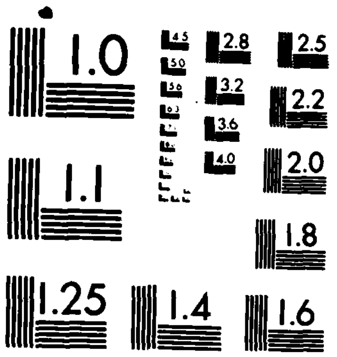
AD-A162 716 GPSG (GENERALIZED PHRASE STRUCTURE GRAMMAR) RECOGNITION IS NP HARD(U) MASSACHUSETTS INST OF TECH CAMBRIDGE ARTIFICIAL INTELLIGENCE LAB E S RISTAD MAR 85 AI-M-837 1/1

UNCLASSIFIED N00014-80-C-0505 F/G 5/7 NL



MICROCOPY RESOLUTION TEST CHART
NATIONAL BUREAU OF STANDARDS-1963-A

2

UNCLASSIFIED

SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

REPORT DOCUMENTATION PAGE		READ INSTRUCTIONS BEFORE COMPLETING FORM
1. REPORT NUMBER 837	2. GOVT ACCESSION NO.	3. RECIPIENT'S CATALOG NUMBER
4. TITLE (and Subtitle) GPSG- Recognition is NP Hard		5. TYPE OF REPORT & PERIOD COVERED A.I. Memo 837
		6. PERFORMING ORG. REPORT NUMBER
7. AUTHOR(s) Eric Sven Ristad		8. CONTRACT OR GRANT NUMBER(s) N0014-80-C-0505
9. PERFORMING ORGANIZATION NAME AND ADDRESS Artificial Inteligence Laboratory 545 Technology Square Cambridge, MA 02139		10. PROGRAM ELEMENT, PROJECT, TASK AREA & WORK UNIT NUMBERS
11. CONTROLLING OFFICE NAME AND ADDRESS Advanced Research Projects Agency 1400 Wilson Blvd. Arlington, VA 22209		12. REPORT DATE March, 1985
		13. NUMBER OF PAGES 9
14. MONITORING AGENCY NAME & ADDRESS(if different from Controlling Office) Office of Naval Research Information Systems Arlington, VA 22217		15. SECURITY CLASS. (of this report) Unclassified
		15a. DECLASSIFICATION/DOWNGRADING SCHEDULE

16. DISTRIBUTION STATEMENT (of this Report)

Distribution is unlimited.

17. DISTRIBUTION STATEMENT (of the abstract entered in Block 20, if different from Report)

18. SUPPLEMENTARY NOTES

None

19. KEY WORDS (Continue on reverse side if necessary and identify by block number)

GPSG
Parsing
Complexity
Natural Language
Linguistics
Natural Language Parsing

20. ABSTRACT (Continue on reverse side if necessary and identify by block number)

Proponents of Generalized Phrase Structure Grammar(GPSG) often cite its weak context-free generative power as proof of the computatoinal tractabilty of GPSG-Recognition. It is well known that context-free languages can be parsed in O( ) time by a wide range of algorithms. Hence, it might be thought that GPSG's weak context-free generative power should guarantee that it too is efficiently parsible. This widely-assumed GPSG "efficient parsibilty" result is false: A reduction from 3-Satsisfiabilty proves that GPSG-Recognition is in the class NP-hard, and likely to be intractable. Crucially, GPSG-Recogition is a function of an input string and a grammar, and the GPSG metarule grammar can result in an arbitrarily large finite set of derived context-free rules. It is further demonstrated that a central object in GPSG theory, the metarule inherently results in an intractable recognition problem, even when severely constrained. The implications for linguistics and natural language parsing are discussed

-A-

MASSACHUSETTS INSTITUTE OF TECHNOLOGY
ARTIFICIAL INTELLIGENCE LABORATORY

A.I. Memo No. 837 March, 1985

GPSG-Recognition is NP-Hard

Eric Sven Ristad

cu h

ABSTRACT: Proponents of Generalized Phrase Structure Grammar (GPSG) often cite its weak context-free generative power as proof of the computational tractability of GPSG-Recognition. It is well known that context-free languages can be parsed in $O(n^3)$ time by a wide range of algorithms. Hence, it might be thought that GPSG's weak context-free generative power should guarantee that it too is efficiently parsible. This widely-assumed GPSG "efficient parsibility" result is false: A reduction from 3-Satisfiability proves that GPSG-Recognition is in the class NP-hard, and likely to be intractable. Crucially, GPSG-Recognition is a function of an input string and a grammar, and the GPSG metarule grammar can result in an arbitrarily large finite set of derived context-free rules. It is further demonstrated that a central object in GPSG theory, the metarule, inherently results in an intractable recognition problem, even when severely constrained. The implications for linguistics and natural language parsing are discussed.

This report describes research done at the Artificial Intelligence Laboratory of the Massachusetts Institute of Technology. Support for the Laboratory's artificial intelligence research has been provided in part by the Advanced Research Projects Agency of the Department of Defense under Office of Naval Research contract N00014-80-C-0505. The author is indebted to Ed Barton, Robert Berwick, Noam Chomsky, Wayne Davis, James Higginbotham, and Scott Weinstein for assistance in writing this paper.

85 12 26 025

1 Complexity of GPSG-Recognition

Gazdar(1981) proposes restricting the class of generative grammars for natural languages to a subset of those with weak context-free generative power. Since context-free languages can be parsed in $O(n^3)$ time, Gazdar argues this would provide "the beginnings of an explanation for the obvious, but largely ignored, fact that humans process the utterances they hear very rapidly."[1] In this paper, I prove Gazdar's appeal to general mathematical results on context-free parsing fails: Generalized Phrase Structure Grammar (GPSG) does not have the desired "efficient parsibility" result unless other constraints are added to the theory. As things currently stand, GPSG, Lexical Functional Grammar (LFG), and Transformational Grammar (TG) all have a general "intractable parsibility" result.

A Generalized Phrase Structure Grammar (GPSG) consists, at core, of a finite set of context-free base rules plus a finite set of metarules. Metarules are functions from base rules to sets of base rules.[2] For example, the metarule

$$[\underset{\substack{[+\text{fin}]\\ [+\text{aux}]}}{VP} \rightarrow V\ VP] \implies [VP \rightarrow V\ ADVP\ VP]$$

stipulates that for every context-free rule expanding a finite VP as a finite auxiliary and a nonfinite VP, there will be an identical rule expanding the VP as before, except with an adverb between the auxiliary and nonfinite VP.

Unconstrained metarule application may generate infinite sets of rules and describe arbitrary languages.[3] To preserve both the context-free weak generative power of a GPSG grammar and the supposedly attendant computational benefits, some formal constraints on metarules have been proposed in the GPSG literature. One proposal is to constrain variables in the metarule pattern to be "abbreviatory variables," i.e. variables that can only stand for strings in a finite and extrinsically determined range.[4] While this constraint can affect the extensional language of the grammar in linguistically unmotivated and arbitrary ways, I adopt the constraint in my proof for the purposes of examining its computational implications. Following most of the work in GPSG, I allow metarules to apply recursively to their own outputs, subject to the restriction that the set of rules derived from metarule application is finite. This constraint preserves the context-free generative power of the formalism.

Using only a minimal GPSG grammar, I prove that the recognition problem for GPSG — is a string an element of the language generated by the grammar? — is NP-hard. A problem $T(x)$ is NP-hard if it is at least as hard

[1] Gasdar(1981) p.155.

[2] See Sag et.al.(1984)

[3] For a discussion of this and some proposals for restricting metarules, see Shieber et.al.(1983)

[4] See Shieber et.al.(1983) for a discussion of Gasdar's 1982 proposal.

computationally as any problem in NP: If we had a subroutine that solved T in polynomial time, then we could write a program to solve any problem in NP in polynomial time on a deterministic Turing machine. Since it is widely believed (though not proved) that the hardest problems in NP (NP-complete problems) cannot be solved in polynomial time, NP-hardness is considered equivalent to computational intractability.

To prove that GPSG-Recognition is NP-hard, I reduce 3-Satisfiability (3-SAT), an NP-complete problem, to GPSG-Recognition. A reduction works by converting instances of 3-SAT into instances of GPSG-Recogntion in polynomial time. If we had a polynomial time subroutine for solving GPSG-Recognition, then we could also solve 3-SAT in polynomial time, simply by converting into the GPSG-Recognition problem. Since we know 3-Satisfiability is the hardest problem in NP (NP-complete), then the reduction shows that GPSG-Recognition is at least as hard as all problems in NP (NP-hard).

The 3-Satisfiability problem is: Given a boolean formula in conjunctive normal form (CNF) where each clause contains exactly three variables, is there some assignment of truths to the formula's variables that will make the formula true? An example of a satisfiable 3-CNF boolean formula with five clauses (where + indicates disjunction) is:

$$(a+b+c)(\bar{a}+d+g)(z+\bar{g}+\bar{y})(b+c+d)(\bar{a}+\bar{d}+\bar{z})$$

1.1 GPSG-Recognition is NP-hard

Theorem 1: GPSG-Recognition is NP-hard.

Proof: On input 3-CNF formula F of length m using n variables, reduce 3-SAT to GPSG-Recognition in polynomial time.

Let

w	be the string resulting from removing all parenthesis and disjunct symbols from F
q	be the n variables in some canonical order, where q_i denotes the i^{th} variable
x_i	denote the i^{th} symbol in the string x
S	be the distinguished start symbol
X, Y, Z, p	be special symbols denoting members of $\{0,1\}^*$
$A, B, 0, 1, \#, \vert$	be special symbols

We construct a GPSG grammar G s.t. the special symbol # is an element of $L(G)$ iff F is satisfiable. G contains

1. $n+1$ base rules:

the i^{th} rule: $[A \rightarrow w \mid 1^i 0^{n-i}]$ where $0 \leq i \leq n$

2. the ten metarules

$$
\begin{array}{lll}
[A \rightarrow w \mid XaYbZ] & \Longrightarrow & [A \rightarrow w \mid XbYaZ] \\
 & & \text{where } a, b \in \{0,1\} \\
[A \rightarrow w \mid x] & \Longrightarrow & [B \rightarrow w' \mid x] \\
 & & \text{where if } w_i = \overline{q_j} \text{ then } w'_i = \begin{cases} 1 & \text{if } x_j = 0 \\ 0 & \text{if } x_j = 1 \end{cases} \\
[B \rightarrow w' \mid x] & \Longrightarrow & [S \rightarrow \#w''] \\
 & & \text{where if } w'_i = q_j \text{ then } w''_i = x_j \\
[S \rightarrow \#001p] & \Longrightarrow & [S \rightarrow \#p] \\
[S \rightarrow \#010p] & \Longrightarrow & [S \rightarrow \#p] \\
[S \rightarrow \#100p] & \Longrightarrow & [S \rightarrow \#p] \\
[S \rightarrow \#011p] & \Longrightarrow & [S \rightarrow \#p] \\
[S \rightarrow \#101p] & \Longrightarrow & [S \rightarrow \#p] \\
[S \rightarrow \#110p] & \Longrightarrow & [S \rightarrow \#p] \\
[S \rightarrow \#111p] & \Longrightarrow & [S \rightarrow \#p]
\end{array}
$$

The first metarule, in conjunction with the $n+1$ base rules, generates all possible truth assignments for the n variables. The second metarule instantiates negated variables, and the third metarule instantiates unnegated variables. The last seven metarules determine whether the assignment that was guessed satisfies the formula F. All variables in the ten metarules are either constants (e.g. w) or "abbreviatory variables" (e.g. p) which stand for a string in $\{0,1\}^*$ of length less than m. If these variables were removed, $O(m)$ new rules would replace each existing rule which makes use of an "abbreviatory variable."

The rule set resulting from metarule application is guaranteed finite because no metarule derives rules longer than its input rules. The reduction can be performed in polynomial time (exactly $O(m^2)$ time) because there are n base rules of length $O(n)$, $n < m$, and only a constant number of metarules. *Q.E.D.*

Example reduction:

<u>**Let**</u>

$$
\begin{array}{lll}
F & = & (a+b+c)(a+b+\bar{c})(a+\bar{b}+c)(a+\bar{b}+\bar{c})(\bar{a}+b+c)(\bar{a}+b+\bar{c})(\bar{a}+\bar{b}+c) \\
w & = & abcab\bar{c}a\bar{b}ca\bar{b}\bar{c}\bar{a}bc\bar{a}b\bar{c}\bar{a}\bar{b}c \\
q & = & abc
\end{array}
$$

The grammar we construct for F contains

1. $n + 1$ base rules

$$[A \to w \mid 000] \quad [A \to w \mid 100] \quad [A \to w \mid 110] \quad [A \to w \mid 111]$$

2. metarule 1 generates the remaining possible truth assignments, deriving 4 additional rules

$$[A \to w \mid 010] \quad [A \to w \mid 001] \quad [A \to w \mid 101] \quad [A \to w \mid 011]$$

3. metarule 2 instantiates negated variables, deriving 8 additional rules

$$\begin{array}{c}
[B \to abcab1a1ca111bc1b111c \mid 000] \\
[B \to abcab1a1ca110bc0b101c \mid 100] \\
[B \to abcab1a0ca010bc0b100c \mid 110] \\
[B \to abcab0a0ca000bc0b000c \mid 111] \\
[B \to abcab1a0ca011bc1b110c \mid 010] \\
[B \to abcab0a1ca101bc1b011c \mid 001] \\
[B \to abcab0a1ca100bc0b001c \mid 101] \\
[B \to abcab0a0ca001bc1b010c \mid 011]
\end{array}$$

4. metarule 3 instantiates unnegated variables, deriving 8 additional rules

$$\begin{array}{c}
[S \to \#00000101001110010110] \\
[S \to \#10010111011100000101 0] \\
[S \to \#11011110010101001100 0] \\
[S \to \#11111010110001101000 1] \\
[S \to \#01001100000111011110 0] \\
[S \to \#00100001101010110011 1] \\
[S \to \#10110011111000100001 1] \\
[S \to \#01101000100011111010 1]
\end{array}$$

5. metarules 4 through 10 check if the assignment guessed satisfies the formula F, generating 28 additional rules

$[S \rightarrow \#101110111000001010]$ $[S \rightarrow \#111100101010011000]$
$[S \rightarrow \#110101100011010001]$ $[S \rightarrow \#011000001110111100]$
$[S \rightarrow \#000011010101100111]$ $[S \rightarrow \#100111110001000011]$
$[S \rightarrow \#010001000111110101]$

$[S \rightarrow \#110111000001010]$ $[S \rightarrow \#100101010011000]$
$[S \rightarrow \#101100011010001]$ $[S \rightarrow \#000001110111100]$
$[S \rightarrow \#111110001000011]$ $[S \rightarrow \#001000111110101]$

$[S \rightarrow \#111000001010]$ $[S \rightarrow \#101010011000]$
$[S \rightarrow \#100011010001]$ $[S \rightarrow \#110001000011]$
$[S \rightarrow \#000111110101]$

$[S \rightarrow \#000001010]$ $[S \rightarrow \#010011000]$
$[S \rightarrow \#011010001]$ $[S \rightarrow \#001000011]$

$[S \rightarrow \#011000]$ $[S \rightarrow \#010001]$
$[S \rightarrow \#000011]$

$[S \rightarrow \#000]$ $[S \rightarrow \#001]$

$[S \rightarrow \#]$

Note $\# \in L(G)$ because the rule $[S \rightarrow \#]$ was derived, so the formula is satisfiable (with the assignment $abc = 111$, as the reader can verify).

1.2 Generative Power and Computational Complexity

At first glance, a proof that GPSG-Recognition is NP-hard appears to contradict the context-free generative power result noted above. After all, there exist a wide range of algorithms capable of recognizing context-free languages in time $O(n^3)$.

The connection between weak generative power and efficient parsibility is not as direct as was evidently assumed. The crux of the matter is that a GPSG grammar M can result in a context-free rule set M' of any size whatsoever, subject only to the restriction that it be finite. M' can be of size $O(2^k)$ where $|M| = k$, or even significantly larger. Context-free parsers like the Earley algorithm actually run in time $O(|G|^2 \cdot n^3)$ where $|G|$ is the context-free grammar size and n the input length, so the hypothetical GPSG grammar M will be recognized in time $O(4^k n^3)$. The exponential term will clearly dominate the Earley algorithm complexity in the reduction above because $|M|$ is a function of the length m of the input formula F. Even if the GPSG grammar is held constant, the exponential increase in derived grammar size will result in an astronomical constant factor, which in turn will dominate the real-world performance of the Earley algorithm for all expected inputs (i.e. those of a million words or less).

1.3 Restricting Metarule Application

Since the central problem is that a metarule is capable of deriving any finitely large set of rules, including exponentially large ones, we must further constrain metarule application if we wish to solve the GPSG-Recognition problem in polynomial time (i.e. obtain an efficient parsibility result).

Let us imagine severely restricting the derivational power of metarules as follows:[5]

1. a metarule may only operate once in the derivation of a given rule (i.e. not recursively on its own output).

2. a metarule is a function from one context-free rule to another context-free rule (and not from a rule to a set of rules).

3. all metarule variables either are constants (e.g. w) or stand for a single symbol and can only have two possible values (e.g. x can only stand for the symbols $0, 1$). Note that this is more restrictive than Gazdar's abbreviatory variables constraint.

4. no unrecoverable deletion occurs in the derivation of a given context-free rule.

5. no two metarules or basic rules are identical either in pattern or function.

Even in such a severely restricted system, GPSG-Recognition will be NP-hard. An $O(m^3)$ time reduction is included in appendix A. Since each metarule application can double the size of the grammar G, and there are now $O(m^2)$ metarules, the resulting grammar will still be size $O(|G| \cdot 2^{m^2})$, or still exponentially larger than G, and the Earley algorithm recognition time will be $O((|G| \cdot 2^{m^2})^2 m^3) = O(4^{m^2} m^9)$.

A list of restrictions necessary to remove GPSG-Recognition from the class NP-hard is:[6]

[5]As far as I know, these three restrictions together go beyond anything discussed in the openly available GPSG literature.

[6]In order to guarantee that these four restrictions are sufficient, GPSG must be completely and exactly formally specified, in a manner which ensures that proliferation of categories will not make the recognition problem intractable. Another aspect of current GPSG formulations which make them NP-hard — and probably intractable — is the Immediate Dominance/Linear Precedence (ID/LP) formalism. See Barton(1984) for a proof.

Note that the linguistically untenable restriction of prohibiting metarule variables of any kind is probably sufficient, when coupled with ID/LP restrictions, to guarantee polynomial time recognition. Such a restriction would mean that a metarule, which may only "match" one basic rule, can only derive exactly one rule. The size of the derived context-free rule

1. strictly bounded "chaining" only a constant number of metarules, fixed in advance for all GPSG grammars, can operate in the derivation of a given context-free rule.
2. each metarule may derive a rule set only polynomially bigger than its input rule set.
3. a metarule may only use "abbreviatory variables."
4. metarules are unable to apply to other metarules.

The simplest effective restriction, and one that is not much different from the list of restrictions above, is to remove metarules from GPSG altogether. Since metarules can only derive polynomially large rule sets and limited chaining will make metarule interaction very unpredictable, the linguist would be better off writing out the context-free rule set in advance, and ignoring metarules altogether.

On the other hand, eliminating metarules poses its own problems. Typical GPSG systems result in very large sets of derived rules — literally trillions of rules — and writing out these context-free rule sets in advance would not be feasible.[7] Furthermore, a context-free grammar lacking metarules fails to capture linguistically significant relations between rules. This significant (and apparently unresolvable) metarule-inclusion issue plagues us only if we maintain the "context-free weak generative power" framework. The obvious next step is to abandon that framework.

1.4 Conclusion

The moral of my proof is that as far as we know, it is not possible to appeal to general mathematical results to rescue "efficient parsibility" results. Specific constraints on the particular representations postulated by linguistic theory are needed to explain efficient parsibility. This does not imply that GPSG theory is without merit: on the contrary, I have merely shown that its particular efficient parsibilty thesis cannot be maintained. Generalized Phrase Structure Grammar, Lexical Functional Grammar, and Transformational Grammar are all probably intractable in an abstract mathematical sense, and each theory must search elsewhere for an explanation of efficient parsibility, if one is to be given at all.[8]

set would be the size of the basic rule set plus the number of metarules. This restriction is linguistically unmotivated because it fails to capture linguistically important generalisations. For example, any metarule applying to singular and plural sentences would have to be replicated at least twice: once to handle the singular case, and once to handle the plural case.

[7]See Shieber(1983) p.137

[8]See Berwick and Weinberg(1984) for a discussion of LFG complexity.

The first order of business is to discover an explanatory computational theory of natural language, which identifies the goal of the computation, explains why it is appropriate, and describes the logic of the strategy by which it can be carried out. Once we develop the computational theory and more clearly understand the language faculty, then we can devise algorithms and even postulate neurological mechanisms which implement the algorithms. No magical mathematical result is likely to rescue us from this hard work.

Metagrammatical devices should, I believe, be avoided in principle for both linguistic and computational reasons. Such devices try to describe some regularity or generality not expressible in the core grammar. A more powerful approach would discover a representation or principle that explains the given phenomenon, rather than merely describing it. Instead of resorting to stipulatory language descriptions, linguists should search for grammatical principles and representations that combine in an explanatory theory.

2 References

Barton, E. (1984). "On the Complexity of ID/LP Parsing," A.I. Memo No. 812, M.I.T. Artificial Intelligence Laboratory, Cambridge, Mass.

Berwick, R. and A. Weinberg (1984). *The Grammatical Basis of Linguistic Performance.* M.I.T. Press: Cambridge.

Earley, J. (1970). "An Efficient Context-Free Parsing Algorithm," *Communications of the ACM* 13.2, pp.94–102.

Garey, M., and D. Johnson (1979). *Computers and Intractability.* W. H. Freeman and Co.: San Francisco.

Gazdar, G. (1981) "Unbounded Dependencies and Coordinate Structure," *Linguistic Inquiry* 12, pp.155–184.

Marr, D. (1982). *Vision.* W. H. Freeman and Co.: San Francisco.

Sag, I., G. Gazdar, T. Wascow, S. Weisler (1984) "Coordination and How to Distinguish Categories," *CSLI Report CSLI-84-3.*

Shieber, S. (1983) "Direct Parsing of ID/LP Grammars," *Linguistics and Philosophy* 7, pp.135–154.

Shieber, S., S. Stucky, H. Uszkoreit, and J. Robinson (1983). "Formal Constraints on Metarules," *1983 ACL Conference Proceedings*, pp.22–27.

A Reduction from 3-SAT to GPSG-Recognition

The reduction makes use of metarule schemata solely to increase clarity of exposition.

Theorem 2: GPSG-Recognition is NP-hard, even when severely constrained.

Proof: On input 3-CNF formula F of length m using n variables, reduce 3-SAT to GPSG-Recognition in polynomial time.

<u>Let</u>

w	be the string resulting from removing all parenthesis and disjunct symbols from F
q	be the n variables in some canonical order, where q_i denotes the i^{th} variable
y^i	denote the string in $\{y\}^*$ of length i
x_i	denote the i^{th} symbol in the string x
S	be the distinguished start symbol
X, Y, Z, p	be special symbols denoting members of $\{0,1\}^*$
$A, B, 0, 1, \#, \vert$	be special symbols

As above, we construct a GPSG grammar G s.t. the special symbol # is an element of $L(G)$ iff F is satisfiable. G contains

1. $n+1$ base rules:

$$\text{the } i^{th} \text{ rule: } [A \rightarrow w \mid 1^i 0^{n-i}] \quad \text{where } 0 \leq i \leq n$$

2. the metarules

   (a) $(n+1)^2$ metarules, to generate all possible truth assignments, described by the schema

$$[A \rightarrow w \mid x^i a x^j b x^k] \implies [A \rightarrow w \mid x^i b x^j a x^k]$$

   where $a, b, x \in \{0,1\}$, $0 \leq i, j \leq n$, and $k = n - i - j - 2$.

   (b) two metarules, to instantiate negated and unnegated variables

$$[A \rightarrow w \mid x] \implies [B \rightarrow w' \mid x]$$
$$\text{where if } w_i = \overline{q_j} \text{ then } w'_i = \begin{cases} 1 & \text{if } x_j = 0 \\ 0 & \text{if } x_j = 1 \end{cases}$$
$$[B \rightarrow w' \mid x] \implies [S \rightarrow \# w'']$$
$$\text{where if } w'_i = q_j \text{ then } w''_i = x_j$$

(c) $7n$ metarules, to check that the guessed assignment satisfies F, described by the 7 schemata

$$\begin{array}{lcl} [S \rightarrow \#001x^{3k}] & \Longrightarrow & [S \rightarrow \#x^{3k}] \\ [S \rightarrow \#010x^{3k}] & \Longrightarrow & [S \rightarrow \#x^{3k}] \\ [S \rightarrow \#100x^{3k}] & \Longrightarrow & [S \rightarrow \#x^{3k}] \\ [S \rightarrow \#011x^{3k}] & \Longrightarrow & [S \rightarrow \#x^{3k}] \\ [S \rightarrow \#101x^{3k}] & \Longrightarrow & [S \rightarrow \#x^{3k}] \\ [S \rightarrow \#110x^{3k}] & \Longrightarrow & [S \rightarrow \#x^{3k}] \\ [S \rightarrow \#111x^{3k}] & \Longrightarrow & [S \rightarrow \#x^{3k}] \end{array}$$

where $0 \le k \le n-1$

The result of applying the two "instantiating" metarules or any metarule described by the last metarule schemata is to change the basic rule's righthand side so that the metarule will not apply to it again.

I now prove that all possible truth assignments can be generated by the first metarule schema above, subject to the restriction that a metarule may only operate once in the derivation of a given rule. This is equivalent to proving that we can generate all binary numbers from 0 to $2^n - 1$ inclusive, using only the $n+1$ binary numbers

$$1^i 0^{n-i} \qquad \text{where} \quad 0 \le i \le n$$

and $(n+1)^2$ metarules described by the schema

$$[x^i a x^j b x^k] \quad \Longrightarrow \quad [x^i b x^j a x^k]$$

where $a, b, x \in \{0,1\}$, $0 \le i, j \le n$, and $k = n - i - j - 2$. These metarules perform the action of exchanging any two given bit positions in a binary number of length n.

<u>Let</u>

x	be a binary number with k $1's$ in its binary representation, $0 \le x < 2^n$
$y = 1^k 0^{n-k}$	be the k^{th} binary number

The algorithm found in figure 1, expressed in a generic programming language, derives x from y using the metarules. No metarule can be applied twice because i and j are different everytime a metarule is applied (in line 6). In any given derivation, clearly at most $\lceil \frac{n}{2} \rceil$ metarules are applied (see line 1), and exactly $min(k, n-k)$ metarules are applied in any given derivation. *Q.E.D.*

Figure 1: Algorithm to derive x from y using metarules.

```
procedure = derive(x,y)
1:    for $i = 1$ to $\lceil \frac{n}{2} \rceil$
2:        if $x_i \neq y_i$ then do
3:            for $j = n$ to 0 step $-1$
4:                if $x_j \neq y_j$ then goto 6
5:            next $j$
6:            apply metarule $[x^{i-1}ax^{j-i-1}bx^{n-j}] \Longrightarrow [x^{i-1}bx^{j-i-1}ax^{n-j}]$ to $y$,
                  switching bit positions $i$ and $j$ in the number $y$
7:    next $i$
```

Example derivation:

Let

$x = 0010011110$ be a binary number with 5 $1's$ in its binary representation.

$y = 1111100000$ be the 5th binary number

y	metarule used	next y
1111100000	$[axxxxxxxbx] \Longrightarrow [bxxxxxxxax]$	0111100010
0111100010	$[xaxxxxxbxx] \Longrightarrow [xbxxxxxaxx]$	0011100110
0011100110	$[xxxaxxbxxx] \Longrightarrow [xxxbxxaxxx]$	0010101110
0010101110	$[xxxxabxxxx] \Longrightarrow [xxxxbaxxxx]$	0010011110