n^k$ respectively.

Then, construct a new grammar $G_n^{k''} = (\Sigma'', V'', P'', S'')_n^k$ such that

1. $\Sigma'' = \Sigma \cap \Sigma'$
2. $V_k'' = V_k \times V_k'$
3. The construction for $P''$ is given below.
4. $S'' = (S, S')$

$P''$ is constructed as follows:

1. If $A \rightarrow a \in P$ and $B \rightarrow a \in P$ where $a \in \Sigma \cap \Sigma'$, then
   
   $$(A, B) \rightarrow a \in P''$$

2. If $A \rightarrow B \in P$ and $C \rightarrow D \in P'$ where $A \in V_i$, $B \in V_{i+1}$, $C \in V_i'$ and $D \in V_{i+1}'$, then
   
   $$(A, C) \rightarrow (B, D) \in P''$$

3. If $A \rightarrow X_i^j \in P$ and $B \rightarrow X_i^j \in P'$, then
   
   $$(A, B) \rightarrow X_i^j \in P''$$

4. If $A \rightarrow a[nB_n] \in P$ and $C \rightarrow a[nD_n] \in P'$ where $a \in \Sigma \cap \Sigma'$, then
   
   $$(A, C) \rightarrow a[n(B, D)_n] \in P''$$

5. If $A \rightarrow B[iC_i] \in P$ and $D \rightarrow E[iF_i] \in P'$, then
   
   $$(A, D) \rightarrow (B, E)[i(C, F)_i] \in P''$$

for all $n > i > 0$.

6. These are all the elements of $P''$.

If $\gamma \in \text{language}(G^n_k)$, then pick any derivation for $\gamma$ in $G^n_k$. It is easy to show by induction that a derivation for $\gamma$ in $G^n_k$ is obtained if each nonterminal in this derivation is
replaced by the corresponding nonterminal from \( V \), that is
the left hand projection of the nonterminal pair. A similar
inductive proof using the right hand projection of the pair
yields a derivation for \( Y \) in \( G_n^{k'} \). Therefore,

\[
\text{language}(G_n^{k''}) \subseteq \text{language}(G_n^k) \cap \text{language}(G_n^{k'})
\]

In order to prove containment the other way, a slightly
stronger proof will be presented. Namely, if \( A \in V_i \),
\( B \in V_i' \), \( A \Rightarrow^* \gamma \) and \( B \Rightarrow^* \gamma \) then

\[
(A,B) \Rightarrow^* \gamma
\]

This will be proved by induction on the length of the
derivation.

Basis: If the length of the derivations is one, then

\[
(A,B) \Rightarrow \gamma
\]

by construction.

Induction step: Assume that this is true for all derivations
of length less than \( m \). If either of the derivations is of
length \( m \), where \( m > 1 \), then

1. If \( \gamma \in R_n^{i+1} \), then

\[
A \Rightarrow C \Rightarrow^* \gamma
\]

and

\[
B \Rightarrow D \Rightarrow^* \gamma
\]
Since C $\Rightarrow \gamma$ and D $\Rightarrow \gamma$ are both of length less than m,

$$(C,D) \Rightarrow \gamma$$

But by construction $(A,B) \Rightarrow (C,D)$ so that

$$(A,B) \Rightarrow \gamma$$

2. $\gamma$ cannot be an $X^j_k$ since the derivation is longer than one.

3. If $\gamma = a[i \beta_i]$ and $n > i > 0$, then

$$A \Rightarrow C[i \ D \ i] \Rightarrow a[i \beta_i]$$

and

$$B \Rightarrow E[k \ F \ i] \Rightarrow a[i \beta_i]$$

where C $\Rightarrow \alpha$, D $\Rightarrow \beta$, E $\Rightarrow \alpha$ and F $\Rightarrow \beta$.

Therefore, by induction hypothesis $(C,E) \Rightarrow \alpha$ and $(D,F) \Rightarrow \beta$ and by construction

$$(A,B) \Rightarrow (C,E)[i(D,F)_i]$$

Therefore,

$$(A,B) \Rightarrow \gamma$$

and the hypothesis is true in this case also.

4. If $\gamma = a[n \beta_n]$, then

$$A \Rightarrow a[n \ D \ n] \Rightarrow a[n \beta_n]$$
and

\[ B \Rightarrow a[k \ F \ i] \Rightarrow a[i \ \beta \ i] \]

where \( D \Rightarrow \beta \) and \( F \Rightarrow \beta \). Therefore, by induction hypothesis \((D,F) \Rightarrow \beta \) and by construction

\[(A,B) \Rightarrow a[i(D,F)i]\]

Therefore,

\[(A,B) \Rightarrow \gamma \]

and the hypothesis is true in this case also.

Therefore, since these are the only possible cases, the hypothesis is proved by induction.

If \( \gamma \in \text{language}(G^k_n) \) and \( \gamma \in \text{language}(G^k_n)' \), then \( S \Rightarrow \gamma \) and \( S' \Rightarrow \gamma \), therefore, by the conjecture,

\[(S,S') \Rightarrow \gamma\]

and \( \gamma \in \text{language}(G^k_n)^" \), so the theorem is proved.

---

A corollary to theorem 59 is

Corollary 61:

If \( L_1 \) and \( L_2 \) are two languages from \( \text{ALH}^k_n \), then there is a third language \( L_3 \) such that \( L_3 \in \text{ALH}^k_n \) and

\[ L_3 = L_1 \cup L_2 \]
Proof:
Construct the union of the corresponding hypertree grammar.
This gives $L_3$ as its extracted language.

A similar corollary to theorem 60 is not given because it is not true. Consider the languages

$$\{a^n b^n c^m\}_{n \geq 1, m \geq 1}$$

and

$$\{a^m b^n c^n\}_{n \geq 1, m \geq 1}$$

Both of these are context free and, therefore, in $ALH^1_2$.
However, the intersection of these two languages is the language

$$\{a^n b^n c^n\}_{n \geq 1}$$

which is not context free. A proof of closure under intersection with regular sets will now be presented.

Theorem 62:
If $L \in ALH^k_n$ and $M \in ALH^k_{k'}$, then

$$L \cap M \in ALH^k_n$$

Proof:
Construct a normal grammar $G_n^1 = (\Sigma, V, P, S)_n^1$ for $L$ and another
grammar $G_k^1' = (\Sigma', V', P', S')_k^1$ for $M$. In each case, construct
the grammar such that $X$s only appear in productions of the
form $A \rightarrow X^j_k$ and there are no productions of the form $A \rightarrow B$
where $A$ and $B$ are in $V_i$ for some $i$. Construct
$G_n'' = (\Sigma'', V'', P'', S'')_n^1$ as follows:

1. $\Gamma'' = \Gamma \cap \Gamma'$.

2. $V_i'' = V_i X V_i' X N X P(\forall X V X N)$ for all
$k \geq i > 0$ where $\forall$ is the set of all the $X$s used in $G_n^1$
and $N$ is the set $\{1...n\}$.

3. The construction of $P''$ is given below.

4. $S''$ does not occur elsewhere.

Construct $P''$ as follows:

1. If $A \rightarrow a \in P$ and $A' \rightarrow a \in P'$ where $a \in \Sigma \cap \Sigma'$, then

$(A, A', n, a) \rightarrow a \in P''$

for all $a \in \forall X V X N$.

2. If $A \rightarrow X^j_k \in P$ and $A' \rightarrow X^j_k \in P'$ (Note that this
implies that $k' \leq k$), then

$(A, A', k', a) \rightarrow X^j_k \in P''$

for all $a \in \forall X V X N$.
3. If $A \rightarrow x^j_k \in P$ and $k' \geq k$, then

$$(A, X, I, a) \rightarrow x^j_k \in P'$$

for all $X \in V_{k'}^i$, $\alpha \in V \times V' \times N$ such that

$$(X^j_{k'}, X, I) \in \alpha.$$  

4. If $A \rightarrow B \in P$ and $A' \rightarrow B' \in P'$ where $A \in V_i$, $B \in V_{i+1}'$, $A' \in V_i'$ and $B' \in V_{i+1}'$ for some $k \geq i > 0$, then

$$(A, A', I, a) \rightarrow (B, B', I, a) \in P'$$

for all $\alpha \in V \times V' \times N$ and $I \in N$.

5. If $A \rightarrow B \in P$ where $A \in V_i$ and $B \in V_{i+1}$ for some $n \geq i > k$, then

$$(A, X, I, a) \rightarrow (B, X, I, a) \in P'$$

for all $X \in V_{k'}^i$, $\alpha \in V \times V' \times N$ and $I \in N$.

6. If $A \rightarrow a[n \cdot B \cdot n] \in P$ and $A' \rightarrow a[n \cdot B' \cdot n]$ where $a \in \Sigma^*_n$, $B \in V_i$ and $B' \in V_{n-1}'$, then

$$(A, A', I, a) \rightarrow a[n \cdot (B, B', I, a) \cdot n] \in P'$$

for all $n > I > 0$. Also,

$$(A, A', n-1, a) \rightarrow a[n \cdot (B, B', n, a) \cdot n] \in P'$$

for all $\alpha \in V \times V' \times N$. Note that in this case $n = k$.  

7. If \( A \rightarrow a[n \ B \ n] \in P \) and \( n > k \), then
\[
(A, X, I, \alpha) \rightarrow a[n \ (B, X, I, \alpha) \ n] \in P''
\]
for all \( n > I > 1 \). Also,
\[
(A, X, n-1, \alpha) \rightarrow a[n \ (B, X, n, \alpha) \ n] \in P''
\]
for all \( X \in V_k' \) and \( \alpha \in V \times V' \times X \times N \).

8. If \( A \rightarrow B[k', C k',] \in P \) and \( A' \rightarrow B'[k', C' k'] \in P' \), then
\[
(A, A', I, \alpha) \rightarrow (B, B', J, \alpha)[k', (C, C', K, \alpha) \ k'] \in P''
\]
for all \( J \in N, K \in N \) and \( \alpha \in V \times V' \times X \times N \) such that \( I = \min(k', K, J) \).

9. If \( A \rightarrow B[k', C k',] \in P \) and \( n > k' > k \), then
\[
(A, X, I, \alpha_A) \rightarrow (B, X, I, \alpha_B)[k', E_{k'-1}(C, P_C, \alpha_C) \ k'] \in P''
\]
for all \( k' > I > 0, X \in V_k' \) and \( \alpha_A, \alpha_B \) and \( \alpha_C \) are subsets of \( V \times V' \times X \times N \) such that
\[
\alpha_B = (\alpha_A - X_{k'} X V' X N) \cup \{(X_j^{j'}, C, I) \mid (j, C, I) \in P_C\}
\]
\[
\alpha_C = \alpha_A
\]
\[
P_C = \{j \mid X_j^j \in V \} \times V' \times X \times N
\]

The function \( E \) which returns a set is defined below.
\[
(A, X, k', \alpha) \rightarrow (B, X, I, \alpha)[k', (C, X, J, \alpha) \ k'] \in P''
\]
for all $\alpha \in V \times V' \times N$, $n \geq I > k'$ and $n \geq J > k'$.

$$(A, X, J, \alpha) \rightarrow (B, X, I, \alpha)[k'] (C, X, J, \alpha) [k'] \in P'$$

for all $\alpha \in V \times V' \times N$, $n \geq I \geq k'$ and $k' \geq J > 0$.

10. $S^* \rightarrow (S, S', I, \emptyset) \in P'$ for all $I \in N$.

11. These are all the elements of $P'$.

E is a set of finite functions $\{E_{k''}\}_{n \geq k'' > 0}$ such that $E_{k''}(A, p, \alpha)$ is the smallest set such that

1. If $A \rightarrow B[k'' C k''] \in P$, then

$$E_{k''+1}(B, p, B, \alpha)[k'' E_{k''-1}(C, p, C', \alpha) [k''] \subseteq E_{k''}(A, p, \alpha)$$

for all $\alpha \in V \times V' \times N$, $p_B$ and $p_C$ such that

$$p = p_B \cup \{(k'' j, C', I) \mid (j, C, I) \in P_C\}$$

$$p_B \subseteq P^{k''+1}_i \times V' \times N$$

and

$$p_C \subseteq P^{k''-1}_i \times V' \times N$$

such that if $(j, C, I) \in p_B \cup p_C$ then $j$ is a suffix of a path in $V \cap X_i$.

2. If $A \rightarrow B \in P$, then

$$E_{k''+1}(B, p, \alpha) \subseteq E_{k''}(A, p, \alpha)$$

where $A \in V_{k''}$ and $B \in V_{k''+1}$ for all $p \in P^{k''+1}_i \times V' \times N$ such that if $(j, C, I)$ is in $p$ then $j$ is the suffix of a path in $V \cap X_i$. 
3. If \( C \in V_k^n \), then

\[
E_{k^n}(C, \phi, \alpha) = \{(C, X, I, \alpha) \mid X \in V' \text{ and } I \in N\}
\]

and

\[
E_{k^n}(C, \{(\lambda, C', I')\}, \alpha) = \{(C, C', I', \alpha)\}
\]

for all \( C' \in V_{k''} \) if \( k'' \leq k \) or \( C' \in V_k' \) if \( n \geq k'' > k \), all \( I' \in N \) and \( \alpha \subseteq V \times V' \times N \).

Note that if \( (A, A', I, \alpha) \in V'' \) and \( (A, A', I, \alpha) \rightarrow^* \beta \) then

\( A \rightarrow^\ast \beta \) in \( G_n^1 \) and \( A' \rightarrow^\ast \) extract\(_k\)(\( \beta' \)) where \( \beta' \) is the same as \( \beta \) except that each \( X_i^j \) that references a subhypertree that is not contained in \( \beta \) is replaced by \( C' \) where \( (X_i^j, C', I) \in \alpha \).

Also, due to the construction if \( (X_i^j, C', I) \in \alpha \) then during the frontiering operation \( X_i^j \) will be replaced with a hypertree that was generated from a nonterminal \( (C, C', I, \alpha) \) for some \( C \) and \( \alpha \). Also, the \( I \) represents the level of the first frontiering operation for \( \beta' \) that will return a single element of \( I \). Each of these fact can be established by simple induction and, therefore, \( G_n^{n''} \) is a grammar for \( L \cap M \) and the theorem is proved.

...  

This last theorem suggests another normal form which may be used to simplify some proofs. This normal form is called "completed" since no \( X \) will ever copy just a piece of the subhypertree generated by a given nonterminal. That is, each subhypertree remains "complete".
Definition 56:

A grammar \( G_n^k \) is said to be a completed grammar if during any extract operation on any element of \( \text{language}(G_n^k) \) a piece of the structure generated by any nonterminal is copied then all of the structure generated from that nonterminal is copied.

The following theorem proves this to be a useful concept.

Theorem 63:

If \( G_n^k = (\Sigma, \mathcal{V}, \mathcal{P}, \mathcal{S})_n^k \) is a regular grammar, then there is a completed grammar \( G_n^k' = (\Sigma', \mathcal{V}', \mathcal{P}', \mathcal{S}')_n^k \) such that

\[
\text{language}(G_n^k) = \text{language}(G_n^k').
\]

Proof:

Construct \( G_n^k \) such that it is a normal grammar and there are no productions of the form \( A \rightarrow B \) where \( A \) and \( B \) are in \( \mathcal{V}_i \) for some \( i \). Also, construct \( G_n^k \) such that all \( X \)s are in productions of the form \( A \rightarrow X_i^j \). Construct \( G_n^k' \) as follows:

(Note that this construction is quite similar to that used in theorem 62.)

1. \( \Sigma' = \Sigma \).
2. \( \mathcal{V}_i' = \mathcal{V}_i \times N \times \mathcal{P}(\mathcal{V} \times N) \) where \( \mathcal{V} \) is the set of all \( X \)s that are used in \( G_n^k \) and \( N \) is the set \( \{1...n\} \).

Also, \( S' \in \mathcal{V}_k \).
3. The construction of \( \mathcal{P}' \) is given below.
4. \( S' \) does not occur elsewhere.
Construct $P'$ as follows:

1. If $A \to a \in P$ where $a \in \Sigma$, then

$$(A,n,a) \to a \in P'$$

for all $\alpha \in \mathcal{V} \times \mathbb{N}$.

2. If $A \to X^a_k, \in P$, then

$$(A,I,\alpha) \to X^a_k \in P'$$

for all $\alpha \in \mathcal{V}$ such that $(X^a_k,I) \in \alpha$.

3. If $A \to B \in P$, then

$$(A,I,\alpha) \to (B,I,\alpha) \in P'$$

for all $\alpha \in \mathcal{V} \times \mathbb{N}$.

4. If $A \to a[A_n B_n] \in P$, then

$$(A,I,\alpha) \to a[A_n (B,I,\alpha)] \in P'$$

for all $n > I > 0$ and $\alpha \in \mathcal{V} \times \mathbb{N}$. Also,

$$(A,n-1,\alpha) \to a[N_n (B,n,\alpha)] \in P'$$

5. If $A \to B[k' C_{k'}] \in P$ and $n > k' > 0$, then

$$(A,I,\alpha_A) \to (B,I,\alpha_B)[_{k'} E_{k'-1}(C,P_C,\alpha_C)]_{k'} \in P'$$

for all $k' > I > 0$, $\alpha_A$, $\alpha_B$ and $\alpha_C \in \mathcal{V} \times \mathbb{N}$ such that
\[ a_B = (a_A - X_{k'} \times N) \cup \{(X_{k'}^j, i) \mid (j, i) \in P_C\} \]
\[ a_C = a_A \]
\[ P_C \subseteq \{j \mid X_{k'}^j \in V\} \times N \]

The function \(E\) is defined below.

\[(A, k', \alpha) \rightarrow (B, I, \alpha) \{k' \mid (C, J, \alpha) \} \in P' \]

for all \(\alpha \in V \times N\), \(n \geq I \geq k'\) and \(n \geq J > k'\).

\[(A, J, \alpha) \rightarrow (B, I, \alpha) \{k' \mid (C, J, \alpha) \} \in P' \]

for all \(\alpha \in V \times N\), \(n \geq I \geq k'\) and \(k' \geq J > 0\).

6. \(S' \rightarrow (S, I, V) \in P'\) for all \(I \in N\).

7. These are all the elements of \(P'\).

\(E\) is a set of finite functions \(\{E_{k''}\}_{n \geq k'' > 0}\) such that

\(E_{k''}(A, p, \alpha)\) is the smallest set such that

1. If \(A \rightarrow B[k'' \mid k'' \leq n] \in P\), then

\[ E_{k''+1}(B, p_B, \alpha) \{k'' \mid E_{k''-1}(C, p_C, \alpha) \} \subseteq E_{k''}(A, p, \alpha) \]

for all \(\alpha\), \(p_B\) and \(p_C\) such that

\[ p = p_B \cup \{(k'' j, i) \mid (j, i) \in P_C\} \]
\[ p_B \subseteq P_{k''+1} \times N \]

and

\[ p_C \subseteq P_{k''-1} \times N \]
such that if \((j,I) \in P_B \cup P_C\) then \(j\) is the suffix of a path in \(V \cap X_i\).

2. If \(A \cup B \in P\), then

\[
E_{k'' + 1}(B,p,a) \subseteq E_{k''}(A,p,a)
\]

where \(A \in V_{k''}\) and \(B \in V_{k'' + 1}\) for all \(p \in P_{i}^{k'' + 1} \times N\) such that if \((j,I) \in p\) then \(j\) is the suffix of a path in \(V \cap X_i\) and \(a \in V \times N\).

3. If \(C \in V_{k''}\), then

\[
E_{k''}(C,B,a) = \{(C,I,a)\}
\]

for all \(a \in V \times N\) and \(B = (\lambda) \cup \{\lambda \mid j \in P_{i}^{k''}, j \notin P_{i}^{k''+1}\}\) and \(j\) is the suffix of a path in \(V \cap X_i\) \(\times \{I\}\).

Notice that if \(B = \emptyset\) then the corresponding structure is dropped when the \(i^{th}\) level frontier is taken. If \(B = \{\lambda\}\), then the structure is copied when the \(i^{th}\) level frontier is taken. If \(B \neq \emptyset\) and \(B \neq \{\lambda\}\), then the \(i^{th}\) frontier is possibly not defined since this corresponds to possibly having an \(X\) which doesn't reference anything. As in the proof for theorem 62, note that \((A,I,a) \Rightarrow B\) if and only if \(A \Rightarrow B\) and the set of \(Xs\) which reference anything outside of \(B\) is a subset of \(a\). The number associated with the \(X\) is the level of frontier which first produces a singleton.
Also, the number associated with the nonterminal is the level of frontier that first transforms the resulting structure into a singleton. These facts can be established by induction on the length of the production. Therefore, $S \Rightarrow^* \beta$ if and only if $(S, I, V) \Rightarrow^* \beta$ for some $I$ so that

$$\text{language}(G^k_n) = \text{language}(G^k'_n)$$

It could also be established that $G^k_n$ is complete since the $E$ functions force nonterminals to be expanded until all the paths that are possible are included in one production. Therefore, the theorem is proved.

---

Corollary 64:

If $L \in \mathcal{ALH}_n^k$, then there is a grammar

$$G^k_n' = (\Sigma', V', P', S')$$

for $L$ such that if

$$a \in \text{language}(G^k_n')$$

then $\text{extract}_k(a)$ exists.

Proof:

Construct the completed grammar for $L$ as in theorem 63 except replace step 3 of the construction of the $E$ function with

3. If $C \in V_n^k$ and $k > k'' > 0$, then

$$E_{k''}(C, \beta, a) = \emptyset$$
for all \( \alpha \in \mathcal{V} \times \mathcal{N} \) and \( \beta \in (\{\lambda\} \cup \{j \mid j \in P^k_i \), \( j \in P^{k+1}_i \) and \( j \) is the suffix of a path in \( \mathcal{V} \cap X_i \}) \times \{I\} \) where \( \beta \neq \phi \) and \( \beta \notin \{\lambda\} \times \mathcal{N} \).

4. If \( C \in \mathcal{V}_k \) and \( n > k \), then

\[
E^{(C,\phi,\alpha)}_k = \{(C,I,\alpha) \mid I \in \mathcal{N}\}
\]

and

\[
E^{(C,\{\lambda\},I,\alpha)}_k = \{(C,I,\alpha)\}
\]

The meaning of the ordered triples is exactly the same as in the theorem. If an \( X \) were generated in the original grammar which would force the desired extract operation to be undefined, then the element could not be generated by \( G_n^{k'} \), since ultimately a call to \( E^{(C,\beta,\alpha)}_k \) would be made in the construction of \( G_n^{k'} \) to generate the required productions, but this is equal to \( \phi \) so the needed productions would not be generated.

---

Theorem 65:

For any language, \( L \in A L R_n^{k'} \), there is a grammar \( G_{n'}' = (\Sigma',\mathcal{V}',P',S') \) such that \( V_i = \phi \) for all \( n > i \geq k \) and

\[
L = \text{extract}^k(G_{n'}')
\]

Proof:
Construct a completed grammar $G^k_n = (\Sigma, V, P, S)_n^k$ for $L$.

Construct $G^k_n$ as follows:

1. $\Sigma' = \Sigma \cup \{\ast\}$.
2. $V_i' = \bigcup_{n \geq i \geq k} V_i$, $V_i' = \emptyset$ if $n > i \geq k$ and $V_i' = V_i$ if $k > i > 0$.
3. The construction for $P'$ is given below.
4. $S' = S$.

Construct $P'$ as follows:

1. If $v \rightarrow \alpha \in P$ and $v \in V_i$ where $k > i > 0$ or $i = n$, then
   
   $v \rightarrow \alpha \in P'$

2. If $v \rightarrow \alpha \in P$ and $v \in V_i$ where $n > i \geq k$, then
   
   $v \rightarrow \ast[n \ast[i-1 \ast[i+1 \ast \ldots \ast[i+l \ast \ldots \ast[n-l \ast n]] n] \in P'$

A simple inductive proof would show that

$L = \text{extract}_{k}(G^k_n)$

---

Corollary 66:

If $L \in ALH_{n}^l$, then there is a grammar

$G^k_n = (\Sigma, V, P, S)_n^k$ such that $V_i = \emptyset$ if $i \neq n$ and

$L = \text{extract}_{1}(G^k_n)$

Proof:

Substitute 1 into theorem 65.
Corollary 57:

If $L \in \mathcal{AL}_{n}^{k}$ and $G_{n}^{k'} = (\Sigma, V, P, S)^{k'}$ is a grammar for $L$ where $k' > k$, then there is a grammar $G_{n}^{k} = (\Sigma', V', P', S')^{k}$ such that

$L = \text{extract}_{k}(G_{n}^{k})$

Proof:

This follows directly from the theorem.

---

Theorem 68:

If $L \in \mathcal{AL}_{n}^{k}$ and $G_{n}^{k'} = (\Sigma, V, P, S)^{k'}$ is a grammar for $L$, then there is a grammar $G_{n}^{k'} = (\Sigma', V', P', S')^{k'}$ such that

$\left[ k' \left[ k-1 \ldots \left[ k'' \left[ L \right] \right] \ldots \left[ k-1 \right] \right] \right]_{k} = \text{extract}_{k}(G_{n}^{k})$

where $k'+1 \geq k'' \geq 1$.

Proof:

Construct $G_{n}^{k'}$ as follows:

1. $\Sigma' = \Sigma \cup \{\ast\}$.
2. $V_{i}' = V_{i}$ if $n > i > 0$. $V_{n}' = V_{n} \cup \{S'\}$.
3. $P' = P \cup \{S' \rightarrow \ast_{n} \ldots \ast_{k''} \ast_{k-1} \ldots \ast_{k} \}$.
4. $S'$ does not occur elsewhere.

A simple induction over $k$ will prove the theorem.
Note that these theorem and corollaries allow \( L \) to be treated as though there were a grammar \( G^k_n \) for it if \( L \in \text{ALH}^k_n \) in most cases.

The following theorem will show that in fact it is not necessary to consider all the possible \( X \)s which might occur in a grammar. But first it is necessary to define a subset of the \( X \)s that will be proved to be sufficient in most cases.

Definition 57:

The set of linear paths of level \( n \), denoted \( \text{LP}_n \), is defined as

\[
\text{LP}_n = \{(n-1))^n
\]

Definition 58:

The set of linear \( X \)s of level \( n \), denoted \( \text{LX}_n \), is defined as

\[
\text{LX}_n = \{X^n_n | j \in \text{LP}_n\}
\]

Definition 59:

A grammar \( G^k_n \) is said to be linear to level \( i \) if \( X^j_i, \) appears in \( G^k_n \) and \( i' > i \) implies that \( X^j_{i'} \in \text{LX}_i \).
Definition 60:

A grammar $G_n^k$ is said to be linear if it is linear to level 1.

Theorem 69:

If $L \in ALH_n^k$, then there is a linear grammar to level $k+1$, $G_n^k = (\Sigma, V, P, S)^{k'}$, such that

$L = extract_k(G_n^k)$

Proof:

Construct a completed grammar $G_n^k = (\Sigma', V', P', S')^k$ for $L$ as was done in theorem 63. Construct $G_n^{k'}$ as follows:

1. $\Sigma = \Sigma' \cup \{\ast\}$.
2. $V = V'$.
3. The construction for $P$ is given below.
4. $S = S'$.

Order the $X$s at each level greater than $k$. Call the first $X_k''(0)$, the second $X_k''(1)$, and so forth for each $n \geq k'' > k$.

Construct $P$ as follows:

1. If $A \to B \in P'$ where $B \in V' \cup \Sigma$, then

   $A \to B \in P$

2. If $A \to a[n\alpha_n] \in P'$ where $a \in \Sigma$, then

   $A \to a[n\alpha_n] \in P$
3. If \( A \rightarrow B[i] \alpha[i] \in \mathcal{P}' \) and \( n \geq i > k \) where \( B \in \mathcal{V}_{i+1}' \), \( B = (X,I,\beta) \) and \( i \geq I > 0 \), then

\[
A \rightarrow B[i] C_0[i-1] C_1[i-1] \cdots C_m[i-1] \in \mathcal{P}
\]

where \( C_j = * \) if \( (X_i(j),I) \) is not in \( \beta \) for any \( I \), and if \( (X_i(j),I) \) is in \( \beta \) then \( C_i \) is the nonterminal from \( \alpha \) that is lead to by the path in \( X_i(j) \). If \( n \geq I > i \), then

\[
A \rightarrow *[i] \alpha[i] \in \mathcal{P}
\]

4. If \( A \rightarrow B[i] \alpha[i] \in \mathcal{P}' \) and \( k \geq i > 0 \), then

\[
A \rightarrow B[i] \alpha[i] \in \mathcal{P}
\]

5. If \( A \rightarrow X_i^j \in \mathcal{P}' \), then

\[
A \rightarrow X_i(i-1)^m \in \mathcal{P}
\]

where \( X_i^j = X_i(m) \).

Simple induction would show that these two grammars define the same language.

\[\cdots\]

Corollary 70:

There is a linear grammar for all string languages in the algebraic language hierarchy.

Proof:
Substitute 1 for \( k' \) in theorem 69. Note that \( X_1^\lambda \) is linear and the only \( X \) that can appear in strings.

Corollary 71:

There is a linear grammar for all tree languages in the algebraic language hierarchy.

Proof:

Substitute 2 for \( k \) in theorem 69. Note that only \( X_1^\lambda \) and elements of \( X_2 \) can appear on trees. Both of these types of \( X \)s must be linear.

Corollary 72:

If \( L \in \text{ALH}_n^3 \) and \( \text{frontier}_{2}^{0}(L) \) exists for each element of the language \( L \), then there is a linear grammar for \( L \).

Proof:

Substitute 3 for \( k' \) in theorem 69. Note that if \( X_3^j \) is in \( \alpha \in L \) then \( \text{frontier}_{2}^{0}(L) \) is not defined since ultimately this will require that \( \text{frontier}_{2}^{3}(X_3^j) \) be evaluated and this frontier is not defined.

Previously it was mentioned that context free languages where in \( \text{ALH}_2^1 \). So far, this has not been established.
formally, so this will be shown in one of the next three theorems. This correspondence is not direct, though, in that the hypertree languages have some extraneous symbols — the Xs and the brackets. It will be assumed, therefore, that the brackets and level 1 Xs will be deleted from the sentences to arrive at the resulting language. Note that no level 2 or higher Xs can occur.

Theorem 73:

Subject to the above constraints, $L \in \text{ALH}^1_1$ if and only if there is a regular grammar for $L$.

Proof:

Construct a normal regular hypertree grammar $G^1_1 = (\Sigma, \mathit{V}, \mathit{P}, S)^1$ for $L$. Then, construct the regular grammar $G' = (\Sigma', \mathit{V}', \mathit{P}', S')$ such that

1. $\Sigma' = \Sigma$.
2. $\mathit{V}' = \mathit{V}_1$.
3. $\mathit{P}'$ is given below.
4. $S' = S$.

Construct $\mathit{P}'$ as follows:

1. If $A \rightarrow a \in \mathit{P}$ where $A \in \mathit{V}_1$ and $a \in \Sigma$, then

   $A \rightarrow a \in \mathit{P}'$

2. If $A \rightarrow B \in \mathit{P}$, then

   $A \rightarrow B \in \mathit{P}'$
3. If $A\to a[B_1] \in P$, then

$$A \to aB \in P'$$

4. If $A\to a[X_1^1] \in P$, then

$$A \to a \in P'$$

The correspondence is easy to show via induction on the length of the derivation. The proof of containment in the other direction is similar.

---

Theorem 74:

Subject to the above constraints, $L = \text{ALH}^1$ if and only if there is a context free grammar for $L$.

Proof:

If $L$ is a context free language, construct a grammar $G = (E,V,P,S)$ for language with productions of the form

$$A \to BC$$
$$A \to a$$

where $A$, $B$ and $C$ are nonterminals and "a" is a terminal.

Construct $G_2' = (E',V',P',S')_2$ such that

1. $E' = E \cup \{\ast\}$.
2. $V_2' = V$, $V_1' = \emptyset$.
3. $P'$ is given below.
4. $S' = S$. 
$P'$ is constructed as follows:

1. If $A \rightarrow BC \in P$, then

   $$A \rightarrow *[\Sigma B[1C[1X_1^11_1^1]_2] \in P']$$

2. If $A \rightarrow a \in P$, then

   $$A \rightarrow a \in P'$$

A simple, though involved, induction will show these two grammars generate the same language.

If $L \in \text{ALH}_2^1$, then construct a normal grammar $G_2^2 = (\Sigma, V, P, S)_2$ for language subject to the constraints that $X$s only appear in production of the form $A \rightarrow X_j$ and there are no productions of the form $A \rightarrow B$ where $A$ and $B$ are either both from $V_1$ or both from $V_2$. Construct $G' = (\Sigma', V', P', S')$ as follows:

1. $\Sigma' = \Sigma$.

2. $V' = (V_1 \cup V_2) \times \{1,2,3\} \cup S'$

3. $P'$ is given below.

4. $S'$ does not occur elsewhere.

Construct $P'$ as follows:

1. If $A \rightarrow a \in P$, then

   $$(A,1) \rightarrow a \in P'$$

2. If $A \rightarrow a[2B_2] \in P$, then
(A,3)→(B,1) ∈ P'
(A,2)→(B,2) ∈ P'
(A,3)→(B,3) ∈ P'

3. If A→B ∈ P, then

(A,1)→(B,1) ∈ P'
(A,2)→(B,2) ∈ P'
(A,3)→(B,3) ∈ P'

4. If A→X^λ ∈ P, then

(A,2)→λ ∈ P'

5. If A→B[1C] ∈ P, then

(A,3)→(B,1)(C,1) ∈ P'
(A,2)→(B,1)(C,2) ∈ P'
(A,3)→(B,1)(C,3) ∈ P'
(A,3)→(B,2)(C,1) ∈ P'
(A,2)→(B,2)(C,2) ∈ P'
(A,3)→(B,2)(C,3) ∈ P'
(A,3)→(B,3) ∈ P'

6. {S'→(S,1), S'→(S,2), S'→(S,3)} ∈ P'.

7. These are all the elements of P'.

Note that productions of the form A → X^j_2 are dropped since if an X^j_2 appears in the resulting hypertree the frontier is
not defined. It can be easily seen that the two grammars
define the same language if it is noted that (A,1) *=> 'γ if
and only if A *=> γ and γ is one symbol only. (A,2) *=> γ
if and only if A *=> γ₁ and \text{frontier}^0(γ₁) = γX₁^{λ}. Also,
(A,3) *=> γ if and only if A *=> γ₁ and \text{frontier}^0(γ₁) = γ
(with no trailing \text{X}_1^{λ}). A simple inductive proof shows the
two languages are the same.

- • -

Theorem 75:

Subject to the above constraints, \( L \in \text{ALH}^1_i \) if and only
if there is an IO macrogrammar for \( L \).

Proof:
If \( L \) is an IO macrolanguage, then construct an IO
macrogrammar \( G = (I,F,V,\rho,S,\mathcal{P}) \) where \( \text{language}(G) = L \), \( I \) is
the set of terminals, \( F \) is the set of nonterminals, \( V \) is the
set of arguments, \( \rho \) is the ranking function for \( F \), \( S \) is the
start symbol and \( \mathcal{P} \) is the set of productions. Construct a
hypertree grammar \( G_3^3 = (E',V',\mathcal{P}',S')_3 \) as follows:

1. \( E' = I \cup \{\ast\} \)
2. \( V_1' = \emptyset \), \( V_2' = \emptyset \) and \( V_3' = F \).
3. \( \mathcal{P}' \) is given below.
4. \( S' = S \).

For each production, \( \alpha \to \beta \) in \( \mathcal{P} \), there is one production in
\( \mathcal{P}' \) constructed as follows:
1. The nonterminal in \( \alpha \) is the nonterminal on the left hand side of the production.

2. For each occurrence of an argument on the right side of the production, substitute an \( X \) such that, if it were the first argument, then substitute \( X^1 \), the second argument, \( X^2 \), the third argument \( X^3 \), etc.

3. Bracket each right hand side with an "\*[3" on the left and an "3]" on the right.

4. Change each ( to an [2 and each ) to an ]2.

5. Bracket each argument on the right hand side with an \*[2 on the left and an ]2 on the right.

6. Add \[1 and \]1 as needed to make the strings in the production legal. Add \( X^1 \) wherever it can legally be added.

For example, the grammar

\[
S \to A(1) \\
A(a) \to A(aa) \\
A(a) \to a
\]

yields the grammar

\[
S \to \*3A[2^*2^1[1 X^1 I 2][1 X^1 I 2]3] \\
A \to \*3A[2^*2^1 X^2^[1 X^1 I 1][1 X^1 I 1]2][1 X^1 I 2]3] \\
A \to \*3 X^2_3
\]
It can be seen by induction that these two grammars give the same language. Notice in the latter case that if the same argument is copied in twice that it must be copied in as the same thing since the hypertree is completely built before the frontier is taken.

If \( L \in \text{ALH}_3^1 \), then construct a completed grammar \( G_3^3 = (\Sigma, V, P, S)_3 \) for \( L \) as was done in theorem 63. Construct an IO macrogrammar \( G' = (\Sigma', F', V', \rho, S', P') \) as follows:

1. \( \Sigma' = \Sigma \).
2. \( F' = (S', D) \cup V \times \{1, 2\} \times \{1, 2, 3\} \times \{0, 1, 2, 3\}^m \).
3. \( V' = \{a_1, a_2, \ldots, a_m\} \) where each \( a_i \) does not appear elsewhere, and \( m \) is the maximum number of ones in a single path in the grammar.
4. \( \rho' \) maps all of \( V \) to \( m \), and \( S' \) to zero.
5. \( S' \) does not occur elsewhere.
6. \( P' \) is constructed as shown below.

Notice that for each nonterminal in this grammar there are two projections associated with the nonterminal and one projection for each possible \( X_2 \) that might exist in the grammar. The meaning of these projections are

1. The first projection:
   1. This nonterminal produces a singleton.
   2. This nonterminal does not produce a singleton.
2. The second projection:
1. This nonterminal produces a singleton after the first frontier operation.
2. This nonterminal produces a string which ends in an $X_1^\lambda$ after the first frontier operation.
3. This nonterminal produces a string which does not end in $X_1^\lambda$.

3. Each projection associated with a path: For each argument, 1, 2 and 3 mean the same as above except that the associated argument produces the string, etc. A zero means the associated argument is not to be used.

Therefore,

1. If $A\rightarrow a \in P$ where $A \in V_3$ and $a \in \Sigma$, then

$$(A, 1, 1, X_1 \ldots X_m)(a_1, a_2 \ldots a_m) \rightarrow a \in P'$$

for all $X_1 \ldots X_m$.

2. If $A\rightarrow a[3B_3] \in P$, then

$$(A, 2, 2, X_1 \ldots X_m)(a_1 \ldots a_m) \rightarrow (B, 2, 2, X_1 \ldots X_m)(a_1 \ldots a_m) \in P'$$

$$(A, 2, 1, X_1 \ldots X_m)(a_1 \ldots a_m) \rightarrow (B, 1, 1, X_1 \ldots X_m)(a_1 \ldots a_m) \in P'$$

$$(A, 2, 1, X_1 \ldots X_m)(a_1 \ldots a_m) \rightarrow (B, 2, 1, X_1 \ldots X_m)(a_1 \ldots a_m) \in P'$$

$$(A, 2, 3, X_1 \ldots X_m)(a_1 \ldots a_m) \rightarrow (B, 2, 3, X_1 \ldots X_m)(a_1 \ldots a_m) \in P'$$
for all $X_1 \ldots X_m$.

3. If $A \rightarrow B \in P$ where $A \in V_2$ and $B \in V_3$, then

$$(A, 1, 1, X_1 \ldots X_m)(a_1 \ldots a_m) \rightarrow (B, 1, 1, X_1 \ldots X_m)(a_1 \ldots a_m) \in P'$$

$$(A, 2, 1, X_1 \ldots X_m)(a_1 \ldots a_m) \rightarrow (B, 2, 1, X_1 \ldots X_m)(a_1 \ldots a_m) \in P'$$

$$(A, 2, 2, X_1 \ldots X_m)(a_1 \ldots a_m) \rightarrow (B, 2, 2, X_1 \ldots X_m)(a_1 \ldots a_m) \in P'$$

$$(A, 2, 3, X_1 \ldots X_m)(a_1 \ldots a_m) \rightarrow (B, 2, 3, X_1 \ldots X_m)(a_1 \ldots a_m) \in P'$$

for all $X_1 \ldots X_m$.

4. If $A \rightarrow X^j_2 \in P$ and $|j| = i$, then

$$(A, 2, X^j_1, X_1 \ldots X_m)(a_1 \ldots a_m) \rightarrow a_i \in P'$$

for all $X_1 \ldots X_m$ except $X_i = 0$. If $X_i = 0$, no production is generated. Note that the second projection must equal $X_i$.

5. If $A \rightarrow B[2C_2] \in P$ and

$$C = C_1[1C_2[1C_3 \ldots [1C_i 1] \ldots 1]_1]$$

where each $C_j \in V_2$ except $C_{m+1}$ which is an element of $V_1$ and $m+1 \geq i > 0$, then

$$(A, 2, 3, X_1 \ldots X_m)(a_1 \ldots a_m) \rightarrow (C_1, Z_1, Y_1, X_1 \ldots X_m)(a_1 \ldots a_m) \ldots$$
\[(C_j, Z_j, 3, X_1 \ldots X_m) \in P'\]

if \(i \geq j > 0\).

\[(A, 2, 2, X_1 \ldots X_m)(a_1 \ldots a_m) \rightarrow (C_1, Z_1', Y_1, X_1 \ldots X_m)(a_1 \ldots a_m) \ldots\]
\[(C_m', Z_1', 2, X_1 \ldots X_m)(a_1 \ldots a_m) \in P'\]

if \(B \Rightarrow a\) where \(a \in \Sigma\).

\[(A, 2, 2, X_1 \ldots X_m)(a_1 \ldots a_m) \rightarrow (C_1, Z_1', Y_1, X_1 \ldots X_m)(a_1 \ldots a_m) \ldots\]
\[(C_1', Z_1', Y_1, X_1 \ldots X_m)(a_1 \ldots a_m), D, \ldots D) \in P'\]
\[(A, 2, 2, X_1 \ldots X_m)(a_1 \ldots a_m) \rightarrow (B, 2, 2, Y_1 \ldots Y_m, 0 \ldots 0)(\ldots\)
\[(C_1', Z_1', Y_1, X_1 \ldots X_m)(a_1 \ldots a_m) \ldots) \in P'\]
\[(A, 2, 3, X_1 \ldots X_m)(a_1 \ldots a_m) \rightarrow (B, 2, 3, Y_1 \ldots Y_m, 0 \ldots 0)(\ldots\)
\[(C_1', Z_1', Y_1, X_1 \ldots X_m)(a_1 \ldots a_m) \epsilon P'\]

for all \(X_1 \ldots X_m, Y_1 \ldots Y_m\) and \(Z_1 \ldots Z_m\). Note that if \(i = m\) or \(m+1\) the sequence of zeros and \(D\)'s above is null.

6. If \(A \Rightarrow B \in P\) where \(A \in \mathcal{V}_1\) and \(B \in \mathcal{V}_2\), then

\[(A, 1, 1, X_1 \ldots X_m)(a_1 \ldots a_m) \rightarrow (B, 1, 1, X_1 \ldots X_m)(a_1 \ldots a_m) \in P'\]
\[(A, 2, 1, X_1 \ldots X_m)(a_1 \ldots a_m) \rightarrow (B, 2, 1, X_1 \ldots X_m)(a_1 \ldots a_m) \in P'\]
for all possible combinations of $X_1 \ldots X_m$.

7. If $A \rightarrow X_1 \in P$, then

$$(A, 2, 2, X_1, \ldots, X_m)(a_1, \ldots, a_m) \rightarrow (B, 2, 2, X_1, \ldots, X_m)(a_1, \ldots, a_m) \in P'$$

for all $X_1 \ldots X_m$.

8. If $A \rightarrow B[C_1] \in P$, then

$$(A, 2, 2, X_1 \ldots X_m)(a_1, \ldots, a_m) \rightarrow (B, 1, 1, X_1 \ldots X_m)(a_1, \ldots, a_m) \rightarrow (C, 2, 2, X_1 \ldots X_m)(a_1, \ldots, a_m) \in P'$$

$$(A, 2, 2, X_1 \ldots X_m)(a_1, \ldots, a_m) \rightarrow (B, 2, 1, X_1 \ldots X_m)(a_1, \ldots, a_m) \rightarrow (C, 2, 2, X_1 \ldots X_m)(a_1, \ldots, a_m) \in P'$$

$$(A, 2, 2, X_1 \ldots X_m)(a_1, \ldots, a_m) \rightarrow (B, 2, 2, X_1 \ldots X_m)(a_1, \ldots, a_m) \rightarrow (C, 2, 2, X_1 \ldots X_m)(a_1, \ldots, a_m) \in P'$$

$$(A, 2, 3, X_1 \ldots X_m)(a_1, \ldots, a_m) \rightarrow (B, 1, 1, X_1 \ldots X_m)(a_1, \ldots, a_m) \rightarrow (C, 1, 1, X_1 \ldots X_m)(a_1, \ldots, a_m) \in P'$$

$$(A, 2, 3, X_1 \ldots X_m)(a_1, \ldots, a_m) \rightarrow (B, 1, 1, X_1 \ldots X_m)(a_1, \ldots, a_m) \rightarrow (C, 2, 3, X_1 \ldots X_m)(a_1, \ldots, a_m) \in P'$$

$$(A, 2, 3, X_1 \ldots X_m)(a_1, \ldots, a_m) \rightarrow (B, 2, 1, X_1 \ldots X_m)(a_1, \ldots, a_m) \rightarrow (C, 2, 3, X_1 \ldots X_m)(a_1, \ldots, a_m) \in P'$$
\[(C,1,1,X_1 \ldots X_m)(a_1 \ldots a_m) \in P'\]

\[(A,2,3,X_1 \ldots X_m)(a_1 \ldots a_m) \rightarrow (B,1,1,X_1 \ldots X_m)(a_1 \ldots a_m)\]
\[(C,2,1,X_1 \ldots X_m)(a_1 \ldots a_m) \in P'\]

\[(A,2,3,X_1 \ldots X_m)(a_1 \ldots a_m) \rightarrow (B,2,1,X_1 \ldots X_m)(a_1 \ldots a_m)\]
\[(C,2,1,X_1 \ldots X_m)(a_1 \ldots a_m) \in P'\]

\[(A,2,3,X_1 \ldots X_m)(a_1 \ldots a_m) \rightarrow (B,2,1,X_1 \ldots X_m)(a_1 \ldots a_m)\]
\[(C,2,3,X_1 \ldots X_m)(a_1 \ldots a_m) \in P'\]

\[(A,2,3,X_1 \ldots X_m)(a_1 \ldots a_m) \rightarrow (B,2,3,X_1 \ldots X_m)(a_1 \ldots a_m)\]
\[(C,2,3,X_1 \ldots X_m)(a_1 \ldots a_m) \in P'\]

for all \(X_1 \ldots X_m\).

9. These are all the elements of \(P'\).

It can be shown by induction that these two grammars define the same language.

---

At this point, a pumping lemma will be introduced. This will be used eventually to prove that the algebraic language hierarchy is a true hierarchy, that is, it has an infinite number of distinct levels.

Before this is proven, the count function will be formally introduced as a means of determining the "length" of a hypertree.
Definition 61:

Count is a family of functions, denoted \( \text{Count}_n^k \), such that

\[
\text{Count}_n^k : H_n^k(\mathcal{I}, \mathcal{V}) \to \mathbb{N}
\]

where \( \mathbb{N} \) is the set of positive integers. The values for the count function are given by

(a) \( \text{Count}_n^{n+1}(\mathcal{I}) = 1 \) if \( n \geq 0 \).

(b) \( \text{Count}_n^0(\mathcal{I}) = \text{Count}_n^1(\mathcal{I}) \) if \( n \geq 0 \).

(c) \( \text{Count}_n^k(\mathcal{I}) = \text{Count}_n^{k+1}(\mathcal{I}) \) if \( \mathcal{I} \in H_n^{k+1} \).

\( \text{Count}_n^k(\mathcal{I}) = 1 \) if \( \mathcal{I} \in X_k \cup V_k \).

\( \text{Count}_n^k(\mathcal{I}) = \text{Count}_n^{k+1}(\mathcal{I}) + \text{Count}_n^{k-1}(\mathcal{I}) \) if \( \mathcal{I} = a[k\alpha,k] \) if \( n \geq k > 0 \).

Notice that count function returns the sum of the number of nodes in a hypertree, the number of Xs in the hypertree plus the number of variables in the given hypertree. Notationally, \( \text{Count}_n \) will refer to \( \text{Count}_n^0 \), count will refer to \( \text{Count}_n^0 \), where \( n' \) is larger than any level under consideration. In each case, a simple proof will show that the domain and codomain given is correct.
At this point, a pumping lemma will be given.

**Theorem 76:**

If $L$ is a regular hypertree language of level $n$ and degree $k$ and $j$ is a positive integer, then there exists an $N$ such that if $\gamma \in L$ and $\text{Count}(\gamma) > N$ then there exists $P_1 P_2 \ldots P_j$ such that $P_1 \in H_n^k$ and $P_2 \ldots P_j \in H_n^{k'}$ such that nonterminal $T$ occurs exactly once in $P_1 \ldots P_{j-1}$ and not all in $P_j$. Also, $T$ is the only nonterminal appearing, $\prod_{j \geq 1 > 1} \text{Count}(P_i)) < N$ and if $L'$ is the language with productions

$$S' \rightarrow P_1$$
$$T' \rightarrow P_i$$

for all $j \geq i > 1$ then $L' \subset L$ and

$$S' \Rightarrow \gamma_1 \Rightarrow \gamma_2 \ldots \Rightarrow \gamma_j$$

where $\gamma_j = \gamma$ and $\gamma_{i-1} \Rightarrow \gamma_i$ uses the $i^{th}$ production above.

**Proof:**
Assume $G_n^k = (\Sigma, V, P, S)_n^k$ is a normal grammar for $P$. Proceed as in the standard proof for a context sensitive except choose $N$ large enough that one of the nonterminals have to occur $j-1$ times.

---
Note that if j is taken as one or two the result is trivial (in fact, N could be taken as zero). If three is chosen for j, then in the string case this becomes the standard pumping lemma for regular sets. Also, in the tree case, this suggests the pumping lemma for context free sets.

To prove that the algebraic language hierarchy is a true hierarchy a language will be presented at each level which cannot belong to the next lower level. A few lemmas must be proven first.

Lemma 77:

If $A \in H_n^k$, then

$$\text{Count}(\text{frontier}_{n-1}^k(A)) < 2^{\text{Count}(A)}$$

Proof by induction on the induction variable:

Basis: If $m = 0$, then $\text{Count}(A) = 1$, and in any event $A = \text{frontier}_{n-1}^k(A)$.

The lemma then becomes $1 < 2^1$ which is obviously true.

Induction step: Assume that the lemma is true for all $m' < m$. Let $A \in H_{n,m}^k$. Then, one of the following is true:

1. If $k = n+1$, then the lemma is true as above.

2. If $k = n$ and $A = [\beta]_{n,n}$, then

$$\text{frontier}_{n-1}^n(A) = \text{frontier}_{n-1}^{n-1}(\beta)$$

and by the induction hypothesis.
Count(frontier_{n-1}^n(A)) = Count(frontier_{n-1}^{n-1}(\beta))
< 2^\text{Count(\beta)}

Also, notice that Count(A) = Count(\beta)+1, so that

Count(frontier_{n-1}^n(A)) < 2^\text{Count(A)}

3. If k = 0, then

\text{frontier}_{n-1}^0(\gamma) = \text{frontier}_{n-1}^1(\gamma)

Therefore,

\text{count(frontier}_{n-1}^0(\gamma)) < 2^{\text{count(\gamma)}}

by induction hypothesis.

4. If n > k > 0 and \gamma \in X_k, then

\text{frontier}_{n-1}^k(\gamma) = \gamma

so

\text{count(frontier}_{n-1}^k(\gamma)) < 2^{\text{count(\gamma)}}

since both count(frontier_{n-1}^k(\gamma)) and count(\gamma) are one.

5. If n > k > 0 and A \in H_{n,m-1}^n, then

\text{frontier}_{n-1}^k(A) = \text{frontier}_{n-1}^{n-1}(A)

and by the induction hypothesis

\text{Count(frontier}_{n-1}^k(A)) < 2^{\text{Count(A)}}

6. If n-1 > k > 0 and A = \alpha[k^\beta_k], then
\[
\text{frontier}_{n-1}^{k}(A) = \text{frontier}_{n-1}^{k+1}(\alpha)[\text{frontier}_{n-1}^{k-1}(\beta),]
\]

and by the induction hypothesis

\[
\text{Count}(\text{frontier}_{n-1}^{k}(A)) < 2^{\text{Count}(\alpha)} + 2^{\text{Count}(\beta)} < 2^{\text{Count}(\alpha)} \times 2^{\text{Count}(\beta)} < 2^{\text{Count}(A)}
\]

7. If \( k = n-1 \), \( A = \alpha[n-1 \beta n-1] \) and \( \alpha \notin \Sigma \), then

\[
\text{frontier}_{n-1}^{k}(A) = \text{apply}_{n-1}^{n-1}(\text{frontier}_{n-1}^{n}(\alpha), \text{frontier}_{n-1}^{n-2}(\beta))
\]

But by induction hypothesis

\[
\text{Count}(\text{frontier}_{n-1}^{n}(\alpha)) < 2^{\text{Count}(\alpha)}
\]

\[
\text{Count}(\text{frontier}_{n-1}^{n-2}(\beta)) < 2^{\text{Count}(\beta)}
\]

The worst possible case is if every symbol in \( \alpha \) copies all of \( \beta \). In this case, the count of the result is the product of the counts of the frontiers. That is

\[
\text{Count(\text{frontier}_{n-1}^{k}(A))} \\
\leq \text{Count(\text{frontier}_{n-1}^{n}(\alpha))} \times \text{Count(\text{frontier}_{n-1}^{n-2}(\beta))}
\]

\[
< 2^{\text{Count}(\alpha)} \times 2^{\text{Count}(\beta)}
\]

\[
< 2^{\text{Count}(\alpha)} + \text{Count}(\beta)
\]

But \( \text{Count}(A) = \text{Count}(\alpha) + \text{Count}(\beta) \) by definition, so
Count(frontier\(^k\)\(_{n-1}(A)\)) < 2^{\text{Count}(A)}

8. If \(k = n-1\), \(A = \alpha[n-1 \beta n-1]\) and \(\alpha \in \Sigma\), then

\[
\text{frontier}\(^k\)\(_{n-1}(A) = \alpha[n-1 \text{ frontier}\(^n-2\)\(_{n-1}(\beta) n-1]\)
\]

and by induction hypothesis

\[
\text{Count}((\text{frontier}\(^n-2\)\(_{n-1}(\beta)\)) < 2^{\text{Count}(\beta)}
\]

which means that

\[
\text{Count}(\text{frontier}\(^k\)\(_{n-1}(A)\)) = \text{Count}(\text{frontier}\(^n-2\)\(_{n-1}(\beta)\)) + 1
\]

\[< 2^{\text{Count}(\beta) + 1}
\]

But since it is known that \(\text{Count}(\beta) \geq 1\) it must be true that

\[
\text{Count}(\text{frontier}\(^k\)\(_{n-1}(A)\)) < 2^{\text{Count}(\beta) + 2^{\text{Count}(\beta)}}
\]

\[< 2^{\text{Count}(\beta) + 1}
\]

\[< 2^{\text{Count}(A)}
\]

Therefore, since in every case above

\[
\text{Count}(\text{frontier}\(^k\)\(_{n-1}(A)\)) < 2^{\text{Count}(A)}
\]

the lemma is true by induction.

- • -
This last lemma in connection with the pumping lemma will be used in a proof that there exists at least one language at each level of the hierarchy that is not in that hierarchy. In order to describe this language, a special function needs to be defined.

Definition 62:

E is a function such that

\[ E : \mathbb{N} \times \mathbb{N} \rightarrow \mathbb{N} \]

where \( \mathbb{N} \) is the set of positive integers. The values are given by the recursive definition

(a) \( E(0, m) = m \)

(b) \( E(n+1, m) = 2^{E(n, m)} \)

Note that \( E(0, m) = m, E(1, m) = 2^m \) etc. This will be used to define the language described above.

Lemma 78:

The language \( L = \{ a^x \mid x = E(n, m) \text{ for all } m \} \) where \( n \) is fixed cannot be in \( \mathcal{ALH}_k \) where \( k \leq n+1 \).

Proof:

Assume that \( L \in \mathcal{ALH}_k \) where \( k \leq n+1 \). Then, there must be a grammar \( G^k \) such that

\[ L = \text{extract}_1(G^k) \]
Let $L' = \text{language}(G_k^k)$. Without loss of generality, assume that $\text{extract}_1(\gamma)$ exists for all $\gamma \in L'$. Let $N$ be chosen as described in the pumping lemma for $j = 4$. Let $\gamma$ be the smallest hypertree such that $\text{extract}_1(\gamma) = a^{E(n,m)}$ and $a^{E(n,m)}$ is not the frontier of any hypertree with fewer than $N$ nodes. Note that $\text{Count}(\gamma) > N$. Therefore, there exist $P_1, P_2, P_3,$ and $P_4$ such that there is a sublanguage $L'$ with productions

\[
\begin{align*}
S' &\rightarrow P_1 \\
T &\rightarrow P_2 \\
T &\rightarrow P_3 \\
T &\rightarrow P_4
\end{align*}
\]

and $S' =\Rightarrow \gamma_1 =\Rightarrow \gamma_2 =\Rightarrow \gamma_3 =\Rightarrow \gamma$ by using the productions in order. Note that neither $P_2$ nor $P_3$ can be dropped by the frontier operation since if they were $S' =\Rightarrow P_1 =\Rightarrow \gamma_2' =\Rightarrow \gamma'$ where the dropped substructure is not used would result in a shorter element of $L'$ for $a^{E(n,m)}$. Therefore, consider the sequence of elements of $L'$ generated by

\[
S' =\Rightarrow P_1 =\Rightarrow \gamma_2'' =\Rightarrow \gamma_3'' =\Rightarrow \gamma''
\]

such that the first production is used the first time, followed by $i$ applications of the second rule, followed by the third and fourth rule. Clearly this is a subset of $L'$. 
Since $P_2$ does not drop $P_3$, this will yield an increasing sequence from $L$ such that the elements of the second sequence are arrived at by taking $extract_2$'s of the first sequence. Let the first sequence be represented by

$$\gamma_0, \gamma_1, \gamma_2, \ldots$$

where $\gamma_1 = \gamma$ and the second sequence be represented by

$$\gamma_0', \gamma_1', \gamma_2', \ldots$$

where $\gamma_i' = extract_2(\gamma_i)$ for all $i$. Then, form a third sequence

$$\gamma_0'', \gamma_1'', \gamma_2'', \ldots$$

such that $\gamma_i'' = \text{frontier}_1^0(\gamma_i')$. Also, clearly

$$\text{Count}(\gamma_i'') \geq E(n, m+i-1)$$

if $i > 1$ since this must be an increasing sequence in $L$. Also,

$$\text{Count}(\gamma_i') = \text{Count}(P_1) + i \times \text{Count}(P_2) + \text{Count}(P_3) + \text{Count}(P_4)$$

or in other words $\text{Count}(\gamma_i') = K_1 + K_2 \times i$ where $K_1$ and $K_2$ are constants. To perform the $extract_2$ requires $k-2$ frontier operations, so an obvious induction using lemma 77 would show that

$$\text{Count}(\gamma_i'') \leq E(k-1, K_1 + K_2 \times i)$$
The final tree frontier can only drop nodes -- that is, it cannot duplicate nodes -- so

\[ \text{Count}(\mathcal{L}_i^\prime) \leq \text{Count}(\mathcal{L}_i^\prime) \]

So it must be true that

\[ E(n,m+i-1) < E(k-1,K_1+K_2*i) \]

or, by another short induction

\[ E(n-K+1,m+i-1) < E(0,K_1+K_2*i) \]
\[ < K_1+K_2*i \]

But this last statement is clearly false since the left side of this inequality is exponential whereas the right side is linear. Therefore, the lemma must be true since assuming it is not true leads to a contradiction.

---

At this point, it will be shown that the language given in lemma 78 is in $ALH_{n+2}^1$. This will be done by inductively showing a grammar for the language.

Lemma 79:

The language given in lemma 78 is in $ALH_{n+2}^1$.

Proof by induction on $n$:

Basis: If $n = 0$, then
S → A
A → *[2a[1 A 1]2] | a

is a grammar for the language.

Induction step: Assume that a grammar $G^n_n$ exists for the
language at level $n-1$. Then, rotate this grammar 90
degrees. That is, increment every number in the grammar by
1 to form a new grammar for a "linear" tree. For example,
the above grammar would give

$$S → A$$
$$A → *[3a[2 A 2]3] | a$$

Next, for each "a" substitute


The "*[n...3]" is used to hold the structure
"*[3^[2^X^λ^2[1^X^λ[1^λ[1]2]3]3]3" together until the frontier $^0$ phase of the extract. During the frontier $^0$ phase, each $X^λ_2$
will pull in the entire structure following it which will create a full binary tree. All of this copying will result in a frontier of length $2^q$ where $q$ is the length of the corresponding element of the original grammar. Each element of the resulting string frontier would be $X^λ_2$ if it were
legal. Therefore, another alternation must be made to substitute an "a" for the $x_2^\lambda$ in each case. This is to add a new start symbol, $S'$, and the production

$$
S' \rightarrow \ast[n \ast[n-1\ldots\ast[3S_3][2a_2]_3]\ldots_n]
$$

Frontiering will keep this structure together until the frontier$^0_3$, when the "a" will be substituted for the $x_2^\lambda$ mentioned above. Each of the facts stated above could be established by an induction proof, and therefore, this altered grammar is the required grammar.

- - -

The previous two lemmas then make the proof of hierarchy easy.

Theorem 80:

$$
\text{ALH}^1_n \subset \text{ALH}^1_{n+1}
$$

and

$$
\text{ALH}^1_n \neq \text{ALH}^1_{n+1}
$$

for all $n > 0$.

Proof:

Clearly this is true for $n = 1$ (regular and context free language). At all other levels, containment is obvious, and the previous two lemmas give an example of language in each level that is not in the previous level.

- - -
Corollary 81:

\[ \text{ALH}_n^k \subset \text{ALH}_{n+1}^k \]

and

\[ \text{ALH}_n^k \neq \text{ALH}_{n+1}^k \]

for all \( n \geq k \geq 1 \).

Proof:

Containment is obvious. Suppose that

\[ \text{ALH}_n^k = \text{ALH}_{n+1}^k \]

for some \( n \) and \( k \). Then, it would be easy to show by induction that

\[ \text{ALH}_n^1 = \text{ALH}_{n+1}^1 \]

which contradicts the theorem.

---

The proof that all members of \( \text{ALH}_n^1 \) are context sensitive will be deferred until chapter 8 due to the length of the proof. It will be shown here, however, that \( U_{n>0} \text{ALH}_n^1 \) is not equal to the set of context sensitive language. This will be done by diagonalization of the above hierarchy of language. That is,

\[ \{ a_\gamma | \gamma = E(n,1) \}_{n>0} \]

is context sensitive, but not any level of the hierarchy.
Lemma 82:

The language

\[ L = \{ a_y \mid y = E(m,1) \text{ for } m > 0 \} \]

is not in \( \mathcal{ALH}_n^{1} \) for any \( n \).

Proof:

This proof parallels exactly the proof of lemma 78.

- - -

Lemma 83:

The language given above is context sensitive.

Proof:

This proof consists of demonstrating a linearly bounded grammar for the above language. Let \( G \) be the grammar

\[
\begin{align*}
S & \rightarrow <(>A<)> \\
A<) & \rightarrow RD<) > \\
AR & \rightarrow RD \\
<) R & \rightarrow <) A \\
AD & \rightarrow DAA \\
<) D & \rightarrow <) \\
A<) & \rightarrow a \\
Aa & \rightarrow aa \\
<)a & \rightarrow a
\end{align*}
\]

It is easy to show by induction that the above grammar generates the desired language. It can also easily be seen
that if \( \gamma \in \text{language}(G) \) then there is a derivation bounded by \(|\gamma| + 3\). Note that the grammar works by generating each successive element of the language as a string of As with \(<(>) and \(<>) around it. At each point, \( A<>) \) goes to \( RD<>) \) will go to the next element whereas "A<>)" goes to "a" will force the production to stop with a string of "a"'s. Note, the three extra symbols are the \(<(>, <)> and the D.

---

Theorem 84:

The set of languages \( \bigcup_{n>0}^{\infty} \text{ALH}_n^1 \) is not equal to the set of context sensitive languages.

Proof:

This follows directly from the last two lemmas.

---

This concludes the chapter on miscellaneous theorems about the hierarchies. Chapter 8 will be dedicated to showing that the algebraic language hierarchy is contained in the set of context sensitive languages.
CONTAINMENT IN CONTEXT SENSITIVE

This chapter will show that any language in the algebraic language hierarchy is, in fact, contained in the set of context sensitive languages. This will be done by "programming" a frontier taking algorithm using unrestricted grammars. It will be shown that this is a linearly bounded operation -- that is, it is linearly bounded in both its input and its output. After that, it will be shown that for any grammar, an equivalent grammar can be found which yields the same string language such that the intermediate frontiers are linearly bounded. This, then, will be a linearly bounded grammar for the resulting language, and, therefore, the language must be context sensitive.

Rather than treat the general case of hypertree grammars, only grammars which meet the following three restrictions will be handled:

1. The set of nonterminals contains only those nonterminals which appear in the productions.

2. The set of terminals contains only those terminals which appear in the productions.

3. If B is a nonterminal from $V_k$, then there is a $\psi$ such that $B \Rightarrow^* \psi$ and $\psi \in H_n^k$. That is, all nonterminals derive something.
It should be rather obvious that these "restrictions" are not restrictions at all, since a simple algorithm would generate a grammar following these restrictions from any grammar.

Throughout this chapter a special set, which is denoted $V$, is used. This is a finite set containing all the symbols from the initial hypertree grammar which are viewed as terminals in resulting grammar. This will include elements of $I$, brackets and $X$'s that occur in the initial hypertree grammar. It is finite since each production is of finite length and there are only a finite number of production. Also, throughout this development the symbols $n$, $k$ and $k'$ will be used. It assumed that these are integers such that $n' \geq n \geq 0$, $n \geq k > 0$ and $n+1 \geq k' \geq 0$ unless otherwise stated, where $n'$ is the level of the original grammars. That is, if the production

$$<A_l^{k}>_n \alpha <) \rightarrow \alpha$$

where $\alpha \in \{A,B\}$ is given, and the grammar given is of level 2, this corresponds to saying that

$$<A_l^{2}> A <) \rightarrow A$$

$$<A_l^{2}> B <) \rightarrow B$$

$$<A_l^{1}> A <) \rightarrow A$$
\[
\langle A_{12}^1 \rangle B \langle \rangle \rightarrow B \\
\langle A_{11}^1 \rangle A \langle \rangle \rightarrow A \\
\langle A_{11}^1 \rangle B \langle \rangle \rightarrow B 
\]

are all productions in the grammar.

Similarly in the "search" function, the symbol "j" appears. "j" is used to stand for any postfix of a path which occurs in V. It is assumed that any nonterminal or terminal which is presented in this grammar does not occur in the original grammar unless specifically stated. Therefore, given a grammar \((\Sigma, V, P, S)^n\), the additional productions needed to make this an unrestricted grammar are given according to the function which is to be performed by the productions. Each of these "functions" will be discussed in the following paragraphs. Note that this corresponds to writing a program in a particular applicative language.

The miscellaneous functions given in table 5 are self-evident and will not be discussed separately.

Table 6 gives an implementation of the "over" function. This function is called by placing the nonterminal \(\langle \text{over}_{n}^{k} \rangle\) on the left side of a string of characters with the \(\langle \text{over}_{n}^{k} \rangle\) preceded by a nonterminal from the set V'. The effect of the routine is to place that nonterminal on the right hand
TABLE 5. Miscellaneous Routines

**Dup:**
\[ \langle \text{Dup} \rangle \langle \rangle \rightarrow \langle \rangle \langle \rangle \]

**Blitz:**
\[ \langle \text{Blitz} \rangle \alpha \rightarrow \langle \text{Blitz} \rangle \]

\[ \text{if } \alpha \in V \cup V \cup \{ \langle \rangle , \langle , \rangle \} \rightarrow \langle \text{end} \rangle \]
\[ \langle \text{Blitz} \rangle \langle \text{end} \rangle \langle \rangle \rightarrow \lambda \]

**Switch:**
\[ \langle \text{Switch}_{k} \rangle \langle \rangle \rightarrow \langle \rangle _{k} \]

\[ \text{where } n' \geq k > 0 \]

**Die:**
\[ \langle \text{Die} \rangle \langle \rangle \rightarrow \langle \rangle \]

where \langle \text{Dup} \rangle, \langle \text{End} \rangle, \langle \text{Switch}_{k} \rangle \text{ and } \langle \text{Die} \rangle \text{ are elements of } V' \text{ (see the over routine)}

\begin{align*}
\text{side of a hypertree of level } n \text{ and degree } k. \text{ This is done by use of the terminal } \langle \text{post}^{k'}_{n} \rangle \text{ where the } k' \text{ and } n \text{ are always equal to the level and degree of hypertrees that the nonterminal and } \langle \text{over}^{k}_{n} \rangle \text{ have passed over. If at any point,}
\end{align*}
TABLE 6. The Over Routine

<table>
<thead>
<tr>
<th></th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>a.</td>
<td>$\beta&lt;\text{over}_n^k\alpha \rightarrow \alpha\beta&lt;\text{over}_n^k&lt;\text{past}_n^{n+1}&gt;$</td>
</tr>
<tr>
<td></td>
<td>$\text{if } \alpha \in \Psi$</td>
</tr>
<tr>
<td>b.</td>
<td>$\beta&lt;\text{over}_n^k&lt;\text{past}_n^1&gt; \rightarrow \beta&lt;\text{over}_n^k&lt;\text{past}_n^0&gt;$</td>
</tr>
<tr>
<td>c.</td>
<td>$\beta&lt;\text{over}_n^k&lt;\text{past}_n^{k+1}&gt; \rightarrow \beta&lt;\text{over}_n^k&lt;\text{past}_n^k&gt;$</td>
</tr>
<tr>
<td></td>
<td>$\beta&lt;\text{over}<em>n^k&lt;\text{past}<em>n^{k'}&gt;[</em>{k'} \rightarrow [</em>{k'} \beta&lt;\text{over}_n^k&lt;\text{P}_n^{k'}&gt;$</td>
</tr>
<tr>
<td></td>
<td>$\beta&lt;\text{over}_n^k&lt;\text{past}_n^{k''-1}&gt;&lt;\text{P}<em>n^{k'}&gt;&lt;</em>{k''}] \rightarrow _{k''}][\beta&lt;\text{over}_n^k&lt;\text{past}_n^{k'}&gt;$</td>
</tr>
<tr>
<td></td>
<td>$&lt;\text{P}_n^{k'}\alpha \rightarrow \alpha&lt;\text{P}_n^{k'}&gt;$</td>
</tr>
<tr>
<td></td>
<td>$\text{if } \alpha \in V U V U {&lt;},&lt;\text{frontier}_n^{k''}}$</td>
</tr>
<tr>
<td>(frontier)</td>
<td>$\beta&lt;\text{over}_n^k&lt;\text{frontier}_n^{k'}&gt; \rightarrow &lt;\text{frontier}_n^{k'}&gt;\beta&lt;\text{over}_n^k&lt;\text{P}_n^{k'}&gt;$</td>
</tr>
<tr>
<td></td>
<td>$\beta&lt;\text{over}<em>{n+1}^k&lt;\text{past}</em>{n+1}^{k'}&gt;&lt;\text{P}_n^{k'}&gt;&lt;&lt;) \rightarrow &lt;)\beta&lt;\text{over}_n^k&lt;\text{past}_n^{k'}&gt;$</td>
</tr>
<tr>
<td></td>
<td>$\beta&lt;\text{over}<em>{n+1}^k&lt;\text{past}</em>{n+1}^{k'}&gt;&lt;\text{P}_n^{k'}&gt;&lt;&lt;) \rightarrow &lt;)\beta&lt;\text{over}_n^k&lt;\text{past}_n^{n+1}&gt;&gt;$</td>
</tr>
<tr>
<td></td>
<td>$&lt;\text{P}_n^{k'}\alpha \rightarrow \alpha&lt;\text{P}_n^{k'}&gt;$</td>
</tr>
<tr>
<td></td>
<td>$\text{if } \alpha \in V U {&lt;\text{frontier}_n^{k''},&lt;}$</td>
</tr>
<tr>
<td>(finish)</td>
<td>$\beta&lt;\text{over}_n^k&lt;\text{past}_n^{k'}&gt; \rightarrow \beta$</td>
</tr>
</tbody>
</table>
TABLE 7. The Search Routine

<search\textsuperscript{k}_n>j \rightarrow <mark><setup\textsuperscript{k}_n>j>

<setup\textsuperscript{k}_n>j\alpha \rightarrow \alpha<setup\textsuperscript{k}_n>j
  \text{if } \alpha \in V \cup \{<frontier\textsuperscript{k'}_n>,<\}\}

<setup\textsuperscript{k}_n>j, \rightarrow \langle, \rangle, <Sl\textsuperscript{k}_n>j<Dup><over\textsuperscript{n-1}_n>

a. <Sl\textsuperscript{n}_n>j \rightarrow <move><S5><over\textsuperscript{n}_n>
  <move>\alpha \rightarrow \alpha<return>\alpha<move>
  \text{where } \alpha \in V \cup \{<frontier\textsuperscript{k'}_n>,<\}\}

\beta<return> \rightarrow \alpha<return>\beta
  \text{where } \beta \in V \cup \{<frontier>,<\>,<,>\}

<mark>\alpha<return> \rightarrow \alpha<mark>

<mark> \rightarrow \lambda

<move><S5><\rangle> \rightarrow \lambda

b. <Sl\textsuperscript{0}_n>j \rightarrow <Sl\textsuperscript{1}_n>j

c. <Sl\textsuperscript{k}_n>j \rightarrow <S2\textsuperscript{k}_n>j<over\textsuperscript{k+1}_n>
  \text{if } j = k_i

<S2\textsuperscript{k}_n>[k \rightarrow [k]<Sl\textsuperscript{k-1}_n>j<switch\textsuperscript{k}_n><over\textsuperscript{k-1}_n>

<S1\textsuperscript{k}_n>j \rightarrow <Sl\textsuperscript{k+1}_n>j<S3\textsuperscript{k}_n><over\textsuperscript{k+1}_n>
  \text{if } j \in P\textsuperscript{k+1}_n

<S3\textsuperscript{k}_n><\rangle> \rightarrow \langle\rangle

<S3\textsuperscript{k}_n>[k \rightarrow \langle][k<S4\textsuperscript{k}_n><over\textsuperscript{k-1}_n>

<S4\textsuperscript{k}_n>[k] \rightarrow \langle k\rangle

\text{where } <S2\textsuperscript{k}_n>, <S3\textsuperscript{k}_n>, <S4\textsuperscript{k}_n> \text{ and } <S5> \text{ are elements of } V'. 
the terminal $\text{past}^k_n$, indicating that a hypertree of level $n$ and degree $k$ has been passed over, an alternative production $\beta<\text{over}^k_n><\text{past}^k_n> \rightarrow \beta$ allows the routine to end with the $\beta$ deposited on the right end of a hypertree. Notice that the over function will work even if a frontier function, indicated by $\text{frontier}^k_n$, is included in the string without requiring the function to be evaluated. It is the responsibility of the calling routine to insure that the proper hypertree is chosen. For example, "$B<\text{over}^n>\text{a}[n\text{ a}^n_n]\text{]}$ could place the $B$ after either the "$a$ to yield "$aB[\text{a }n\text{ a}^n_n]\text{]}$ or after the "$n\text{]}$ to yield "$a[\text{a }n\text{ a}^n_n]B$" since both "$a$ and "$a[\text{a }n\text{ a}^n_n]\text{]}$ are legal hypertrees of level $n$. Quite often if the wrong hypertree is chosen the $B$ gets stranded and can never be eliminated from the resulting structure so that no output element is generated.

The search routine is given in table 7. It works on a string structure of the form

$$<\text{search}^k_n> \gamma_1 <> \gamma_2 <>$$

where $\gamma_1$ is an unknown string of elements of $\mathcal{V}$ and nonterminals, and $\gamma_2$ is the argument to the search. The reason for this arrangement is that the argument may have to be used by other search routine calls and, therefore, must be held independently of the calls. If the argument were
TABLE 8. The Apply Routine

\[
\begin{align*}
\langle \text{apply}_n^k \rangle & \rightarrow \langle A_2_n \rangle \langle A_1_n^k \rangle \langle A_4 \rangle \langle \text{over}_n^k \rangle \\
\langle A_2_n \rangle \alpha & \rightarrow \alpha \langle A_2_n \rangle \\
& \quad \text{if } \alpha \in \Sigma. \\
\langle A_2_n \rangle \langle , \rangle & \rightarrow \langle \text{Blitz} \rangle \langle \text{end} \rangle \langle \text{over}_n^{n-1} \rangle \\
\langle A_4 \rangle \langle , \rangle & \rightarrow \langle , \rangle \langle , \rangle \\
\text{a. } \langle A_1_n^{n+1} \rangle \alpha \langle \rangle & \rightarrow \alpha \\
& \quad \text{if } \alpha \in \Sigma. \\
\text{b. } \langle A_1_n^0 \rangle & \rightarrow \langle A_1_n^1 \rangle \\
\text{c. } \langle A_1_n^k \alpha \rangle & \rightarrow \alpha \\
& \quad \text{if } \alpha \in X_k \cap \Sigma. \\
\langle A_1_n^n \alpha \rangle & \rightarrow \langle \text{search}_{n-1} \rangle \\
& \quad \text{if } \gamma = X_j \text{ where } \gamma \in \Sigma. \\
\langle A_1_n^k \rangle & \rightarrow \langle A_1_n^{k+1} \rangle \langle A_3_n^k \rangle \langle \text{over}_n^{k+1} \rangle \\
\langle A_3_n^k \rangle \langle , \rangle & \rightarrow \langle , \rangle \\
\langle A_3_n^k \rangle & \langle k \rightarrow \langle , \rangle \langle k \rangle \langle A_1_n^{k-1} \rangle \langle \text{switch}_k \rangle \langle \text{over}_n^{k-1} \rangle \\
\end{align*}
\]

where \langle A_4 \rangle \text{ and } \langle A_3_n^k \rangle \text{ are in } V'.

\[\]

copied to be held adjacent to the \( \text{search}^{k^j}_n \), then the resulting grammar would not be linearly bounded. Beyond this the implementation is straightforward. The nonterminal \( \text{setup}^{k^j}_n \) is used to change the string structure to

\[
\langle \text{mark} \rangle \ y_1 \langle , \rangle \ <\text{Sl}^{k^j}_n \rangle \ y_2 \rangle \langle \rangle
\]

Then, the \( \langle \text{Sl}^{k^j}_n \rangle \) nonterminal is used to do the search that is requested. When the "j" is exhausted and \( \langle \text{Sl}^{k^j}_n \rangle \) is encountered, the search ends and the nonterminal \( \langle \text{move} \rangle \) is used to send the characters selected back to the \( \langle \text{mark} \rangle \). \( \langle \text{mark} \rangle \) can disappear, but if not all the characters are moved yet a nonterminal \( \langle \text{return} \rangle \) will get stranded. Note that \( \langle \text{frontier}^{k^j}_n \rangle \) and \( \langle \rangle \) are included in the characters that can be moved so that the frontier functions do not have to be evaluated before the search function is used.

The apply function is given in table 8. It works on the structure

\[
\langle \text{apply}^{k^j}_n \rangle \ y_1 \langle , \rangle \ y_2 \langle \rangle
\]

The initial productions transforms this into

\[
\langle A_2^\langle n \rangle \langle A_1^{k^j}_n \rangle \ y_1 \langle )\rangle \langle , \rangle \ y_2 \rangle \langle \rangle
\]

which should look something like the string required for the search routine. The nonterminal \( \langle A_1^{k^j}_n \rangle \) is used to do the
TABLE 9. The Frontier Routine

a. \( <\text{frontier}_{n+1}^n \alpha() > \rightarrow \alpha \)
   \( \text{if } \alpha \in \Sigma. \)
   \( <\text{frontier}_{n+1}^n \alpha[n+1] \rightarrow <\text{frontier}_{n}^n > <F_1^0 > <\text{over}_{n+1}^n > \)
   \( \text{if } \alpha \in \Sigma. \)
   \( <F_{1n}^n > [n+1] () ) \rightarrow () > \)

b. \( <\text{frontier}_0^n > \rightarrow <\text{frontier}_1^n > \)

c. \( <\text{frontier}_k^n \alpha() > \rightarrow \alpha \)
   \( \text{if } \alpha \in X_k \cap \Sigma. \)
   \( <\text{frontier}_{k+1}^n > <F_2^k > <\text{over}_{n+1}^n > \)
   \( \text{if } k < n. \)
   \( <F_{2k}^n() > \rightarrow () > \)
   \( <F_{2k}^n > [k \rightarrow (]> [k <\text{frontier}_{k-1}^n > \text{switch}_k <\text{over}_{n+1}^n > \)
   \( <\text{frontier}_{n}^n > \rightarrow <\text{frontier}_{n}^n > <\text{Die} > <\text{over}_{n+1}^n > \)
   \( <\text{frontier}_{n}^n > \alpha[n+1] \rightarrow \)
   \( <\text{apply}_{n}^n > <\text{frontier}_{n}^n > <F_3^0 > <\text{over}_{n+1}^n > \alpha[n+1] \)
   \( \text{if } \alpha \in \Sigma. \)
   \( <F_3^0 > [n \rightarrow ()]> <\text{frontier}_{n}^n > <F_4 > <\text{over}_{n}^n > \)
   \( <F_4 > [n() ] ) \rightarrow () > () > () > \)
   \( <\text{frontier}_{n}^n > \alpha[n \rightarrow \alpha[n <\text{frontier}_{n}^n > \text{switch}_n <\text{over}_{n}^n > \)
   \( \text{if } \alpha \in \Sigma. \)

where \( <F_1^0 > , <F_2^k > , <F_3 > > \text{ and } <F_4 > \text{ are in } \Sigma'. \)
actual work of the apply function, where the second argument is always separated as per the discussion of the search routine. The actual implementation is very simple.

Table 9 gives the productions for the frontier routine. This is an extremely straightforward implementation of this function, and need not be discussed here.

TABLE 10. The String Routine

\[
\begin{align*}
S' & \rightarrow <\text{string}_{n}, S>) \\
<string_{n}> & \rightarrow <\text{string}_{n-1}>, <\text{frontier}_{n-1}^{0}>, <\text{Dup}>^{0}, <\text{over}^{0}_{n}> \\
<string_{1}> & \rightarrow <S1>, <\text{over}^{1}_{1}> \\
<S1> & \rightarrow \lambda
\end{align*}
\]

where the original grammar is \((L,V,P,S)^{k}_{n}\) and

\[S'\] is the new start symbol.

String is a routine which is used to call the frontier function the required number of times to get the string frontier. The implementation of the string routine is given in table 10. At this point, the start symbol of the initial grammar is referenced so that the initial hypertree can be generated.
The proofs involving these routines require the use of the following data structure which is based on hypertrees.

Definition 63:

\( F_{n,m}^k \) is the smallest set such that

(a) \( F_{n,m}^{n+1} = \emptyset \)

if \( n \geq 0 \) and \( m \geq 0 \).

(b) \( F_{n,0}^k = X_k \)

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

(c) \( F_{n,m+1}^k = F_{n,m}^{k+1} \cup F_{n,m}^k \cup F_{n,m}^{n+1} [ F_{n,m}^{k-1} \]

\( \cup \text{frontier}^k_{n} F_{n+1,m}^k \)\)

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

(d) \( F_{n,m}^0 = F_{n,m}^1 \)

if \( n > 0 \) and \( m \geq 0 \).
Notice that if the frontier function works properly, and \( \gamma \in F_{n,m}^k \) then \( \gamma \Rightarrow \gamma' \) where \( \gamma' \in H_{n}^k \). The purpose of this data structure is to allow the definition of hypertrees where not all of the frontier functions have been evaluated.

In view of this definition, the following lemmas summarize the use of these functions.

**Lemma 85:**

If \( \gamma \in F_{n,m}^k \cap (V \cup \{<\text{frontier}_n^k,\langle\rangle>\})^* \) and \( \beta \in V' \), then

\[
\beta^{<\overline{\gamma}^k_n>} \Rightarrow \gamma \Rightarrow \gamma \beta^{<\overline{\gamma}^k_n<past'_n>}
\]

and there is a derivation of width less than or equal to \( 2|\gamma|+3 \).

**Proof by induction on m:**

**Basis:** If \( m = 0 \), then

\[
F_{n,0}^k = X_k
\]

and the derivation is given in table 11.

**Induction step:** Assume that if \( m' < m \) then

\[
\beta^{<\overline{\gamma}^k_n>} \Rightarrow \gamma \Rightarrow \gamma \beta^{<\overline{\gamma}^k_n<past'_n>}
\]

where \( \gamma \in F_{n,m'}^k \). If \( \gamma \in F_{n,m}^k \), then
TABLE 11. Over Derivation for $\gamma \in F_{n,0}^{k'}$

<table>
<thead>
<tr>
<th>Line</th>
<th>Width</th>
</tr>
</thead>
<tbody>
<tr>
<td>1. $\beta &lt;\overset{k}{\gamma}_n$</td>
<td>4</td>
</tr>
<tr>
<td>2. $\gamma \beta &lt;\overset{k}{\gamma}<em>n &lt;\text{past}</em>{n}^{k'}$</td>
<td></td>
</tr>
</tbody>
</table>

1. If $k' = n+1$, then $\gamma \in \Xi$ and the derivation is given in table 12.

TABLE 12. Over Derivation for $\gamma \in \Xi$

<table>
<thead>
<tr>
<th>Line</th>
<th>Width</th>
</tr>
</thead>
<tbody>
<tr>
<td>1. $\beta &lt;\overset{k}{\gamma}_n$</td>
<td>4</td>
</tr>
<tr>
<td>2. $\gamma \beta &lt;\overset{k}{\gamma}<em>n &lt;\text{past}</em>{n}^{n+1}$</td>
<td></td>
</tr>
</tbody>
</table>

2. If $n \geq k' > 0$ and $\gamma \in F_{n,m-1}^{k+1}$, then the derivation is given in table 13. Notice that in going from line 1 to line 2 the induction hypothesis had to be used.

3. If $n \geq k > 0$ and $\gamma \in F_{n,m-1}^{k}$, then the lemma is true by induction hypothesis.
TABLE 13. Over Derivation for $\gamma \in F_{n, m-1}^{k+1}$

<table>
<thead>
<tr>
<th>Line</th>
<th>Width</th>
</tr>
</thead>
<tbody>
<tr>
<td>1. $\beta &lt;\overline{k}\gamma_n$</td>
<td>$2</td>
</tr>
<tr>
<td>2. $\gamma\beta &lt;\overline{k}&lt;\text{past}\n^{k+1}_n$</td>
<td>$</td>
</tr>
<tr>
<td>3. $\gamma\beta &lt;\overline{k}&lt;\text{past}\n^{k'}_n$</td>
<td></td>
</tr>
</tbody>
</table>

4. If $n \geq k > 0$ and $\gamma \in F_{n, m-1}^{k+1}$, then the derivation is given in table 14. Notice in this table $|\gamma| = |\gamma_1|+|\gamma_2|+2$ therefore $2|\gamma|+3 = 2|\gamma_1|+2|\gamma_2|+7$.

5. If $n \geq k > 0$ and $\gamma \in \text{frontier}_n^{k}F_{n+1, m-1}^{k}$, then the derivation is given in table 15. Here, $2|\gamma|+3 = 2|\gamma_1|+7$.

6. If $k = n$ and $\gamma \in \text{frontier}_n^{k+1}F_{n+1, m-1}^{k+1}$, then the derivation would be similar to that given in table 15.

7. If $k = 0$, then the derivation is given in table 16. Notice that between lines 1 and 2 is a use of the derivation given for one of the previous parts of this induction step.

Note that in each case the maximum width is less than $2|\gamma|+3$ so that the lemma is proved.

- - -
Corollary 86:

If $\gamma \in F_{n,m}^{k} (\forall U (\langle \text{frontier}_{n}^{k}, \rangle)^{*} \text{ and } \beta \in \forall'$, then

$$\beta^{\overline{k}_{n}} \gamma \Rightarrow \gamma \beta$$

and there is a derivation of width less than or equal to $2|\gamma|+3$.

Proof:
TABLE 15. Over for $\gamma \in \text{<frontier}_n^{k,F_{n+1,m-1}>}$

<table>
<thead>
<tr>
<th>Line</th>
<th>Width</th>
</tr>
</thead>
<tbody>
<tr>
<td>1. $\beta \text{&lt;over}<em>{n}^{k}\text{&lt;frontier}</em>{n}^{k'}\gamma_1&gt;$</td>
<td>$</td>
</tr>
<tr>
<td>2. $\text{&lt;frontier}<em>{n}^{k'}\beta \text{&lt;over}</em>{n+1}^{k}\text{&lt;P2}_{n}^{k'}\gamma_1&gt;$</td>
<td>$</td>
</tr>
<tr>
<td>3. $\text{&lt;frontier}<em>{n}^{k'}\beta \text{&lt;over}</em>{n+1}^{k}\gamma_1\text{&lt;P2}_{n}^{k'}$</td>
<td>$2</td>
</tr>
<tr>
<td>4. $\text{&lt;frontier}<em>{n}^{k'}\gamma_1\beta \text{&lt;over}</em>{n+1}^{k}\text{&lt;past}<em>{n+1}^{k'}\text{&lt;P2}</em>{n}^{k'}$</td>
<td>$</td>
</tr>
<tr>
<td>5. $\text{&lt;frontier}<em>{n}^{k'}\gamma_1\beta \text{&lt;over}</em>{n}^{k}\text{&lt;past}_{n}^{k'}$</td>
<td></td>
</tr>
</tbody>
</table>

TABLE 16. Over Derivation for $\gamma \in F_{n,m}^0$

<table>
<thead>
<tr>
<th>Line</th>
<th>Width</th>
</tr>
</thead>
<tbody>
<tr>
<td>1. $\beta \text{&lt;over}_{n}^{k}\gamma$</td>
<td>$2</td>
</tr>
<tr>
<td>2. $\gamma\beta \text{&lt;over}<em>{n}^{k}\text{&lt;past}</em>{n}^{l}$</td>
<td>$</td>
</tr>
<tr>
<td>3. $\gamma\beta \text{&lt;over}<em>{n}^{k}\text{&lt;past}</em>{n}^{0}$</td>
<td></td>
</tr>
</tbody>
</table>

This follow directly from lemma 85.
Definition 64:
If \( n+1 \geq k \geq 0 \) then \( S_n^k \) is a function,
\[
S_n^k : H_n^k \rightarrow U_{m \geq 0} F_{n,m}^k
\]
such that if \( S_n^k(\gamma) = \psi \), then
\[
<\text{frontier}_n^k \gamma *> \Rightarrow \psi
\]
and \( <\text{frontier}_n^k \gamma *> \) is the only frontier nonterminal to occur in \( \psi \) and no \( <\text{frontier}_n^k \gamma *> \) nonterminal is expanded.

Lemma 87:
If \( \gamma \in H_{n+1,m}^k \) and \( S_n^k(\gamma) = \psi \), then
\[
<\text{frontier}_n^k \gamma *> \Rightarrow \psi
\]
has a derivation with a width of less than \( 2|\gamma|+5 \).
Also, the length of \( \psi \) may be less than or equal to \( 2|\gamma|+1 \).

Proof by induction on \( m \):
Basis: If \( m = 0 \), the derivation is given in table 17. In this case, \( 2|\gamma|+5 = 7 \).
Induction step: Assume that the lemma is true for \( m' < m \).
If \( \gamma \in H_{n,m'}^k \), then
1. If \( k = n+1 \), then the derivation given in either table 18 or 19. In table 18, \( 2|\gamma|+5 = 7 \) and, in table 19, \( 2|\gamma|+5 = 2|\gamma_2|+7 \).

2. If \( k = n \), then the string is already in the required form.

3. If \( n > k > 0 \), then either the derivation given in table 20 holds, or the derivation given in table 21 is true or \( \gamma \in H^k_{n,m-1} \) in which case the lemma is true by
<table>
<thead>
<tr>
<th>Line</th>
<th>Width</th>
</tr>
</thead>
<tbody>
<tr>
<td>1. (&lt;\text{frontier}<em>{n}^{n+1}\gamma</em>{1}[n+1\gamma_{2}n+1]&lt;&gt;)</td>
<td>$</td>
</tr>
<tr>
<td>2. (&lt;\text{frontier}<em>{n}^{n}&lt;\text{Fl}</em>{n}&gt;&lt;\text{over}<em>{n+1}^{n}\gamma</em>{2}n+1]&lt;&gt;)</td>
<td>$2</td>
</tr>
<tr>
<td>3. (&lt;\text{frontier}<em>{n}^{n}\gamma</em>{2}&lt;\text{Fl}_{n}^{n}n+1]&lt;)</td>
<td>$</td>
</tr>
<tr>
<td>4. (&lt;\text{frontier}<em>{n}^{n}\gamma</em>{2}&lt;&gt;)</td>
<td></td>
</tr>
</tbody>
</table>

induction hypothesis. In table 21, notice that

$2|\gamma|+5 = 2|\gamma_{1}|+2|\gamma_{2}|+9$.

4. If $k = 0$, then the derivation is given in table 22.

Since in each of these cases the width is less than $2|\gamma|+5$, that part of the lemma is proved by induction. Inspection will also show that the output $\psi$ has a length less than $2|\gamma|+1$ in all cases, so this part of the lemma is proved by induction.
TABLE 20. $S_n^k$ Derivation for $\xi \in H_{n,m-1}^{k+1}$

<table>
<thead>
<tr>
<th>Line</th>
<th>Width</th>
</tr>
</thead>
<tbody>
<tr>
<td>1. $&lt;\text{frontier}_n^k&gt;\xi&lt;&gt;()$</td>
<td>$</td>
</tr>
<tr>
<td>2. $&lt;\text{frontier}_n^{k+1}&gt;&lt;\overline{F}<em>2^k&gt;&lt;\overline{\text{over}}</em>{n+1}^{k+1}&gt;\xi&lt;&gt;()$</td>
<td>$2</td>
</tr>
<tr>
<td>3. $&lt;\text{frontier}_n^{k+1}&gt;\xi&lt;\overline{F}_2^k&gt;&lt;&gt;()$</td>
<td>$</td>
</tr>
<tr>
<td>4. $&lt;\text{frontier}_n^{k+1}&gt;\xi&lt;&gt;()$</td>
<td>$2</td>
</tr>
<tr>
<td>5. $\psi$</td>
<td></td>
</tr>
</tbody>
</table>

Lemma 88:

If $\xi_1 \in (\nu U \{<\text{frontier}_n^k>,<>,<>,><\})^*$ and $\xi_2 \in F_n^k \cap (\nu U \{<\text{frontier}_n^k>,<>,><\})^*$, then

$<\text{mark}\xi_1<S_1^k><\xi_2><>() > \Rightarrow \psi \xi_1 \xi_2$

where $\psi \in F_{n,m}^n \cap (\nu U \{<\text{frontier}_n^k>,<>,><\})^*$ has a derivation with a width of less than $|\xi_1|+2|\xi_2|+6$.

Proof by induction on $m$: 
TABLE 21. \( S^k_n \) for \( \gamma \in H^{k+1}_{n+1,m-1}[k H^{k-1}_{n+1,m-1} k] \)

<table>
<thead>
<tr>
<th>Line</th>
<th>Width</th>
</tr>
</thead>
<tbody>
<tr>
<td>1. (&lt;\text{frontier}_n^k&gt;\gamma_1[k \gamma_2 k]&lt;)</td>
<td>(</td>
</tr>
<tr>
<td>2. (&lt;\text{frontier}_n^{k+1}&gt;\gamma_2&lt;k \text{switch}<em>k&lt;over^{k+1}</em>{n+1}&gt;\gamma_1[k \gamma_2 k]&lt;)</td>
<td>(2</td>
</tr>
<tr>
<td>3. (&lt;\text{frontier}_n^{k+1}&gt;\gamma_2&lt;k \text{frontier}^k[k \gamma_2 k]&lt;)</td>
<td>(</td>
</tr>
<tr>
<td>4. (&lt;\text{frontier}<em>n^{k+1}&gt;\gamma_1[k \text{frontier}</em>{k}^{k-1}&lt;\text{switch}<em>k&lt;over</em>{n+1}^{k-1}&gt;\gamma_2 k]&lt;)</td>
<td>(</td>
</tr>
<tr>
<td>5. (&lt;\text{frontier}<em>n^{k+1}&gt;\gamma_1[k \text{frontier}</em>{n}^{k-1}&gt;\gamma_2&lt;\text{switch}&gt; k]&lt;)</td>
<td>(</td>
</tr>
<tr>
<td>6. (&lt;\text{frontier}<em>n^{k+1}&gt;\gamma_1[k \text{frontier}</em>{n}^{k-1}&gt;\gamma_2&lt;]&lt;k)</td>
<td>(2</td>
</tr>
<tr>
<td>7. (\psi_1[k \text{frontier}_{n}^{k-1}&gt;\gamma_2&lt;]&lt;k)</td>
<td>(2</td>
</tr>
<tr>
<td>8. (\psi_1[k \gamma_2 k])</td>
<td></td>
</tr>
</tbody>
</table>

Basis: If \( m = 0 \), then \( \gamma_2 \in \gamma_n^k, 0 = X_k \). But there are no
TABLE 22. $S_n^0$ Derivation for $\gamma \in H_{n,m}^0$

<table>
<thead>
<tr>
<th>Line</th>
<th>Width</th>
</tr>
</thead>
<tbody>
<tr>
<td>1. $&lt;$frontier$_n^0$$\gamma&lt;$)</td>
<td>$</td>
</tr>
<tr>
<td>2. $&lt;$frontier$_n^1$$\gamma&lt;$)</td>
<td>$2</td>
</tr>
<tr>
<td>3. $\psi$</td>
<td></td>
</tr>
</tbody>
</table>

derivations for this case, so the hypothesis must be false and the lemma is true by default in this case.

Induction step: Assume that the lemma is true for $m' < m$.

If $\gamma \in F_{n,m'}^k$, then

1. If $k = n$, then the derivation is given in table 23. Note that between steps 3 and 4 the nonterminal $<\text{return}>$ occurs. It is assumed that this derivation only allows one $<\text{return}>$ at a time.

2. If $n > k > 0$ and $j = ki$, then the derivation is given in table 24. Note that if $\gamma_2 \in F_{n,m-1}^{k+1}$ then the $<S2_n^k>$ in step 3 will be stranded so that case is not possible. Also $|\gamma_1|+2|\gamma_2|+6 = |\gamma_1|+2|\gamma|+2|\beta|+10$.

3. If $n > k > 0$, $\gamma_2 \in F_{n,m-1}^{k+1}$ and $j \in F_n^{k+1}$, then the derivation is given in table 25.
### TABLE 23. \(<S1^nn>_ Derivation for \(\gamma_2 \in F^n_{n,m}\)

<table>
<thead>
<tr>
<th>Line</th>
<th>Width</th>
</tr>
</thead>
<tbody>
<tr>
<td>1. (&lt;\text{mark}&gt;\gamma_1 \langle S1^n_j \rangle \gamma_2 &lt;\text{mark}&gt;)&gt;</td>
<td>(</td>
</tr>
<tr>
<td>2. (&lt;\text{mark}&gt;\gamma_1 &lt;\text{move}&gt; &lt; S5 &lt;\text{over}^n \rangle \gamma_2 &lt;\text{mark}&gt;)&gt;</td>
<td>(</td>
</tr>
<tr>
<td>3. (&lt;\text{mark}&gt;\gamma_1 &lt;\text{move}&gt; \gamma_2 &lt;\text{mark}&gt;)&gt;</td>
<td>(</td>
</tr>
<tr>
<td>4. (\gamma_2 &lt;\text{mark}&gt;\gamma_1 \gamma_2 &lt;\text{move}&gt; &lt; S5 &lt;\text{mark}&gt;)&gt;</td>
<td>(</td>
</tr>
<tr>
<td>5. (\gamma_2 \gamma_1 \gamma_2 &lt;\text{move}&gt; &lt; S5 &lt;\text{mark}&gt;)&gt;</td>
<td>(</td>
</tr>
<tr>
<td>6. (\gamma_2 \gamma_1 \gamma_2)</td>
<td></td>
</tr>
</tbody>
</table>

4. If \(n > k > 0\), \(\gamma_2 \in F^{k+1}_{n,m-1} [k F^{k-1}_{n,m-1} k]\) and \(j \in F^{k+1}_{n}\), then the derivation is given in table 26. In this case, \(|\gamma_1| + 2 |\gamma_2| + 6 = |\gamma_1| + 2 |\alpha| + 2 |\beta| + 10\).

5. If \(n > k > 0\) and \(\gamma_2 \in F^k_{n,m-1}\), then the lemma is true by induction hypothesis.

6. If \(k = 0\), then the derivation is given in table 27. Note that as before the previous cases are used here.
<table>
<thead>
<tr>
<th>Line</th>
<th>Width</th>
</tr>
</thead>
<tbody>
<tr>
<td>1. &lt;mark&gt;$\gamma_1 S_1^k j \alpha[k \beta_k]$&lt;/mark&gt;</td>
<td>$</td>
</tr>
<tr>
<td>2. &lt;mark&gt;$\gamma_1 S_2^k n \alpha[k \beta_k]$&lt;/mark&gt;</td>
<td>$</td>
</tr>
<tr>
<td>3. &lt;mark&gt;$\gamma_1 a S_2^k n [k \beta_k]$&lt;/mark&gt;</td>
<td>$</td>
</tr>
<tr>
<td>4. &lt;mark&gt;$\gamma_1 a[k S_1^k l \alpha[\text{switch}_k] \beta[k \beta_k]$&lt;/mark&gt;</td>
<td>$</td>
</tr>
<tr>
<td>5. &lt;mark&gt;$\gamma_1 a[k S_1^k l \alpha[\text{switch}_k \beta_k]$&lt;/mark&gt;</td>
<td>$</td>
</tr>
<tr>
<td>6. &lt;mark&gt;$\gamma_1 a[k S_1^k l \alpha[\beta_k]$&lt;/mark&gt;</td>
<td>$</td>
</tr>
<tr>
<td>7. $\psi_1 a[k \beta_k]$</td>
<td></td>
</tr>
</tbody>
</table>

Note that since $\text{<frontier}$ is the only frontier nonterminal to be in $\gamma_2$, $\gamma_2$ cannot equal $\text{<frontier}$ if $n > k > 0$. The case where $k = n$ is handled in step 1. Therefore, the lemma is proved.
TABLE 25. \(<S1^k_n>\) Derivation for \(\gamma_2 \in P^{k+1}_n\) and \(j \in P^k_n\)

<table>
<thead>
<tr>
<th>Line</th>
<th>Width</th>
</tr>
</thead>
<tbody>
<tr>
<td>1. (&lt;\text{mark}&gt;\gamma_1&lt;S1^k_n&gt;&lt;\gamma_2&lt;)&gt;</td>
<td>(</td>
</tr>
<tr>
<td>2. (&lt;\text{mark}&gt;\gamma_1&lt;S1^{k+1}_n&gt;&lt;S3^k_n&gt;&lt;\text{over}^k_n&gt;&lt;\gamma_2&lt;)&gt;</td>
<td>(</td>
</tr>
<tr>
<td>3. (&lt;\text{mark}&gt;\gamma_1&lt;S1^{k+1}_n&gt;&lt;\gamma_2&lt;S3^k_n&gt;&lt;)&gt;</td>
<td>(</td>
</tr>
<tr>
<td>4. (&lt;\text{mark}&gt;\gamma_1&lt;S1^{k+1}_n&gt;&lt;\gamma_2&lt;)&gt;</td>
<td>(</td>
</tr>
<tr>
<td>5. (\psi\gamma_1\gamma_2)</td>
<td></td>
</tr>
</tbody>
</table>

Lemma 89:

If \(\gamma_1 \in P^{k^k}_n, m \cap V^*, \gamma_2 \in (V \cup \{<\text{frontier}^k_n>,<\})^*, \) and \(\gamma_3 \in \cup_{n \geq 0} P^{n-1}_n, m \cap (V \cup \{<\text{frontier}^n_n>,<\})^*, \) then

\(<A1^k_n><\gamma_1><\gamma_2<>><\gamma_3><) \Rightarrow \psi\gamma_2<>><\gamma_3<>|

where \(\psi \in (V+\{<\text{frontier}^n_n>,<\})^*\) has a derivation of width

\(2|\gamma_1|+|\gamma_2|+2|\gamma_3|+|\psi|+6\)

Proof by induction on \(m:\)

Basis: If \(m = 0\) and \(k < n,\) then the derivation is given in table 28. If \(k = n,\) then the derivation is given in table
TABLE 26. \( S_1^k \) for \( \gamma_2 \in \text{P}^{k+1}_n \{ x \}_{n \in \text{P}^{k-1}_n} \) and \( j \in \text{P}^{k+1}_n 

<table>
<thead>
<tr>
<th>Line</th>
<th>Width</th>
</tr>
</thead>
<tbody>
<tr>
<td>1. ( &lt;\text{mark}&gt;\gamma_1 &lt;S_1^k_{n}\gamma_{j}&gt; \alpha[k \beta k]\ &lt;\text{mark}&gt; )</td>
<td>(</td>
</tr>
<tr>
<td>2. ( &lt;\text{mark}&gt;\gamma_1 &lt;S_1^{k+1}<em>{n}\gamma</em>{j}&gt; &lt;S_3^k_{n}\gamma_{j}&gt; \alpha[k \beta k]\ &lt;\text{mark}&gt; )</td>
<td>(</td>
</tr>
<tr>
<td>3. ( &lt;\text{mark}&gt;\gamma_1 &lt;S_1^{k+1}<em>{n}\gamma</em>{j}&gt; \alpha &lt;S_3^k_{n}\gamma_{j}&gt; \alpha[k \beta k]\ &lt;\text{mark}&gt; )</td>
<td>(</td>
</tr>
<tr>
<td>4. ( &lt;\text{mark}&gt;\gamma_1 &lt;S_1^{k+1}<em>{n}\gamma</em>{j}&gt; \alpha[j] &lt;S_4^k_{n}\gamma_{j}&gt; \alpha[k \beta k]\ &lt;\text{mark}&gt; )</td>
<td>(</td>
</tr>
<tr>
<td>5. ( &lt;\text{mark}&gt;\gamma_1 &lt;S_1^{k+1}<em>{n}\gamma</em>{j}&gt; \alpha[j] &lt;S_4^k_{n}\gamma_{j}&gt; \alpha[k \beta k]\ &lt;\text{mark}&gt; )</td>
<td>(</td>
</tr>
<tr>
<td>6. ( &lt;\text{mark}&gt;\gamma_1 &lt;S_1^{k+1}<em>{n}\gamma</em>{j}&gt; \alpha[j] &lt;k \beta k\ &lt;\text{mark}&gt; )</td>
<td>(</td>
</tr>
<tr>
<td>7. ( \psi \gamma_1 \alpha[k \beta k]\ &lt;\text{mark}&gt; )</td>
<td>(</td>
</tr>
</tbody>
</table>

29. In either case, \( 2|\gamma_1| + |\gamma_2| + |\psi| + 6 = |\gamma_2| + 2 |\gamma_3| + |\psi| + 8 \). Note that in table 29 between line 7 and 8 the width is \( |\gamma_2| + 2 |\gamma_3| + 8 \). This is less than \( 2|\gamma_1| + |\gamma_2| + 2 |\gamma_3| + |\psi| + 6 \) since \( |\gamma_1| \geq 1 \) and \( |\psi| \geq 1 \).

Induction step: Assume that the lemma is true for all \( m' < m \). If \( \gamma_1 \in \text{H}_{n, m}^k \), then one of the following 5 cases is true:
TABLE 27. \( <S_{1n}^0> \) Derivation

\[
<\text{mark}>y_1<\text{mark}>S_{1n}^0y_2<</mark>
\]

\[
|y_1| + |y_2| + 3
\]

\[
<\text{mark}>y_1<\text{mark}>S_{1n}^1y_2<</mark>
\]

\[
|y_1| + 2|y_2| + 6
\]

\[
\phi y_1 y_2
\]


TABLE 28. \( <A_{1n}^k> \) Derivation for \( y_1 \in X_k \) and \( k < n \)

<table>
<thead>
<tr>
<th>Line</th>
<th>Width</th>
</tr>
</thead>
<tbody>
<tr>
<td>1. ( &lt;A_{1n}^k&gt;y_1&lt;y_2,y_3&lt;&lt;/mark&gt; )</td>
<td>(</td>
</tr>
<tr>
<td>2. ( y_1y_2,y_3&lt;&lt;/mark&gt; )</td>
<td></td>
</tr>
</tbody>
</table>

1. If \( k = n+1 \), then the derivation is given in table 30. Here, \( 2|y_1| + |y_2| + 2|y_3| + |\psi| + 6 = |y_2| + 2|y_3| + |\psi| + 8 \).

2. If \( n \geq k > 0 \) and \( y_1 \in H_{n,m-1}^{k+1} \), then the derivation is given in table 31.

3. If \( n \geq k > 0 \) and \( y_1 \in H_{n,m-1}^{k} \), then the lemma is true by induction hypothesis.
TABLE 29. \(<A_1^n_n> Derivation for \(r_1 = X_j^j\)

<table>
<thead>
<tr>
<th>Line</th>
<th>Width</th>
</tr>
</thead>
<tbody>
<tr>
<td>1. (&lt;A_1^n_n&gt;y_1^&lt;)&gt;y_2^&lt;),y_3^&lt;)&gt;</td>
<td>(</td>
</tr>
<tr>
<td>2. (&lt;\text{search}_n^{n-1}&gt;y_2^&lt;),y_3^&lt;)&gt;</td>
<td>(</td>
</tr>
<tr>
<td>3. (&lt;\text{mark}&gt;\text{setup}_n^{n-1}&gt;y_2^&lt;),y_3^&lt;)&gt;</td>
<td>(</td>
</tr>
<tr>
<td>4. (&lt;\text{mark}&gt;y_2^&lt;\text{setup}_n^{n-1}&gt;y_2^&lt;),y_3^&lt;)&gt;</td>
<td>(</td>
</tr>
<tr>
<td>5. (&lt;\text{mark}&gt;y_2^&lt;,\text{Dup}&lt;\text{over}_n^{n-1}&gt;y_3^&lt;)&gt;</td>
<td>(</td>
</tr>
<tr>
<td>6. (&lt;\text{mark}&gt;y_2^&lt;,\text{Dup}&lt;y_3^&lt;)&gt;</td>
<td>(</td>
</tr>
<tr>
<td>7. (&lt;\text{mark}&gt;y_2^&lt;,\text{Y}_3^&lt;)&gt;</td>
<td>(</td>
</tr>
<tr>
<td>8. (\psi y_2^&lt;,y_3^&lt;)&gt;</td>
<td>(</td>
</tr>
</tbody>
</table>

4. If \(n \geq k > 0\) and \(v \in H_{n,m-1}^{k+1} [H_{n,m-1}^{k-1} k]\), then the derivation is given in table 32. Here,
\[2|y_1|+|y_2|+|\psi|+6 = 2|\alpha|+2|\beta|+|y_2|+2|y_3|+|\psi_1|+|\psi_2|+12.\]

5. If \(k = 0\), then the derivation is given in table 33.
**TABLE 30.** \( A_{n}^{n+1} \) Derivation

<table>
<thead>
<tr>
<th>Line</th>
<th>Width</th>
</tr>
</thead>
<tbody>
<tr>
<td>1. ( A_{n}^{n+1} \gamma_{1} \gamma_{2} \gamma_{3} )</td>
<td>(</td>
</tr>
<tr>
<td>2. ( \gamma_{1} \gamma_{2} \gamma_{3} )</td>
<td>-</td>
</tr>
</tbody>
</table>

**TABLE 31.** \( A_{n}^{k} \) Derivation for \( \gamma_{1} \in H_{n,m-1}^{k+1} \)

<table>
<thead>
<tr>
<th>Line</th>
<th>Width</th>
</tr>
</thead>
<tbody>
<tr>
<td>1. ( A_{n}^{k} \gamma_{1} \gamma_{2} \gamma_{3} )</td>
<td>(</td>
</tr>
<tr>
<td>2. ( A_{n}^{k+1} A_{3}^{k} \overline{\gamma}<em>{1} \gamma</em>{2} \gamma_{3} )</td>
<td>( 2</td>
</tr>
<tr>
<td>3. ( A_{n}^{k+1} \gamma_{1} A_{3}^{k} \gamma_{2} \gamma_{3} )</td>
<td>(</td>
</tr>
<tr>
<td>4. ( A_{n}^{k+1} \gamma_{1} \gamma_{2} \gamma_{3} )</td>
<td>( 2</td>
</tr>
<tr>
<td>5. ( \gamma_{2} \gamma_{3} )</td>
<td>-</td>
</tr>
</tbody>
</table>

In each case, the required bound is correct. Therefore, the lemma is proved.
TABLE 32. $A_{n}^{k}$ for $\gamma_{1} \in H_{n,m-1}^{k+1} H_{n,m-1}^{k-1}$

<table>
<thead>
<tr>
<th>Line</th>
<th>Width</th>
</tr>
</thead>
<tbody>
<tr>
<td>1. $A_{n}^{k} a_{k}^{(1)} \gamma_{2}^{(1)} \gamma_{3}^{(1)}$</td>
<td>$</td>
</tr>
<tr>
<td>2. $A_{n}^{k+1} A_{n}^{k+1} a_{k}^{(2)} \gamma_{2}^{(2)} \gamma_{3}^{(2)}$</td>
<td>$2</td>
</tr>
<tr>
<td>3. $A_{n}^{k+1} A_{n}^{k+1} a_{k}^{(3)} \gamma_{2}^{(3)} \gamma_{3}^{(3)}$</td>
<td>$</td>
</tr>
<tr>
<td>4. $A_{n}^{k+1} A_{n}^{k+1} a_{k}^{(4)} A_{n}^{k+1} a_{k}^{(5)} \gamma_{2}^{(4)} \gamma_{3}^{(4)}$</td>
<td>$</td>
</tr>
<tr>
<td>5. $A_{n}^{k+1} A_{n}^{k+1} a_{k}^{(6)} A_{n}^{k+1} a_{k}^{(7)} \gamma_{2}^{(6)} \gamma_{3}^{(6)}$</td>
<td>$</td>
</tr>
<tr>
<td>6. $A_{n}^{k+1} A_{n}^{k+1} a_{k}^{(8)} A_{n}^{k+1} a_{k}^{(9)} \gamma_{2}^{(8)} \gamma_{3}^{(8)}$</td>
<td>$2</td>
</tr>
<tr>
<td>7. $\psi_{1}^{(1)} A_{n}^{k+1} a_{k}^{(11)} \gamma_{2}^{(11)} \gamma_{3}^{(11)}$</td>
<td>$2</td>
</tr>
<tr>
<td>8. $\psi_{1}^{(12)} A_{n}^{k+1} a_{k}^{(13)} \gamma_{2}^{(13)} \gamma_{3}^{(13)}$</td>
<td></td>
</tr>
</tbody>
</table>
Table 33. $A_{ln}^0$ Derivation

| Line | $<A_{ln}^0\gamma_1\gamma_2\gamma_3>$ | Width $|\gamma_1|+|\gamma_2|+|\gamma_3|+4$ |
|------|---------------------------------|----------------------------------|
| 1.   | $<A_{ln}^0\gamma_1\gamma_2\gamma_3>$ | $|\gamma_1|+|\gamma_2|+|\gamma_3|+4$ |
| 2.   | $<A_{ln}^1\gamma_1\gamma_2\gamma_3>$ | $2|\gamma_1|+|\gamma_2|+2|\gamma_3|+|\psi|+6$ |
| 3.   | $\psi\gamma_2\gamma_3>$ |                                      |

Lemma 90:
If $\gamma \in H_{n+1,m}'$, then

$$<\text{frontier}_{n}^k\gamma> \Rightarrow \psi$$

where $\psi \in \mathcal{V}^*$ then there is a derivation of width less than

$$4|\gamma|+5|\psi|+3$$

Proof by induction on $m$:

Basis: If $m = 0$, then the derivation is given in table 34.

Table 34. Frontier Derivation for Basis

<table>
<thead>
<tr>
<th>Line</th>
<th>$&lt;\text{frontier}_{n}^k\gamma&gt;$</th>
<th>Width $3$</th>
</tr>
</thead>
<tbody>
<tr>
<td>1.</td>
<td>$&lt;\text{frontier}_{n}^k\gamma&gt;$</td>
<td>$3$</td>
</tr>
<tr>
<td>2.</td>
<td>$\gamma$</td>
<td></td>
</tr>
</tbody>
</table>
Induction step: Assume that the lemma is true for all \( m' < m \). If \( \gamma \in H_{n+1,m}' \), then

1. If \( k = n+1 \) and \( \gamma \in I \), then the derivation is given in table 34.

2. If \( k = n+1 \) and \( \gamma = \alpha_{n+1} \beta_{n+1} \), then the derivation is given in table 35. Here, 
\[
4|\gamma| + 5|\psi| + 3 = 4|\beta| + 5|\psi| + 15.
\]

TABLE 35. Frontier Derivation for \( \gamma = \alpha_{n+1} \beta_{n+1} \)

<table>
<thead>
<tr>
<th>Line</th>
<th>Width</th>
</tr>
</thead>
<tbody>
<tr>
<td>1. (&lt;\text{frontier}^{n+1}<em>{n} \alpha</em>{n+1} \beta_{n+1} \rangle&gt;)</td>
<td>(</td>
</tr>
<tr>
<td>2. (&lt;\text{frontier}^{n}<em>{n} &lt;F \rangle</em>{n} &lt;\text{over}^{n}<em>{n+1} \beta</em>{n+1} \rangle&gt;)</td>
<td>(2</td>
</tr>
<tr>
<td>3. (&lt;\text{frontier}^{n}<em>{n} \beta &lt;F \rangle</em>{n+1} \rangle&gt;)</td>
<td>(</td>
</tr>
<tr>
<td>4. (&lt;\text{frontier}^{n}_{n} \beta \rangle&gt;)</td>
<td>(4</td>
</tr>
<tr>
<td>5. (\psi)</td>
<td></td>
</tr>
</tbody>
</table>

3. If \( k = n \) and \( \gamma \in H_{n+1,m-1} \), then the derivation is given in table 36.
<table>
<thead>
<tr>
<th>Line</th>
<th>Width</th>
</tr>
</thead>
<tbody>
<tr>
<td>1. <code>&lt;frontier^n&gt; y &lt;)</code></td>
<td>$</td>
</tr>
<tr>
<td>2. <code>&lt;frontier^{n+1}&gt; die &lt;over^{n+1}&gt; y &lt;)</code></td>
<td>$2</td>
</tr>
<tr>
<td>3. <code>&lt;frontier^{n+1}&gt; y &lt;die&gt;</code></td>
<td>$</td>
</tr>
<tr>
<td>4. <code>&lt;frontier^{n+1}&gt; y &lt;)</code></td>
<td>$4</td>
</tr>
<tr>
<td>5. $\psi$</td>
<td></td>
</tr>
</tbody>
</table>

4. If $k = n$ and $C \in H_{n+1,m-1}^n$, then the lemma is true by induction hypothesis.

5. If $k = n$, $\gamma = \alpha[\beta_1 \ldots \beta_n]$ and $\alpha \in \Sigma$, then the derivation is given in table 37. Here, $4|\gamma| + 5|\psi| + 3 = 4|\beta| + 5|\psi| + 30$.

6. If $k = n$, $\gamma = \alpha[\beta_1 \ldots \beta_n]$ and $\alpha \notin \Sigma$, then the derivation is given in table 38. Here, $4|\gamma| + 5|\psi| + 3 = 4|\alpha| + 4|\beta| + 5|\psi| + 11$. Notice that $\psi_0$ is the frontier of $\alpha$ and must be shorter than or equal to $|\psi|$ since each "X" that is replaced in $\psi_0$ is replaced with at least one character. $\psi_1 = S_{n}^{n-1}(\beta)$ and by lemma 87.
TABLE 37. Frontier for $k = n$, $\gamma = \alpha[n \beta n]$ and $\alpha \in \Sigma$.

<table>
<thead>
<tr>
<th>Line</th>
<th>Width</th>
</tr>
</thead>
<tbody>
<tr>
<td>1. $\langle \text{frontier}_n^n \alpha[n \beta n] \rangle$</td>
<td>$</td>
</tr>
<tr>
<td>2. $\alpha[n \langle \text{frontier}_n^{n-1} \rangle \langle \text{switch}_n \rangle \langle \text{over}_n^{n-1} \beta \rangle n]$</td>
<td>$2</td>
</tr>
<tr>
<td>3. $\alpha[n \langle \text{frontier}_n^{n-1} \rangle \beta \langle \text{switch}_n \rangle n]$</td>
<td>$</td>
</tr>
<tr>
<td>4. $\alpha[n \langle \text{frontier}_n^{n-1} \rangle \beta \rangle_n]$</td>
<td>$4</td>
</tr>
<tr>
<td>5. $\alpha[n \psi_2 n]$</td>
<td></td>
</tr>
</tbody>
</table>

$|\psi_1| \leq 2|\beta| + 1$

Between step 11 and step 12 $\psi_2$ is the same as $\psi_1$ except that the frontier has been taken of everything that is to be copied into $\psi_0$. The width is calculated as follows. Let

$|\psi_1| = \psi_{11} + 2\psi_{12} + \psi_{13}$

where $\psi_{11}$ is the length of the portion of $\psi_1$ which is not copied, $\psi_{12}$ is the number of frontier operations in that portion of $\psi_1$ that is copied, and $\psi_{13}$ is the
total of the length of the arguments to the frontier operations in that portion of \( \psi_1 \) that is not copied. Let

\[ |\psi_2| = \psi_{11} + \psi_{22} \]

where \( \psi_{22} \) is the sum of the lengths of all of the frontiers in \( \psi_2 \) that are copied. Let \( X \) be the number of "X"'s in \( \psi_0 \). Then, if the frontiers are taken one at a time the width must be less than

\[ |\psi_0| + \psi_{11} + 4\psi_{13} + 5\psi_{22} + 8 \]

It would be easy to show that \( \psi_{11} \leq 2(\beta - \psi_{13}) + 1 \) using lemma 87 as a guide. Therefore, a bound is

\[ |\psi_0| + 2|\beta| + 2\psi_{13} + 5\psi_{22} + 9 \]

Since \( \psi_{13} \leq |\beta| \) another bound is

\[ |\psi_0| + 4|\beta| + 5\psi_{22} + 9 \]

Now, consider the two possible cases:

1. If \( \psi_0 = X_n^j \), then a bound is \( 4|\beta| + 5\psi_{22} + 10 \) since \( |\psi_0| = 1 \). Also, \( \psi_{22} \leq |\psi| \) so another bound is \( 4|\beta| + 5|\psi| + 10 \).

2. If \( \psi_0 \neq X_n^j \), then it can be shown by induction that \( |\psi_0| \geq 2X \). It is also true that
|ψ| ≥ |ψ₀| + ψ₂₂ - X since all of ψ₀ except the X's appears in ψ, and by definition all of ψ₂₂ appears in ψ. Therefore,

\[
2|ψ| ≥ 2|ψ₀| + 2ψ₂₂ - 2X \\
≥ |ψ₀| + 2ψ₂₂ + (|ψ₀| - 2X) \\
≥ |ψ₀| + 2ψ₂₂
\]

Therefore,

\[
5|ψ| ≥ |ψ₀| + 5ψ₂₂
\]

and a possible bound is 4|β| + 5|ψ| + 9.

In either case, a bound is 4|β| + 5|ψ| + 10.

Between 12 and 13 the bound is 2|ψ₀| + 2|ψ₂| + |ψ| + 7 by lemma 89. But |ψ| ≥ |ψ₀| and

\[
|ψ₂| = ψ₁₁ + ψ₂₂ \\
≤ ψ₁₁ + |ψ| \\
≤ |ψ| + 2|β| + 1
\]

by lemma 87 so a bound is 4|β| + 5|ψ| + 9. Similarly for the rest of the table.
TABLE 38. Frontier for \( k = n \), \( \gamma = \alpha[n, \beta_n] \) and \( \alpha \notin \Sigma \).

<table>
<thead>
<tr>
<th>Line</th>
<th>Width</th>
</tr>
</thead>
<tbody>
<tr>
<td>1. (&lt;\text{frontier}_n^n \alpha[n, \beta_n]&gt;))</td>
<td>(</td>
</tr>
<tr>
<td>2. (&lt;\text{apply}_n^n \text{frontier}_n^n \text{F}_3^n &gt; \text{over}^{n+1}_n \alpha[n, \beta_n]&gt;))</td>
<td>(2</td>
</tr>
<tr>
<td>3. (&lt;\text{apply}_n^n \text{frontier}_n^n \alpha &lt; \text{F}_3^n &gt; [n, \beta_n]&gt;))</td>
<td>(</td>
</tr>
<tr>
<td>4. (&lt;\text{apply}_n^n \text{frontier}_n^n \alpha &lt; \text{F}_3^n &gt; [n, \beta_n]&gt;))</td>
<td>(</td>
</tr>
<tr>
<td>5. (&lt;\text{apply}_n^n \text{frontier}_n^n \alpha &lt; \text{F}_4^n &gt; [n, \beta_n]&gt;))</td>
<td>(</td>
</tr>
<tr>
<td>6. (&lt;\text{apply}_n^n \text{frontier}_n^n \alpha &lt; \text{F}_4^n &gt; [n, \beta_n]&gt;))</td>
<td>(4</td>
</tr>
<tr>
<td>7. (&lt;\text{apply}_n^n \psi_0 &lt; \text{F}_4^n &gt; [n, \beta_n]&gt;))</td>
<td>(</td>
</tr>
<tr>
<td>8. (&lt;\text{A}_2^n &lt; \text{A}_1^n &lt; \text{A}_4^n &gt; \text{over}^{n}_n \psi_0 &lt; \text{frontier}_n^{n-1} \beta&gt;))</td>
<td>(</td>
</tr>
</tbody>
</table>
7. If $n > k > 0$ and $\gamma \in H_{n+1,m-1}^{k+1}$, then the derivation is given in the table 39.

8. If $n > k > 0$ and $\gamma \in H_{n+1,m-1}^k$, then the lemma is true by induction hypothesis.
TABLE 39. Frontier Derivation for $\gamma \in E_n^{k+1}$

<table>
<thead>
<tr>
<th>Line</th>
<th>Width</th>
</tr>
</thead>
<tbody>
<tr>
<td>1. $&lt;\text{frontier}_n^k \gamma &gt;$</td>
<td>$</td>
</tr>
<tr>
<td>2. $&lt;\text{frontier}_n^{k+1} &gt; \text{F}_2^k &lt;\text{over}_n^{k+1} \gamma &gt;$</td>
<td>$2</td>
</tr>
<tr>
<td>3. $&lt;\text{frontier}_n^{k+1} &gt; \gamma \text{F}_2^k \gamma &gt;$</td>
<td>$</td>
</tr>
<tr>
<td>4. $&lt;\text{frontier}_n^{k+1} &gt; \gamma &gt;$</td>
<td>$4</td>
</tr>
<tr>
<td>5. $\psi$</td>
<td></td>
</tr>
</tbody>
</table>

9. If $n > k > 0$ and $\gamma = \alpha[\gamma \beta_1 \beta_2 \ldots \beta_k]$, then the derivation is given in table 40. Note that $4|\gamma| + 5|\psi| + 3 = 4|\alpha| + 4|\beta| + 5|\psi_1| + 5|\psi_2| + 21$.

10. If $k = 0$, then the derivation is given in table 41. In this case, note that the derivations given in the previous steps must be used to complete the proof.

Now, since each of these are bound by $4|\gamma| + 5|\psi| + 3$, the lemma is proved by induction.
### TABLE 40. Frontier Derivation for $\gamma = a[k \, \beta \, k]$

<table>
<thead>
<tr>
<th>Line</th>
<th>Width</th>
</tr>
</thead>
<tbody>
<tr>
<td>1. <code>&lt;frontier^k_n&gt;a[k \, \beta \, k]&lt; &gt;</code></td>
<td>$</td>
</tr>
<tr>
<td>2. <code>&lt;frontier^{k+1}_n&gt;&lt;F2_k&gt;&lt;over^{k+1}_n&gt;a[k \, \beta \, k]&lt; &gt;</code></td>
<td>$2</td>
</tr>
<tr>
<td>3. <code>&lt;frontier^{k+1}_n&gt;&lt;a&lt;F2[k]&gt;&lt;[k \, \beta \, k]&lt; &gt;</code></td>
<td>$</td>
</tr>
<tr>
<td>4. <code>&lt;frontier^{k+1}_n&gt;&lt;a&gt;[k&lt;frontier^{k-1}_n&gt;&lt;switch_k&gt;&lt;over^{k-1}_n}\beta_k]&lt; &gt;</code></td>
<td>$</td>
</tr>
<tr>
<td>5. <code>&lt;frontier^{k+1}_n&gt;&lt;a&gt;[k&lt;frontier^{k-1}_n}\beta&lt;switch_k&gt;\beta_k]&lt; &gt;</code></td>
<td>$</td>
</tr>
<tr>
<td>6. <code>&lt;frontier^{k+1}_n&gt;&lt;a&gt;[k&lt;frontier^{k-1}_n}\beta&lt; &gt;</code></td>
<td>$</td>
</tr>
<tr>
<td>7. <code>&lt;frontier^{k+1}_n&gt;&lt;a&gt;[k \psi_2 k]</code></td>
<td>$4</td>
</tr>
<tr>
<td>8. $\psi_1[k \psi_2 k]$</td>
<td></td>
</tr>
</tbody>
</table>
### TABLE 41. Frontier Derivation for $k = 0$

<table>
<thead>
<tr>
<th>Line</th>
<th>Width</th>
</tr>
</thead>
<tbody>
<tr>
<td>1. $\langle \text{frontier}_n^0, y \rangle$</td>
<td>$</td>
</tr>
<tr>
<td>2. $\langle \text{frontier}_n^1, y \rangle$</td>
<td>$4</td>
</tr>
<tr>
<td>3. $\psi$</td>
<td></td>
</tr>
</tbody>
</table>

**Lemma 91:**

If $y \in H_{n+1,m}$, then

$$\langle \text{frontier}_n^k, y \rangle \Rightarrow \psi$$

where $\psi \in V^*$ then

$$\psi = \text{frontier}_n^k(y)$$

**Proof:**

This can be established by a simple but lengthy induction on $m$. It would be necessary to prove a similar result as a lemma for search and apply, but these will not be written down here.

---

**Lemma 92:**

If $y \in H_n$, then

$$\langle \text{string}_n^k, y \rangle \Rightarrow \psi$$

where $\psi \in V^*$ then $\psi = \text{string}(y)$.
Proof:

This is a simple induction on \( n \) using lemma 91.

---

Lemma 93:

Given the grammar presented, the language is the same as the string language of the original grammar. Also, if a linearly bounded sequence of frontiers can be found for each element of the string language, then the string language has a context sensitive grammar.

Proof:

The first part of this lemma follows directly from lemma 92. The second part requires inductive type proof using lemma 90.

---

This last lemma reduces the problem to one of showing that for every hypertree grammar there is a (possible the same) hypertree grammar which has the same string language and each element of the string language has a linearly bounded set of frontiers. The approach that will be used here is to generate a normalized grammar for the language, then some alternations are made on the language which do not effect the resulting string language. Finally, this grammar is presented as a linearly bounded grammar, proving the language to be context sensitive.
Lemma 94:

If \( G_n^k = (\Sigma, V, P, S)_n \) is a normal hypertree grammar, then there is a grammar \( G'_n^k = (\Sigma', V', P', S')_n \) and a \( K \) such that

\[
\text{string}(G_n^k) = \text{string}(G'_n^k)
\]

and if \( \alpha \in \text{string}(G'_n^k) \) then there is a \( \psi \) such that

\[
\text{extract}_n(\psi) = \alpha
\]

and

\[
\text{extract}_j(\psi) < K|\alpha|
\]

for all \( n \geq j \geq 1 \).

Proof:

Construct a completed linear grammar for \( \text{string}(G_n^k) \) by constructing a linear grammar as was done in corollary 70 in chapter 7. Then, construct a completed grammar as was done in theorem 63 in chapter 7. Call this new grammar \( G'_n^k \). Let

\[
V = \{ X^j_n | X^j_n \text{ appears in } G'_n^k \}
\]

Let \( K_1 \) be the number of characters in the righthand side of the longest production in \( G'_n^k \).

If \( A \Rightarrow \psi \) and \( \alpha = \text{extract}_i(\psi) \) where \( \alpha \in \Sigma U \bigcup_{n>0} X_n^j \), then define \( G(A, \psi, \alpha) \) as the number of characters in the longest \( \psi_1 \) such that

\[
\psi_1 = \text{extract}_j(\psi)
\]

for some \( j \). Let

\[
K_2 = \max_{A, \alpha} (\min_{\psi} (G(A, \psi, \alpha)))
\]
If $A \Rightarrow \psi$, $\alpha = \text{string}(\psi)$ and $\psi$ is the smallest hypertree for which this is true, then a limit on the length of $\text{extract}_j(\psi)$ is

$$((|V|K_2+K_1+4)|V||V||V|)^{j-1}|\alpha|$$

Proof by induction on $j$:

Let $K_3 = (|V|K_2+K_1+4)|V||V||V|$.

Basis: If $j = 1$, then

$$\text{extract}_j(\psi) = \alpha$$

so that

$$|\text{extract}_j(\psi)| = |\alpha|$$

$$\leq K_3^0|\alpha|$$

$$\leq K_3^{j-1}|\alpha|$$

Induction step: Assume that if $j' < j$ then

$$|\text{extract}_{j'}(\psi)| \leq K_3^{j'-1}|\alpha|$$

Notice that in the definition of frontier only two lines result in a frontier that is shorter than the frontiered hypertree. These are

1. $\text{frontier}_n^{n+1}(\alpha[\beta]) = \text{frontier}_n^n(\beta)$.
2. $\text{frontier}_n^n(\alpha[\beta]) = \text{apply}_n^n(\text{frontier}_n^{n+1}(\alpha), \text{frontier}_n^{n-1}(\beta))$. 
Define a function $F$ which maps a hypertree to a set of nonterminals. In particular, $F(\alpha) = \{A \mid \extract_{k'}(\alpha) = \extract_{k'}(\psi')$ where $\psi'$ is that portion of $\alpha$ generated by nonterminal $A\}$. It should be obvious that

$$|F(\alpha)| \leq |V|$$

for all $\alpha$. It should also be obvious that if the nonterminal that generates $\alpha$ is in $F(\alpha)$ then there is a shorter hypertree which could have been used to generate the same frontier as $\alpha$. This limits the maximum length of any recursion which does not change the frontier to $|V|$ in $\psi$. It is possible that the hypertree may be changed by just changing the X's and not changing the length, but there are less than $|V||V|^{|V|}$ possible ways that the X's can be arranged. Note that once two X's become the same during the frontiering process they must remain the same throughout any further recursions. The total number of recursions that need to be allowed, therefore, are fewer than $|V||V|^{|V|}$. Note that the number of characters dropped by one recursion is at most $|V|K_2 + K_1 + 3$ since

1. If the recursion is of type one, then only 3 characters are dropped.

2. If the recursion is of type two, then less than $K_1$ characters can be deleted from the $\beta$ in the
equation since \( \beta \) has a maximum length of \( K_1 \). Also, a maximum of \( |V|K_2 + K_1 + 3 \) characters can be deleted from the \( \alpha \) since

1. If \( \alpha = a[n+1, \beta_1 n+1], \) frontier(\( \beta_1 \)) \( \neq \) \( X_n^j \) and all of \( \beta \) is dropped, then frontier(\( \beta_1 \)) is the hypertree that is copied. Therefore, 6 characters (a, \( [n+1, n+1] \), \( [n', * \text{ and } n] \)) are dropped in this case.

2. If \( \alpha = a[n+1, \beta_1 n+1], \) frontier(\( \beta_1 \)) \( \neq \) \( X_n^j \) and not all of \( \beta \) is deleted, and since by design frontier(\( \beta_1 \)) = frontier(\( a[n, \beta n] \)), therefore, it must be true that the only copying that is allowed is for one \( X \) to replace itself with another \( X \). Each of these \( X \)'s must have been generated by the shortest derivation possible since \( \psi \) is the shortest hypertree. Therefore, each of these hypertrees must have a length of less than \( K_2 \) characters. In addition, there are at most \( |V| \) of these, plus at most \( K_1 \) characters which are dropped from the \( \beta \) directly plus 3 characters (a, \( [n+1, n+1] \)) that are dropped from the \( \alpha \). That is

\[ |V|K_2 + K_1 + 3 \]

characters are dropped.
3. If \( a = a[n+1, b_1, n+1] \) where 
\( \text{frontier}(b_1) = x^j_n \), then at most \( K_2 \) characters are dropped from \( b_1 \) by definition of \( K_2 \) and less than \( K_1 \) characters are dropped from \( b \). 
Therefore, at most \( K_1 + K_2 + 3 \) characters are dropped in this case.

In each of these cases, fewer than 
\[ |V|K_2 + K_1 + 3 \]
characters are dropped so 
\[ |\text{extract}_{j+1}(a)| < K_3 |\text{extract}_j(a)| < (K_3)^{j-1} |a| \]
and the conjecture is proven by induction. Also, if 
\[ K = (K_3)^{n-1} \]
the lemma is proved.

---

Theorem 95:
If \( G_n^k \) is a regular hypertree grammar, then the string language for \( G_n^k \) is context sensitive.

Proof:
This follows directly from lemma 94 and lemma 93.

---
Note that the linear bound on this language potentially quite large, so that it may not always be readily apparent what the context sensitive grammar is, however, given patience, this paper will eventually allow its construction.
CONCLUSION

The results shown, therefore, illustrate that there is an infinite hierarchy of languages called the algebraic language hierarchy. This hierarchy can be expressed via hypertrees -- using either grammars, automata or regular expressions. It has also been shown that this entire hierarchy is not as powerful as the set of context sensitive languages. It is hoped that further research will be done to determine exactly what can and cannot be expressed using this hierarchy. Also, research is currently being done to determine a feasible algorithm for parsing languages in this hierarchy.

Some of the potential usefulness of this hierarchy can be seen by comparing the set of three languages given by Aho and Ullman (Aho and Ullman 1977) modeling various real programming language constructs which are not context free. In each case, a macro grammar can be easily given which generates the language. Although this is greatly oversimplified, perhaps the additional power of the higher levels of the hierarchy can model the more complex real life situation.

Quite probably the concept of hypertrees may find uses which do not directly relate to the specification of programming language. Already, Dr. George Strawn of Iowa
State has found a very practical use for hypertree modeling in the area of idiom matching (Strawn 1982). Trees may be used to effectively encode the parse of a string language in most programming languages. However, recently some languages have been developed which are written in a tree format. It may be possible to parse these languages using a hypertree of level three in place of the parse tree. This last suggestion might be extended to the parse tree itself so that in addition to the parse tree there would be the parse of the parse tree. Lastly, it might be possible to use the concept of hypertrees as a starting point to find other types of grammars, etc. for specifying languages.

Research is already being done to answer the question, "What happens if the paths are redefined to eliminate the restriction that an \( n^{th} \) level path cannot have any \( n \)'s in it?"

Throughout this entire paper it has been assumed that the set of hypertrees did not include the null hypertree. This was just a matter of taste as was indicated in the introduction -- it would have been equally as easy to have a characterization which included a null hypertree. Therefore, an alternative definition of hypertree might be
Definition 65:

The set of hypertrees over $\Sigma$ of level $n$, denoted $H_n(\Sigma)$, is the smallest set such that

(a) $H_{-1}(\Sigma) = \{\lambda\}$

(b) $H_n(\Sigma) = \Sigma \cup X_n \cup \Sigma \{\lambda H_{n-1}(H_n(\Sigma))\}_k$

In this definition, $\lambda$ is the null hypertree. Note that definition of a path would have to be altered to allow a $P_{n-1} = \emptyset$. Although this addition could possibly make some major changes in the proofs, it is felt that this will not significantly effect the final results.

And finally, in this paper an exploration has been made of the IO hierarchy because the author felt it was the most natural. What happens with the OI hierarchy? It can easily be seen that it is a true hierarchy since the language hierarchy given for the IO case will, with minor modifications, show the hierarchy for the OI case. The proof of containment in context sensitive language, although potentially simplified by the proofs in this paper, is by no means a trivial proof. It has been shown that OI macro grammars can model a language which requires variables to be declared. It might, therefore, be possible that the OI hierarchy may model some aspects of real languages ever better than the IO algebraic language hierarchy that has been presented here.
BIBLIOGRAPHY


Baker, B. S. "Composition of Top-down and Bottom-up Tree Transducers". Information and Control 41 (1979): 186-213.


ACKNOWLEDGEMENTS

I would like to acknowledge the contribution of Dr. George O. Strawn to the completion of this paper. His constant advice and insights were greatly appreciated. I would like to also express appreciation to those graduate students who were involved in the various aspects of this research. I also appreciate the contribution of the other members of my graduate committee for being an inspiration to me in various ways. I appreciate the contribution of the other members of the Iowa State faculty who have inspired me through the years.

Also, appreciation is extended to the Iowa State Computer Science Department and Computer Center for the computer time and resources used in the editing of this paper.

Any set of acknowledgements must also include some of the people who contributed by supplying the background that was needed for this project. In particular, Mr. Jack Freeze, my junior high mathematics teacher, who recognized the talent that I had and was willing to inspire me to pursue it. Also, Mr. Leonard Johnsen, my high school mathematics teacher who taught me to think logically.

I would like to also acknowledge the contribution of my parents, whose loving home was a constant source of strength
to me. I particularly appreciated the contribution of my father who often put my educational needs above his own interests. For this I am very grateful.

Last, but far from least, I appreciate the contribution of my wife, Feng-chuan Hsiao. In a practical sense, she entered most of this dissertation onto the computer. It is to her that I dedicate this paper, as I have already dedicated my life.
APPENDIX: LINEARLY BOUNDED GRAMMARS

This appendix is presented to establish a convenient method of proving that a given grammar describes a context sensitive language. This method, called linearly bounded grammar, is used extensively in this paper. Note that this is somewhat similar to the approach taken by Kuroda (Kuroda 1964) and Landweber (Landweber 1963) except that the major thrust in both papers was the development of linearly bounded automata whereas this proof emphasizes the grammars. Also, the idea of "linearly bounded" is much more restrictive in these papers.

In order to simplify the proofs, a normal form is established which is an extension of Chomsky Normal form to unrestricted grammars. In this normal form, productions are allowed to be in the form

\[ A \rightarrow a \]

\[ A \rightarrow BC \]

\[ AB \rightarrow C \]

\[ A \rightarrow \lambda \]
where capital letters represent nonterminals and small letters represent terminals. It will be seen later that an important concept involved in linearly bounded grammars is that of the width of a derivations. To define this, derivation will be formally defined.

Definition 66:
A derivation for $\gamma$ over grammar $G=(\Sigma, V, P, S)$ where $\gamma \in \text{language}(G)$ is a sequence of sentential forms

$$\gamma_0, \gamma_1, \ldots, \gamma_n$$

such that $S = \gamma_0, \gamma_i \Rightarrow \gamma_{i+1}$ for all $n > i \geq 0$ and $\gamma_n = \gamma$.

Definition 67:
The width $w$ of derivation $\gamma_0, \gamma_1 \ldots, \gamma_n$ is defined as

$$w = \max_{n \geq i \geq 0}(|\gamma_i|)$$

It will be shown that the normal form grammar described preserves the width of derivation. This fact simplifies proofs involving linearly bounded grammars.
Lemma 96:

For every grammar $G=(\Sigma, V, P, S)$, there is a corresponding grammar $G'=(\Sigma', V', P', S')$ such that

$$\text{language}(G) = \text{language}(G')$$

and

$$P' \subseteq (V'^{+} \rightarrow V'^{*}) \cup (V' \rightarrow \Sigma')$$

and if there is a derivation for $\gamma$ over $G$ of width $w$, then there is also a derivation for $\gamma$ over $G'$ of width $w$.

Proof:

Construct $G'$ as follows:

1. $\Sigma' = \Sigma$.
2. $V' = V \cup \Sigma''$ where $\Sigma''$ has elements of the form $(a)$ where $a \in \Sigma$.
3. $P'$ is described below.
4. $S'$ is the same as $S$.

$P'$ is constructed as follows:

1. If $a \rightarrow \beta$ is in $P$, then $a' \rightarrow \beta'$ is in $P'$ where $a'$ is $a$ with all terminals replaced by the corresponding element from $\Sigma''$, and $\beta'$ is $\beta$ with its terminals replaced by the corresponding nonterminal.
2. If $a \in \Sigma$ and appears in $P$, then add $(a) \rightarrow a$.
3. These are all the elements of $P'$.

Claim: $G'$ is the required grammar.
Proof:
If $\gamma \in \text{language}(G')$ and $\gamma_0', \gamma_1', \ldots, \gamma_n'$ is a derivation for $\gamma$, then construct a new sequence $\gamma_0'', \gamma_1'', \ldots, \gamma_n''$ such that for all $n' \geq i \geq 0$, $\gamma_i''$ is the same as $\gamma_i'$ except that each element of $\Sigma''$ in $\gamma_i'$ is replaced by the corresponding element from $\Sigma$. Therefore, it must be true that if $n' \geq i \geq 0$ then

$$S \Rightarrow \gamma_i''$$

since, by induction

Basis: If $i = 0$, then $\gamma_0'' = S$ but

$$S \Rightarrow S$$

by definition.

Induction step: Assume that if $i' \leq i$ then

$$S \Rightarrow \gamma_i''$$

Since $\gamma_i' \Rightarrow \gamma_{i+1}'$, it must be true that the production used in the construction of $\gamma_{i+1}'$ either come from the first half of the construction for $P'$ or the second half. If it is from the second half, that is, of the form

$$(a) \rightarrow a$$

then $\gamma_i'' = \gamma_{i+1}''$ since $\gamma_i''$ and $\gamma_{i+1}''$ differ only in that the chosen element in $\Sigma''$ is replaced by its corresponding element in $\Sigma$. If it is from the first half, then
\[ \gamma_i^{\prime\prime} \Rightarrow \gamma_{i+1}^{\prime\prime} \]

since by definition \( \gamma_i = \gamma_{i1}\alpha'\gamma_{i3} \) and

\[ \gamma_{i+1} = \gamma_{i1}\beta'\gamma_{i3} \]

where \( \gamma' \Rightarrow \beta' \in P' \). But, by construction this implies that

\[ \alpha \Rightarrow \beta \in P \]

where \( \alpha \) is the same as \( \alpha' \) except elements of \( I'' \) are replaced by the corresponding elements of \( \Sigma \) etc. Also,

\[ \gamma_i^{\prime\prime} = \gamma_{i1}^{\prime\prime} \alpha \gamma_{i3}^{\prime\prime} \]

and

\[ \gamma_{i+1}^{\prime\prime} = \gamma_{i1}^{\prime\prime} \beta \gamma_{i3}^{\prime\prime} \]

where \( \gamma_{i1}^{\prime\prime} \) and \( \gamma_{i3}^{\prime\prime} \) are the same as \( \gamma_{i1} \) and \( \gamma_{i3} \) respectively except the elements of \( I'' \) have been replaced by the corresponding elements of \( \Sigma \). But this implies that

\[ \gamma_i^{\prime\prime} \Rightarrow \gamma_{i+1}^{\prime\prime} \]

But since \( S \Rightarrow \gamma_i^{\prime\prime} \) it must be true that

\[ S \Rightarrow \gamma_{i+1}^{\prime\prime} \]

and by induction the statement that

\[ S \Rightarrow \gamma_1^{\prime\prime} \]
is established. Notice that $\gamma_n'$ does not contain any elements of $\Sigma''$ since it consists only of terminal symbols. Therefore, $\gamma_n' = \gamma_n''$ and since $\gamma_n' = \gamma$, $\gamma \in \text{language}(G)$

Therefore,

$$\text{language}(G') \subset \text{language}(G)$$

If $\gamma \in \text{language}(G)$ and $\gamma_0, \gamma_1, ..., \gamma_n$ is a derivation for $\gamma$, then construct a new sequence

$$\gamma_0', \gamma_1', ..., \gamma_n', \gamma_{n+1}', ..., \gamma_n''$$

such that if $n \geq i \geq 0$ then $\gamma_i'$ is the same as $\gamma_i$ except that each element of $\Sigma$ in $\gamma_i$ is replaced by the corresponding element of $\Sigma''$. If $n' \geq i \geq n$, then $\gamma_{i+1}$ is the same as $\gamma_i$ except one element of $\Sigma''$ is replaced by the corresponding element of $\Sigma$. Also,

$$\gamma_n', \in \Sigma^*$$

Notice that the construction of $\gamma_{n+1}'$, to $\gamma_n'$ is not unique. This new sequence is a derivation over $G'$ for $\gamma$ and, further, the width of this derivation is the same as the original derivation, since by induction

Basis: if $i = 0$ then $\gamma_0 = S$. and since $S \in V$, $\gamma_0' = S$. Therefore,

$$S \Rightarrow \gamma_0'$$
Induction step: Assume that if $i' \leq i$ then

$$S \Rightarrow \gamma_i'$$

By definition,

$$\gamma_i = \gamma_{i1} \alpha \gamma_{i2}$$

and

$$\gamma_{i+1} = \gamma_{i1} \beta \gamma_{i2}$$

where $\alpha \Rightarrow \beta \in P$. Therefore,

$$\gamma_i' = \gamma_{i1}' \alpha' \gamma_{i3}'$$

and

$$\gamma_{i+1}' = \gamma_{i1}' \beta' \gamma_{i3}'$$

where $\gamma_{i1}' = \gamma_{i1}$ with all the elements of $I$ replaced by the corresponding element of $\Sigma'$ etc. But if that is the case, then $\alpha' \Rightarrow \beta' \in P'$ by construction, so that

$$\gamma_i' \Rightarrow \gamma_{i+1}'$$

Since $S \Rightarrow \gamma_i'$ by induction hypothesis,

$$S \Rightarrow \gamma_{i+1}'$$

If $n' > i \geq n$, then

$$\gamma_i' \Rightarrow \gamma_{i+1}'$$

since the production $(a)\Rightarrow a \in P'$ by construction. Therefore,

$$S \Rightarrow \gamma_i'$$
in this case also. Therefore, by induction, $S \Rightarrow \gamma'_i$ for all $n' \geq i \geq 0$. Since $\gamma'_n \in \Sigma^*$, this new sequence is a derivation. Notice that $\gamma'_n$ is the same as $\gamma_n$ except that all elements of $\Sigma$ are replaced by elements of $\Sigma'$. Also, due to construction, $\gamma'_n$ is the same as $\gamma_n'$ except all the elements of $\Sigma'$ are replaced by the elements of $\Sigma$.

Therefore, $\gamma_n = \gamma'_n$ and

$$\gamma_n' = \gamma$$

so that

$$\gamma \in \text{language}(G')$$

and

$$\text{language}(G) \subset \text{language}(G')$$

Notice that in the above derivation,

$$|\gamma_i| = |\gamma'_i|$$

for all $n \geq i \geq 0$ so that

$$\max_{n \geq i \geq 0}(|\gamma_i|) = \max_{n \geq i \geq 0}(|\gamma'_i|)$$

also when $n' \geq i \geq n$

$$|\gamma_i| = |\gamma'_{i+1}|$$

it must be true that

$$\max_{n \geq i \geq 0}(|\gamma_i|) = \max_{n' \geq i \geq 0}(|\gamma'_i|)$$
Therefore, the width of the two derivations is the same, and the final statement in the lemma is proved.

- - -

Lemma 97:

Given a grammar $G = (\Sigma, V, P, S)$ where

$$P \subseteq (U_{n \geq i > 0} V^i \rightarrow U_{n \geq i \geq 0} V^i) \cup (V \rightarrow \Sigma)$$

with $n > 2$ then there is a grammar $G' = (\Sigma', V', P', S')$ such that

$$P' \subseteq (U_{n > i > 0} V'^i \rightarrow U_{n > i \geq 0} V'^i) \cup (V' \rightarrow \Sigma)$$

such that

$$\text{language}(G) = \text{language}(G')$$

and if $\gamma \in \text{language}(G)$ has a derivation of width $w$ then there is a derivation for $\gamma$ in $G'$ with a width of $w$.

Proof:

Construct $G'$ as follows:

1. $\Sigma' = \Sigma$.
2. $V' = V \cup V X V$.
3. The construction for $P'$ is given below.
4. $S' = S$.

$P'$ is constructed as follows:

1. If $i < n$, $j < n$ and

$$\alpha_1 \alpha_2 \ldots \alpha_i \beta_1 \beta_2 \ldots \beta_j \in P$$

then this production is also in $P'$. 
2. If \( i = n, j < n \) and 

\[ \alpha_1 \alpha_2 \ldots \alpha_i \beta_1 \beta_2 \ldots \beta_j \in P \]

then the productions \( \alpha_1 \alpha_2 \rightarrow (\alpha_1 \alpha_2) \) and 

\[ (\alpha_1 \alpha_2) \alpha_3 \ldots \alpha_i \beta_1 \beta_2 \ldots \beta_j \]

are in \( P' \).

3. If \( i < n, j = n \) and the production given above is in \( P \), then the productions \( \beta_1 \beta_2 \rightarrow \beta_1 \beta_2 \) and 

\[ \alpha_1 \alpha_2 \ldots \alpha_i \rightarrow (\beta_1 \beta_2) \beta_3 \ldots \beta_j \]

are in \( P' \).

4. If both \( i = n, j = n \) and the given production is in \( P \), then the productions \( \alpha_1 \alpha_2 \rightarrow (\alpha_1 \alpha_2) \), 

\( (\beta_1 \beta_2) \rightarrow \beta_1 \beta_2 \) and 

\[ (\alpha_1 \alpha_2) \alpha_3 \ldots \alpha_i \rightarrow (\beta_1 \beta_2) \beta_3 \ldots \beta_j \]

are in \( P' \).

5. These are all the elements of \( P' \).

The proof that this is the required grammar follows the proof of lemma 96 quite closely so only an outline will be given here. If \( \gamma \in \text{language}(G') \) and 

\[ \gamma_0, \gamma_1 \ldots \gamma_j \]

is a derivation for \( \gamma \), then construct a new sequence
such that each nonterminal of the form \((a_1a_2)\) is replaced by the corresponding \(a_1a_2\). A simple inductive proof will then establish that

\[ S \Rightarrow \gamma_i' \]

using grammar \(G\) for all \(i\). Since \(\gamma = \gamma_j = \gamma_j'\),

\[ S \Rightarrow \gamma \]

and

\[ \text{language}(G') \subset \text{language}(G) \]

If \(\gamma \in \text{language}(G)\) and

\[ \gamma_0, \gamma_1, \ldots, \gamma_j \]

is a derivation for \(\gamma\), then construct a derivation for \(\gamma\) over \(G'\) by inserting the following sentential forms into the sequence:

1. If \(\gamma_i \Rightarrow \gamma_{i+1}\) and

\[
\begin{align*}
\gamma_i &= \gamma_{i1} a_1 a_2 \ldots a_k \gamma_{i3} \\
\gamma_{i+1} &= \gamma_{i1} \beta_1 \beta_2 \ldots \beta_j \gamma_{i3}
\end{align*}
\]

where

\[ a_1 a_2 \ldots a_k \Rightarrow \beta_1 \beta_2 \ldots \beta_j \in P \]
If $k < n$ and $j < n$ then do not insert anything between $a_i$ and $a_{i+1}$.

2. If $k = n$ and $j < n$ in the above equations, then insert

$$\gamma_{i1} (a_1a_2)a_3 \ldots a_k \gamma_{i3}$$

between $\gamma_i$ and $\gamma_{i+1}$.

3. If $k < n$ and $j = n$ in the equations given above, then insert

$$\gamma_{i1} (b_1b_2)b_3 \ldots b_j \gamma_{i3}$$

between $\gamma_i$ and $\gamma_{i+1}$.

4. If $k = n$ and $j = n$ in the above equations, then insert

$$\gamma_{i1} (a_1a_2)a_3 \ldots a_k \gamma_{i3}$$

and

$$\gamma_{i1} (b_1b_2)b_3 \ldots b_j \gamma_{i3}$$

between $\gamma_i$ and $\gamma_{i+1}$ in order.

If this new sequence is relabeled as

$$\gamma_0', \gamma_1', \ldots, \gamma_j'$$
then a simple induction would establish that \( \gamma_i' \rightarrow \gamma_{i+1}' \) so that this is a derivation. Since all of the newly added sentential forms are shorter than one or the other of the adjacent sentential forms, the width of this production must be the same as the width of the original production.

---

Lemma 98:

Given a grammar \( G = (\Sigma, V, P, S) \) such that

\[
P \subseteq (V^+ \rightarrow V^*) \cup (V \rightarrow \Sigma)
\]

then there is a grammar \( G' = (\Sigma', V', P', S') \) such that

\[
P' \subseteq (\bigcup_{2 \geq i > 0} V^{i+1} \rightarrow \bigcup_{2 \geq i \geq 0} V^i) \cup (V' \rightarrow \Sigma)
\]

such that

\[
\text{language}(G) = \text{language}(G')
\]

and if \( \gamma \in \text{language}(G) \) has a derivation \( D \) of width \( w \), then there is a derivation for \( \gamma \) in \( G' \) such that the second derivation has a width of \( w \) also.

**Proof:**

This is easily established by a simple induction using lemma 97.

---
Lemma 99:

Given a grammar $G = (\Sigma, V, P, S)$ such that

$$P \subseteq (\bigcup_{i \geq 0} V^i) \cup (V \rightarrow \Sigma)$$

then there is a grammar $G' = (\Sigma', V', P', S')$ where

$$P' \subseteq (VV \rightarrow V) \cup (V \rightarrow VV) \cup (V \rightarrow \lambda) \cup (V \rightarrow \Sigma) \cup (V \rightarrow \Sigma)$$

such that

$$\text{language}(G) = \text{language}(G')$$

and width is preserved.

Proof:

Note that

$$P \subseteq (V \rightarrow VV) \cup (V \rightarrow V) \cup (V \rightarrow \lambda) \cup (VV \rightarrow VV)$$

$$\cup (VV \rightarrow V) \cup (VV \rightarrow \lambda) \cup (V \rightarrow \Sigma)$$

therefore, only productions of the form $AB \rightarrow CD$ and $AB \rightarrow \lambda$ need to be eliminated. Construct $G'$ as follows:

1. $\Sigma' = \Sigma$.

2. $V' = V \cup (VV \rightarrow VV) \cup (VV \rightarrow \lambda)$.

3. The construction of $P'$ is given below.

4. $S' = S$.

$P'$ is constructed as follows:
1. If \( \alpha \rightarrow \beta \in P \) and \( |\alpha| = 1 \) or \( |\beta| = 1 \), then \( \alpha \rightarrow \beta \) is an element of \( P' \).

2. If \( \alpha \rightarrow \beta \in P \), \( |\alpha| \neq 1 \) and \( |\beta| \neq 1 \), then \( \alpha \rightarrow (\alpha \rightarrow \beta) \) and \((\alpha \rightarrow \beta) \rightarrow \beta \) are elements of \( P' \).

Note that the offending productions are eliminated since these are the only productions where \( |\alpha| \neq 1 \) and \( |\beta| \neq 1 \).

To prove that this is the required grammar, use the same approach as was used in lemma 97. If \( \gamma \in \text{language}(G') \) with derivation

\[
\gamma_0, \gamma_1, \ldots, \gamma_n
\]

then replace any nonterminal \((\alpha \rightarrow \beta)\) added in step two of the construction with \( \alpha \) (note \( \beta \) could have been used) to form a new sequence

\[
\gamma_0', \gamma_1', \ldots, \gamma_n'
\]

An inductive proof establishes that \( S \Rightarrow^{*} \gamma_i' \) using grammar \( G \) so

\[
\text{language}(G') \subset \text{language}(G)
\]

If \( \gamma \in \text{language}(G) \) and

\[
\gamma_0, \gamma_1, \ldots, \gamma_n
\]

is a derivation for \( \gamma \), then construct a new sequence
by inserting a new sentential form between any two
sentential forms which use one of the offending production.
The new sentential form is

\[ \gamma_i \alpha \beta \gamma_j \]

where \( \alpha \rightarrow \beta \) is the offending production, \( \gamma_i = \gamma_i \alpha \gamma_i \beta \gamma_j \) are the sentential forms. A simple proof
would show that this is the required derivation. Proving
the lemma.

---

Lemma 100:

Given a grammar \( G = (\Sigma, V, P, S) \) such that

\[ P \subseteq (V \rightarrow V) \cup (V \rightarrow \Sigma) \cup (V' \rightarrow V) \cup (V \rightarrow \lambda) \]

there is a grammar \( G' = (\Sigma', V', P', S') \) where

\[ P' \subseteq (V' \rightarrow V') \cup (V' \rightarrow V' \rightarrow V') \cup (V' \rightarrow \Sigma) \cup (V \rightarrow \lambda) \]

such that

\[ \text{language}(G) = \text{language}(G') \]

and width is preserved.

Proof:

Let \( \text{Right}(G) = \{ X | X \in V \text{ and } X \text{ is on the right hand side of a } \}
production of the form \( A \rightarrow B \text{ in } G \} \). Proof by induction on
the number of elements in \( \text{Right}(G) \).
Basis: If right(G) is null, then G has no production of the form A → B where A and B are nonterminals, so G itself is the required grammar.

Induction step: Assume that if Right(G) has fewer than n elements there is a grammar with the desired properties. If Right(G) has n elements, then choose one element from Right(G), call it X. Form the set

$$L_X = \{ a | a \in V \text{ and } a \to X \in P \} \setminus \{ X \}$$

Construct a new grammar G' = (Σ', V', P', S') as follows:

1. Σ' = Σ.
2. V' = V.
3. The construction of P' is given below.
4. S' = S.

P' is constructed as follows:

1. If a→β ε P and a→β is not of the form γ → X where γ is an element of V, then a→β is in P'.
2. If X→β ε P and a ε L_X, then the production a → β is in P'.
3. If Xβ→γ ε P and a ε L_X, then the production aβ → γ is in P'.
4. If βX→γ ε P and a ε L_X, then the production βa → γ is in P'.
5. If $XX \rightarrow \gamma \in P$ and also $\alpha$ and $\beta \in \mathcal{L}_X$, then the production $\alpha \beta \rightarrow \gamma$ is in $P'$.

Observe that there are no production of the form $\alpha \rightarrow X$ where $\alpha \in V$ in $G'$, and, therefore, $\text{Right}(G') = \text{Right}(G') - \{X\}$.

If $\gamma \in \text{language}(G)$ with derivation

$$\gamma_0, \gamma_1 \ldots \gamma_n$$

then form a new sequence

$$\gamma_0', \gamma_1' \ldots \gamma_n'$$

by deleting all members of the sequence $\gamma_{i+1}$ where

$$\gamma_i = \gamma_i \alpha \gamma_i$$

$$\gamma_{i+1} = \gamma_i \alpha \gamma_{i+1}$$

$\alpha \in V$ and $\alpha \rightarrow X \in P$. It can be shown through a simple inductive proof that the second sequence is a derivation over $G'$ for $\gamma$. Therefore,

$$\text{language}(G) \subset \text{language}(G')$$

Also, notice that each of the deleted sentential forms is the same length as the preceding sentential form in the first sequence. The result is that the width of the new derivation is the same as the width of the original derivation.

If $\gamma \in \text{language}(G')$ with derivation
construct a derivation over \( G \) for \( \gamma \) by inserting

\[
\gamma_1 X \gamma_3
\]

into the sequence between \( \gamma_i \) and \( \gamma_{i+1} \) where

\[
\gamma_i = \gamma_{i1} \alpha \gamma_{i3} \\
\gamma_{i+1} = \gamma_{i1} \beta \gamma_{i3}
\]

and \( \alpha \to \beta \) is one of the production in \( P' \) that is not in \( P \).
A simple inductive proof would then establish that this is a legal derivation for \( \gamma \) over \( G \) so that

\[
\text{language}(G') \subset \text{language}(G)
\]

and the lemma is proved by induction.

---

**Theorem 101:**

Given a grammar \( G = (\Sigma, V, P, S) \) then there is a grammar \( G' = (\Sigma', V', P', S') \) such that

\[
P' \subset (V'V' \to V') \cup (V' \to V'V') \cup (V' \to \Sigma') \cup (V' \to \lambda)
\]

where

\[
\text{language}(G) = \text{language}(G')
\]

and if \( \gamma \in \text{language}(G) \) has a derivation of width \( w \) then there is a derivation over \( G' \) for \( \gamma \) of width \( w \).

**Proof:**
By lemma 96 there is a grammar \( G_1 = (\Sigma_1, V_1, P_1, S_1) \) such that

\[
P_1 \subseteq (V_1^+ \rightarrow V_1^*) \cup (V_1' \rightarrow \Sigma_1')
\]

width is preserved and \( \text{language}(G) = \text{language}(G) \). By lemma 98 there is a grammar \( G_2 = (\Sigma_2, V_2, P_2, S_2) \) such that

\[
P_2 \subseteq (V_2 \rightarrow \Sigma_2^i \rightarrow V_2^i) \cup (V_2' \rightarrow \Sigma_2^i') \cup (V_2 \rightarrow \Sigma_2)
\]

width is preserved and \( \text{language}(G_2) = \text{language}(G) \). By lemma 99 there is a grammar \( G_3 = (\Sigma_3, V_3, P_3, S_3) \) such that

\[
P_3 \subseteq (V_3 \rightarrow V_3^i) \cup (V_3 \rightarrow V_3^i \rightarrow V_3^i) \cup (V_3 \rightarrow \Sigma_3^i) \cup (V_3 \rightarrow \Sigma_3)
\]

width is preserved and \( \text{language}(G_3) = \text{language}(G) \). Finally, by lemma 100 there is a grammar \( G_4 = (\Sigma_4, V_4, P_4, S_4) \) such that

\[
P_4 \subseteq (V_4 \rightarrow V_4^i) \cup (V_4 \rightarrow V_4^i \rightarrow V_4^i) \cup (V_4 \rightarrow \Sigma_4^i) \cup (V_4 \rightarrow \Sigma_4)
\]

width is preserved and \( \text{language}(G_4) = \text{language}(G_3) \). This last grammar, \( G_4 \), is the desired grammar.

Note that this last theorem has the required normal form.

As an example of the normal form, consider the grammar

\[
S \rightarrow aTbc \\
T \rightarrow aTbU \\
T \rightarrow \lambda \\
Ub \rightarrow bU \\
Uc \rightarrow cc
\]
This grammar defines the language \( \{a^n b^n c^n\}_{n>0} \). If the first transformation given in lemma 96 is applied to this grammar, then the grammar becomes

\[
\begin{align*}
S & \rightarrow ATBC \\
T & \rightarrow ATBU \\
T & \rightarrow \lambda \\
UB & \rightarrow BU \\
UC & \rightarrow CC \\
A & \rightarrow a \\
B & \rightarrow b \\
C & \rightarrow c
\end{align*}
\]

If \( n \) is set to 4 and the transformation given in lemma 97 is applied, then the grammar becomes

\[
\begin{align*}
S & \rightarrow VBC \\
V & \rightarrow AT \\
T & \rightarrow WBU \\
W & \rightarrow AT \\
T & \rightarrow \lambda \\
UB & \rightarrow BU \\
UC & \rightarrow CC \\
A & \rightarrow a \\
B & \rightarrow b \\
C & \rightarrow c
\end{align*}
\]
If \( n \) is set to 3 and the transformation is applied again, the grammar becomes

\[
S \rightarrow XC \\
X \rightarrow VB \\
V \rightarrow AT \\
T \rightarrow YU \\
Y \rightarrow WB \\
W \rightarrow AT \\
T \rightarrow \lambda \\
UB \rightarrow BU \\
UC \rightarrow CC \\
A \rightarrow a \\
B \rightarrow b \\
C \rightarrow c
\]

Again, the transformation given in lemma 99 produces the grammar

\[
S \rightarrow XC \\
X \rightarrow VB \\
V \rightarrow AT \\
T \rightarrow YU \\
Y \rightarrow WB \\
W \rightarrow AT \\
T \rightarrow \lambda
\]
Since there are no productions of the form $A \rightarrow B$ where $A$ and $B$ are in $V$, this is the required grammar. If the element $aabbcc$ of the language is selected and the derivation

$$S, aTbc, aaTbUbc, aabUbc, aabbUc, aabbcc$$

is selected, note that width of the derivation is 7. A derivation in the transformed grammar for the same element of width 7 is

$$S, XC, VBC, ATBC, AYUBC, AWBUBC, AATBUBC, AABUBC, AABZC, AABBUC, AABBD, AABBCC, aABBCC, aaBBCC, aabBCC, aabBCc, aabbcC, aabbcc$$

Notice that the first, fourth, seventh, eighth, tenth and twelfth sentential form in the sequence are the same as in the original sequence except all of the terminals are replaced by the respective nonterminals.
Now that a normal form for a grammar has been developed, this form will be used to simplify the proof of the theorem that this appendix is written to provide. This theorem states that if the elements of a language all have a linearly bounded derivation, then the language is context sensitive. It should be noted that only one derivation for each element need be linearly bounded. Other derivations may exist which may or may not be linearly bounded, but they are irrelevant. This fact is used extensively in the containment proof given in this paper.

Lemma 102:

Given a grammar \( G = (\Sigma, V, P, S) \) which is in normal form such that if \( \gamma \in \text{language}(G) \) there exists at least one derivation for \( \gamma \) of width \(|\gamma|\) then there is a grammar \( G' = (\Sigma', V', P', S') \) such that

\[ \text{language}(G) = \text{language}(G') \]

and \( G' \) is context sensitive.

Proof:

Construct grammar \( G' = (\Sigma', V', P', S') \) as follows:

1. \( \Sigma' = \Sigma \).
2. \( V' = V \cup \{ S', Z \} \).
3. The construction for \( P' \) follows.
4. \( S' \) does not appear elsewhere.

\( P' \) is constructed such that

1. \( S' \to S'Z \) and \( S' \to S \) are in \( P' \).
2. If AB-C \in P, then AB \rightarrow CZ is in P'.
3. If A \rightarrow BC \in P, then AZ \rightarrow BC is in P'.
4. If A \rightarrow a \in P, then A \rightarrow a is in P'.
5. If A \rightarrow \lambda \in P, then A \rightarrow Z is in P'.
6. If A \in V' U \Sigma', then AZ \rightarrow ZA and ZA \rightarrow AZ are in P'.

Claim: G' is the required context sensitive language. Note that G' is context sensitive since each production added to P' has the same number of tokens on each side of the arrow with the exception of the very first production.

If \gamma \in \text{language}(G') with derivation

\gamma_0, \gamma_1, \ldots, \gamma_n

then construct a new sequence

\gamma'_0, \gamma'_1, \ldots, \gamma'_n

by deleting all occurrences of the nonterminal Z and replacing the nonterminal S' by S. A simple induction proof will show that S \Rightarrow \gamma'_i over G for all n \geq i \geq 0.

Therefore, in particular,

S \Rightarrow \gamma'_n

and

\text{language}(G') \subset \text{language}(G)
Note that the fact that $G$ is linearly bounded is not used in this part of the proof, so that if this construction is performed on a grammar which is not bounded by $|\gamma|$ then the grammar describes a context sensitive subset of the original grammar $G$.

If $\gamma \in \text{language}(G)$ and

$$\gamma_0, \gamma_1, \ldots, \gamma_n$$

is a derivation for $\gamma$ of width $|\gamma|$, then construct a new sequence as follows:

1. Before $\gamma_0$ insert the sequence

$$S', S'Z, S'ZZ, \ldots, S'|\gamma|-1$$

2. Expand each sentential form in the original sequence to length $|\gamma|$ by putting as many occurrences of $Z$ on the end of the sentential form as are needed. Note that this step is not possible if the original derivation has a width of more than $|\gamma|$.

3. If $\gamma_i = \gamma_{i1} \alpha \gamma_{i3}$ and $\gamma_{i+1} = \gamma_{i1} \beta \gamma_{i3}$ where $\alpha \rightarrow \beta \in P$ and is of the form $AB \rightarrow C$, then insert the sequence of sentential forms

$$\gamma_{i1} \beta Z \gamma_{i3} Z^W$$
$\gamma_{i1} \beta \gamma_{i1}^{1}Z\gamma_{i3}^{2} \cdots \gamma_{i3}^{n'}Z^{w}$

$\gamma_{i1} \beta \gamma_{i1}^{1} \gamma_{i3}^{2} \cdots \gamma_{i3}^{n'-1}Z\gamma_{i3}^{n'}Z^{w}$

each of length $|\gamma|$ between $\gamma_i'$ and $\gamma_{i+1}'$, where

to $w = |\gamma| - |\gamma_{i1}'$ and $\gamma_{i3} = \gamma_{i3}^{1}\gamma_{i3}^{2} \cdots \gamma_{i3}^{n'}$.

4. If the production is of the form $A \rightarrow BC$, then

insert the sequence of sentential forms

$\gamma_{i1}A\gamma_{i3}^{1}\gamma_{i3}^{2} \cdots \gamma_{i3}^{n'-1}Z\gamma_{i3}^{n'}Z^{w}$

$\gamma_{i1}AZ\gamma_{i3}^{1}\gamma_{i3}^{2} \cdots \gamma_{i3}^{n'}Z^{w}$

$\gamma_{i1}AZ\gamma_{i3}^{1}\gamma_{i3}^{2} \cdots \gamma_{i3}^{n'}Z^{w}$

each of length $|\gamma|$ between $\gamma_i'$ and $\gamma_{i+1}'$, where

$w = |\gamma| - |\gamma_{i1}'$ and $\gamma_{i3} = \gamma_{i3}^{1}\gamma_{i3}^{2} \cdots \gamma_{i3}^{n'}$.

5. If the production is of the form $A \rightarrow a$, do not

insert anything between $\gamma_i'$ and $\gamma_{i+1}'$. 
6. If the production is of the form $A \rightarrow \lambda$, then insert the sequence of sentential forms

$$\gamma_i \gamma_i^1 \gamma_i^2 \ldots \gamma_i^{n-1} \gamma_i^w$$

between $\gamma_i$ and $\gamma_{i+1}'$. Label this new sequence as

$$\gamma_0', \gamma_1', \ldots, \gamma_n'$$

This new sequence is a derivation for $\gamma$ over the grammar $G'$, since

1. The first symbol is $S'$, the start symbol for $G'$.
2. If $\gamma_i'$ was added in step one, then the first production in the construction of $P'$ is used to generate $\gamma_i'$.
3. If $\gamma_i'$ is the last sentential form in this set, then the second production is used to generate $\gamma_{i+1}'$. 
4. The sentential forms added in steps 3 to 6 above are applications of the productions added in step 6 of the construction. In general, they move the Zs to where they are needed in order for the productions added in steps 2 through 5 of the construction to be applied.

Since $\Sigma' = \Sigma$ and the last item in the second sequence is equal to $x_n = \gamma$, an element of $\Sigma^*$, $\gamma$ must be an element of $L(G)$ and

$$language(G) = language(G')$$

so the lemma is proved.

---

Note that $\lambda$ cannot be an element of these languages, unless an ad hoc statement is made that the production

$$S \to \lambda$$

is allowed, and is simply carried along as an exception in each case. This is also true of context sensitive (and even context free) languages since the general characterization (production of the form $\alpha \to \beta$ where $|\alpha| \leq |\beta|$) does not allow the null string.

An example of a grammar which is not context sensitive but is bounded as defined in lemma 102 is as follows:
Note that this grammar defined the language

\[ \{a^n b^n c^n\}_{n>0} \]

Since Ub \rightarrow X is a production, this is not a context sensitive grammar. However, the X must eventually be expanded by X \rightarrow bU so that the intermediate sentential forms must all be shorter than the final generated string.

Lemma 103:

Given a Grammar \( G = (\Sigma, \Gamma, P, S) \) in normal form then there is a grammar \( G' = (\Sigma', \Gamma', P', S') \) such that if \( \gamma \in \text{language}(G) \) with a derivation of width \( w \) then there is a derivation over \( G' \) for \( \gamma \) of width \( w' \) such that

\[ w' \leq \max((w+1)/2, |\gamma|, \vert \gamma \vert) \]

and \( \text{language}(G) = \text{language}(G') \).

Proof:
Construct $G' = (\Sigma', V', P', S')$ as follows:

1. $\Sigma' = \Sigma$.
2. $V' = V \cup (VU\Sigma) \times (VU\Sigma) \cup (VU\Sigma) \times (VU\Sigma) \times (VU\Sigma)$.
3. The construction of $P'$ is given below.
4. $S' = S$.

$P'$ is constructed as follows:

1. If $AB \rightarrow C \in P$ and $D$ and $E$ are elements of $V \cup \Sigma$, then $(AB) \rightarrow C$, $(DA)(BE) \rightarrow (D(CE))$ and $A(BE) \rightarrow (CE)$ are in $P'$.
2. If $A \rightarrow BC \in P$ and $D \in V \cup \Sigma$, then $(AD) \rightarrow (BCD)$, $(DA) \rightarrow (DCB)$ and $A \rightarrow (BC)$ are in $P'$.
3. If $A \rightarrow \lambda \in P$ and $D \in V \cup \Sigma$, then $(AD) \rightarrow D$, $(DA) \rightarrow D$ and $A \rightarrow \lambda$ are in $P'$.
4. If $A \rightarrow a \in P$ and $D \in V \cup \Sigma$, then $(AD) \rightarrow (aD)$, $(DA) \rightarrow (Da)$ and $A \rightarrow a$ are in $P'$.
5. If $A$, $B$, $C$, $D$ and $E$ are elements of $V \cup \Sigma$, then:

$$
(AB)C \rightarrow A(BC)
$$

$$
(AB)(CDE) \rightarrow (ABC)(DE)
$$

$$
A(BCD) \rightarrow (AB)(CD)
$$

$$
(ABC) \rightarrow A(BC)
$$

$$
A(BC)D \rightarrow (AB)(CD)
$$

are in $P'$.
6. If $A$ and $B \in \Sigma$, then $(AB) \rightarrow AB$ are in $P'$.

These are arranged in table 42 for easy understanding.
TABLE 42. Construction for Lemma 103

<table>
<thead>
<tr>
<th>Start of Production</th>
</tr>
</thead>
<tbody>
<tr>
<td>(., )</td>
</tr>
<tr>
<td>(., )</td>
</tr>
<tr>
<td>(., )</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th></th>
<th>(AB)-&gt;C</th>
<th>(DA)(BE)-&gt;D(CE)</th>
<th>A(BD)-&gt;(CE)</th>
<th>(AD)-&gt;(BCD)</th>
<th>(DA)-&gt;(DBC)</th>
<th>A-(BC)</th>
</tr>
</thead>
<tbody>
<tr>
<td>A-&gt;BC</td>
<td>(AD)-&gt;(BCD)</td>
<td></td>
<td></td>
<td>(DA)-&gt;(Da)</td>
<td>(DA)-&gt;(Da)</td>
<td></td>
</tr>
<tr>
<td>A-&gt;</td>
<td>(AD)-&gt;D</td>
<td>(DA)-&gt;D</td>
<td>A-(BC)</td>
<td></td>
<td>(DA)-&gt;(Da)</td>
<td>A-&gt;a</td>
</tr>
<tr>
<td>A-&gt;a</td>
<td>(AD)-&gt;(aD)</td>
<td>(DA)-&gt;(Da)</td>
<td></td>
<td></td>
<td>(DA)-&gt;(Da)</td>
<td>A-&gt;a</td>
</tr>
</tbody>
</table>

If $y \in \text{language}(G')$ with derivation

$$\gamma_0, \gamma_1 \cdots \gamma_n$$

then construct the sequence

$$\gamma_0', \gamma_1' \cdots \gamma_n'$$

where $\gamma_i'$ is the same as $\gamma_i$ except every occurrence of (AB) has been replaced by A B and every occurrence of (ABC) has been replaced by A B C. A simple inductive proof shows that

$$S \Rightarrow \gamma_i$$

so

$$\text{language}(G') \subseteq \text{language}(G)$$

If $y \in \text{language}(G)$ with derivation

$$\gamma_0, \gamma_1 \cdots \gamma_n$$
then construct a new derivation by

1. Grouping the characters from each element \( \gamma_i \) into groups of two and putting parentheses around them. If there are an odd number of characters, leave the first by itself.

2. If \( \gamma_i = \gamma_{i1} \alpha \gamma_{i3} \) and \( \gamma_{i+1} = \gamma_{i1} \beta \gamma_{i3} \) where \( \gamma \rightarrow \beta \in P \), construct a new element \( \gamma_{i\alpha} \) which results from using the appropriate production from table 42 depending on the form of \( \alpha \rightarrow \beta \) and the starting position of the production in \( \gamma_i \). If \( \gamma_{i\alpha} \) is the same as the element corresponding to \( \gamma_{i+1} \), then do not insert anything into the new derivation.

3. If \( \gamma_{i\alpha} \) is not the same as the element corresponding to \( \gamma_{i+1} \), then insert \( \gamma_{i\alpha} \) and as many other sentential forms as are needed to bring the extra nonterminal or ordered triple to the front of the production by using the productions of the form (AB)C \( \rightarrow \) A(BC) and (AB)(CDE) \( \rightarrow \) (ABC)(DE). Then use productions of the form

\[
(ABC) \rightarrow A(BC)
\]
\[
A(BC)D \rightarrow (AB)(CD)
\]
or

\[
A(BCD) \rightarrow (AB)(CD)
\]
to combine the elements which are not ordered pairs at the end. These will generate the element corresponding to $\gamma_{i+1}$.

4. Lastly, use productions of the form $(AB) \rightarrow AB$ to expand the element corresponding to $\gamma_n$ back to $\gamma$.

This can be proved to be the required derivation by a simple inductive proof. Note that the width of this derivation is $(w+1)/2$ if the maximum width is reached at or before the element corresponding to $\gamma_n$, $|\gamma|$ if the maximum is reached at the end or 1 if $w = 1$. Therefore, the lemma is proved.

---

**Lemma 104:**

If $G = (\Sigma, V, P, S)$ is a grammar and $f$ is a function such that if $\gamma \in \text{language}(G)$ there exists at least one derivation for $\gamma$ of width $w$ such that

$$w \leq f(\gamma)$$

then if $\alpha$ is an arbitrary element of $\Sigma^*$ it is possible to determine whether or not $\alpha \in \text{language}(G)$.

**Proof:**

Given a string $\alpha$, construct a sequence as follows

1. $\gamma_1 = \alpha$.

2. Given $\gamma_i$ then the construction for $\gamma_{i+1}$ is given below.

$\gamma_{i+1}$ is constructed as follows:
1. Identify all substrings of $\gamma_i$ which are the right hand sides of productions in $P$.

2. Try to replace the substrings one at a time with the corresponding left hand side of the production to generate $\gamma_{i+1}$ until either $\gamma_{i+1}$ does not fail or all the substrings have been tried.

$\gamma_{i+1}$ can fail for any of the following reasons

1. $|\gamma_{i+1}| > f(\alpha)$.
2. $\gamma_{i+1} = \gamma_k$ for any $1 \leq k \leq i$.
3. All of the possible constructions for $\gamma_{i+1}$ fail and $\gamma_{i+1} \neq S$.

Note that if $G$ is a context free grammar, the substrings are tried in left to right order and strings are allowed to fail for reason 3 above only then this algorithm is a bottom-up backtracking parsing algorithm. If $\gamma_1$ succeeds, then $\alpha$ is in language($G$). It is easy to prove in this case that

$$\gamma_n', \gamma_{n-1}', \ldots, \gamma_1$$

where $\gamma_n = S$ is a parse for $\alpha$. Note that this algorithm terminates since points 1 and 2 above limit the length of the derivation to $f(\alpha)!$. Therefore, it can be determined that an arbitrary element of $\Sigma^*$ is in the bounded grammar so the lemma is proved.
Corollary 105:
Given a grammar \( G = (\Sigma, V, P, S) \) such that if 
\( \gamma \in \text{language}(G) \) then there is a derivation for \( \gamma \) of
width \( w \) such that
\[
  w \leq K_1 |\gamma| + K_2
\]
then it can be determined if \( \lambda \) is an element of the
language.

Proof:
This follows directly from lemma 104.

---

Lemma 106:
Given a grammar \( G = (\Sigma, V, P, S) \) such that if
\( \gamma \in \text{language}(G) \) then there is a derivation for \( \gamma \) of
width \( w \) such that
\[
  w \leq K|\gamma|
\]
where \( K \geq 1 \) then there exists a context sensitive
grammar \( G' = (\Sigma', V', P', S') \) such that
\[
  \text{language}(G) = \text{language}(G')
\]
Proof by induction on \( K \):
Basis: If \( K = 1 \), then by Theorem 101 there is a normal form
grammar that is equivalent to \( G \) and by lemma 102 there is a
 corresponding context sensitive language.
Induction step: Assume that if \( K' < K \) then the lemma is true for a grammar bounded by \( K' \). If \( G = (\Sigma, \Gamma, P, S) \) is a grammar that is bounded by \( K \), then by Theorem 101 there is a normal form grammar for language(\( G \)) with the same bound. By lemma 103 there is a corresponding grammar where the derivations for \( y \) are bounded by

\[
w = \max((K|y|+1)/2, |y|, 1).
\]

Since \( K \geq 2 \), it must be true that

\[
(K|y| + 1)/2 > |y|
\]

\(|y| \geq 1\) because \( \lambda \) is not an element of \( G \) (otherwise the width of a derivation for \( \lambda \) must be zero, but that is not possible). Therefore,

\[
w = (K|y|+1)/2
\]

Since the bound must be an integer, \( w = K|y|/2 \) if \( K|y| \) is even. In particular, if \( K = 2 \), then the new bound is \( K|y|/2 \). But \( K > K/2 \) so by induction hypothesis there is a corresponding context sensitive grammar. If \( K > 2 \), then if a production is bounded by \( (K|y|+1)/2 \), it is also bounded by \( (K+1)|y|/2 \). Since \( K \) is an integer, if \( (K+1)/2 \) is not an integer then \( (K+2)/2 \) is an integer and \( (K+2)|y|/2 \) is a bound for production. Now

\[
K = K/2 + K/2 > K/2 + 1
\]
since $K > 2$ and

$$K > \frac{K + 2}{2} > \frac{K + 1}{2}$$

In any event, there is an integer bound for the grammar given by lemma 103 which is less than $K$, so by induction hypothesis there is a context sensitive grammar for the original language and lemma is proved by induction.

---

**Theorem 107:**

Given a grammar $G = (\Sigma, V, P, S)$ such that if $C \in \text{language}(G)$ then there is a derivation for $I$ of width $w$ such that

$$w \leq K_1 |I| + K_2$$

then there exists a context sensitive grammar $G' = (\Sigma', V', P', S')$ such that

$$\text{language}(G) = \text{language}(G')$$

**Proof:**

$K_1 \geq 1$ since if that were not the case, when $|I|$ becomes greater than $K_2/(1 - K_1)$ then the length of a derivation must be less than $|I|$ which is impossible. If $K_2 < 0$, then $K_1 |I|$ is also a bound. If $\lambda$ is not an element of $G$ (this can be determined by corollary 105), then in all cases,

$$K_1 |I| + K_2 \leq (K_1 + K_2) |I|$$
and by lemma 106 the required context sensitive grammar exists. If $\lambda \in G$, then the constructions used in the proof of lemma 106 will result in equivalent grammar describing the set language($G$) - {$\lambda$}. Therefore, add a new start symbol $S'$ and two productions

$$S' \rightarrow S$$

$$S' \rightarrow \lambda$$

and this is the required grammar.

- - -

For examples of linearly bounded grammars, see chapter 8 of this paper.