On the Lengths of Proofs in the Propositional Calculus

Robert A. Reckhow

A thesis submitted in conformity with the requirements for the degree of Doctor of Philosophy in the Department of Computer Science, University of Toronto, Toronto, Ontario, Canada

© 1975 Robert A. Reckhow

On the Lengths of Proofs in the Propositional Calculus

Robert A. Reckhow

A thesis submitted in conformity with the requirements for the degree of Doctor of Philosophy in the Department of Computer Science, University of Toronto, Toronto, Ontario, Canada

© 1975 Robert A. Reckhow

ABSTRACT

Just as $\mathcal{P} = \mathcal{NP}$ if and only if some $\mathcal{NP}$-complete set is a member of $\mathcal{P}$, the class $\mathcal{NP}$ is closed under complementation if and only if the complement of some $\mathcal{NP}$-complete set is a member of $\mathcal{NP}$. This in turn leads to the fact that $\mathcal{NP}$ is closed under complementation if and only if there exists a system for proving tautologies of the propositional calculus in which each tautology has a (polynomial-time verifiable) proof whose length is no greater than some fixed polynomial in the length of the tautology. Such a system is called a polynomial-bounded verification system.

Most of the important proof systems for the propositional calculus that have been proposed in the literature have been investigated, and two types of results are reported. The first are simulation results that show that if one system is polynomial-bounded, then the simulating system is also polynomial-bounded. Such simulations are shown for all Frege systems (called "Hilbert-type" systems by Kleene), natural deduction systems, and Gentzen systems with cut. Frege system with the substitution rule simulate all other systems studied. The second type of results are lower bounds, where certain systems are shown not to be polynomial-bounded. Lower bounds are reported for regular resolution, semantic trees, analytic tableaux, and other systems, and Tseitin's lower bound for regular resolution is improved and applied to certain systems of regular resolution with limited extension.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF THEOREMS, LEMMAS, AND COROLLARIES

ACKNOWLEDGEMENTS

I would like to thank all of those who helped me in the preparation of this thesis. I greatly appreciated the financial assistance I received from the National Research Council of Canada and the University of Toronto. I would like to thank my supervisor, S. A. Cook, for introducing me to this area of research, and for his advice and encouragement throughout my career as a graduate student. Special thanks are also due to Prof. D. A. Clarke and Dr. R. Solovay for serving as my internal and external readers. Finally, the biggest thanks of all go to my wife, Diane, for the encouragement and patience that kept me going all these years, and for typing this long and difficult manuscript.

Robert A. Reckhow
November 1975

1. Introduction

This thesis is concerned with the computational complexity of proof systems for the propositional calculus. In particular it is an investigation into the relationship between the length of a Boolean formula and the length of the shortest proof of that formula in various proof systems. Besides its intrinsic interest to logicians, this relation is also of importance in the theory of computational complexity, because of its relationship to the $\mathcal{P}$ vs $\mathcal{NP}$ question.

1.1. The $\mathcal{P}$ vs $\mathcal{NP}$ question

An important area of study in computational complexity is the general area of nondeterminism. The question asked is, "under what circumstances are resource-bounded nondeterministic machines able to compute more than deterministic machines with the same resource bounds?" For many problems it is easy to find very efficient nondeterministic algorithms, but straightforward simulation of these algorithms by deterministic machines leads to inefficient solutions.

An early example of a nondeterminism question is known as the "LBA problem". A Turing machine whose storage is limited to some constant times the length of its input is called a *linear-bounded automaton* (LBA). It was shown by Kuroda [Kuroda 1964] that the class of languages accepted by nondeterministic LBAs is the class of context-sensitive languages. It remains an open question, however, whether or

not there exists a context-sensitive language that cannot be accepted by a deterministic LBA. Thus, it remains unknown whether or not nondeterministic LBAs are strictly more powerful than deterministic LBAs.

In a similar vein, but this time concerning computation time rather than storage, Cook [Cook 1971b] defined classes, now called $\mathcal{P}$ and $\mathcal{NP}$. These are the classes of languages accepted (or problems solved) in polynomial time by deterministic and nondeterministic Turing machines, respectively. He observed that a number of combinatorial problems with no known deterministic polynomial-time algorithms have simple polynomial-time nondeterministic algorithms. The obvious question then is, "is $\mathcal{P} = \mathcal{NP}$?" And if $\mathcal{P} \neq \mathcal{NP}$, are any of these problems in $\mathcal{NP} - \mathcal{P}$? Cook answered this second question by showing that certain problems (known as $\mathcal{NP}$-complete problems), including the satisfiability problem for propositional calculus, belong to $\mathcal{P}$ if and only if $\mathcal{NP}$ is identical to $\mathcal{P}$. Since Cook's proof is constructive, a deterministic polynomial-time decision procedure for propositional calculus (if it exists and can be found) would effectively yield deterministic polynomial-time algorithms for all problems in $\mathcal{NP}$, many of which are not now known to have such algorithms. Conversely, if it could be shown that the satisfiability problem for propositional calculus has no deterministic polynomial-time algorithm, then it would follow that $\mathcal{P} \neq \mathcal{NP}$, and as a byproduct would come the result that none of the $\mathcal{NP}$-complete problems has a deterministic polynomial-time algorithm.

1.2. POSSIBLE CLOSURE OF $\mathcal{NP}$ UNDER COMPLEMENTATION AND POLYNOMIAL-BOUNDED VERIFICATION SYSTEMS

Another question that can be asked is whether $\mathcal{NP}$ is closed under complementation (*i.e.* if the complement of every language from $\mathcal{NP}$ is also in $\mathcal{NP}$). If not, then $\mathcal{P} \neq \mathcal{NP}$, since $\mathcal{P}$ is clearly closed under complementation. On the other hand, there are many languages from $\mathcal{NP}$ (including all known $\mathcal{NP}$-complete languages) whose complements are not known to be in $\mathcal{NP}$. It is shown in this thesis that $\mathcal{NP}$ is closed under complementation if and only if the complement of any $\mathcal{NP}$-complete language is in $\mathcal{NP}$. It is furthermore shown that $\mathcal{NP}$ is closed under complementation if and only if there exists a proof system for tautologies of the propositional calculus (where proofs can be verified for correctness in polynomial time) and a polynomial p such that every tautology of length n has a proof of length no greater than $p(n)$. This result motivates the investigation of lengths of proofs in systems for proving propositional tautologies.

1.3. PROOF SYSTEM SURVEY

Logicians have proposed a great number of systems for proving theorems. These systems give certain rules for constructing proofs and for associating a theorem (formula) with each proof. It is important that these rules are much simpler to understand than the theorems. Thus, a proof gives a constructive way of understanding that a theorem is true. A proof system is *sound* if every theorem is true, and it is

complete if every true statement (from a certain class) is a theorem (*i.e.* has a proof).

Some proof systems prove valid formulas, others prove inconsistent formulas, and some prove either. Also, some systems prove only formulas of certain forms, such as conjunctive normal form. Thus, "true statement" in the preceding paragraph could refer to valid formulas, inconsistent sets of clauses, *etc.*, depending on the proof system.

The most straightforward kind of proof system for propositional formulas is the system of truth tables. For a formula in n variables, one simply writes down the 2^n possible different truth assignments to those variables, and verifies that each one makes the formula true (to prove the formula is valid). This is obviously a great deal of work, and several systems have been derived which allow shorter derivations in some examples. These include reduced truth tables [Kleene 1967], semantic trees [Robinson 1968] [Kowalski & Hayes 1969], and enumeration trees [Galil 1975].

The proof system in Mendelson's introductory logic text [Mendelson 1964] is typical of a large class of proof systems called Frege systems. This system constructively proves valid formulas containing only negations and implications. Three valid formulas are distinguished and called *axioms*. Any instance of an axiom is a theorem. Mendelson's only other rule for constructing proofs is *modus ponens*: if A is a theorem and $A \supset B$ is a theorem, then B is a theorem.

By allowing proofs to make use of hypotheses and the deduction theorem (if B can be proved from A, then $A \supset B$ is provable), any Frege system can be made into a "natural deduction" system like those of [Kleene 1967] and [Thomason 1970]. Since the deduction theorem has a long proof, incorporating it into the proof system allows for shorter proofs in many examples.

Another family of proof systems are based on Gentzen's system (see [Kleene 1967]). The objects manipulated by Gentzen's system are *sequents*, which are made up of a list of hypotheses and a list of alternate conclusions for those hypotheses. Gentzen's rules for deriving new sequents from old ones are particularly intuitive, and the notational simplification of [Smullyan 1968] gives rise to a proof system called analytic tableaux that is both easy to understand, and efficient to use on simple examples.

Proof systems based on the resolution principle [Robinson 1965a] have received great attention from computer scientists, as well as logicians. Resolution is particularly suited to mechanization, because it uses only one proof rule, and that rule is easy to implement. A resolution proof is a way of showing that a set of clauses (*i.e.* a formula in conjunctive normal form) is inconsistent. Various restrictions on the form of resolution proofs have been proposed with the hope that they would simplify the task of finding resolution proofs. They include the Davis-Putnam procedure (which actually pre-dates resolution) [Davis & Putnam 1960], the unit preference strategy [Wos *et al.* 1964], set of support [Wos *et al.* 1965],

linear resolution [Loveland 1970], Method I [Cook 1972b], and regular resolution [Tseitin 1968]. Resolution has also been generalized to hyper-resolution [Robinson 1965b], and extension [Tseitin 1968].

1.4. PREVIOUSLY KNOWN RESULTS

Before the work of Tseitin [Tseitin 1968] and Cook [Cook 1971c], little work had been done on the complexity of any of these systems. The size of a truth table can obviously grow exponentially with the size of the formula, but even for such simple systems as semantic trees and analytic tableaux, no nontrivial (*i.e.* more than linear) lower bounds on proof lengths were known. The lengths of proofs for Frege systems and Gentzen-type systems had apparently never been studied. Although efficiency was of primary importance to the designers of resolution theorem provers, most of the results in this area have the form: "Here is an example of a family of (usually predicate) formulas where this resolution strategy found a proof in fewer resolutions than that strategy." Nothing was said about absolute lower bounds, and little attention was paid specifically to propositional calculus.

Tseitin [Tseitin 1968] was the first to obtain nontrivial lower bounds on lengths of proofs. He found a family of sets of clauses for which the length of the shortest regular resolution proof grows faster than any polynomial in the size of the set of clauses. He showed that these same sets of clauses have polynomial-length proofs when the extension rule is allowed.

linear resolution [Loveland 1970], Method I [Cook 1972b], and regular resolution [Tseitin 1968]. Resolution has also been generalized to hyper-resolution [Robinson 1965b], and extension [Tseitin 1968].

1.4. PREVIOUSLY KNOWN RESULTS

Before the work of Tseitin [Tseitin 1968] and Cook [Cook 1971c], little work had been done on the complexity of any of these systems. The size of a truth table can obviously grow exponentially with the size of the formula, but even for such simple systems as semantic trees and analytic tableaux, no nontrivial (*i.e.* more than linear) lower bounds on proof lengths were known. The lengths of proofs for Frege systems and Gentzen-type systems had apparently never been studied. Although efficiency was of primary importance to the designers of resolution theorem provers, most of the results in this area have the form: "Here is an example of a family of (usually predicate) formulas where this resolution strategy found a proof in fewer resolutions than that strategy." Nothing was said about absolute lower bounds, and little attention was paid specifically to propositional calculus.

Tseitin [Tseitin 1968] was the first to obtain nontrivial lower bounds on lengths of proofs. He found a family of sets of clauses for which the length of the shortest regular resolution proof grows faster than any polynomial in the size of the set of clauses. He showed that these same sets of clauses have polynomial-length proofs when the extension rule is allowed.

Tseitin also briefly indicated that proof lengths in the system of resolution with extension are no greater than proof lengths in Gentzen's system, by showing how resolution with extension can "simulate" proofs in Gentzen's system. Cook [Cook 1973] found a family of formulas for which the size of the smallest analytic tableau grows faster than any polynomial in the size of the formula. Kirkpatrick [Kirkpatrick 1974] observed that Tseitin's lower bound for regular resolution also applied to proofs generated by the Davis-Putnam procedure, and he raised the lower bound from $2^{c\sqrt{n}}$ to $2^{c'n/(log(n))^2}$. Finally, after the results of this thesis were announced [Cook & Reckhow 1974], Galil [Galil 1975] showed that enumeration trees (a generalization of semantic trees) can simulate regular resolution, and also extended Tseitin's lower bound to this system.

1.5. SUMMARY OF RESULTS

Figure 1.5.i. is a chart summarizing the major known results concerning lengths of proofs for the propositional calculus. A more detailed version of this chart appears in section 5.2. The boxes indicate proof systems or families of proof systems, and the arrows indicate simulations. The strongest proof systems (*i.e.* those with shortest proofs) are at the top.

A solid arrow $\boxed{S_1} \longrightarrow \boxed{S_2}$ indicates that system S_1 can simulate S_2 in the sense that for some polynomial $p(n)$, for every proof D of length n in system S_2 there is a proof D' in

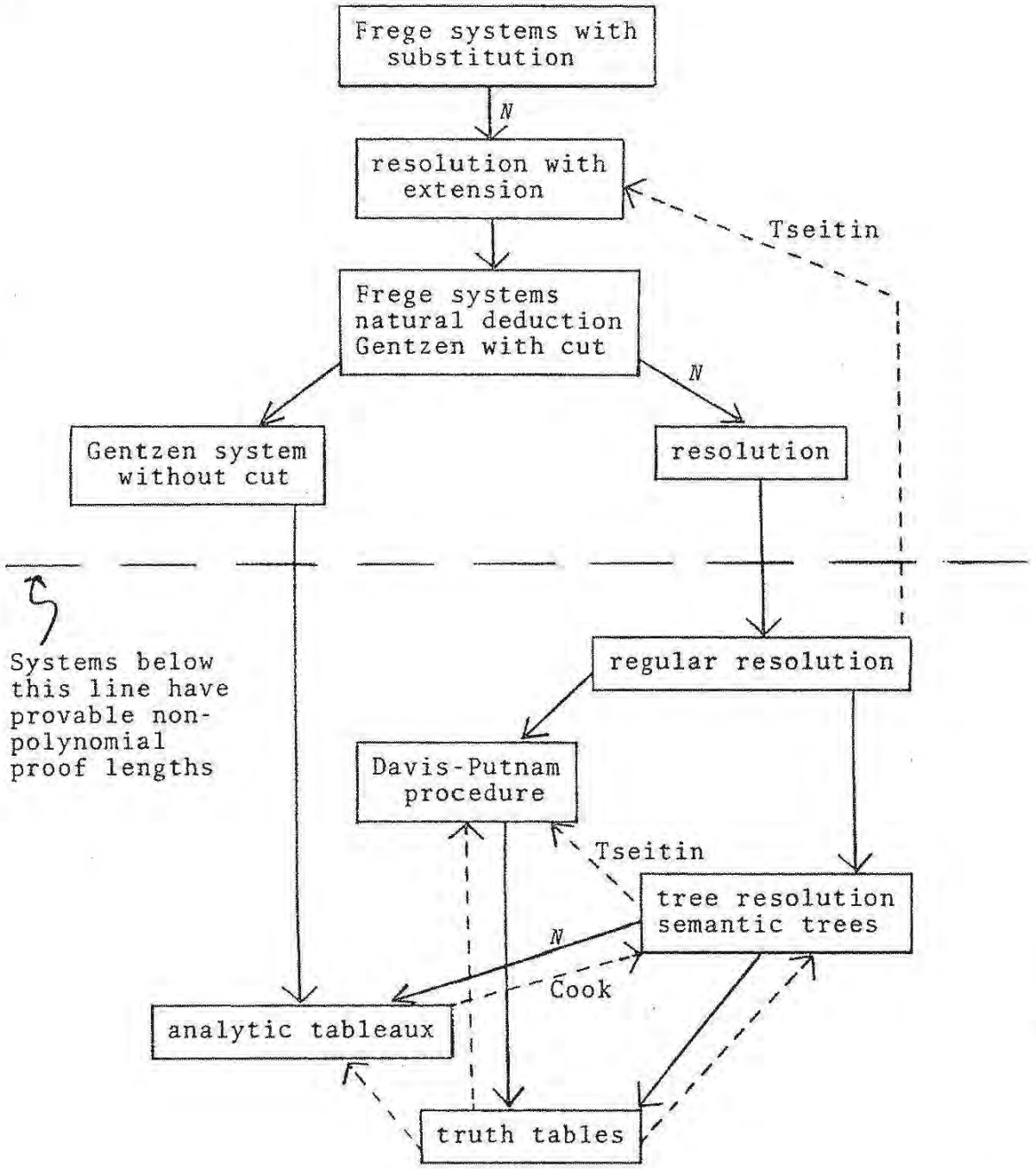


FIGURE 1.5.i.

Brief Summary of Results

system S_1, where the formula proved by D' is a suitable translation of the formula proved by D and the length of D' is bounded by $p(n)$. The translation is only necessary if the systems use different connectives. A dashed arrow $\boxed{S_1}$ ----> $\boxed{S_2}$ indicates that no such simulation is possible. Labels on the arrows indicate the author of the result. The label N indicates that the result is new, and arrows corresponding to trivial results are unlabelled. Where a box contains a family of systems, all systems in the family can simulate each other. All of these results are new.

The dashed horizontal line divides systems with known non-polynomial lower bounds from those with no such bound known. All of the lower bounds shown can be derived as corollaries of Tseitin's lower bound for regular resolution.

The results indicated by the chart are described in more detail in the following three subsections.

1.5.1. Simulation Results

The chart in figure 1.5.i. shows what is now known about proof systems for the propositional calculus partially ordered by the "simulates" relation.

The family of systems in the top box on the chart are Frege systems with the substitution rule added: any instance of a theorem is also a theorem. These systems are the most powerful systems shown on the chart, since any one of them can simulate any other system shown (including other Frege systems with substitution).

The next lower box contains the system of resolution with extension, which was introduced by Tseitin [Tseitin 1968]. This system can simulate all of the other systems shown, except Frege systems with substitution. It has not been shown, however, that resolution with extension cannot simulate Frege systems with substitution. Such a result would necessarily imply a non-polynomial lower bound for proof lengths in the system of resolution with extension (and, thus, everything below extended resolution in the diagram), and would therefore be an important result.

Below this is a box containing Frege systems, natural deduction, and the Gentzen systems with cut. Every system in this class can simulate any other system in the class, regardless of what connectives the systems use for expressing formulas, and regardless of what rules of inference they have. This result is surprising in at least two ways. First, some logical connectives allow much more succinct expression of certain truth functions than others, and one would expect that a proof system designed to deal with these more "powerful" connectives would allow shorter proofs of some formulas. Secondly, the form of inference rules in natural deduction systems and Gentzen-type systems is much less restricted than in Frege systems, and one might expect that certain rules that are not allowed in Frege systems (the deduction theorem, in particular) would lead to much shorter proofs of some formulas. In chapter 5 of this thesis it is shown that while more "powerful" connectives and inference rules might lead to shorter proofs, the improvement can never be more than a polynomial improvement.

Below Frege systems there are a large number of specific proof systems, not all of which are shown in figure 1.5.i. Many of these systems deal only with sets of clauses, and most have provably non-polynomial lower bounds. Many of the simulations among these systems are trivial, because one system is a special case or a simple notational variant of another. The more detailed chart in section 5.2. includes a number of other systems of technical interest that fit into the chart in the vicinity of the dividing line between systems with proven non-polynomial lower bounds and systems without them.

1.5.2. Lower Bounds

The dashed horizontal line in figure 1.5.i. separates systems with proven non-polynomial lower bounds (below) from systems without known non-polynomial lower bounds (above the line). As the chart shows, all of the lower-bound results can be derived from Tseitin's lower bound for regular resolution. Some of these results were derived independently of Tseitin's work, however, by Cook (for analytic tableaux) and Kirkpatrick (for the Davis-Putnam procedure).

Associated with non-polynomial lower bounds, negative results about simulation can usually be derived. For example, Tseitin showed that regular resolution cannot directly simulate resolution with extension, by showing how to construct polynomial-length resolution-with-extension proofs for the sets of clauses that have no polynomial-length regular resolution proofs. Similarly, Cook showed that the formulas (which are actually sets of clauses) that have no polynomial-sized

analytic tableaux have linear-sized tree resolution proofs. (A resolution proof is a tree if each derived clause is used only once to create a new resolvent; clauses that are used more than once must be re-derived for each use.) Tseitin also found a different family of sets of clauses for which there is no polynomial upper bound on lengths of shortest tree resolution proofs, but which have polynomial-length regular resolution proofs. Careful scrutiny of Tseitin's proof reveals that such polynomial-length regular resolution proofs can actually be generated by the Davis-Putnam procedure. Finally, for any system other than complete truth tables, it is easy to find infinite families of n-variable formulas that have proof lengths bounded above by a polynomial in n, thus ruling out the possibility of a direct simulation of that system by complete truth tables.

1.5.3. Original Contributions of this Thesis

Many of the results of this research were first announced in a preliminary report [Cook & Reckhow 1974]. Significant consequences of the publication of that report were that it made researchers in North America aware of the work of Tseitin [Tseitin 1968], and that it stimulated research in the area of proof system complexity (*e.g.* [Galil 1975]).

The definition of "verification system" in subsection 3.3.1. gives a new and precise formalism for proof systems. The connection between verification systems and the P vs NP question, as embodied in lemmas 3.3.2.a. and 3.3.3.a., and corollary 3.3.3.b., is new, as is the formalization of "simulation" of verification systems, as given in subsection 5.1.

The definition of "Frege system" of section 4.2.1. gives new precision to the vaguely-held feeling that these systems (called "Hilbert-type systems" by Kleene [Kleene 1967]) are all very similar. The terms "natural deduction system" and "sequent system" are defined by analogy with Frege systems, and apply to familiar systems as well as new ones invented for the purpose of proving simulation results.

The most important new results, however, are the simulations. Frege systems are shown to be able to simulate other Frege systems, natural deduction systems, and sequent systems. Perhaps the most interesting of these is the simulation of natural deduction. Natural deduction systems can use the deduction theorem as a rule of inference, whereas Frege systems cannot. The usual proof of the deduction theorem seems to indicate that proof lengths can double with each elimination of an application of the deduction theorem. A way was found to eliminate all applications of the deduction theorem from a proof at once, while only increasing the proof length by a polynomial.

The fact that any Frege system can simulate any other Frege system is also a bit surprising. This is because some connectives are more "powerful" than others. For example, the "mod 2 sum" of n arguments can be expressed by a formula of size $O(n)$ if the exclusive-or connective ($\not\equiv$) is allowed. If only the connectives $\{\neg, \vee, \&\}$ are allowed, it first appears that a formula of size $O(2^n)$ is required, but in fact there

is a formula of size $O(n^2)$ that works. It might then seem that Frege systems with $\not\equiv$ cannot simulate Frege systems with $\{\neg,\vee,\&\}$, because they can express theorems that cannot even be expressed, let alone proved, succinctly in the second system. On the other hand, it might seem that Frege systems without $\not\equiv$ could not simulate Frege systems with $\not\equiv$, because, although $\not\equiv$ could not be used to express theorems, its use in expressing intermediate steps of a proof (along with inference rules for manipulating formulas with $\not\equiv$) might allow economies that could not be matched by the system without $\not\equiv$. In subsection 5.3.1. it is shown that neither of these cases holds.

The simulation of resolution by Frege systems is new, as is the simulation of extended resolution by Frege systems with substitution. This last result establishes Frege systems with substitution as no less powerful (modulo a polynomial) than any other system studied here.

The observation that tree resolution and semantic trees (for sets of clauses) are equivalent (modulo a linear factor) established the position of semantic trees within the hierarchy of proof systems. Also, the observation of the fact that the optimal regular resolution proofs in Tseitin's lower-bound theorems can be generated by the Davis-Putnam procedure (without subsumption) establishes for the first time that no simulation of the Davis-Putnam procedure by tree resolution or semantic trees is possible. Finally, Kirkpatrick's stated belief that his techniques can be combined with Tseitin's to raise Tseitin's lower bound for regular resolution is confirmed in paragraph 5.6.2.

1.6. PREVIEW OF THE THESIS

Chapter 2. gives a brief history of the work that has been done in computational complexity pertaining to polynomial-time-bounded computations. The results of Cobham, Cook, and Karp are described in some detail, and a few more recent results are mentioned.

In chapter 3. the more important results of chapter 2. are made precise, and given a uniform notation. These results are then extended in order to prove that $\mathcal{NP}$ is closed under complement if and only if there is a proof system for propositional calculus with a polynomial upper bound on lengths of shortest proofs.

Chapter 4. begins with a description of a formalism for propositional calculus and a few simple theorems. The bulk of the chapter is devoted to describing in more detail a number of proof systems and families of proof systems for propositional logic. The chapter closes with a description of how these systems can be made to fit the formalism introduced in chapter 3. for the complexity-analysis of proof systems.

The major results of this thesis are contained in chapter 5., where the proof systems described in chapter 4. are compared in an attempt to answer the complexity related questions raised in chapter 3. The chapter begins with two specific examples that both typify the results given later in the chapter and motivate the formal definition of what it means for one proof system to simulate another. A more detailed

version of figure 1.5.i. is then given, summarizing the results that follow. Simulation results are given first for Frege systems, followed by natural deduction, Gentzen systems and their derivatives, and resolution and related systems.

The results of the thesis are summarized and conclusions are drawn in chapter 6. The thesis closes with a discussion of the important open questions relating to proof system complexity and suggestions for further research.

2. Problems of Polynomial Difficulty

Most algorithms that are commonly used to solve practical computational problems have running times that grow in proportion to a polynomial in the size of the input. Any algorithm that does not have this property would be impractical for large inputs, since its running time would grow faster than any polynomial in the size of the input. Such considerations have motivated researchers to attempt to determine which problems have algorithms with this property and which problems have no such algorithm.

2.1. COBHAM'S CLASS $\mathcal{L}$

The class of functions that are computable in polynomial time was first discussed by Cobham [Cobham 1965]. He used $\mathcal{L}$ to denote the class of natural number functions that are computable by devices (from some class of computing devices) whose computation times are bounded by polynomials in the lengths (*i.e.* logarithms) of the arguments. He argued that $\mathcal{L}$ is a natural complexity class to study for several reasons.

First, the class $\mathcal{L}$ is relatively insensitive to the class of computing devices under consideration. The same class of functions is obtained whether one considers one-tape Turing machines (see, for example, [Hopcroft & Ullman 1969]), many-tape Turing machines [Hartmanis & Stearns 1965] [Hennie & Stearns 1966], Turing machines with multi-dimensional

tapes and/or many heads per tape, register machines [Shepherdson & Sturgis 1963], random-access machines (provided they cannot multiply two arbitrarily large integers in a single step) [Earley 1970] [Hartmanis 1971] [Cook 1972a] [Cook & Reckhow 1972] [Cook & Reckhow 1973], Schoenhage's storage modification machines, or iterative arrays of finite-state machines. This is a consequence of the fact that for each machine in any one of these classes there is a machine in each of the other classes that can simulate it with no worse than a polynomial time loss. The class $\mathcal{L}$, then, seems to be a reasonable abstraction for the class of functions that can be computed by real computers in polynomial time.

Second, $\mathcal{L}$ is an interesting class because it includes many functions of practical interest, such as addition and multiplication, polynomials, and the integer encodings of many practical computational problems from graph theory and combinatorics. Also, it can be argued that any function not in $\mathcal{L}$ should be labelled as "practically intractable", since the computation time for such a function would grow faster than any polynomial in the length of the input, making computation times impracticably long for large inputs. Similar arguments were advanced by Edmonds [Edmonds 1965], who called an algorithm "good" if its running time increased algebraically (as opposed to exponentially) with the size of its input. He argued that finding "good" algorithms is of practical as well as theoretical interest,

because, he said, "for practical purposes the difference between algebraic and exponential order is often more crucial than the difference between finite and non-finite."

Finally, the class $\mathcal{L}$ has some natural closure properties and an interesting characterization in terms of these closure properties. Cobham observed that $\mathcal{L}$ is closed under explicit transformation, composition, and limited recursion on notation (digit-by-digit recursion, or recursion on length). Limited recursion on notation was discussed by Bennett [Bennett 1962], and may be described as follows: the function f is defined from the functions $g, h_0, \ldots, h_9$, and k by limited recursion on (decimal) notation if

$$f(\vec{x},0) = g(\vec{x})$$

$$f(\vec{x},s_i(y)) = h_i(\vec{x},y,f(\vec{x},y)) \quad (0\le i\le 9,\ i\neq 0 \text{ if } y=0)$$

$$f(\vec{x},y) \le k(\vec{x},y)$$

where s_i is the i^{th} generalized successor: $s_i(y) = 10y+i$. Cobham stated that $\mathcal{L}$ can be characterized as the least class of functions containing the functions s_i and $x^{\ell(y)}$ (where $\ell(z)$ is the (decimal) length of z), and closed under the operations of explicit transformation, composition, and limited recursion on notation.

2.2. COOK'S CLASSES $\mathcal{L}_*$ AND $\mathcal{L}^+$, AND COOK'S THEOREM

A "time-bounded computer" was defined by Cook [Cook 1971a] to be any device that can be simulated by a simple deterministic Turing machine with no more than a polynomial time loss. This gives a name to the family of classes of machines over which Cobham's class $\mathcal{L}$ is invariant. Cook defined $\mathcal{L}_*$ to be the class of sets of strings (over some finite alphabet Σ) that can be recognized by time-bounded computers in polynomial time. That is, the set A is in $\mathcal{L}_*$ if and only if there are a time-bounded computer M and a polynomial P such that for every string x in Σ^*, M halts on input x within time $P(n)$ (where n is the length of x), and M accepts x if and only if x is a member of A. A set A is in $\mathcal{L}_*$ if and only if A is the set of base b notations for a set B of integers whose characteristic function is in Cobham's class $\mathcal{L}$ (where b is the number of symbols in Σ). In the same paper Cook characterized $\mathcal{L}_*$ as the class of languages accepted by two-way multihead (deterministic or nondeterministic) pushdown automata.

In a later paper Cook [Cook 1971b] investigated the class of sets of strings that are accepted by nondeterministic Turing machines in polynomial time. He defined a set S of strings to be *P-reducible* to a set T of strings if some deterministic multitape Turing machine with an oracle for T recognizes S in polynomial time. The set {DNF tautologies} was defined to be the set of strings that encode tautologies of propositional calculus that are in disjunctive normal form.

Cook then proved that if a set S of strings is accepted by some nondeterministic Turing machine within polynomial time, then S is P-reducible to {DNF tautologies}. In other words, if a deterministic polynomial-time recognition procedure were available for recognizing {DNF tautologies}, then that procedure could be used as a subroutine to provide deterministic polynomial-time recognition procedures for every set of strings that has a nondeterministic polynomial-time acceptor. On the other hand, since it appeared likely to Cook that nondeterministic polynomial-time acceptors are strictly more powerful than deterministic polynomial-time recognizers, he suggested that {DNF tautologies} would be a likely candidate for an interesting set that is not in $\mathcal{L}_*$.

In the same paper Cook showed that {DNF tautologies} is P-reducible to two other interesting sets: (isomorphic) {subgraph pairs} and D_3, DNF tautologies with at most three conjuncts per term. He observed that each of these sets, along with {primes} and {isomorphic graph pairs}, is P-reducible to {DNF tautologies}, since each set (or its complement) is accepted in polynomial time by some nondeterministic Turing machine. Such machines use the "guess and verify" technique. For example, a nondeterministic polynomial-time acceptor for the complement of the set of {primes} would, through a sequence of nondeterministic moves, write two random numbers on its work tape ("guess"), and then deterministically multiply the two numbers and compare their product with the input ("verify"), all in polynomial

time. If the comparison succeeded, the machine would accept the input, and if not, it would abort.

Using strings to represent n-tuples of natural numbers in m-adic or b-ary notation, Cook defined $\mathcal{L}_*$ *relations* as those relations that are recognized in polynomial time by deterministic Turing machines. He used $\mathcal{L}^+$ to denote the class of relations accepted in polynomial time by nondeterministic Turing machines. He then characterized $\mathcal{L}^+$ as the relations of the form

$$(\exists y \leq g_k(x))\ R(\vec{x},y)$$

where $g_k(\vec{x}) = 2^{(\ell(\max \vec{x}))^k}$, $\ell(z)$ is the dyadic length of z, and $R(\vec{x},y)$ is an $\mathcal{L}_*$ relation. He remarked that $\mathcal{L}^+$ is also the same as Bennett's class of extended positive rudimentary relations [Bennett 1962].

Drawing an analogy with recursive function theory, Cook observed that $\mathcal{L}_*$ is the analog of the recursive sets and $\mathcal{L}^+$ is the analog of the recursively enumerable sets within the polynomial-time domain. Then {DNF tautologies} can be seen as the analog of the halting problem, since it has the complete $\mathcal{L}^+$ degree (in the sense of P-reducibility) just as the halting problem has the complete r.e. degree (in the sense of Turing reducibility). He noted that the diagonal argument that shows the halting problem is not recursive apparently cannot be easily adapted to show that {DNF tautologies} is not in $\mathcal{L}_*$.

2.3. KARP'S COMPLETE PROBLEMS

In a paper that popularized and extended Cook's work, Karp [Karp 1973] used P to denote Cook's class $\mathcal{L}_*$. He used Π to denote the class of functions from Σ^* into Σ^* that are computable in polynomial time by one-tape Turing machines. The class of two-place relations over Σ^* that are recognized in polynomial time by deterministic one-tape Turing machines was denoted by $P^{(2)}$. If $L^{(2)}$ is a relation in $P^{(2)}$ and p is a polynomial, then the language (set of strings) L defined by

$L = \{x \mid$ there exists y such that $\langle x,y\rangle \in L^{(2)}$ and $lg(y) \leq p(lg(x))\}$

(where $lg(z)$ is Karp's notation for the length of z), is the language derived from $L^{(2)}$ by *p-bounded existential quantification*. Karp then defined NP to be the class of languages derivable from languages in $P^{(2)}$ by polynomial-bounded existential quantification, and proved NP is also the class of languages accepted by nondeterministic Turing machines that operate in polynomial time.

While Cook's definition of P-reducibility is the polynomial time analog of Turing reducibility from recursive function theory [Rogers 1967], Karp used a polynomial-time analog of many-one reducibility. He said language L is *reducible* to language M if there is a function f from Π such that $f(x) \in M$ if and only if $x \in L$. A language L is said to be *(polynomial) complete* if L is a member of NP and every language in NP is reducible to L. As with Cook's notion of

P-reducibility, if L_1 is in P and L_2 is reducible to L_1 (in Karp's sense), then L_2 is in P. Consequently, if L is any complete language, then L is in P if and only if P and NP are the same class.

Karp defined the language SATISFIABILITY to be the set of encodings into Σ^* of satisfiable sets of clauses (propositional formulas in conjunctive normal form that are true under at least one truth assignment). Cook's theorem, then, asserts that SATISFIABILITY is complete. Karp's main contribution was to show that about twenty other language recognition problems arising from practical problems in graph theory, discrete optimization, and combinatorics are also complete. These include such problems as 0-1 INTEGER PROGRAMMING, CLIQUE, SET COVERING, UNDIRECTED HAMILTON CIRCUIT, CHROMATIC NUMBER, STEINER TREE, and JOB SEQUENCING. Each of these languages is defined precisely in Karp's paper, but the names should give an idea of the variety of problems that are complete.

2.4. RECENT WORK

Cook's and Karp's papers were followed by a flood of papers related to the *P vs NP* question. (See, for example [SIGACT 1973, 1974], [SWAT 1972, 1973].) Some papers concentrated on showing new problems to be complete. It is now clear that the class of complete problems includes a vast number of practical problems from discrete mathematics. It is common for researchers, when confronted with a new

combinatorial problem with no obvious polynomial-time algorithm, to attempt to show the problem is complete. Once a problem has been shown to be complete it is known to be very difficult (perhaps impossible) to solve efficiently.

There are three important problems from *NP* with no known polynomial-time algorithms that have not been shown to be complete: the problems of recognizing NONPRIMES and ISOMORPHIC GRAPH PAIRS (both of which were mentioned by Cook [Cook 1971b]), and LINEAR PROGRAMMING. It is interesting to note that, since PRIMES has also been shown to be in *NP* [Pratt 1975], if either PRIMES or NONPRIMES were complete, then *NP* would be closed under complementation. Since the complement of LINEAR PROGRAMMING can also be shown to be in *NP*, the same situation holds for LINEAR PROGRAMMING as for PRIMES.

Other recent papers have investigated subrecursive reducibilities. In addition to general papers about computation-limited reducibilities, there have appeared papers about log-space reducibility and completeness in *P* with respect to log-space reducibility, polynomial-time analogs of truth-table reducibility as well as (Cook's) Turing reducibility and (Karp's) many-one reducibility, and some nondeterministic polynomial-time reducibilities.

One interesting paper [Meyer & Stockmeyer 1972] used a nondeterministic polynomial-time reducibility-like

relation to build a possible analog of Kleene's arithmetical hierarchy [Rogers 1967]. They defined a deterministic oracle reducibility $\leq_p$ like Cook's, and a nondeterministic relation: $L_1 R_n L_2$ if L_1 is accepted by some nondeterministic polynomial-time machine with oracle language L_2. Then they defined the following sequence of classes of languages:

$$\Sigma_0^p = \Pi_0^p = \Delta_0^p = \phi$$

$$\Sigma_{i+1}^p = \{L \mid L R_n L' \text{ for some } L' \in \Sigma_i^p\}$$

$$\Pi_{i+1}^p = \{L \mid \neg L R_n L' \text{ for some } L' \in \Sigma_i^p\} \quad (\neg L \text{ is the complement of } L)$$

$$\Delta_{i+1}^p = \{L \mid L \leq_p L' \text{ for some } L' \in \Sigma_i^p\} .$$

In particular, then, $\Sigma_1^p = NP$ and $\Delta_1^p = P$. The class Π_i^p is the class of languages whose complements are in Σ_i^p. Meyer and Stockmeyer noted that their "hierarchy" has the same inclusion structure as the Kleene hierarchy, but they were unable to show that any of the inclusions are proper. Such a demonstration would have as a corollary that $P \neq NP$. The final result of this paper was the derivation of a sequence $B_1, B_2, \ldots$ of languages such that B_k is complete with respect to $\leq_p$ in the class $\Sigma_k^p \cup \Pi_k^p$.

Another characterization of this potential hierarchy can be based on polynomial-bounded quantification. In this case it is most convenient to think of a language as being a k-place relation over Σ^*, for some k. If L_1 is a $k+1$-place relation over Σ^* and p is a polynomial in k variables, then

the k-place relation L_2 defined by

$L_2=\{\langle x_1,\ldots,x_k\rangle \mid \exists y$ such that $\ell y \le p(\ell x_1,\ldots \ell x_k), \langle x_1,\ldots,x_k,y\rangle \in L_1\}$

(where ℓz is the length of z)

is the language derived from L_2 by *p-bounded existential* (*universal* if $\exists$ is replaced by $\forall$) *quantification*. Then a sequence of language classes can be defined:

$$E_0^P = A_0^P = P$$

$$E_{i+1}^P = \{L \mid \text{for some } L' \in A_i^P,\ L \text{ is derived from } L' \text{ by polynomial-bounded existential quantification}\}$$

$$A_{i+1}^P = \{L \mid \text{for some } L' \in E_i^P,\ L \text{ is derived from } L' \text{ by polynomial-bounded universal quantification}\}.$$

As Cook and Karp have pointed out, $E_1^P = NP = \Sigma_1^P$, and their arguments can be easily extended to show that for all $i \ge 1$, $E_i^P = \Sigma_i^P$ and $A_i^P = \Pi_i^P$.

2.5. DISCUSSION

It is now widely accepted that the *P vs NP* question is one of the most important open questions in computational complexity. As Karp observed, *NP* can be informally described as the class of problems that can be solved by polynomial-depth backtrack search. This is an extremely wide class, and the practical impact would be tremendous if it could be shown that each of these problems could be solved deterministically in polynomial time. On the other hand, it would be a breakthrough for computational complexity if

it could be shown that some language in *NP* has no polynomial-time recognition algorithm, since this would imply that none of the complete languages can be recognized deterministically in polynomial time.

Another question that can be asked is whether *NP* is closed under complementation; that is, does $L \in NP$ imply $\neg L \in NP$? Since *P* is a class of languages defined in terms of deterministic recognition devices, it is closed under complementation. Thus, a proof that *NP* is not closed under complementation would demonstrate that $P \neq NP$.

Although closure of *NP* under complementation would not directly imply $P=NP$, it would have a number of interesting consequences. The "hierarchy" of Meyer and Stockmeyer would collapse to the two classes *P* and *NP*. In particular, this would mean that many languages that are clearly in a higher level of the "hierarchy" but not known to be in *NP*, such as BOOLEAN MINIMIZATION, MAXIMUM CLIQUE, TRAVELING SALESMAN PROBLEM, and TAUTOLOGIES, would be in *NP*. The languages whose complements are in *NP* could be characterized as those languages that can be recognized by a polynomial-depth backtrack search that fails. For example, a pair of isomorphic graphs can be recognized by searching for an isomorphism. It seems unlikely, however, that there is any polynomial-depth backtrack algorithm that searches for something that "proves" two given graphs are not isomorphic. Similarly, satisfiability of a Boolean expression can be demonstrated by exhibiting a satisfying truth assignment.

Demonstrating that such an expression is unsatisfiable seems to require a proof whose length is potentially exponential in the length of the expression. Thus, it would be nearly as surprising to find that *NP* is closed under complementation as it would be to find that *P=NP*.

3. Polynomial-Bounded Verification Systems

This chapter establishes a consistent notation and gives precise definitions and proofs of the results upon which the rest of this thesis is based. The definitions and notation are based on those introduced by [Cook 1971b], [Karp 1973] and [Ladner *et al.* 1974] and the standardizing efforts of [Knuth 1974a,b]. The results of section 3.3. first appeared in [Cook & Reckhow 1974], and the notion of "super proof system" from that paper has been generalized here to the concept of polynomial-bounded verification systems.

3.1. NOTATION AND BASIC DEFINITIONS

The concern here is with the complexity of recognition by time-bounded computers of sets of strings over some finite alphabet. For the sake of definiteness, the computing model used will be one-tape Turing machines as formalized in [Hopcroft & Ullman 1969], and the reference alphabet will be {0,1}. The customary notation {0,1}* is used to denote the set of all finite strings of 0's and 1's. A *language* is a subset of {0,1}*, and a function from {0,1}* into {0,1}* is called a *string function*. If x is a string from {0,1}*, then ℓx denotes the *length* (number of symbols) of x. If L is a language, then $\neg L$ denotes its complement with respect to {0,1}*.

3.1.1. Polynomial-Time-Bounded Computations

If $T(n)$ is a function on the natural numbers, then language L is *recognized in time* $T(n)$ if there is a one-tape deterministic Turing machine M such that for every string w (from $\{0,1\}^*$), M with input w halts with its tape blank[†] within $T(\ell w)$ steps, halting in the ACCEPT state if $w \in L$ and in the REJECT state if $w \notin L$. Language L' is *accepted (nondeterministically) in time* $T'(n)$ if there is a one-tape nondeterministic Turing machine M' such that for every string $x \in L'$ there is a computation of M' on input x that halts in the ACCEPT state within $T'(\ell x)$ steps, and for every string $y \notin L'$ there is no computation of M' on input y that halts. (Note: nondeterministic Turing acceptors have no REJECT state; they either ACCEPT or compute forever.)

[†] The restriction that the recognizer must halt with blank tape can be satisfied by adding end markers to the machine's tape alphabet, along with states to maintain the markers and to erase them and all symbols between them before the machine halts. These extra operations can increase the machine's running time by at most a linear factor, and provided $T(n) > n$, the speedup theorem [Hartmanis & Stearns 1965] can be applied to give a machine that simulates the original machine and halts with blank tape within $T(n)$ steps.

String function F is *computed in time* $T''(n)$ if there is a one-tape deterministic Turing machine M'' such that for every string z, M'' with input z halts within $T''(\ell z)$ steps with $F(z)$ on its tape. It is a trivial observation that a language is recognized in time $T(n)$ if and only if its characteristic function is computed in time $T(n)$. Whenever the recognizer enters the ACCEPT (REJECT) state, the machine computing the characteristic function prints a 1 (0) and halts, and *vice versa*. (Recall that a recognizing Turing machine must halt with a blank tape.)

A language is *recognized (accepted) in polynomial time* if there is a polynomial $P(n)$ such that the language is recognized (accepted) in time $P(n)$. Define $\mathcal{P}$ to be the class of languages recognized in polynomial time, and $\mathcal{NP}$ to be the class of languages accepted in polynomial time. A class of languages C is said to be *closed under complements* (or *complementation*) when $L \in C$ if and only if $\neg L \in C$.

A string function is *computed in polynomial time* if there is a polynomial P'(n) such that the function is computed in time P'(n). Let $\mathcal{PF}$ be the class of string functions that are computed in polynomial time. A string function $f \in \mathcal{PF}$ is called a *transformation*.

3.1.1.1. Insensitivity to the Computing Model

A class of *time-bounded computers* is any class of abstract computing devices such that whenever there is a device in the class that computes string function F in time $T(n)$, there is an integer k such that F is computed in time $(T(n))^k$ (by a one-tape Turing machine). A time-bounded computer recognizes a language by computing its characteristic function. For most classes of time-bounded computers there is an obvious generalization to nondeterministic computations, and in these cases the concept of nondeterministic acceptance of languages by time-bounded computers can be formalized in a straightforward way.

A direct consequence of the definition of time-bounded computers is that a function is computed (a language is recognized, or accepted) in polynomial time (by a one-tape Turing machine) if and only if it is computed (respectively, recognized, or accepted) in polynomial time by a device from some class of time-bounded computers. Thus the classes $\mathcal{P}$, $\mathcal{NP}$, and $\mathcal{PF}$ would be the same if they had been formalized in terms of any class of time-bounded computers (as long as machines from the class can simulate Turing machines in polynomial time) rather than one-tape Turing machines. In fact, the same classes $\mathcal{P}$, $\mathcal{NP}$, and $\mathcal{PF}$ would be obtained if they had been formalized in terms of the union of all classes of time-bounded computers.

As noted in section 2.1., the family of classes of time-bounded computers is a rich one. Most of the common abstract models of computing are classes of time-bounded computers. These include various models of random-access computers, list processing machines, and numerous extensions to the basic Turing machine model.

3.1.1.2. Insensitivity to the Alphabet and Notation

In addition to its insensitivity to the class of computing devices considered, the theory of polynomial-time-bounded computations is insensitive to the alphabet used. Given any finite alphabet A, there is a straightforward encoding function e which maps A^* one-to-one into $\{0,1\}^*$ such that e and its left inverse can be computed in polynomial time.

Membership in $\mathcal{P}$ or $\mathcal{NP}$ of the encoding into $\{0,1\}^*$ of various combinatorial problems is also relatively insensitive to the way the encoding is done. That is, the set of strings obtained by a "reasonable" encoding of some combinatorial problem is a member of $\mathcal{P}$ (or $\mathcal{NP}$) if and only if the set obtained from any other "reasonable" encoding of the same problem is in $\mathcal{P}$ (or $\mathcal{NP}$). Although this is not a mathematically precise statement, its intuitive meaning can be clarified with the aid of a few examples.

Problems related to the recognition of graphs with certain properties can be encoded by representing graphs either by adjacency matrices, incidence matrices, or lists

of adjacencies. The matrices can be linearized either in row order or column order, and adjacency lists may be structured in several ways. Given an encoding x of a graph G in any of these forms, it is easy to transform x into an encoding of G in either of the other forms in polynomial time.

Language recognition problems involving integers can be encoded by representing integers in any radix notation, such as decimal, binary, or dyadic. In this context, unary notation is not "reasonable", since the length of the unary notation for a number grows exponentially with the length of the radix notations for it. In contrast, any two radix notations differ in length by only a constant factor, and can be translated one into the other in polynomial time.

Logical formulas are conceptually trees with logical connectives at the internal nodes and logical variables (atoms) at the leaves. To encode such formulas as strings over a finite alphabet, variables can be distinguished by integer labels in radix notation, and the trees can be linearized by either prefix, postfix, or infix notation. Again it is not hard to see that there are polynomial-time algorithms for transforming a formula from any one of these forms to any other.

3.1.1.3. Cobham's Thesis

Paragraphs 3.1.1.1. and 3.1.1.2. have discussed the ways in which $\mathcal{P}$ and $\mathcal{NP}$ are insensitive to the computing model used, the input alphabet, and the particular way in which combinatorial problems are encoded as language recognition problems. This shows that there is a polynomial-time analog of Church's Thesis from recursive function theory. Perhaps the title "Cobham's Thesis" could be given to the statement that giving a polynomial-time algorithm for the recognition (acceptance) by any time-bounded computer of some "reasonable" encoding of a combinatorial problem into strings over any finite alphabet constitutes a proof that the problem belongs to $\mathcal{P}$ ($\mathcal{NP}$). Although Cobham never made exactly this statement, his pioneering paper [Cobham 1965] was the first to put forth the major ideas contained in it.

These observations are well known, and it has become customary in the literature to use rather informal arguments to show that certain computations can be performed in polynomial time. As a consequence of Cobham's Thesis, informal arguments will be used in this thesis. There will be few specific references to Turing machines, and alphabets will grow to whatever size allows convenient expression of the problems considered. Encodings into finite alphabets will often be left unspecified. As with Church's Thesis all informal arguments that appeal to Cobham's Thesis could be made precise, but only at the expense of a great deal of tedious, confusing, and irrelevant detail.

3.1.2. Polynomial-Time Reducibilities

If L is a language, then an *L-oracle machine* is a one-tape (deterministic or nondeterministic) Turing machine with an additional write-only *query tape* and three special states: TEST, YES, and NO. Whenever the machine enters the TEST state with x on its query tape, the next state is YES if $x \in L$ and NO if $x \notin L$. In either case the query tape is erased, and the machine assumes its next state in a single step, as if the machine's query about the membership of x in L were answered by an "oracle" with full knowledge of L. The meanings of recognition and acceptance by oracle machines in polynomial time are obvious.

Let L' and L'' be languages. Then language L' *reduces (in polynomial time)* to L'' (written $L' \leq_T^P L''$) if some (deterministic) L''-oracle machine recognizes L' in polynomial time. Language L' *transforms (in polynomial time)* to L'' (written $L' \leq_m^P L''$) if there is some function f from $\mathcal{PF}$ such that for every $x \in \{0,1\}^*$, $x \in L'$ if and only if $f(x) \in L''$. Note that $L' \leq_m^P L''$ implies $L' \leq_T^P L''$.

Since $\leq_T^P$ and $\leq_m^P$ are transitive and reflexive, the relations $\equiv_T^P$ and $\equiv_m^P$, defined by $L' \equiv_T^P (\equiv_m^P) L''$ if $L' \leq_T^P (\leq_m^P) L''$ and $L'' \leq_T^P (\leq_m^P) L'$, are equivalence relations. Denote by $deg_T^P(L)$ $(deg_m^P(L))$ the equivalence class of L with respect

to $\equiv_T^P$ ($\equiv_m^P$), called the *polynomial T-degree* (m-*degree*) of L. It is easy to see that $\mathcal{P} = deg_T^P(\phi) = deg_m^P(\phi)$. (The superscript P, representing polynomial-time reducibilities, will be dropped whenever the resulting ambiguity causes no confusion.)

If C is a class of languages and $\propto$ is a transitive and reflexive relation on languages, then language L is *C-hard with respect to* $\propto$ if for every L' in C, $L' \propto L$. Language L is *C-complete with respect to* $\propto$ if $L \in C$ and L is C-hard with respect to $\propto$. If L is C-complete with respect to $\propto$, $L' \in C$, and $L \propto L'$, then L' is C-complete with respect to $\propto$. In fact, $deg^{\propto}(L)$ is the subclass of all languages that are C-complete with respect to $\propto$.

Since $L' \leq_m L''$ implies $L' \leq_T L''$, $deg_m(L) \subseteq deg_T(L)$. Thus, if L is C-complete with respect to $\leq_m$, then L is also C-complete with respect to $\leq_T$. The terms *$\mathcal{NP}$-hard* and *$\mathcal{NP}$-complete* will be assumed to mean with respect to $\leq_T$, and *$\mathcal{NP}$-hard (complete) by transformation* will mean with respect to $\leq_m$. It is an open question whether $\leq_m$ differs from $\leq_T$ on $\mathcal{NP}$, although it was shown in [Ladner *et al.* 1974] that they differ on the class of exponential-time-recognizable sets.

Meyer and Stockmeyer [Meyer & Stockmeyer 1972] defined a related reducibility-like notion for the purposes of building a "hierarchy" of classes above $\mathcal{P}$ and $\mathcal{NP}$

(see section 2.4). Since it is not clear whether their relation is transitive, the more neutral notation R_T^{NP} is used. Language L' is *Meyer-Stockmeyer derived from* L'' (written $L' R_T^{NP} L''$) if there is a (nondeterministic) L''-oracle machine that accepts L' in polynomial time.

3.2. $\mathcal{P}$ *vs* $\mathcal{NP}$ AND COOK'S THEOREM

The importance of studying $\mathcal{NP}$-complete languages stems from the following observation.

LEMMA 3.2.a.

The classes $\mathcal{P}$ and $\mathcal{NP}$ are identical if and only if some $\mathcal{NP}$-complete language is in $\mathcal{P}$.

Proof

If $\mathcal{P}$ and $\mathcal{NP}$ are identical, then every language in $\mathcal{P}$ is $\mathcal{NP}$-complete, since $\mathcal{NP} = \mathcal{P} = deg_T(\phi)$.

Conversely, suppose language L is $\mathcal{NP}$-complete and in $\mathcal{P}$. Then, since $L \in \mathcal{P}$, there is a Turing machine M that recognizes L in polynomial time. Since L is $\mathcal{NP}$-complete, then given any language $L' \in \mathcal{NP}$, there is an L-oracle machine M' that recognizes L' in polynomial time. A deterministic polynomial-time recognizer for L' can be constructed from M' by replacing calls to the oracle by calls to M. Since this construction can be done for every $L' \in \mathcal{NP}$, $\mathcal{P}$ and $\mathcal{NP}$ are identical. ☒ 3.2.a.

This lemma shows that either all $\mathcal{NP}$-complete languages have polynomial-time recognition algorithms or none of them does. The $\mathcal{NP}$-complete class contains languages derived from many important combinatorial problems. The following theorems indicate the extent of the class of $\mathcal{NP}$-complete languages.

THEOREM 3.2.b. (Cook) [Cook 1971b]

Let SATISFIABILITY be the set of formulas of propositional calculus that are satisfiable. The language SATISFIABILITY is $\mathcal{NP}$-complete by transformation.

THEOREM 3.2.c. (Cook, Karp, and others)

By virtue of the fact that SATISFIABILITY transforms to it, each of the following languages is $\mathcal{NP}$-complete by transformation.

CNF SATISFIABILITY: [Cook 1971b] satisfiable formulas in conjunctive normal form (CNF)

3CNF SATISFIABILITY: [Cook 1971b] satisfiable CNF formulas with no more than three literals per clause.

SUBGRAPH: [Cook 1971b] pairs of graphs where the first is isomorphic to a subgraph of the second

CLIQUE: [Cook 1971b][Karp 1973] pairs where the first is an integer k and the second is a graph that contains a clique (complete subgraph) on k nodes

CHROMATIC NUMBER: [Karp 1973] pairs where the first is an integer k and the second is a graph that is k-colourable

3-COLOURABILITY: [Garey *et al.* 1974] graphs that are 3-colourable

PLANAR 3-COLOURABILITY: [Garey *et al.* 1974] planar 3-colourable graphs

PLANAR DEGREE 4 3-COLOURABILITY: [Garey *et al.* 1974] planar graphs of node degree no greater than four that are 3-colourable

DIRECTED HAMILTON CIRCUIT: [Karp 1973] Hamiltonian directed graphs

UNDIRECTED HAMILTON CIRCUIT: [Karp 1973] Hamiltonian graphs

0-1 INTEGER PROGRAMMING: see [Karp 1973]

SET PACKING: see [Karp 1973]

SET COVERING: see [Karp 1973]

3-DIMENSIONAL MATCHING: see [Karp 1973]

JOB SEQUENCING: see [Karp 1973]

REGISTER ALLOCATION: see [Sethi 1973]

It appears that most combinatorial problems with obvious nondeterministic polynomial-time algorithms and no known deterministic polynomial-time algorithms can be shown to be $\mathcal{NP}$-complete. The five main exceptions to this general rule are GRAPH ISOMORPHISM (pairs of isomorphic graphs), PRIMES (integers that are prime), NONPRIMES (composite integers), LINEAR INEQUALITIES (pairs where the first is an integer matrix C and the second is an integer vector d such that $Cx \geq d$ has a rational solution), and the complement of LINEAR INEQUALITIES. Each of these problems is in $\mathcal{NP}$;

for all except PRIMES, which was proven by Pratt [Pratt 1975], and the complement of LINEAR INEQUALITIES, which relies on duality theory, the "guess and verify" algorithm is readily apparent from the statement of the problem. None of these problems, however, has been shown to be $\mathcal{NP}$-complete, and none is known to have a deterministic polynomial-time recognition algorithm. In this respect, PRIMES - NONPRIMES and LINEAR INEQUALITIES and its complement are particularly interesting, since they are complementary pairs. They are the only known natural pairs of complementary languages where both members of the pair are known to be in $\mathcal{NP}$ but not known to be in $\mathcal{P}$. As the results of the next section will show, if either PRIMES, NONPRIMES, or LINEAR INEQUALITIES were proven to be $\mathcal{NP}$-complete, then the complement of *every* language from $\mathcal{NP}$ would also be in $\mathcal{NP}$.

3.3. POSSIBLE CLOSURE OF $\mathcal{NP}$ UNDER COMPLEMENTS

Since it is defined in terms of deterministic recognition devices, the class $\mathcal{P}$ is closed under complements. A deterministic recognizer for a language becomes a recognizer for its complement when the rôles of its ACCEPT and REJECT states are interchanged. Similarly, the class $\mathcal{P}$ is closed under finite union, intersection, and difference of languages.

On the other hand, it is unknown whether $\mathcal{NP}$ is closed under complements. A proof that $\mathcal{NP}$ is not closed

under complements would settle the $\mathcal{P}$ vs $\mathcal{NP}$ question in the negative. If $\mathcal{NP}$ were closed under complements, however, a number of other surprising results would follow, all based on the following lemma.

LEMMA 3.3.a.

If $\mathcal{NP}$ is closed under complements, then the Meyer-Stockmeyer "hierarchy" [Meyer & Stockmeyer 1972] (see section 2.4.) consists of at most the two classes $\mathcal{P}$ and $\mathcal{NP}$.

Proof

If $\mathcal{NP}$ is closed under complements, then $\Sigma_1^p=\Pi_1^p=\mathcal{NP}$. If it can be shown that closure of $\mathcal{NP}$ under complements implies $\Sigma_{i+1}^p=\Sigma_i^p$, the result follows by induction.

Suppose $L\in\Sigma_{i+1}^p$ and $\Sigma_i^p=\mathcal{NP}$. Then there is a language $L'\in\Sigma_i^p$ such that some nondeterministic L'-oracle machine M accepts L in polynomial time. Since $\mathcal{NP}=\Sigma_i^p$ is assumed to be closed under complements, there are nondeterministic polynomial acceptors M' for L' and M'' for $\neg L'$. A nondeterministic polynomial-time acceptor M° for L can now be constructed from M, M', and M''. Machine M° is identical to M except when M enters the TEST state. It then nondeterministically chooses to simulate either M' or M'' with M's query tape contents as input. If M' ACCEPT's, then M° enters M's YES state, and if M'' ACCEPT's, then M° enters M's NO state. Since M, M', and M'' are polynomial-time-bounded, M° runs in polynomial time also. By construction M° accepts L, so $L\in\mathcal{NP}$. Since this construction

can be carried out for arbitrary $L \in \Sigma^P_{i+1}$, $\Sigma^P_{i+1} = \Sigma^P_i$. Then by induction on i it can be shown that for $i \geq 1$, $\Sigma^P_i = \Pi^P_i = \mathcal{NP}$.

∎ 3.3.a.

As a consequence of this lemma, closure of $\mathcal{NP}$ under complements would imply that such problems as NON-ISOMORPHIC GRAPH PAIRS, TRAVELING SALESMAN, MAXIMUM CLIQUE, TAUTOLOGIES, and BOOLEAN MINIMIZATION, which obviously fall somewhere in the Meyer-Stockmeyer "hierarchy" but are not known to be in $\mathcal{NP}$, are all members of $\mathcal{NP}$.

3.3.1. Verification Systems

It would be surprising to find that all of these problems from the higher levels of the "hierarchy" are in $\mathcal{NP}$, because membership in $\mathcal{NP}$ is tantamount to possession of a polynomial-time "guess and verify" algorithm. There is no intuition that suggests the existence of any system in which there is a polynomial-time-verifiable "guess" that would "prove" that two graphs are not isomorphic, some salesman's tour is minimal, or some Boolean expression has no shorter equivalent.

The intuitive idea of a "guess and verify" algorithm can be formalized very neatly. If L is a language, then a *(polynomial-time) verification system* for L is a function $F \in \mathcal{PF}$ which maps $\{0,1\}^*$ *onto* L. If F is a verification system for L and $F(w)=x$, then intuitively w is the "guess" and the (polynomial-time) computation of $F(w)$ is the

"verification" that $x \in L$. If $F(w)=x$, then w is called a *derivation* of x in the system F.

Possession of a verification system is not sufficient, however, to guarantee that language L belongs to $\mathcal{NP}$. This is because, although the verification can be performed in time bounded by a polynomial in the length of the guess, the length of guess itself may not be bounded by any polynomial in the length of the string being verified. In fact, a language L has a verification system if and only if L is recursively enumerable.

3.3.2. Polynomial-Bounded Verification Systems and Membership in $\mathcal{NP}$

If F is a verification system for L, then F is said to be *polynomial(-length)-bounded* if there is a polynomial p such that for each string $x \in L$ there is a string w such that $\ell w \leq p(\ell x)$ and $F(w)=x$. That is, F is polynomial-bounded if there is a polynomial upper bound on the lengths of shortest derivations of strings in L.

LEMMA 3.3.2.a.

Language L is a member of $\mathcal{NP}$ if and only if L has a polynomial-bounded verification system.

Proof

Suppose $L \in \mathcal{NP}$ and let M be a $p(n)$-time-bounded acceptor for L. Without loss of generality, it may be assumed that whenever M makes a move, there are no more than two possible successor states. A machine can be converted

to this form without increasing its running time by more than a constant factor. A polynomial-bounded verification system F for L can then be described in terms of the machine M' that computes F. Given input $x\#y$ (suitably encoded in $\{0,1\}^*$), M' simulates M on input x. Whenever M makes a nondeterministic move, M' consults the next symbol of y to determine which choice to make. If M halts in the ACCEPT state, then M' outputs x, and if M fails to halt before y is exhausted, then M' outputs α, some fixed (short) member of L. It is clear that if M' accepts x in $p(\ell x)$ steps, then there is a y of length no greater than $p(\ell x)$ such that $F(x\#y)=x$. Also, M' can be constructed so that it always halts in time bounded by $q(\ell y)$, where q is a fixed (low-degree) polynomial. This is enough to ensure that F is a polynomial-bounded verification system for L.

Conversely, suppose F is a polynomial-bounded verification system for L. Then a nondeterministic polynomial-time acceptor for L nondeterministically writes the string w on its tape, computes $F(w)$ in polynomial time, compares $F(w)$ with the input string x, and, if the comparison is successful, accepts the input. Thus $L \in \mathcal{NP}$. ☒ 3.3.2.a.

This lemma confirms the intuitive feeling that "guess and verify" as formalized by polynomial-bounded verification systems accurately characterizes the computing power of nondeterministic polynomial-time acceptors.

3.3.3. Polynomial-Bounded Verification Systems and Closure of $\mathcal{NP}$ Under Complements

As lemma 3.2.a. shows, the $\mathcal{NP}$-complete languages are canonical forms of the $\mathcal{P}$ vs $\mathcal{NP}$ question, since if L is $\mathcal{NP}$-complete, then $\mathcal{P}=\mathcal{NP}$ if and only if $L\in\mathcal{P}$. It turns out that the complements of the $\mathcal{NP}$-complete languages have a similar relationship to the question of the closure of $\mathcal{NP}$ under complements.

LEMMA 3.3.3.a.

If L is $\mathcal{NP}$-complete, then $\mathcal{NP}$ is closed under complements if and only if $\neg L\in\mathcal{NP}$.

Proof

If L is $\mathcal{NP}$-complete, then $L\in\mathcal{NP}$. Thus if $\mathcal{NP}$ is closed under complements, then $\neg L\in\mathcal{NP}$.

The proof of the converse is similar to the proof of lemma 3.3.a. Suppose L is $\mathcal{NP}$-complete and $\neg L\in\mathcal{NP}$. Let L' be any language in $\mathcal{NP}$. If it can be shown that $\neg L'\in\mathcal{NP}$, then $\mathcal{NP}$ must be closed under complements.

Since L and $\neg L$ are in $\mathcal{NP}$, there are polynomial-time acceptors M_0 and M_1 for L and $\neg L$ respectively. Since L is $\mathcal{NP}$-complete and $L'\in\mathcal{NP}$, there is an L-oracle machine M_2 that recognizes L' in polynomial time. A polynomial-time acceptor M_3 for $\neg L'$ can then be constructed from M_0, M_1, and M_2 by the method used in the proof of lemma 3.3.a. Machine M_3 is identical to M_2, except that when M_2 enters the TEST state, M_3 nondeterministically chooses to simulate

either M_0 or M_1 with M_2's query tape contents as input. If M_0 accepts its input, then M_3 continues in M_2's YES state, and if M_1 accepts, then M_3 enters M_2's NO state. This simulation continues until M_2 halts. If M_2 rejects its input, then M_3 accepts; otherwise, M_3 does not halt. It is clear by this construction that M_3 accepts $\neg L'$ in polynomial time, so that $\neg L' \in \mathcal{NP}$. ☒ 3.3.3.a.

COROLLARY 3.3.3.b.

$\mathcal{NP}$ is closed under complements if and only if for some (also for every) $\mathcal{NP}$-complete language L there exists a polynomial-bounded verification system for $\neg L$.

This corollary is an immediate consequence of lemmas 3.3.2.a. and 3.3.3.a. It says that one way to attack the question of whether or not $\mathcal{NP}$ is closed under complements is to determine whether or not the complement of some $\mathcal{NP}$-complete language has a polynomial-bounded verification system.

3.4. SUMMARY AND DISCUSSION

The question of the closure of $\mathcal{NP}$ under complements is an important one, both for its relevance to the $\mathcal{P}$ vs $\mathcal{NP}$ question and for its own intrinsic interest. One promising approach to this question is through polynomial-bounded verification systems. Verification systems are conceptually simple, and have an intuitive appeal. They lead to a natural characterization of $\mathcal{NP}$ (lemma 3.3.2.a.) and to an alternate form of the question of the closure of $\mathcal{NP}$ under complements (corollary 3.3.3.b.).

As a consequence of corollary 3.3.3.b., the question of closure of $\mathcal{NP}$ under complements is reduced to the investigation of verification systems for the complement of any particular $\mathcal{NP}$-complete language. An obvious choice for this language is TAUTOLOGIES of the propositional calculus. The many systems that logicians have proposed for proving tautologies are obvious candidates for polynomial-bounded verification systems. In fact, almost all of these proof systems can be easily made into verification systems, by regarding the proofs as strings over some alphabet and having the verification system map a proof into the formula proved. If some string does not code a valid proof, the verification system maps it into some fixed tautology such as $p \vee \neg p$. For most proof systems it is easy to see how to check a string to see that it codes a valid proof and find the formula proved in polynomial time. All that remains is to check the verification systems derived from these proof systems to see if they are polynomial-bounded.

Although no polynomial-bounded verification system for TAUTOLOGIES has been found in this way, a number of interesting results have been obtained. First, a number of proof systems have been proven not to be polynomial-bounded, and thus can be eliminated from further consideration. Second, many of the remaining systems have been compared, and simulation results have been obtained which show that one system is polynomial-bounded only if another system is polynomial-bounded. Thus, attention should be concentrated

on the second system, since it is "at least as powerful" as the first system.

Proof systems for the propositional calculus will be surveyed in chapter 4., and chapter 5. follows with the lower bound and simulation results.

4. Proof Systems for the Propositional Calculus

This chapter surveys the major types of systems that have been proposed for proving theorems in the propositional calculus. Some of the systems described here are specific to the propositional calculus, but others are derived from systems that were proposed for the predicate calculus. Since the propositional calculus is a subsystem within the predicate calculus, any proof system for the predicate calculus contains within it a proof system for propositional logic.

Of the systems reported here, some are designed to prove tautologies, some are to be used to prove inconsistent formulas, and still others can be used to prove both valid and inconsistent formulas. All of these systems can be regarded as proof systems for tautologies if a proof that the negation of a formula is inconsistent is accepted as a proof that the formula itself is a tautology.

The first section of this chapter describes a language for the propositional calculus and gives the standard definitions and basic results of the theory. The next section contains subsections describing each of the major types of proof systems that have been proposed for propositional logic. The chapter closes with a description of how these systems can be made into verification systems and some concluding remarks.

4.1. PROPOSITIONAL CALCULUS

Let $\mathbb{B}=\{\mathsf{T},\mathsf{F}\}$. The symbols T and F represent the *truth values*, or propositional constants, *true* and *false*, respectively. A *truth function* of n variables is a function $f:\mathbb{B}^n\to\mathbb{B}$, sometimes called an *n-ary* truth function.

The basic objects of propositional calculus are *atoms* (propositional variables), which will be denoted by lower-case letters: p, q, r, *etc.*, and propositional *connectives*. Each connective $\star$ has associated with it a nonnegative integer n^*, called its *arity*, and an n^*-ary truth function f^*, called its *semantic function*. When the meaning is clear from context, the symbol for the connective $\star$ is sometimes used to denote the semantic function f^*.

A *formula* is a finite, rooted, ordered, labelled tree that either consists of a single node whose label is an atom or has some connective $\star$ (called its *principal connective*) labelling its root and n^* formulas (called its *principal subformulas*) as its maximal proper subtrees. Any subtree of a formula (including the formula itself) is called a *subformula*. Formulas will be denoted by upper-case letters: A, B, C, *etc.* If A is a formula, then the set of subformulas of A is denoted by $sub(A)$, and the set of subformulas of A which are atoms, called the *atoms of* A, is denoted by $at(A)$. Sets of formulas will be denoted by upper-case Greek letters: Γ, Δ, Θ, Λ, *etc.* Both *sub* and *at* may also be applied to sets of formulas.

If $\mathcal{A}$ is a set of atoms, then a *truth assignment* to $\mathcal{A}$ is a function $\tau: \mathcal{A} \to \mathbb{B}$. If n is the cardinality of $\mathcal{A}$, then there are 2^n distinct truth assignments to $\mathcal{A}$. Given any ordering of $\mathcal{A} = \langle p_1, \ldots, p_n \rangle$, there is a one-to-one correspondence between truth assignments to $\mathcal{A}$ and n-tuples from $\mathbb{B}^n$ given by $\tau \leftrightarrow \langle \tau(p_1), \ldots, \tau(p_n) \rangle$. It will be convenient to assume that any set of atoms has a natural ordering (lexicographic, for instance) so that this correspondence can be exploited.

The truth function *expressed* by formula A is an n-ary truth function f_A, where n is the cardinality of $at(A) = \langle p_1, \ldots, p_n \rangle$. The function f_A is defined inductively on the subformulas of A as follows:

1) If C is p_j, then $f_C(x_1, \ldots, x_n) = x_j$
2) If the principal connective of C is $*$ and the principal subformulas of C are $B_1, \ldots, B_{n^*}$, then

$$f_C(x_1, \ldots, x_n) = f^*(f_{B_1}(x_1, \ldots, x_n), \ldots, f_{B_{n^*}}(x_1, \ldots, x_n)).$$

If τ is a truth assignment to some set of atoms $\mathcal{A}$, then τ gives a function from formulas A for which $at(A) \subseteq \mathcal{A}$ into $\mathbb{B}$, defined by $\tau(A) = f_A(\tau)$. Thus, truth assignments extend to formulas in a natural way.

Truth assignment τ *satisfies* A if $\tau(A) = \mathsf{T}$, and *falsifies* A if $\tau(A) = \mathsf{F}$. Formula A is *satisfiable* (also *consistent*) if there is a truth assignment that satisfies A, and A is *falsifiable* if there is a truth assignment that

falsifies A. If A is not satisfiable, then A is said to be *unsatisfiable* (also *inconsistent*). If A is not falsifiable, then A is *valid* (also a *tautology*). A formula must be either satisfiable or falsifiable, and a formula that is both satisfiable and falsifiable is called *contingent*. Note that A is valid if and only if f_A is the constant function $\top$ (*i.e.* every truth assignment makes A true). Dually, A is inconsistent if and only if f_A is the constant function $\bot$ (and every truth assignment makes A false).

A set of formulas Γ *logically implies* formula A (denoted $\Gamma \models A$) if every truth assignment to $at(\Gamma \cup \{A\})$ that does not falsify any formula in Γ satisfies A. Note that this definition says that for $\Gamma = \phi$ (the empty set), $\phi \models A$ (usually shortened to $\models A$) if and only if A is a tautology. Formulas A and B are *logically equivalent* (denoted $A \sim B$) if $A \models B$ and $B \models A$.

If $A_1,\ldots,A_k$ are formulas and $q_1,\ldots,q_k$ are distinct atoms, then the *substitution* $\sigma = \frac{A_1,\ldots,A_k}{q_1,\ldots,q_k}$ is the mapping from formulas to formulas such that $\sigma(B)$ (usually written $B\sigma$) is the formula obtained by (simultaneously) replacing all occurrences of each q_i in B by the corresponding A_i. The result of this substitution is necessarily a formula, since a formula can appear anywhere that an atom can appear in a formula. The formula $B\sigma$ is said to be an *instance* of

B (under σ). The application of a substitution to a set of formulas results in the substitution being applied separately to each formula. If Γ is a formula or set of formulas, then a *renaming* for Γ is a special kind of substitution $\rho = \frac{r_1,\ldots,r_k}{q_1,\ldots,q_k}$, where $\{r_1,\ldots,r_k\}$ are distinct atoms disjoint from the set $(at(\Gamma)-\{q_1,\ldots,q_k\})$. If ρ is a renaming for A, then there is a suitable reordering of the arguments of f_A such that $f_{A\rho}=f_A$ (reordered).

The connectives most commonly encountered are $\neg$ (negation), $\vee$ (disjunction), $\&$ (conjunction), $\supset$ (implication), and $\equiv$ (equivalence). For completeness, all nontrivial connectives of arity no greater than two will be introduced. The two nullary connectives are T (where $f^{\mathsf{T}}=\mathsf{T}$), and F (where $f^{\mathsf{F}}=\mathsf{F}$). The unary connective is $\neg$, where $\neg(\mathsf{T})=\mathsf{F}$ and $\neg(\mathsf{F})=\mathsf{T}$. The other three unary truth functions are the two constant functions (which can be represented by T and F) and the identity function. The binary connectives and the table of values for their semantic functions are given in table 4.1.i. Notice that ten of the possible sixteen binary truth functions are present. The six missing functions are the two constant functions (which can be represented by T and F), the two projection functions, and the two negated projection functions (which can be represented using $\neg$).

x_1	x_2	$*:$	$\vee$	$\subset$	$\supset$	$\equiv$	$\&$	$\mid$	$\not\equiv$	$\not\supset$	$\not\subset$	$\downarrow$
T	T		T	T	T	T	T	F	F	F	F	F
T	F		T	T	F	F	F	T	T	T	F	F
F	T		T	F	T	F	F	T	T	F	T	F
F	F		F	T	T	T	F	T	F	F	F	T

TABLE 4.1.i.

Binary Connectives and their Semantic Functions

It will also prove convenient to introduce four connectives of arity greater than two. For each $n \geq 0$ the n-ary connectives V_n, E_n, $\&_n$, and S_n have the following semantic functions:

for $n \geq 1$:

$V_n(x_1,\ldots,x_n) = F$ if and only if $x_1=\ldots=x_n=F$.

$E_n(x_1,\ldots,x_n) = F$ if and only if the number of $x_i=F$ is odd.

$\&_n(x_1,\ldots,x_n) = T$ if and only if $x_1=\ldots=x_n=T$.

$S_n(x_1,\ldots,x_n) = T$ if and only if the number of $x_i=T$ is odd.

$V_0=S_0=F$

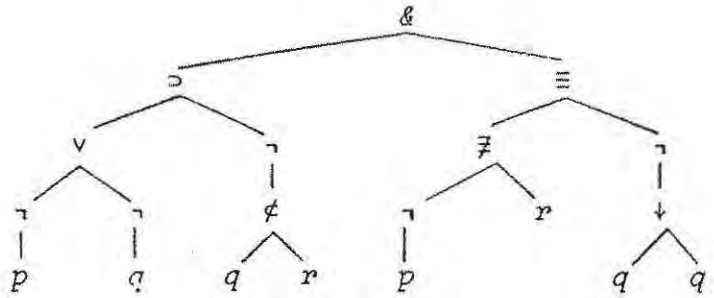
$E_0=\&_0=T$.

Note in particular that $V_2=\vee$, $E_2=\equiv$, $\&_2=\&$, $S_2=\not\equiv$, and $V_1=E_1=\&_1=S_1=$ the identity function. Note also that for $n \geq 2$, V_n, E_n, $\&_n$, and S_n can be represented by iterated formulas in $\vee$, $\equiv$, $\&$, and $\not\equiv$ respectively. For example, $V_n(x,\ldots,x_n) = \vee(x_1,\vee(x_2,\ldots\vee(x_{n-1},x_n)\ldots))$. Since the semantic functions for $\vee$, $\equiv$, $\&$, and $\not\equiv$ are commutative and associative, the order of the arguments and their parenthesization are irrelevant. Thus V, E, $\&$, and S (subscript n will be dropped whenever this causes no confusion) are well-defined when applied to sets.

A set κ of connectives is *adequate* if for every truth function θ there is a formula with only connectives from κ that expresses θ. It can be shown that, for connectives of arity no greater than two, there are 26 minimally adequate

sets of connectives ($|$, $\downarrow$, $\neg\vee$, $\neg\subset$, $\neg\supset$, $\neg\&$, $\neg\neq$, $\neg\not\equiv$, $T\not\equiv$, $T\not\subset$, $F\subset$, $F\supset$, $\subset\not\equiv$, $\supset\not\equiv$, $\not\equiv\equiv$, $\not\subset\equiv$, $\subset\not\equiv$, $\subset\not\subset$, $\supset\not\equiv$, $\supset\not\subset$, $T\vee\not\equiv$, $T\&\not\equiv$, $F\vee\equiv$, $F\&\equiv$, $\vee\equiv\not\equiv$, and $\&\equiv\not\equiv$) and 4 maximally inadequate sets of connectives ($TF\vee\&$, $TF\neg\equiv\not\equiv$, $T\vee\subset\supset\&\equiv$, and $F\vee\&\not\equiv\not\subset\not\equiv$).

Formulas in nullary, unary, and binary connectives are customarily represented by strings in a fully-parenthesized infix notation. For example,



is represented by $(((\neg p\vee\neg q)\supset\neg(q\not\subset r))\&((\neg p\not\equiv r)\equiv\neg(q\downarrow q)))$.

A *literal* (represented by lower-case Greek letters: ξ, ζ, *etc.*) is an atom p or the negation of an atom $\neg p$. The *complement* of ξ (denoted $\bar{\xi}$) is $\neg p$ if ξ is p, and it is p if ξ is $\neg p$. A *disjunction* is a formula whose principal connective is $\bigvee_n$, and a *conjunction* is a formula whose principal connective is $\&_n$. A *clause* is a set of literals, where the implied connective is disjunction. The implied connective for a *set of clauses* is conjunction. A set of clauses is sometimes said to be in *conjunctive normal form* (*CNF*). Every formula is logically equivalent to some formula in conjunctive normal form.

The following basic facts about formulas will prove useful: ("iff" abbreviates "if and only if".)

1. $\models A$ iff $\neg A$ is inconsistent.
2. $A \models B$ iff $\models A \supset B$.
3. $\neg\neg A \sim A$.
4. If $\Gamma \models A$, then $\Gamma\sigma \models A\sigma$.
5. If $\Gamma \models B_1, \ldots, \Gamma \models B_n$, and $\{B_1, \ldots, B_n\} \models A$, then $\Gamma \models A$.
6. If $\Gamma \models A$, then $\Gamma \cup \{B\} \models A$.
7. $A \models \mathsf{T}$ iff $\models A$.
8. $A \models \mathsf{F}$ iff A is inconsistent.
9. $A_1, \ldots, A_n \models B$ iff $\models (\&_n(A_1, \ldots, A_n) \supset B)$.

4.2. PROOF SYSTEMS

Any formula is either valid, inconsistent, or contingent, and the purpose of proof systems is to determine in which of these three classes a given formula lies. Since A is valid if and only if $\neg A$ is inconsistent, any procedure which distinguishes either tautologies from falsifiable formulas or unsatisfiable formulas from satisfiable ones is adequate to perform this three-way partition of formulas. Such a procedure can be built using a proof system for verifying valid or inconsistent formulas and truth assignment evaluations for verifying falsifiable or satisfiable formulas.

Most proof systems presented in the literature on mathematical logic operate on restricted classes of formulas, such as formulas containing only certain connectives, or

formulas of certain restricted forms (*e.g.* sets of clauses). To show that such a system has general applicability, then, one must show that the restricted class of formulas is adequate to express all truth functions, and one must also show how to translate an arbitrary formula into an equivalent one in the restricted class. References in the following subsections describing the major families of proof systems indicate only where descriptions of these systems can be found in the literature, and do not necessarily cite original sources.

4.2.1. Frege-type Systems

One of the most important classes of proof systems is built around the ideas in a system of Frege [Frege 1879]. In systems of this type, certain formulas of simple form, called *axioms*, can be recognized as tautologies by inspection. Then certain rules are used to derive other formulas from the axioms and previously derived formulas. The rules are such that if the original formulas are valid, then the derived formula is also valid. If the axioms and rules of such a system are chosen properly, then every tautology will have some derivation. In this case, the system is said to be *complete*. A proof that some formula A is a tautology is a derivation of A from the axioms according to the rules of the system.

There are many proof systems in the literature that follow this general outline. They can all be treated together, however, under the general concept of *Frege system*, which is given a precise definition below.

A *rule of inference* is a pair $\langle\Gamma,A\rangle$, written $R=\Gamma\rightarrow A$, where Γ is a (possibly empty) finite set of formulas and A is a formula. Rule R is said to be *a rule in* κ if A and the formulas in Γ contain only connectives from the set κ. Rule R is *sound* if $\Gamma\models A$. By fact 4. at the end of section 4.1., if R is sound, then $\Gamma\sigma\models A\sigma$, for any substitution σ. If R is sound and $\Gamma=\phi$, then all substitution instances of A are tautologies, and R is often called an *axiom(scheme)*. If $R=\{A_1,\ldots,A_k\}\rightarrow B$ is a rule of inference and $C_1,\ldots,C_k,D$ are formulas, then D is *inferred* from $C_1,\ldots,C_k$ by R if there is a substitution σ such that $C_1=A_1\sigma,\ldots,C_k=A_k\sigma$, and $D=B\sigma$. If R is sound and D is inferred from $C_1,\ldots,C_k$ by R, then $C_1,\ldots,C_k\models D$.

An *inference system* is a pair $I=\langle\kappa,\mathcal{R}\rangle$, where κ is a set of connectives, and $\mathcal{R}$ is a finite set of sound rules of inference in the connectives κ. A *derivation* in I of formula B from the set of formulas Γ is a sequence $D=\langle A_1,\ldots,A_n\rangle$ of formulas in κ such that for each i, $1\leq i\leq n$, A_i is inferred from formulas in $\Gamma\cup\{A_1,\ldots,A_{i-1}\}$ by some rule in $\mathcal{R}$, and $A_n=B$. The notation $\Gamma\vdash_I A$ *via* D means D is a derivation of A from Γ in inference system I, and $\Gamma\vdash_I A$ means there is a D such that $\Gamma\vdash_I A$ *via* D. When I is clear from context, $\Gamma\vdash A$ will be used for $\Gamma\vdash_I A$.

Since $\models$ is a transitive relation and the rules of I are sound, it follows that $\Gamma \models A$ whenever $\Gamma \vdash_I A$. System I is *complete* if $\vdash_I A$ for every tautology A in the connectives κ. System I is *implicationally complete* if $\Gamma \vdash_I A$ for every Γ and A in the connectives κ for which $\Gamma \models A$.

If D is a derivation, then $D\sigma$ is the sequence of formulas obtained by applying σ to each formula in D. It is useful to observe that, since rules of inference are transparent to substitution, if $\Gamma \vdash_I A$ *via* D and if σ is a substitution in κ, then $\Gamma\sigma \vdash_I A\sigma$ *via* $D\sigma$.

A *Frege system* is an implicationally complete inference system $F=\langle \kappa, \mathcal{R} \rangle$, where κ is adequate. An *s-Frege system* (Frege system with substitution) is a Frege system $F=\langle \kappa, \mathcal{R} \rangle$ whose derivations allow inferences according to the rules in $\mathcal{R}$ (as in a Frege system) and also according to the *substitution rule*: if σ is a substitution in κ, then $A\sigma$ can be inferred from A. Note that the substitution rule is not even sound, in general, so that an s-Frege system is not implicationally sound. (For example $\neg p$ is obtained from p by the substitution rule, but is certainly not a logical consequence of it.) In a derivation of a tautology from $\Gamma=\phi$, however, all formulas in the derivation are tautologies. Since $A\sigma$ is valid whenever A is valid, then, the substitution rule does give valid inferences in this special case. Therefore s-Frege systems will be used only for deriving tautologies from no hypotheses.

The notion of a Frege system is intended to describe the essential characteristics of the propositional fragments of the deductive systems found in most textbooks on mathematical logic. Although Kleene calls such systems "Hilbert-type systems", the first was probably Frege's original system [Frege 1879], which had six axiom schemes and the rule *modus ponens*: $\{p,(p\supset q)\}\rightarrow q$. A very common system described by Mendelson [Mendelson 1964] is the system $M=\langle\kappa,\mathcal{R}\rangle$, where $\kappa=\{\neg,\supset\}$, and $\mathcal{R}=\{\rightarrow(p\supset(q\supset p)),\rightarrow((p\supset(q\supset r))\supset((p\supset q)\supset(p\supset r))),\rightarrow((\neg p\supset\neg q)\supset(q\supset p)),\{p,p\supset q\}\rightarrow q\}$. Other Frege systems can be found in [Hilbert & Ackermann 1950], [Kleene 1952,1967], [Mendelson 1964], and [Shoenfield 1967].

The idea behind s-Frege systems is that once a theorem has been proven, substitution instances of it can be used as hypotheses (effectively, as additional axioms) to prove further theorems. Most logic texts use this technique in their informal development of propositional theories. Although there is an effective procedure for eliminating all uses of the substitution rule in a Frege system derivation (by re-deriving each substitution instance of a theorem), this procedure could potentially cause an exponential increase in the length of the resulting derivation.

Since the set of connectives for a Frege system must be adequate, and since the system must be complete with respect to those connectives, Frege systems have the kind of general applicability described in the beginning of section 4.2. There are mechanical procedures for translating

any given formula into a logically equivalent formula in any adequate set of connectives. With such a procedure for translating formulas into its connectives, then, a Frege system becomes a complete proof system for all formulas.

4.2.2. Natural Deduction

A consequence of the implicational soundness of Frege systems is the *Deduction Theorem*: if $\Gamma, A \vdash B$, then $\Gamma \vdash$ "$A \supset B$", where "$A \supset B$" is any formula logically equivalent to $(A \supset B)$ in the connectives of the Frege system. In the development used in most logic texts, the deduction theorem is difficult to prove, and is used to show the completemess of the system. The important thing to note about this theorem, however, is that it provides a new kind of inference rule. This rule allows one to infer something about the derivability of a certain formula from a certain set of formulas by showing a (possibly easier) derivation of a slightly different formula from a slightly different set of hypotheses. Thus, there is the potential, at least, for much shorter derivations by appealing to this rule.

This observation has led to the development of proof systems from Frege systems and the deduction theorem which appear to yield more "natural" proofs (see [Kleene 1967], [Fitch 1952], [Thomason 1970]). These systems, which are often called *natural deduction systems*, can all be cast in the following more general framework.

A *line* is a pair $L=\Gamma\vdash A$, where Γ is a finite set of formulas and A is a formula. Under any truth assignment, L takes on the same truth value as $(\&(\Gamma)\supset A)$, so that the concepts of satisfiability, validity, logical consequence, *etc.*, for lines are well-defined. Substitutions apply to lines in the natural way, and if Δ is a set of formulas, then the *restriction* of L to the *environment* Δ is the line $\Delta L=(\Delta\cup\Gamma)\vdash A$. Substitutions and restrictions can also be applied to sets of lines. If Λ is a set of lines, L is a line, Γ is an environment (set of formulas), and σ is a substitution, then observe that $\Lambda\models L$ implies that $\Gamma\Lambda\sigma\models\Gamma L\sigma$. A *rule* in a natural deduction system is a pair $R=\Lambda\rightarrow L$, where Λ is a set of lines, and L is a line. Rule R is *sound* if $\Lambda\models L$. Line L' is *inferred* from the set of lines Λ' by rule R if there are a substitution σ and an environment Γ such that $\Lambda'=\Gamma\Lambda\sigma$ and $L'=\Gamma L\sigma$. A *natural inference system* is a pair $N=\langle\kappa,\mathcal{R}\rangle$, where κ is a set of connectives, and $\mathcal{R}$ is a finite set of sound natural deduction rules. The definitions of *derivation*, *completeness*, and *implicational completeness* for such systems are analogous to those for Frege systems, with lines taking the place of formulas. A *proof* of the validity of A is a derivation of the line $\vdash A$. A *natural deduction (ND) system* is an implicationally complete natural inference system with an adequate set of connectives. By analogy to s-Frege systems, *s-ND systems* are defined to be ND systems whose derivations also allow applications of the substitution rule. As with s-Frege systems, s-ND

systems are only used in derivations with no hypothesis lines.

One way of deriving ND systems is by adding the deduction theorem to a Frege system. If F is a Frege system, then if each formula A in each rule of F is replaced by the line $\vdash A$, a set of ND rules is obtained. To obtain a complete ND system, add the rules $p\vdash q \rightarrow \vdash "p\supset q"$ (deduction theorem), $\rightarrow p\vdash p$, and $p\vdash q \rightarrow p,r\vdash q$. To see that this system N is implicationally complete, suppose $\Gamma_1\vdash A_1,\ldots,\Gamma_n\vdash A_n \models \Delta\vdash B$. Then $\&(\Gamma_1)\supset A_1,\ldots,\&(\Gamma_n)\supset A_n \models \&(\Delta)\supset B$, so that there is a derivation D in the system F (which is implicationally complete) such that $"\&(\Gamma_1)\supset A_1",\ldots,"\&(\Gamma_n)\supset A_n" \vdash_F "\&(\Delta)\supset B"$ *via* D, where the conjunctions and implications are suitably represented in the connectives of F. In particular, $"\&(C_1,\ldots,C_k)\supset A_i"$ is represented by $"C_1\supset(\ldots(C_k\supset A_i)\ldots)"$. Then, using the rules copied from F, a derivation D' can be constructed by changing each formula C in D into the line $\vdash C$. For $1\leq i\leq n$, the derivation D_i can be constructed by repeated application of the deduction theorem rule, so that $\Gamma_i\vdash A_i \vdash_N \vdash "\&(\Gamma_i)\supset A_i"$ *via* D_i. Let $D^{\supset}$ be a derivation such that $p,"p\supset q" \vdash_F q$ *via* $D^{\supset}$, so that $C,"C\supset B" \vdash_F B$ *via* $D^{\supset}\frac{C,B}{p,q}$. Then, by changing each formula A of $D^{\supset}\frac{C,B}{p,q}$ to the line $\Gamma,C\vdash A$, a derivation $\hat{D}$ is obtained, where $\Gamma,C\vdash C$, $\Gamma,C\vdash "C\supset B" \vdash \Gamma,C\vdash B$ *via* $\hat{D}$. By the last two rules, $\Gamma,C\vdash C$ can be derived in a single step, and $\Gamma,C\vdash "C\supset B"$ can be derived from $\Gamma\vdash "C\supset B"$ in a single step, giving a derivation $\tilde{D}$ of $\Gamma,C\vdash B$ from $\Gamma\vdash "C\supset B"$.

Using this idea repeatedly, a derivation D'' of $\Delta\vdash B$ from "$\&(\Delta)\supset B$" can be obtained. Putting this all together, the derivation $D_1 \ldots D_n D' D''$ is a derivation of $\Delta\vdash B$ from $\Gamma_1\vdash A_1, \ldots, \Gamma_n\vdash A_n$.

Another way to derive ND systems is to devise two rules for each connective: one for introducing it in the consequent formula of the line concluding the rule, and one for eliminating it. For example, the deduction theorem gives a rule for $\supset$-introduction and *modus ponens* is $\supset$-elimination. The following table gives possible introduction and elimination rules for some common connectives.

connective	introduction	elimination
$\neg$	$p\vdash q,\ p\vdash\neg q \rightarrow \vdash\neg p$	$\vdash\neg\neg p \rightarrow \vdash p$
$\vee$	$\vdash p \rightarrow \vdash p\vee q$ $\vdash p \rightarrow \vdash q\vee p$	$\vdash p\vee q,\ p\vdash r,\ q\vdash r \rightarrow \vdash r$
$\supset$	$p\vdash q \rightarrow \vdash p\supset q$	$\vdash p,\ \vdash p\supset q \rightarrow \vdash q$
$\&$	$\vdash p,\ \vdash q \rightarrow \vdash p\&q$	$\vdash p\&q \rightarrow \vdash p$ $\vdash p\&q \rightarrow \vdash q$
$\equiv$	$p\vdash q,\ q\vdash p \rightarrow \vdash p\equiv q$	$\vdash p\equiv q,\ \vdash p \rightarrow \vdash q$ $\vdash p\equiv q,\ \vdash q \rightarrow \vdash p$

Some authors consider the *antecedent* Γ of the line $L=\Gamma\vdash A$ to be a sequence, rather than a set, of formulas. In their formulations of natural deduction systems, rules only operate on formulas at the tail of this sequence. This convention can often save some writing in pencil-and-paper use of the system, since environment formulas, which usually

do not change from one line of a derivation to the next, need not be recopied in each line of the derivation. This saving is obtained, however, at the expense of a restriction on the allowed form of derivations.

Again, as with Frege systems, the existence of procedures for translating formulas into equivalent ones in any adequate set of connectives insures that any ND system has applicability to all formulas.

4.2.3. Sequential Calculus and Tableaux

The logical next step (although historically it came earlier) in successive generalizations of Frege systems is to allow sets of formulas to the right of the $\vdash$-sign as well as to the left. The symmetry of these systems leads to tableau methods that provide a systematic method of searching for a derivation. According to [Kleene 1967], the first of these systems was introduced by Gentzen in 1932 and 1934-35, with some preliminary work having been done by Hertz in 1929. Since then refinements and modifications to Gentzen's system have appeared in the work of Beth (1955) Hintikka (1955), Schütte (1956), and Kanger (1957). These refinements culminated in the very elegant "analytic tableau" method of [Smullyan 1968]. The two types of systems that will be described here are a generalization of the Gentzen-type systems as described in [Kleene 1967] and Smullyan's analytic tableaux. The intermediate systems are mostly notational variants of Gentzen systems, and could easily be formalized as such.

4.2.3.1. Gentzen-type Systems

A *sequent* is an ordered pair $S=\Delta\vdash\Gamma$ of sets of formulas. (Note: Kleene uses $\rightarrow$ in place of $\vdash$ and a horizontal line in place of $\rightarrow$. The present notation is used, however, for uniformity with the closely related ND systems.) Under any truth assignment S takes on the truth value of $(\&(\Delta)\supset\vee(\Gamma))$, so that the concepts of satisfiability, validity, logical consequence, *etc.*, for sequents are well-defined. The empty sequent "$\vdash$" is always false, and is not allowed. A *sequent scheme* has the form $\Lambda_1,\ldots,\Lambda_m,\Delta\vdash\Gamma,\Theta_1,\ldots,\Theta_n$, where the Λ_i and Θ_i are variables whose values range over sets of formulas and Δ and Γ are particular sets of formulas. An *instance* of such a scheme is obtained by applying a substitution σ to the formulas in Δ and Γ and then consistently replacing each of the variables by a set of formulas. For example, the sequent $(r\vee t),s,(p\supset q),(p\supset r)\vdash((p\supset q)\&(p\supset r)),(r\vee t),s$ is an instance of the scheme $\Gamma,p,q\vdash(p\&q),\Gamma$. The set $\mathcal{S}$ of sequent schemes *logically implies* scheme S (written $\mathcal{S}\models S$) if every instance of $\mathcal{S}$ logically implies (in the usual sense) the corresponding instance of S (*i.e.* under the same substitution and replacement of variables).

A *sequent rule* is a pair $R=\mathcal{S}\rightarrow S$, where $\mathcal{S}$ is a set of sequent schemes and S is a sequent scheme. Rule R is *sound* if $\mathcal{S}\models S$. A *sequent inference system* is a pair $G=\langle\kappa,\mathcal{R}\rangle$ where κ is a set of connectives and $\mathcal{R}$ is a finite set of sound sequent rules in the connectives κ.

The notions of *derivation*, *completeness*, and *implicational completeness* for sequent inference systems are defined in an analogous way to Frege systems, with sequents taking the place of formulas. A derivation of the sequent $\vdash A$ is a *proof* of the validity of A, and a derivation of $A\vdash$ proves A inconsistent. Finally, a *sequent system* is an implicationally complete sequent inference system $S=\langle \kappa, \mathcal{R} \rangle$, where κ is adequate.

Gentzen's system, as described in [Kleene 1967], had two rules for each connective: one for introducing it on the right, or *consequent*, side of a sequent ("introduction"), and one for introducing it on the left, or *antecedent*, side ("elimination"). Although such rules could be devised for any connectives, the rules for the five connectives Kleene considered are given below.

connective	introduction rule
$\neg$	$\{\Delta, p\vdash\Gamma\} \rightarrow \Delta\vdash\neg p,\Gamma$
$\vee$	$\{\Delta\vdash p, q,\Gamma\} \rightarrow \Delta\vdash(p\vee q),\Gamma$
$\supset$	$\{\Delta, p\vdash q,\Gamma\} \rightarrow \Delta\vdash(p\supset q),\Gamma$
$\&$	$\{\Delta\vdash p,\Gamma \;;\; \Delta\vdash q,\Gamma\} \rightarrow \Delta\vdash(p\,\&\,q),\Gamma$
$\equiv$	$\{\Delta, p\vdash q,\Gamma \;;\; \Delta, q\vdash p,\Gamma\} \rightarrow \Delta\vdash(p\equiv q),\Gamma$

connective	elimination rule
$\neg$	$\{\Delta \vdash p, \Gamma\} \rightarrow \Delta, \neg p \vdash \Gamma$
$\vee$	$\{\Delta, p \vdash \Gamma \ ; \ \Delta, q \vdash \Gamma\} \rightarrow \Delta, (p \vee q) \vdash \Gamma$
$\supset$	$\{\Delta \vdash p, \Gamma \ ; \ \Delta, q \vdash \Gamma\} \rightarrow \Delta, (p \supset q) \vdash \Gamma$
$\&$	$\{\Delta, p, q \vdash \Gamma\} \rightarrow \Delta, (p \& q) \vdash \Gamma$
$\equiv$	$\{\Delta, p, q \vdash \Gamma \ ; \ \Delta \vdash p, q, \Gamma\} \rightarrow \Delta, (p \equiv q) \vdash \Gamma$

With the addition of the *axiom* rule, $\rightarrow \Delta, p \vdash p, \Gamma$, any set of pairs of these introduction and elimination rules (or the appropriate rules for other connectives) gives a complete sequent inference system for the connectives concerned. If the set of connectives is adequate, then the system is called a *basic Gentzen system*. The fact that basic Gentzen systems are not implicationally complete arises from the fact that introduction and elimination rules are *analytic*. That is, every formula in any sequent on the left side of a rule also occurs (possibly as a subformula) in the sequent on the right side of the rule. Thus, no sequent can be derived which does not contain as subformulas all of the formulas in the sequents from which it was derived.

Two rules which do not make the system implicationally complete, but nevertheless seem to allow for shorter derivations in some examples, are the *thinning* rules:

thinning introduction: $\Delta \vdash \Gamma \rightarrow \Delta \vdash p, \Gamma$

thinning elimination: $\Delta \vdash \Gamma \rightarrow \Delta, p \vdash \Gamma$.

Systems with an introduction and an elimination rule for each connective, the axiom rule, and the two thinning rules will be called *Gentzen systems with thinning*.

An analytic Gentzen-like system for deriving inconsistent sets of clauses, called the *Gentzen system for sets of clauses*, can be defined as follows. The *axiom* rule of this system is $\rightarrow\Delta,\xi,\bar{\xi}\vdash$ (where ξ is any literal), which replaces the axiom and $\neg$-elimination rules of basic Gentzen systems. The conjunction elimination rule is $\Delta,\&(S_1),\&(S_2)\vdash \rightarrow \Delta,\&(S_1 \cup S_2)\vdash$, where S_1 and S_2 are sets of clauses. Note that if S_i consists of a single clause C, then $\&(C)$ is the same as C itself. The disjunction elimination rule is $\{\Delta,\vee(C_1)\vdash \ ; \ \Delta,\vee(C_2)\vdash\} \rightarrow \Delta,\vee(C_1 \cup C_2)\vdash$, where C_1 and C_2 are clauses. The disjunction $\vee(\xi)$ is considered to be the same as ξ, for any literal ξ. A derivation in this system of the sequent $S\vdash$ constitutes a *proof* of the inconsistency of the set of clauses S. The system is complete in the sense that every inconsistent set of clauses has a proof. This system may also be augmented to include the thinning elimination rule.

Basic Gentzen systems become implicationally complete with the addition of the *cut* rule:

$$\{\Delta\vdash p,\Gamma \ ; \ \Lambda,p\vdash\Theta\} \rightarrow \Delta,\Lambda\vdash\Gamma,\Theta.$$

This rule is not analytic, since p appears on the left but not on the right. Basic Gentzen systems with the cut rule added, called *Gentzen systems with cut*, are examples of sequent systems. Note that these systems do not need the thinning rules, since the same effect can be achieved using the cut rule.

In [Kleene 1967] sequents are pairs of sequences (rather than sets) of formulas, so rules must be introduced to allow such structural operations as reordering lists, introducing multiple copies of formulas, and eliminating duplicate copies of formulas. Such operations detract from the basic elegance of Gentzen systems, and will not be discussed.

In [Kleene 1967], derivations in Gentzen systems are trees with sequents at the nodes. Two sequents are adjacent if the second comes from the first (and possibly others) by the application of one rule. The tree format means that (in principle, at least) if a sequent S is used more than once in a derivation, then separate copies of the derivation of S must be supplied for each use of S. This obvious wastefulness is eliminated by letting derivations be sequences, rather than trees, of sequents.

4.2.3.2. Analytic Tableaux

Although it may be wasteful to re-derive a sequent each time it is used, a tree format for derivations in basic Gentzen systems allows for economies of a different sort. In derivations in Gentzen systems, formulas must be copied over many times as other formulas are being built up using the introduction and elimination rules. Smullyan [Smullyan 1968] reduced this copying with his method of analytic tableaux. Information about the sequents is distributed along the branches of the derivation tree, and

at any step the only formulas that must be written down are those that have changed from the previous step.

Smullyan's system uses the connectives $\neg, \&, \vee$, and $\supset$, although it is not hard to see how to extend it to include any other connectives. According to Smullyan, an *analytic tableau* for a formula F is a rooted labelled binary tree constructed as follows:

1. F is placed at the root.
2. For any node C in the tableau,
   a) If C is $\neg\neg B$, then B may be appended to the end of any branch (a *branch* is a path from the root to a leaf) through C.
   b) If C is of *type* α:

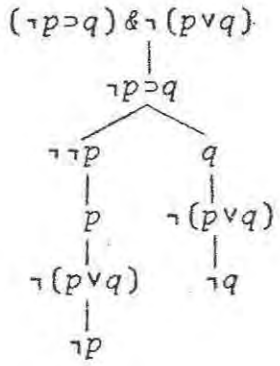
      i.e. if C is $A\&B$, α_1 is A, and α_2 is B;
      or C is $\neg(A \vee B)$, α_1 is $\neg A$, and α_2 is $\neg B$;
      or C is $\neg(A \supset B)$, α_1 is A, and α_2 is $\neg B$

      then either α_1 or α_2 may be appended to the end of any branch through C.
   c) If C is of *type* β:

      i.e. if C is $\neg(A\&B)$, β_1 is $\neg A$, and β_2 is $\neg B$;
      or C is $A \vee B$, β_1 is A, and β_2 is B;
      of C is $A \supset B$, β_1 is $\neg A$, and β_2 is B

      then any branch through C may be extended by adding both β_1 and β_2 as new branches.
3. Any branch that contains both a formula C and its negation $\neg C$ is said to be *closed*. A tableau is *closed* if every branch of it is closed.

As an example, a closed tableau for $(\neg p \supset q) \& \neg (p \vee q)$ is:



a formula F is unsatisfiable if and only if there is a closed analytic tableau for F.

There is a direct correspondence between analytic tableaux and derivations in basic Gentzen systems. It is not hard to see that Smullyan's α-rules are derived from Gentzen's $\&$-elimination, $\vee$-introduction, and $\supset$-introduction, while the β-rules come from $\&$-introduction, $\vee$-elimination, and $\supset$-elimination. The rule for double negations replaces both $\neg$-introduction and $\neg$-elimination. The definition of a closed branch corresponds to Gentzen's axiom rule. With these correspondences, it is not hard to see how a closed analytic tableau for A corresponds to a basic Gentzen derivation of the sequent $A \vdash$.

Analytic tableaux for sets of clauses are constructed as follows:

1. The CNF formula is placed at the root.
2. A single branch is constructed from the root, containing each of the clauses.

3. For any clause $C=\{\xi_1,\ldots,\xi_n\}$, any branch of the tableau may be extended by adding all of $\xi_1,\ldots,\xi_n$ as new branches.
4. Any branch which contains a literal ξ and its complement $\bar{\xi}$ is *closed*.

Analytic tableaux provide an elegant proof system that is satisfying to apply on simple examples. This elegance comes mostly from the restriction of the system to tree-format derivations, however, and as the results of Chapter 5 will show, this restriction leads to very in inefficient derivations in certain cases.

4.2.4. Consensus and Resolution

An operation called *consensus* was introduced by Quine [Quine 1955] as a method to help find the minimum disjunctive normal form for a formula. It was adapted by [Dunham & North 1962] as a computer method for establishing the validity of formulas in disjunctive normal form.

The dual of consensus is *resolution*, which was introduced by Robinson [Robinson 1965a] as part of a proof method for the predicate calculus. It is simplest to regard a *clause* to be a set of literals, where the implied connective is disjunction. The notation $C\xi$ is used to mean the clause $(C\cup\{\xi\})$, and CD means $(C\cup D)$. The empty clause is denoted by $\square$ and, consistent with the convention for $\bigvee_0$, it has the truth value F. Two clauses are said to *clash* if

one contains exactly one literal whose complement is in the other. If two clauses clash, their *resolvent* is defined to be the clause obtained by removing the clashing pair of literals from their union. That is, the resolvent of $C\xi$ and $D\bar{\xi}$ is CD. The resolvent of a pair of clauses is a logical consequence of their conjunction. A resolution *derivation* D of a clause C from a set of clauses S (denoted $S \vdash_r C$ *via* D) is a sequence of clauses, each of which is either a clause from S or a resolvent of two previous clauses in the sequence. If D is a resolution derivation of C from S, then $S \models C$. Robinson's completeness theorem says that if S is inconsistent then $S \vdash_r \square$. Thus, resolution is a complete system for proving the inconsistency of formulas in conjunctive normal form. Since there is an effective procedure to translate any formula into a logically equivalent one in CNF, then, resolution can be used as a general proof system.

Resolution derivations can be regarded as trees or as directed acyclic graphs (*dags*). In either case, the nodes are labelled with clauses, and the two edges into any clause C come from the two clauses (called the *parent* clauses) which clashed to form C as their resolvent. If $C\xi$ and $D\bar{\xi}$ clash to form the resolvent CD, then the edge from $C\xi$ to CD is labelled ξ, and the edge from $D\bar{\xi}$ to CD is labelled $\bar{\xi}$. In the dag formulation, there is one node for each clause in the linear-format derivation, but in the tree formulation, the restriction of out-degree no greater than one requires that the entire derivation of each clause must be repeated

each time the clause is used. If there is a path in a derivation from C to D, then C is an *ancestor* of D and D is a *descendant* of C. If C is a parent of D, then D is an *immediate descendant* of C. The *leaves* (clauses with no ancestors) are the original clauses, and the *roots* (clauses with no descendants) are the derived clauses.

4.2.4.1. Restricted Forms of Resolution

Various attempts at devising efficient resolution-based decision procedures (*i.e.* procedures for finding derivations) have led to a number of restrictions to the resolution rule that still allow the resulting proof procedure to be complete.

The first of these appeared before resolution [Davis & Putnam 1960], but is actually a restricted resolution procedure. In resolution terminology, the Davis-Putnam procedure works as follows: Given a set of clauses S', let $S=S'$, and

> Choose an atom p that appears in S. Form all possible resolvents of clauses that clash on the literals p and $\bar{p}$. If $\square$ is formed, S' is inconsistent. If not, add the new resolvents to S and delete from S all clauses that contain p or $\bar{p}$. If S is empty, S' is consistent. Otherwise repeat this procedure for a new atom p. Continue until S' is found to be either consistent or inconsistent.

The procedure is guaranteed to terminate, since at each step all occurrences of some atom p are removed from S. The number of new clauses generated at any step could potentially be very large, though. The subsumption rule can be added to the Davis-Putnam procedure to cut down this proliferation of clauses somewhat.

If clause C_1 is a subset of C_2, then C_1 is said to *subsume* C_2. Since C_2 is a logical consequence of C_1, C_2 can be discarded. The *subsumption rule* then states: delete from S any clause that is subsumed by some other clause in S. Incorporation of the subsumption rule into the Davis-Putnam procedure can lead to much shorter proofs.

The Davis-Putnam procedure produces resolution derivations of a very restricted form. A Davis-Putnam derivation is completely determined once the original set of clauses and the order of elimination of the atoms are specified. When the derivation is viewed as a dag, the sequence of atoms (literals with $\neg$'s removed) labelling the edges along any path is a subsequence of the elimination sequence. Furthermore, since every permissible resolvent must be formed, Davis-Putnam derivations often contain clauses which make no contribution to the derivation of $\square$ (*i.e.* are not ancestors of $\square$).

A relaxation of these restrictions was given by Tseitin [Tseitin 1968], who defined a resolution derivation to be *regular* if, when it is viewed as a dag, no path contains

the same edge label twice or any literal from the clause at the end of the path as an edge label. That is, a derivation is not regular whenever some literal is removed from a clause in passing from a parent to resolvent, and then that same literal is re-introduced into a descendant clause. Every Davis-Putnam derivation (either with or without subsumption) is regular. Conversely, there are many regular resolution derivations that cannot be generated by the Davis-Putnam procedure, either with or without subsumption.

The regularity restriction is a natural one, and would intuitively seem to lead one toward shorter derivations rather than longer ones. As Tseitin himself puts it [Tseitin 1968, p.118]

> "The regularity condition can be interpreted as a requirement for not proving intermediate results in a form stronger than that in which they are later used. (If A and B are disjunctions such that $A \subseteq B$, then A may be considered to be the stronger assertion of the two; if the derivation of a disjunction containing a variable ξ involves the annihilation of the latter, then we can avoid this annihilation, some of the disjunctions in the derivation being replaced by "weaker" disjunctions containing ξ. In addition to the appearance of ξ, the disappearance of other variables is possible in the course of such a rearrangement.)"

As the results of Chapter 5 will show, the construction described by Tseitin for removing irregularities from a derivation can be used to remove all irregularities from a tree-form resolution derivation without any increase in the size of the derivation. The situation for linear or dag-form derivations is less clear, though, since in this case Tseitin's

method of eliminating irregularities could produce greatly lengthened derivations. On the other hand, it remains an open question whether or not there exists an inconsistent set of clauses whose shortest regular resolution derivation is longer than its shortest non-regular derivation. If not, then as far as lengths of derivations are concerned, regularity would be no restriction at all.

In connection with efforts at predicate calculus theorem proving, numerous resolution strategies have been devised which, at the propositional calculus level, amount to restrictions on the allowed form of resolution derivations. These strategies include unit preference [Wos *et al.* 1964], set of support [Wos *et al.* 1965], semantic resolution [Slagle 1967], linear resolution [Loveland 1970], and linear resolution with merging [Anderson & Bledsoe 1970]. Also of interest is Robinson's hyper-resolution [Robinson 1965b], based on a generalization of the resolution rule.

Perhaps the most successful of these restricted resolution theorem provers for the propositional calculus is Cook's Method I [Cook 1972b]. The method, which is motivated by the method of semantic trees (see subsection subsection 4.2.5.), can best be described by a program. The input to this program is a set of clauses S. The program uses a stack of literals $L=<\xi_1,\ldots,\xi_n>$ and auxiliary variables S_1 and S_2 for sets of clauses, C for a clause, and ξ for a literal. In addition, in order to be completely

specified, Method I requires a *literal selection strategy* $\mathcal{L}$ and a *resolution selection strategy* $\mathcal{R}$. A "pseudo-Algol" program for Method I might look like this:

```
begin
   if □∈S then S is unsatisfiable;
   n←0;
   repeat
      n ← n+1;
      ξ_n ← choose according to L any literal from S such
            that neither ξ_n nor ξ̄_n is in L;
      S_1 ← {clauses C∈S | (∀ξ∈C) ξ̄∈L};
      if S_1≠φ then begin
         ξ_n ← ξ̄_n;
         S_2 ← {clauses C∈S | (∀ξ∈C) ξ̄∈L};
         while S_2≠φ do
            C ← choose according to R any resolvent of some
                clause in S_1 with some clause in S_2;
            if C=□ then S is unsatisfiable;
            n ← maximum integer j such that ξ̄_j∈C;
            ξ_n ← ξ̄_n;
            S ← S∪{C};
            S_1 ← {C};
            S_2 ← {clauses C∈S | (∀ξ∈C) ξ̄∈L}
         end
      end
   until n = the number of distinct atoms in S;
   S is satisfiable
end
```

A *partial truth assignment* to a set Γ of formulas is a map $\pi: at(\Gamma) \to \{\mathsf{T}, \mathsf{F}, \mathsf{U}\}$ (U is for "undefined"). The map π can be extended to $sub(\Gamma)$ by the recursive equations

$\pi(\mathsf{T}) = \mathsf{T}$, $\pi(\mathsf{F}) = \mathsf{F}$,

$$\pi(\neg A) = \begin{cases} \mathsf{T} \text{ if } \pi(A) = \mathsf{F} \\ \mathsf{F} \text{ if } \pi(A) = \mathsf{T} \\ \mathsf{U} \text{ if } \pi(A) = \mathsf{U} \end{cases}$$

$\pi(A * B) = \pi(A) * \pi(B)$, where $\pi(A) * \pi(B)$ is given by table 4.1.i., if $\pi(A) \neq \mathsf{U}$ and $\pi(B) \neq \mathsf{U}$,

and if just one of $\pi(A),\pi(B)\neq U$, then $\pi(A)*\pi(B)$ is T or F if this can be determined without knowing the value of the other, and otherwise $\pi(A)*\pi(B)=U$. For example, if $*$ is $\&$ then $F\&U=F$, $U\&F=F$, but $T\&U=U\&T=U\&U=U$. Partial truth assignment π_1 is an *extension* of π_2 if $\pi_1(p)=\pi_2(p)$ whenever $\pi_2(p)\neq U$.

If π is a partial truth assignment, and f is a truth function, then $f(\pi)$ is the truth function obtained by fixing the arguments of f to the values specified by π. The arguments to $f(\pi)$ are the variables to which π assigns the value U. The function $f(\pi)$ is called the *restriction* of f by π. Partial truth assignment π *satisfies* formula A if $f_A(\pi)$ is identically T, and *falsifies* A if $f_A(\pi)$ is identically F.

A truth function f *depends on* its ith argument if there is a partial truth assignment π such that $\pi(p_i)=U$, $\pi(p_j)\neq U$ for $j\neq i$, and $f(\pi)$ is not constant.

The stack L in the Method I program can be regarded as a partial truth assignment to the atoms of S (the one which makes each literal in L true). Each addition to L is an attempt to extend this partial truth assignment toward a truth assignment that satisfies S. Cook's literal selection strategy is designed to move in the desired direction as rapidly as possible, by satisfying the most crucial clauses at each stage.

For any set of literals C, let $\bar{C}$ denote the set of literals whose complements are in C (note that $\bar{C}$ is not equivalent to the negation of C in general), and let $S\backslash C$ denote the set of clauses obtained from S by first deleting all clauses containing literals whose complements are in C and then deleting from what remains all occurrences of literals from C. Any truth assignment to $at(S)$ which makes each literal in C false (*i.e.* makes each literal in $\bar{C}$ true) satisfies S if and only if it satisfies $S\backslash C$. Consequently, $S\backslash C$ is unsatisfiable if $S \models C$, where C is regarded as a clause.

In particular, the truth assignment represented by L can be extended to a satisfying truth assignment for S if and only if $S\backslash \bar{L}$ is satisfiable. Cook's literal selection strategy is first to isolate from $S\backslash \bar{L}$ those clauses whose number of literals is minimum. Then let p be the atom with the most occurrences in this further reduced set of clauses. The literal selected will be p or $\neg p$, whichever occurs most often in this last set of clauses. Ties are to be resolved arbitrarily. Cook suggests that the resolution selection strategy should be: choose the shortest permissible resolvent.

With these selection strategies, Method I has been extensively tested and compared with other resolution strategies. These tests show that Method I produces derivations that are at least as short as those produced by any other strategy in almost all cases. The method very seldom generates clauses that are not later used to derive $\square$.

4.2.4.2. Extended Resolution

Tseitin introduced a new operation to be used in conjunction with resolution, which he called *extension* [Tseitin 1968]. He showed that in certain cases extension can be used to give much shorter derivations than can be obtained without it.

The *extension* operation works as follows. Let S be a set of clauses, let α, β, and γ be literals such that neither α nor $\bar{\alpha}$ occurs in S, and let $*$ be any binary connective. The *extension* of S with respect to α, β, γ, and $*$ consists of the addition to S of the clauses that make up the conjunctive normal form for the formula $(\alpha \equiv (\beta * \gamma))$. The literals α and $\bar{\alpha}$ are said to be *introduced* into S by this extension. (If β or γ did not occur in S before the extension, they are said to be *added* to S, but they are not introduced.) (Note: Tseitin considered only the case where $*$ is $|$, which is sufficient, but the generality here seems more natural.) The extended set of clauses is satisfiable if and only if the original set S is satisfiable. An *extended resolution derivation* of clause C from the set of clauses S (notation: $S \vdash_{er} C$) consists of a sequence of extensions of S followed by a resolution derivation of C from this extended set of clauses.

THEOREM 4.2.4.2.a.

If S is a set of clauses with $\square \notin S$ and C is a clause, then $S \models C$ if and only if there is an extended resolution derivation D such that $S \vdash_{er} C$ *via* D and no literal in C is introduced *via* extension in D.

Proof

i) Soundness. Suppose $S \vdash_{er} C$. Let S' be the extension of S from which C is derived by resolution alone, and let τ be an arbitrary truth assignment that satisfies S. (If no τ satisfies S, then $S \models C$ trivially.) Then τ can be extended to a truth assignment τ' that satisfies S' as follows. If S was extended by adding the clauses equivalent to $(\alpha \equiv (\beta * \gamma))$, then τ' assigns the same truth value to α that τ assigns to $(\beta * \gamma)$. This step-by-step extension of τ to τ' parallels the step-by-step extension of S to S'. By the soundness of resolution, $S' \models C$, so τ' satisfies C. But since τ and τ' assign the same values to literals that occur in S, τ must also satisfy C. Finally, since τ is an arbitrary truth assignment that satisfies S, $S \models C$.

ii) Completeness. Suppose $S \models C$. Then $(S \backslash C) \models \square$. Let D be a resolution derivation of $\square$ from $S \backslash C$, and let D' be the derivation obtained from D by restoring the literals of C to the clauses from which they were deleted. Thus, D' is a resolution derivation from S of some clause $C' \subseteq C$. Let $C - C' = \{\xi_1, \ldots, \xi_n\}$, and let α_0 be some literal in C', so that $C' = C'' \alpha_0$ (If $C' = \square$, then let $C'' = \square$ and let α_0 be one of the parent clauses of C.). Then, let $S_0 = S$, and for $1 \leq i \leq n$, extend S_{i-1} to S_i by adding the clauses $\alpha_i \bar{\alpha}_{i-1}$, $\alpha_i \bar{\xi}_i$, $\bar{\alpha}_i \alpha_{i-1} \xi_i$, where α_i is a new literal. Note that adding these three clauses is the same as "defining" $\alpha_i \sim (\alpha_{i-1} \vee \xi_i)$. Now, for $i = 1, \ldots, n$, resolve $C'' \alpha_{i-1}$ with $\alpha_i \bar{\alpha}_{i-1}$ to obtain $C'' \alpha_i$, giving an n-step derivation D'' of $C'' \alpha_n$ from $(S_n - S) \cup \{C'' \alpha_0\}$. Then, for $i = n, \ldots, 1$, resolve $C'' \alpha_i \xi_{i+1} \ldots \xi_n$ with $\bar{\alpha}_i \alpha_{i-1} \xi_i$ to obtain $C'' \alpha_{i-1} \xi_i \ldots \xi_n$,

giving an n-step derivation D^3 of $C''\alpha_0\xi_1\ldots\xi_n$ from $C''\alpha_n$. If $C'=\square$, then D' with its last step removed is a derivation of $\bar{\alpha}_0$ and α_0, so that $D'D''D^3$ is a derivation of $\bar{\alpha}_0$ and $\alpha_0\xi_1\ldots\xi_n$ from S_n. One more step gives a derivation of $\xi_1\ldots\xi_n=C$. If $C'\neq\square$, then $C''\alpha_0\xi_1\ldots\xi_n=C$, so that $D'D''D^3$ is a derivation of C from S_n. In both cases, an extended resolution derivation of C from S has been constructed. ☒ 4.2.4.2.a.

In the sense of this theorem, then, extended resolution has a kind of implicational completeness. The extension rule allows the introduction of new literals to be used essentially as abbreviations for arbitrary formulas in the original variables. Results about these formulas can be derived by manipulating the abbreviations, often in many fewer steps than the same results could be derived directly.

The ideas inherent in extension can be used to convert an arbitrary formula into a set of clauses in a much more efficient way than straightforward application of deMorgan's laws [Bauer *et al.* 1973] [Tseitin 1968]. Given any formula A, associate with each distinct subformula B whose principal connective is not $\neg$ a unique atom p^B (atoms of A are associated with themselves). If B is $\neg C$ and the literal γ is associated with C, then associate $\bar{\gamma}$ with B. In this way each subformula B has associated with it some literal ξ^B. For every subformula $B=(C_1*C_2)$ whose principal

connective is binary, let $cl(B)$ be the set of clauses which are equivalent to $(p^B \equiv (\gamma_1 * \gamma_2))$, where γ_1 and γ_2 are the literals associated with C_1 and C_2, respectively. Finally, let $def(A)$ be the union (conjunction) of $cl(B)$ over all binary subformulas B of A. It can be shown by induction on the number of connectives of A that a truth assignment τ satisfies $def(A)$ if and only if for every subformula B of A, $\tau(\xi^B)=\tau(B)$. Thus $def(A) \models \xi^A$ if and only if $\models A$. Consequently $def(A) \cup \{\overline{\xi^A}\}$ is an unsatisfiable set of clauses if and only if A is a tautology. This set of clauses can be derived in time (and space) proportional to the length of A, whereas the CNF for $\neg A$ could have a length which is exponential in the length of A. For this reason, the method described here is preferred.

A resolution derivation (without extension) of $\square$ from $def(A) \cup \{\overline{\xi^A}\}$ is called a derivation of A by *resolution with limited extension*, and is denoted $\vdash_{1e} A$. An extended resolution derivation of ξ^A from $def(A)$ is called an *extension derivation* of A, and is denoted $\vdash_e A$.

Note that there is nothing in this development that precludes the use of these ideas with formulas containing connectives of arity greater than two. The only change required is to define $cl(B)$ for formulas B whose principal connective has arity greater than two.

4.2.5. Other Proof Systems

The most primitive proof system is the method of truth tables. Given a formula A with n atoms, a *truth table* for A is a table of truth values with $n+1$ columns and 2^n rows. The first n columns contain the 2^n truth assignments to the n atoms of A, one assignment per row (The atoms of A are assumed to have some natural order, and one atom is assumed to head each of the first n columns.). The entry in the last column of each row is the truth value assigned to A by the truth assignment given in the first n columns of that row. Formula A is valid if the last column of its truth table is all T's, inconsistent if the last column is all F's, and contingent otherwise.

If conventions are made for a standard ordering of all atoms and a standard ordering for truth assignments, then a truth table may be reduced in size by the omission of all but the last column. This *simplified truth table* consists of a list of the truth values assigned to A by each of the possible truth assignments to A, given in their conventional order. But even this simplified table has a length which is exponential in the length of A in cases where the length of A is proportional to the number of atoms in A.

One way to shorten truth tables somewhat, in certain cases at least, was suggested by Kleene [Kleene 1967]. The idea behind Kleene's method is that a truth value can often be assigned to a formula on the basis of only a partial truth assignment. For instance, the formula $(q \supset (p \& r))$ is

true whenever q is false, regardless of the values of p and r. A *reduced truth table* for n-atom formula A is again a table with $n+1$ columns. Each of the first n columns is labelled by an atom of A, and the last column is (implicitly) labelled by A. The first n columns of each row contain a partial truth assignment sufficient to assign a value to A in the last column of that row. A partial truth assignment is specified by giving truth values in each of the first j columns (for some $1 \leq j \leq n$) and leaving the remaining columns blank. The table must contain enough rows so that each of the possible 2^n truth assignments is an extension of some row in the table, and the rows must be arranged in alphabetical order (blank comes before T, which comes before F, for example), so that this condition can be easily verified.

The method of semantic trees relaxes the restriction of a single ordering on the atoms throughout the analysis, by allowing the values of different atoms to go unspecified in different parts of the analysis of some formula A. Semantic trees were discussed in [Robinson 1968] and [Kowalski & Hayes 1969] as a general method for analyzing mechanical proof procedures for the predicate calculus. The special version presented here is useful as a proof system for the propositional calculus.

A *semantic tree* for a formula A is a finite rooted binary tree, with the pair of edges leading out from each node labelled p and $\neg p$ respectively, for some $p \epsilon at(A)$, and such that no branch (*i.e.* path from root to a leaf) has a

pair of complementary literals on it. A branch determines a partial truth assignment π by the conditions: $\pi(p)=T$ if p labels some edge on the branch, $\pi(p)=F$ if $\neg p$ is on the branch, and $\pi(p)=U$ otherwise. The branch whose partial truth assignment is π is *closed* for A if $\pi(A)\neq U$. A *closed semantic tree* for A is a semantic tree for A with every branch closed for A.

It should be clear from the above discussion that truth tables are a special form of reduced truth tables, and, except for differences in notation, both are special cases of semantic trees. Thus, semantic trees are the most general form of these primitive kinds of proof systems.

At the other end of the scale, powerful formal mathematical theories, such as Zermelo-Fraenkel set theory and various formalizations of number theory, can be used as proof systems for propositional formulas. It may be that these theories give shorter proofs of some families of formulas than any of the other proof systems described in this chapter. The extreme generality of these systems makes them very difficult to analyze, however, and nothing more will be said about them here.

4.3. PROOF SYSTEMS AS VERIFICATION SYSTEMS

Any of the proof systems described in this chapter can be made to fit the formal definition of verification systems by representing the proofs in the system by strings

on some finite alphabet, and by having the function F take a proof into the formula proved. If string w does not code a proof, then define $F(w)$ to be some fixed standard provable formula, for example $(p \vee \neg p)$ if the system proves tautologies in the connectives $\vee, \neg$.

Making any particular proof system into a verification system, therefore, entails two requirements. First, an alphabet and encoding for proofs must be given. Second, it must be shown that the function F of the resulting verification system can be computed in polynomial time. In general, most reasonable encodings will work, so the most straightforward encoding is usually the best.

As an example, let $E=\langle K, \mathcal{R} \rangle$ be a Frege system. The alphabet for the encoding is $\{p, _0, _1, (,), \top, \bot, \neg, \vee, \subset, \supset, \equiv, \&, |, \not\equiv, \not\supset, \not\subset, \downarrow\}$. Atoms are encoded by p subscripted by a string of $_0$'s and $_1$'s. Formulas are represented in fully parenthesized form, as indicated in section 4.1. A derivation is represented by concatenating the representations of the formulas that make up the derivation. Since formulas are fully parenthesized, this string can readily be parsed into its constituent formulas.

The function F that maps w into A must be computable in time bounded by a polynomial in the length of w. In order to determine if w codes a well-formed proof in the system E, w must be parsed into a sequence of formulas, each formula must be examined to see that it is a well-formed formula in

the connectives κ (which amounts to no more than a parse of a string according to a context-free grammar), and then it must be verified that each formula is inferred from previous formulas by a rule in $\mathcal{R}$. The first two of these operations can clearly be performed in polynomial time. In order to check that formulas follow according to the inference rules, the parsing information on each formula is used. With this parsing information, it is not hard to decide if a formula is an instance of another formula (the right-hand side of a rule). For each rule for which the given formula is an instance of the right-hand side, the previous formulas in the derivation are searched for the corresponding instances of the left-hand-side formulas. If the search succeeds for any rule, then the given formula is properly derived from previous formulas. Since the number of rules to try for each formula is finite, this entire verification can be done in polynomial time. Finally, the time to output either the last formula in the sequence (if the proof is well-formed) or the standard tautology (if it is not) is clearly small. Thus, the function F can be computed in polynomial time, and so is a verification system for the set of strings that encode tautologies in the connectives κ.

4.4. SUMMARY AND DISCUSSION

This chapter has described all of the important proof systems that have been proposed in the literature for proving formulas in the propositional calculus. It has also been

briefly indicated how each of these systems can be analyzed as a verification system. According to corollary 3.3.3.b. if one of these verification systems is a polynomial-bounded verification system for the complement of some set in $\mathcal{NP}$, then $\mathcal{NP}$ is closed under complements. Conversely, if each of them can be shown not to be polynomial-bounded, this will add more evidence to support the widely-held speculation that $\mathcal{NP}$ is not closed under complements, and thus, $\mathcal{P} \neq \mathcal{NP}$.

It is the aim of this thesis, and the next chapter in particular, to investigate these proof systems to see what can be said about whether or not any of them are polynomial-bounded verification systems. The great number of systems to be evaluated (an infinite number, in fact, since the family of Frege systems is infinite) would make that task unmanageable, however, without a way of comparing the complexity of proof systems. What is needed is a way to prove results about verification systems of the form: if system *F* is polynomial-bounded then system *G* is also polynomial-bounded. The major results of chapter 5 are results of this type.

5. Studies of Proof System Complexity

In this chapter proof systems from chapter 4. are analyzed. The goal of this investigation is to determine if any of these systems are polynomial-bounded verification systems for tautologies. Some systems have been proven not to be polynomial-bounded, and here it is shown by means of polynomial-time-bounded simulations that the complexities of many of these systems are related.

The chapter begins with two examples of such simulations and a formal definition of simulation. The next section summarizes the simulation and lower-bound results, indicating which were known before and which are new. After that are four sections that give the details of results for Frege systems, natural deduction, sequent and tableau systems, and resolution and related systems.

5.1. SIMULATION

The complexity of proof systems can be compared by showing how one proof system "simulates" another. Suppose that F and G are proof systems for the same set of formulas and that there is a uniform procedure A such that, given any proof P in system F, A transforms P into a proof P' which in system G is a proof of the same formula that P proves in system F. It can then be said that by using procedure A, system G is "simulating" system F. If it is furthermore true that the length of P' is bounded by some polynomial in

the length of P, then the simulation is "efficient". To within a polynomial, then, G is "at least as powerful" as F. If F is polynomial-bounded, then G is also polynomial-bounded, or, to view it the other way, if G is not polynomial-bounded, then F cannot be.

5.1.1. Two Examples

This subsection gives two examples of cases where one proof system simulates another. They are presented here in an informal, intuitive way in order to motivate the formal, and somewhat technical, definition of simulation, which is given in the following subsection. In the first example it is shown how one Frege system can simulate another, and the second example shows how a Gentzen system with cut can be simulated by extension.

5.1.1.1. Two Frege Systems

Recall system M from subsection 4.2.1. [Mendelson 1964] uses for its propositional language the connectives $\neg$ and $\supset$. His proof system has 3 axiom schemes:

$$(p \supset (q \supset p))$$

$$((p \supset (q \supset r)) \supset ((p \supset q) \supset (p \supset r)))$$

$$((\neg p \supset \neg q) \supset (q \supset p))$$

and the rule *modus ponens*:

$$p, (p \supset q) \rightarrow q$$

The system is sound and implicationally complete (over all formulas in $\neg$ and $\supset$) and thus is an example of a Frege system.

A system that superficially looks much different was used by [Shoenfield 1967]. The connectives for Shoenfield's system S are $\neg$ and $\vee$, and there is only one axiom scheme:

$(\neg p \vee p)$.

Shoenfield's four propositional inference rules are

$p \rightarrow (q \vee p)$ (expansion rule)

$(p \vee p) \rightarrow p$ (contraction rule)

$((p \vee q) \vee r) \rightarrow (p \vee (q \vee r))$ (associative rule)

$(p \vee q), (\neg p \vee r) \rightarrow (q \vee r)$ (cut rule).

Although seemingly very different in character from Mendelson's system, Shoenfield's system, being sound and implicationally complete, is also a Frege system.

It is not apparent that there is any relationship between proof lengths in these two systems. To begin with, the systems use different connectives, and, thus, it is not even clear what formulas to consider when comparing proof lengths. The most straightforward method of attack is to take advantage of the equivalences $(p \vee q) \sim (\neg p \supset q)$ and $(p \supset q) \sim (\neg p \vee q)$. Using these equivalences, a formula in one set of connectives can be transformed into an equivalent formula in the other set of connectives, without significantly changing its structure or appreciably increasing its length. For example, $(p \vee \neg((q \vee \neg r) \vee (\neg p \vee \neg(q \vee \neg r))))$ becomes $(\neg p \supset \neg(\neg(\neg q \supset \neg r) \supset (\neg\neg p \supset \neg(\neg q \supset \neg r))))$ and $((p \supset (q \supset r)) \supset ((p \supset q) \supset (p \supset r)))$ becomes $(\neg(\neg p \vee (\neg q \vee r)) \vee (\neg(\neg p \vee q) \vee (\neg p \vee r)))$.

Let $t(F)$ be the translation of formula F by this method. It is important to note that $t(F)$ has the same subformula

structure as F. Connectives $\vee$ and $\supset$ have been interchanged, and a $\neg$ has been inserted in each binary subformula. The length of $t(F)$ is less than twice the length of F, since the number of $\neg$'s inserted is no more than the number of $\vee$'s or $\supset$'s already present.

Now suppose D is a derivation of F in system M. Then D is a sequence of formulas in the connectives $\neg$ and $\supset$, such that each formula in the sequence is either an instance of one of the three axioms or follows from previous rules by the rule *modus ponens*, and such that the last formula in the sequence is F. A derivation D' in system S of $t(F)$ can be obtained by a mechanical translation of D. Furthermore, the length of D' will be proportional to the length of D.

To obtain the skeleton of D', each formula in D is translated into an equivalent formula in $\neg$ and $\vee$. This produces a sequence of valid formulas in Shoenfield's connectives. To "flesh out" this skeleton, each formula is preceded by a sub-derivation which "simulates" in system S the inference rule from system M that was used to derive the original formula. The number of steps of this sub-derivation is determined by the inference rule being simulated, and the lengths of the formulas involved are proportional to the lengths of the formulas being simulated. Thus, the total length of D' is proportional to the length of D.

For example, Mendelson's first axiom $(p\supset(q\supset p))$ translates into $(\neg p\vee(\neg q\vee p))$ in Shoenfield's connectives.

Since this formula is a tautology, and since system S is complete, there is a derivation (say D_{ax1}) of $(\neg p \vee (\neg q \vee p))$ in system S. Figure 5.1.1.1.i. gives one possibility for D_{ax1}, a derivation of 9 lines, containing 16 occurences of the atom p and 7 occurrences of q.

A derivation in system M may contain any instance of this axiom, such as $(A \supset (B \supset A))$. In system S this becomes $(\neg t(A) \vee (\neg t(B) \vee t(A)))$, and a derivation of this translated formula can be obtained from D_{ax1} by substituting $t(A)$ for p and $t(B)$ for q. Since the length of D_{ax1} and the number of occurrences of p and q in D_{ax1} are fixed and since translation no more than doubles the length of a formula, the total length of this simulating derivation $(D_{ax1}\frac{t(A)}{p}\ \frac{t(B)}{q})$ is bounded by a constant (independent of A and B) times the length of the original formula $(A \supset (B \supset A))$. Figure 5.1.1.1.ii. is an example of this substitution where A is $(\neg\neg p \supset p)$ and B is $(q \supset (p \supset r))$. Note that each formula in fig. 5.1.1.1.ii. is obtained from the corresponding formula in fig. 5.1.1.1.i. by substituting $t(A)$ for p and $t(B)$ for q, and that the result is a proper derivation in system S.

Mendelson's other axioms can be handled in the same way. The *modus ponens* rule can also be simulated by similar techniques. Since the *modus ponens* rule $(p, (p \supset q) \rightarrow q)$ is sound (a requirement of Frege system rules), q is a logical consequence of p and $(p \supset q)$. Also, since system S is implicationally complete

line number	formula	origin (rule and lines)
1	$(\neg p \vee p)$	axiom
2	$(\neg\neg p \vee \neg p)$	axiom
3	$(p \vee \neg p)$	cut - 1,2
4	$(\neg q \vee (p \vee \neg p))$	expansion - 3
5	$(\neg\neg q \vee \neg q)$	axiom
6	$((p \vee \neg p) \vee \neg q)$	cut - 4,5
7	$(p \vee (\neg p \vee \neg q))$	associative rule - 6
8	$((\neg p \vee \neg q) \vee p)$	cut - 7,1
9	$(\neg p \vee (\neg q \vee p)) = t(p \supset (q \supset p))$	associative rule - 8

FIGURE 5.1.1.1.i.

D_{ax1}: System S simulation of Mendelson's axiom $(p \supset (q \supset p))$

line number	formula	origin
1	$(\neg(\neg\neg\neg p \vee p) \vee (\neg\neg\neg p \vee p))$	axiom
2	$(\neg\neg(\neg\neg\neg p \vee p) \vee \neg(\neg\neg\neg p \vee p))$	axiom
3	$((\neg\neg\neg p \vee p) \vee \neg(\neg\neg\neg p \vee p))$	cut - 1,2
4	$(\neg(\neg q \vee (\neg p \vee r)) \vee ((\neg\neg\neg p \vee p) \vee \neg(\neg\neg\neg p \vee p)))$	expansion - 3
5	$(\neg\neg(\neg q \vee (\neg p \vee r)) \vee \neg(\neg q \vee (\neg p \vee r)))$	axiom
6	$(((\neg\neg\neg p \vee p) \vee \neg(\neg\neg\neg p \vee p)) \vee \neg(\neg q \vee (\neg p \vee r)))$	cut - 4,5
7	$((\neg\neg\neg p \vee p) \vee (\neg(\neg\neg\neg p \vee p) \vee \neg(\neg q \vee (\neg p \vee r))))$	associative - 6
8	$((\neg(\neg\neg\neg p \vee p) \vee \neg(\neg q \vee (\neg p \vee r))) \vee (\neg\neg\neg p \vee p))$	cut - 7,1
9	$(\neg(\neg\neg\neg p \vee p) \vee (\neg(\neg q \vee (\neg p \vee r)) \vee (\neg\neg\neg p \vee p)))$	associative - 8

FIGURE 5.1.1.1.ii.

An instance of D_{ax1}

(another requirement of Frege systems), it must support a derivation of q from the hypotheses p and $t(p \supset q)$. Such a derivation, designated D_{mp}, is shown in figure 5.1.1.1.iii. To simulate the derivation of B from A and $(A \supset B)$, the required derivation is obtained by substituting $t(A)$ for p and $t(B)$ for q into the last five lines of D_{mp}. (The formulas obtained by substituting into the first two lines must have already appeared earlier in the simulation.)

By this method of translating formulas from D and preceding each translated formula by a sub-derivation of that formula, the derivation D' can be constructed. This derivation will be a legal derivation in system S of $t(F)$. Furthermore, the length of D' is no more than some constant times the length of D. All that was needed in order to show this result was 1) $\ell t(A) \leq c \cdot \ell A$, 2) $t(A\frac{B}{p}) = t(A)\frac{t(B)}{p}$, 3) system M is sound, 4) system S is implicationally complete, 5) any substitution of a derivation is also a derivation, and 6) of all the immediate consequences of a formula $(A \supset B)$ by *modus ponens*, only one (namely B) is not a subformula of the other parent formula. These facts (except 6) remain true when the rôles of systems M and S are interchanged, so that system M can simulate system S in the same way. (Although the derivation may grow in length more than linearly, growth is still bounded by a polynomial.)

But this is not quite enough to show that system M is polynomial-bounded if and only if system S is polynomial-bounded. The problem arises because not every formula in $\neg, \vee$

line number	formula	origin (rule and lines)
1	$p \quad =t(p)$	hypothesis
2	$(\neg p \vee q) = t(p \supset q)$	hypothesis
3	$(q \vee p)$	expansion - 1
4	$(\neg q \vee q)$	axiom
5	$(p \vee q)$	cut - 3,4
6	$(q \vee q)$	cut - 5,2
7	$q \quad =t(q)$	contraction - 6

FIGURE 5.1.1.1.iii.

D_{mp}: Simulation of *modus ponens* by system S

is the image under the t function of some formula in $\neg,\supset$ (*i.e.* because t is not onto). For example, $(p \vee (q \vee \neg p))$ is such a formula. There are several ways this problem can be handled.

One way is to translate the given formula twice. This gives a formula that is equivalent to the given formula, but with every left subformula double-negated. For example, $(p \vee (q \vee \neg p))$ becomes $(\neg\neg p \vee (\neg\neg q \vee \neg p))$, which is the image of $(\neg p \supset (\neg q \supset \neg p))$. It must then be shown that the double negations can be removed without adding too much to the length of the derivation.

Other ways involve various methods of changing the translation between connectives so that every formula is the image of some formula under translation. For the simple example here, where a one-to-one correspondence can be maintained between $\vee$ and $\supset$, the simplest solution is to turn the translation functions around, so that they remove negations whenever possible. Thus, $t'(A \supset B) = \left\{ \begin{array}{l} (\neg t(A) \vee t(B)) \text{ if } A \text{ does not begin with } \neg \\ (t(C) \vee t(B)) \text{ if } A \text{ is } \neg C \end{array} \right\}$. Now every formula in $\neg,\vee$ is the image under t' of some formula in $\neg,\supset$, but the t' function no longer is transparent to substitution. That is, it is no longer true that $t'(A\frac{B}{p}) = t'(A)\frac{t'(B)}{p}$. What is true, however, is that if B does not begin with $\neg$, then $t'(A\frac{B}{p}) = t'(A)\frac{t'(B)}{p}$, and if B is $\neg C$, then $t'(A\frac{B}{p}) = t'(A\frac{\neg p}{p})\frac{t'(C)}{p}$. It is also necessary to note that although the t' function may now shorten some formulas, it does not shorten them much. In particular, $\ell A < 2\ell t'(A)$, so that the translated formula is more than half as long as the original.

With this new translation function, the simulation of system M by system S can proceed as before, except that each axiom and the *modus ponens* rule of system M will be simulated by several "standard" derivations in system S. For example, corresponding to $(p \supset (q \supset p))$ there will be derivations (D^{++}_{ax1}, D^{+-}_{ax1}, D^{-+}_{ax1}, and D^{--}_{ax1}) of $(\neg p \vee (\neg q \vee p))$, $(\neg p \vee (q \vee p))$ $(p \vee (\neg q \vee \neg p))$, and $(p \vee (q \vee \neg p))$. The translation of any instance of this axiom will be an instance of one of these four formulas[†1], so taking the proper instance of the proper derivation gives a derivation of that translated formula. The other axioms are handled the same way, with axioms containing n atoms getting no more than 2^n different simulating derivations[†2]. Two simulations of *modus ponens* are required: a derivation of q from p and $\neg p \vee q$ (as before), and a derivation of q from $\neg p$ and $p \vee q$. (The two cases where q is negated are instances of the above two, but neither of the above two is an instance of the other.) Any instance of *modus ponens*, when translated, becomes an instance of one of these two cases.

[†1] Note that in this case, the first formula is an instance of the second, and the third is an instance of the fourth, so that only two really distinct derivations are required.

[†2] In fact, the minimum number of simulating derivations is no greater than 2^k, where k is the number of atoms that appear both as left subformulas of $\supset$ and some other way (*i.e.* negated or as a right subformula of $\supset$).
(*e.g.* q in $((p \supset (q \supset r)) \supset ((p \supset q) \supset (p \supset r)))$, but not p or r).

Now it should be clear that the arguments given earlier about simulation of system M by system S still carry through. But now the t' function is onto, so that if system M is polynomial-bounded, then system S is also polynomial-bounded. To see that this is so, assume that system M is polynomial-bounded. That is, given any tautology F in $\neg,\supset$, there is a derivation D of F whose length is at most $p(\ell F)$. Now let A' be any tautology in $\neg,\vee$, and let A be a formula in $\neg,\supset$ such that $t'(A)=A'$. By construction of the t' function, $\ell A<2\ell A'$. Since system M is assumed to be polynomial-bounded, let D be a derivation of A of length no more than $p(\ell A)$. Translating D by the method described above yields D', a derivation of $t'(A)$, whose length is at most $c\cdot\ell D$, where c is a constant depending only on the proof systems. Thus, D' is a derivation of A' in system S, and the length of D' is bounded by $c\cdot p(2\ell A')$, which is a polynomial in the length of A'. Therefore, system S is also polynomial-bounded.

The same technique can be used to show that system M is polynomial-bounded if system S is polynomial-bounded. Moreover, since the translations and simulation of system M by system S are linear, if both systems are polynomial-bounded, then the polynomial bound for system S can have no higher degree than the bound for system M.

Finally, it should be noted that this particular method of translation was successful only because $\vee$'s and $\supset$'s could be interchanged on a one-for-one basis. Other special techniques are needed to handle the cases where the simulating system has

more connectives than the system being simulated (*e.g.* How can a translation from $\neg,\vee$ to $\neg,\vee,\&$ be made onto?), or where the system being simulated has $\equiv$ or $\not\equiv$ and the simulating system does not (*i.e.* No formula in $\neg,\vee$ that is equivalent to $((\ldots((p_0\equiv p_1)\equiv p_2)\ldots)\equiv p_n)$ has length $O(n)$.).

5.1.1.2. Extension *vs* Gentzen Systems

This section is an expansion of an idea found in [Tseitin 1968].

Recall from paragraph 4.2.3.1. that a *basic Gentzen system* is a complete analytic sequent inference system with two rules for each connective, and that a *Gentzen system with cut* is a sequent system obtained from a basic Gentzen system by adding the cut rule. Also recall from paragraph 4.2.4.2. that a derivation of A by *resolution with limited extension* is a resolution derivation of $\square$ from $def(A)\cup\{\overline{\xi^A}\}$, and that an *extension derivation* of A is a derivation of ξ^A from $def(A)$ by extended resolution. In this section it will be shown that resolution with limited extension can simulate basic Gentzen systems and extension can simulate Gentzen systems with cut.

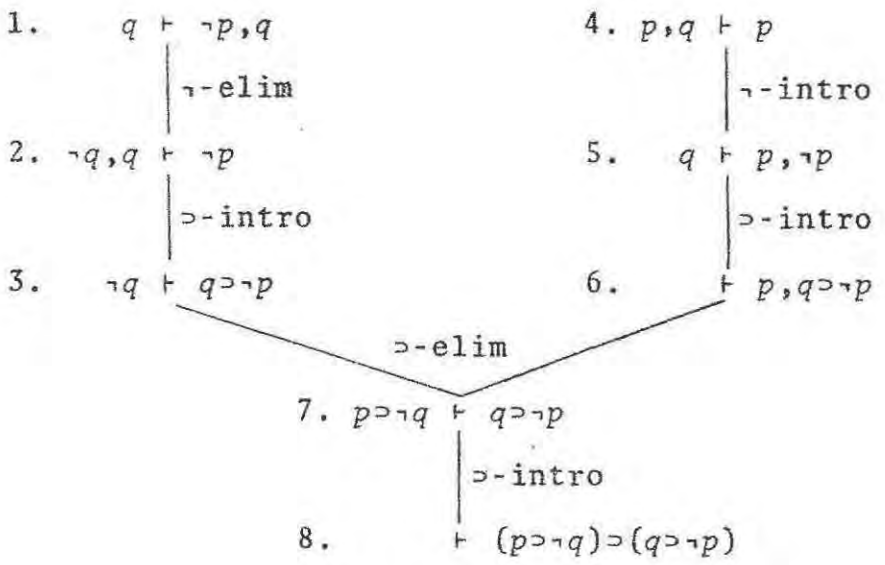
Let A be a tautology, and let D be a derivation of A by a Gentzen system (with or without cut). Using the method of constructing $def(A)$ from A as described in paragraph 4.2.4.2., construct the set of clauses $def(D)$ ($def(D)$ is the union of $def(B)$ over all formulas B that appear in D.). Note that $def(D)=def(A)$ unless D contains some subformula that is not a subformula of A. This latter case can only occur if D

includes applications of the cut rule, since basic Gentzen systems are analytic. In any case, observe that $def(A) \subseteq def(D)$ (since A appears in D), and that the clauses in $def(D)$-$def(A)$ (if any) can be obtained by the extension rule.

Now with each sequent $A_1,\ldots,A_m \rightarrow B_1,\ldots,B_n$ in D associate the clause $\xi^{\overline{A_1}} \ldots \xi^{\overline{A_m}} \xi^{B_1} \ldots \xi^{B_n}$. Refer to figures 5.1.1.2.i. and 5.1.1.2.ii. for two examples. Both show simulations of Gentzen derivations of $((p \supset \neg q) \supset (q \supset \neg p))$ (neither of which is minimal). The first Gentzen derivation is cut-free, while the second derivation contains an application of the cut rule. Sequents are numbered for reference, and the corresponding clauses in the resolution proofs have corresponding numbers with primes.

Note that certain sequents have no corresponding clauses. These fall into two classes. First are the sequents that result from applications of the $\neg$-introduction and $\neg$-elimination rules. The clause corresponding to such a sequent is identical to the clause corresponding to its parent, and thus does not need to be rederived. An example of this is sequent 9. of figure 5.1.1.2.ii., whose associated clause is $\bar{b}\bar{p}\bar{q}$, the same clause that is associated with sequent 8. (Other examples of this situation are sequents 2. and 5. of figure 5.1.1.2.i. and sequent 3. of figure 5.1.1.2.ii., but in these cases, the parent sequent itself has no associated clause.) The other way a sequent can have no associated clause is for that sequent to be an axiom. In this case, the associated clause would

Basic Gentzen Derivation



Simulation by Resolution with Limited Extension

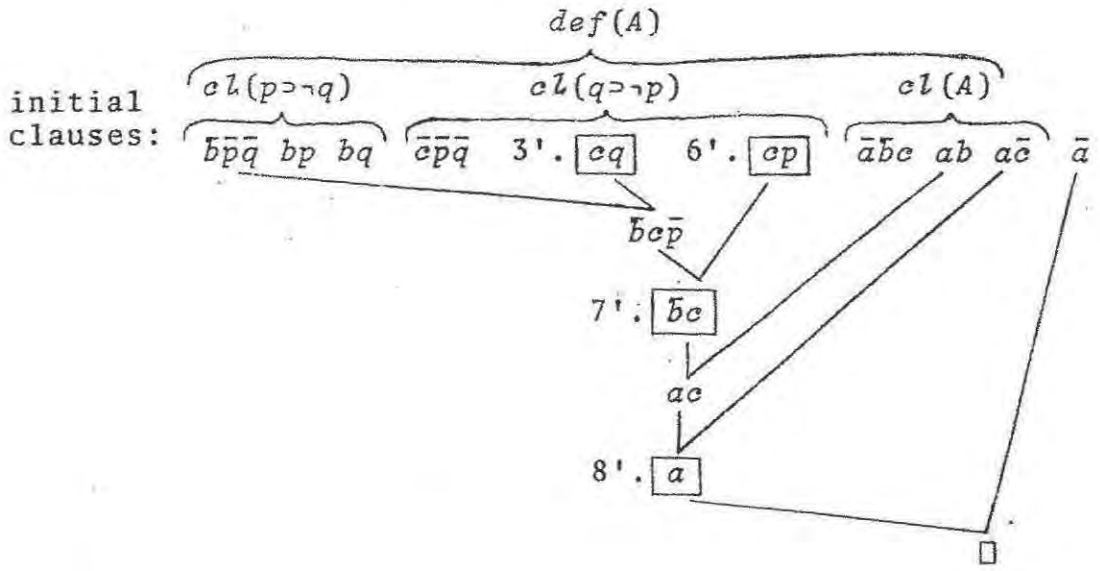
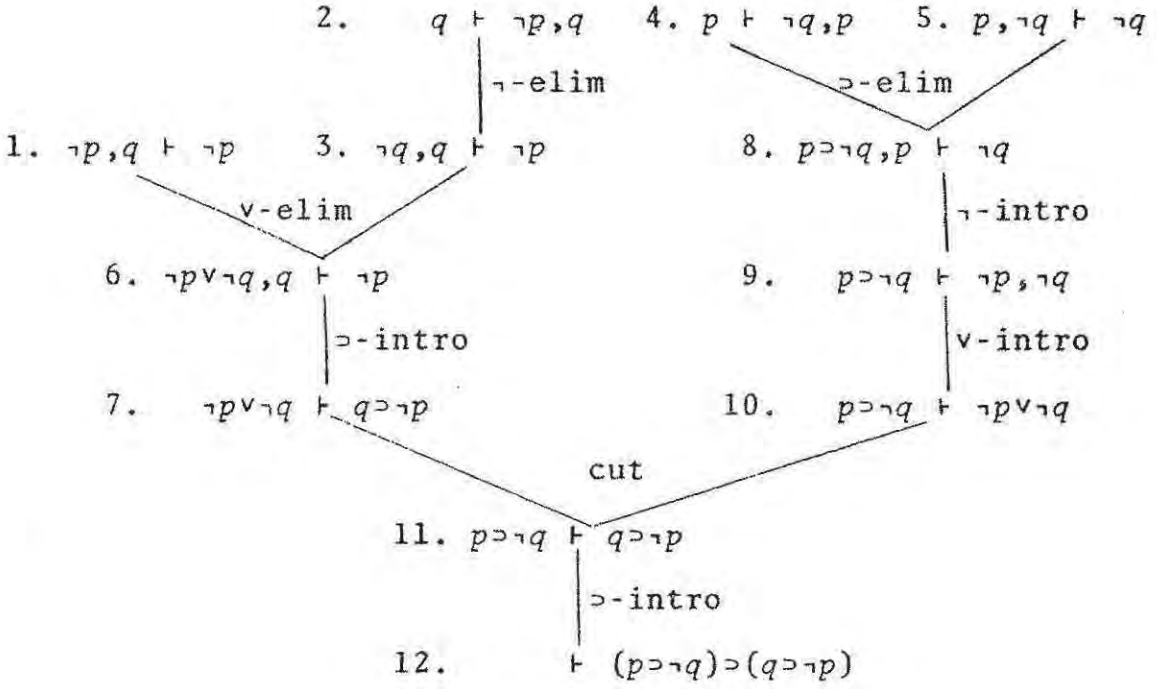


FIGURE 5.1.1.2.i.

Simulation of Basic Gentzen System by Resolution with Limited Extension

Derivation in Gentzen System with Cut



Extension Simulation

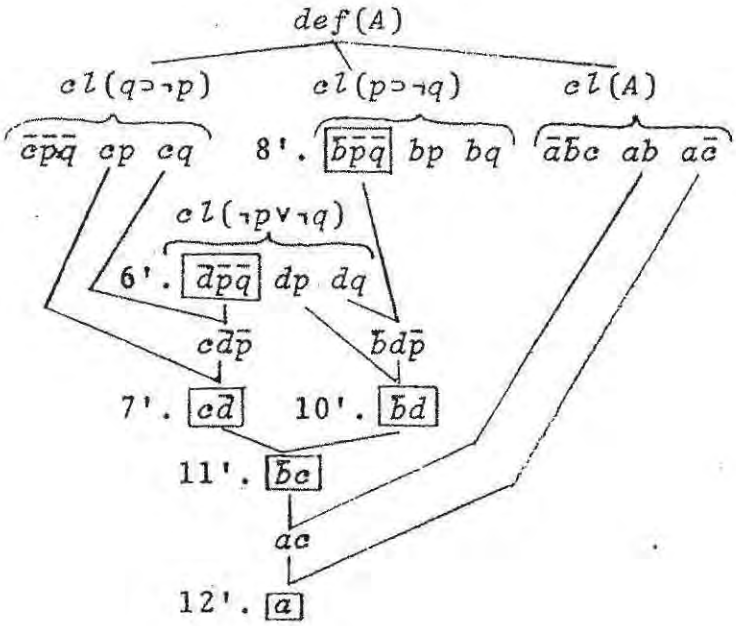


FIGURE 5.1.1.2.ii.

Extension Simulation of Gentzen System with Cut

contain a contradictory pair of literals. Every other sequent has an associated clause.

These clauses (enclosed in boxes in the figures) form the skeleton of a resolution derivation of ξ^A (ξ^A is a in the examples.) from $def(D)$. The skeleton is "fleshed out" with the addition of intermediate resolvents, using the appropriate clauses from $def(D)$. For example, the clause $\bar{b}c$ corresponds to the sequent $p \supset \neg q \vdash q \supset \neg p$. The descendent of this sequent is $\vdash (p \supset \neg q) \supset (q \supset \neg p)$. This sequent is derived by applying $\supset$-introduction to the formula A, and the corresponding clause is a. Using the clauses from $cl(A)$, this resulting clause is derived from the original clause in two resolution steps. Note that the presence of other formulas would not interfere with this resolution derivation (it would just add extra literals to the clauses). Also, it does not matter what formulas B and C the literals b and c represent, so long as a represents $B \supset C$. As a result, $\supset$-introduction is always simulated by two resolutions. Similarly, each of the other basic Gentzen system rules can be simulated by two resolutions (or three in the cases of $\equiv$-elimination and introduction). The cut rule is the same as resolution except that it uses sets of formulas, rather than sets of literals, as clauses. Thus, every application of the cut rule gives rise to exactly one resolution. (See sequent 11. and clause 11'. in figure 5.1.1.2.ii.)

The clauses corresponding to sequents all of whose parents are axioms (or come from axioms by $\neg$-introduction or

$\neg$-elimination) appear already in $def(D)$, or are subsumed by clauses in $def(D)$. (*e.g.* clauses 3'. and 6'. in fig. i, and 6'. and 8'. in fig.ii). To see why this is so, assume, for instance, that sequent $\Delta, C \supset B \vdash \Gamma$ is not an axiom, but is derived by $\supset$-elimination from the axioms $\Delta \vdash C, \Gamma$ and $\Delta, B \vdash \Gamma$. Since these last two sequents are axioms, it must be true that $C \in \Delta$ and $B \in \Gamma$. Therefore, the clause associated with $\Delta, C \supset B \vdash \Gamma$ must contain $\overline{\xi^C} \overline{\xi^{C \supset B}} \xi^B$, which is one of the three clauses included in $def(D)$ as $cl(C \supset B)$: $\overline{\xi^{C \supset B}} \overline{\xi^C} \xi^B$, $\xi^{C \supset B} \xi^C$, $\xi^{C \supset B} \overline{\xi^B}$. The same type of argument applies to all basic Gentzen rules for binary connectives. If a sequent is derived by the cut rule from two axioms, then that sequent itself is an axiom, so the cut rule need never have been applied.

Finally, since the clauses at the leaves of this derivation may subsume some of the clauses corresponding to sequents with axioms for parents, the derivation may need to be pruned (by removing extraneous literals from some clauses) to make it a proper resolution derivation. The result is a resolution derivation D' of ξ^A from $def(D)$, whose length is proportional to the length of D. If D included no applications of the cut rule, then $def(D) = def(A)$, so adding one resolution to D' gives a resolution derivation of $\square$ from $def(A) \cup \{\overline{\xi^A}\}$. This establishes the simulation of basic Gentzen systems by resolution with limited extension. On the other hand, if D included applications of the cut rule, it is still true that $def(A) \subseteq def(D)$, and that $def(D)$ can be obtained from $def(A)$ by repeated applications of the extension rule. Therefore D'

along with those extensions gives an extended resolution derivation of ξ^A from $def(A)$, establishing the simulation of Gentzen systems with cut by extension.

5.1.2. A Formalization of Simulation

The preceding two examples are typical of the results of this thesis. They each show that if shortest proof lengths in one system are bounded by a polynomial, then shortest proof lengths in another system are also bounded by a polynomial. The method of proof is a sort of "simulation", in which provable objects (tautologies, unsatisfiable sets of clauses, valid sequents, *etc.*) from the second system are translated into not-too-much longer provable objects in the first system, and then proofs in the first system are translated into the second system, again without increasing the length by more than a polynomial. These ideas are formalized in the following definition of "simulates", which applies to arbitrary verification systems, as defined in subsection 3.3.1.

Let F_1 be a verification system for L_1 and let F_2 be a verification system for L_2. Thus, $F_1:\{0,1\}^* \xrightarrow{\text{onto}} L_1$, $F_2:\{0,1\}^* \xrightarrow{\text{onto}} L_2$, and $F_1, F_2 \in \mathcal{PF}$. System F_2 *simulates* F_1 if there exist functions $g:L_2 \to L_1$ and $h:\{0,1\}^* \times L_2 \to \{0,1\}^*$, and polynomials p,q such that $\forall x \in \{0,1\}^*$ and $\forall y \in L_2$, $\ell g(y) \leq p(\ell y)$ $\ell h(x,y) \leq q(\ell x, \ell y)$, and $F_2(h(x,y)) = y$ whenever $F_1(x) = g(y)$. If $g, h \in \mathcal{PF}$ (which is the case for all simulations in this thesis), the stronger notion *p-simulates* will be used. (Note: g needs only to be computable in polynomial time for inputs in L_2.)

?? This seems wrong

Intuitively, g is the function that translates formulas from L_2 to L_1, and h translates derivations. (If $L_2 \subseteq L_1$ and g is the identity function, then the simulation is said to be *direct*.) The definition says that F_2 simulates F_1 if there is a way of translating strings (formulas) and a way of translating derivations (proofs) such that by first translating a string, then finding a derivation of that string in system F_1, and finally translating that derivation, one obtains a derivation of the original string in system F_2

If F_2 is polynomial-bounded, then F_2 trivially simulates any other F_1. The function $h(x,y)$ simply ignores x and gives a shortest derivation of y in the system F_2. The function g is unimportant, as long as it is polynomial-length-bounded. Thus, if $L_2 \subseteq L_1$, F_2 directly simulates F_1 in the same trivial way (with g the identity function). Therefore, a corollary of the fact that F_2 cannot (directly) simulate F_1 is the fact that F_2 cannot be polynomial-bounded.

If F_2 cannot directly simulate F_1, it may still be true that F_2 simulates F_1, however, if F_1 is also not polynomial-bounded. For this simulation, the function g translates a string y into another string y', of approximately the same length as y, such that the shortest derivation of y' in system F_1 is "very long". Then the computation of $h(x,y)$ returns a shortest derivation of y in system F_2, provided that x is "long enough" so that the length of this derivation is no greater than $q(\ell x, \ell y)$, for some polynomial q. When the "simulates" definition is applied to g and h, the string x will

always be "long enough", because x must be a derivation of y' in system F_1 .

This simulation involves a very unnatural use of the translation function g, and it only applies in the uninteresting case where F_1 and F_2 are both known not to be polynomial-bounded. For this reason, the strongest type of negative result that can reasonably be expected is of the form: there is no simulation of F_1 by F_2 in which g is of a certain restricted type. In fact, the negative results in this thesis all restrict g to be the identity function.

The following lemma establishes the most important property of simulations between verification systems.

LEMMA 5.1.2.a.

If F_1 and F_2 are verification systems such that F_2 simulates F_1, then F_2 is polynomial-bounded if F_1 is polynomial-bounded.

Proof

Suppose F_1 is a verification system for L_1, F_2 is a verification system for L_2, and there exist g, h, p, q as in the definition of "simulates". Also, assume r is a polynomial upper bound on lengths of shortest derivations in system F_1.

Define $F':L_1 \rightarrow \{0,1\}^*$ so that F' is the first (in lexicographic order by length) derivation of y in system F_1. Then, $\forall y \in L_1$, $\ell F'(y) \leq r(\ell y)$ and $F_1(F'(y))=y$.

Given any $x \in L_2$, let $d(x)=h(F'(g(x)),x)$. Then, $\ell d(x)=\ell h(F'(g(x)),x) \leq q(\ell F'(g(x)),\ell x) \leq q(r(\ell g(x)),\ell x) \leq q(r(p(\ell x)),\ell x$ Also, $F_2(d(x))=F_2(h(F'(g(x)),x))=x$ because $F_1(F'(g(x)))=g(x)$. Thus, $d(x)$ is a derivation of x in system F_2, and the leng·h

of $d(x)$ is bounded above by a polynomial in the length of x. Therefore, F_2 is polynomial-bounded. ⊠5.1.2.a.

An immediate consequence of this lemma is that if F_2 simulates F_1, then to prove that F_1 is not polynomial-bounded, it suffices to prove that F_2 is not polynomial-bounded.

Another important property of (p-)simulation is that the relation is transitive.

LEMMA 5.1.2.b.

If F_2 (p-)simulates F_1 and F_3 (p-)simulates F_2, then F_3 (p-)simulates F_1.

Proof

Let g_1, h_1, p_1, q_1 be from the first simulation, and let g_2, h_2, p_2, q_2 be from the second. Define $g_3(x)=g_1(g_2(x))$, and $h_3(y,x)=h_2(h_1(y,g_2(x)),x)$, so that $\ell g_3(x)\leq p_1(p_2(\ell x))=p_3(\ell x)$, and $\ell h_3(y,x)\leq q_2(q_1(\ell y,p_2(\ell x)),\ell x)=q_3(\ell y,\ell x)$. Now suppose $F_1(y)=g_3(x)=g_1(g_2(x))$. Then, by simulation of F_1 by F_2, $F_2(h_1(y,g_2(x)))=g_2(x)$, and by the simulation of F_2 by F_3, $F_3(h_2(h_1(y,g_2(x)),x))=x$. That is, $F_3(h_3(y,x))=x$ whenever $F_1(y)=g_3(x)$. This establishes that $\langle g_3,h_3\rangle$ gives a simulation of F_1 by F_3. If $g_1,g_2,h_1,h_2\in\mathcal{PF}$, then $g_3,h_3\in\mathcal{PF}$, so that F_3 (p-)simulates F_1. ⊠5.1.2.b.

5.1.3. Notation and Terminology for Simulations

In order to facilitate the description of various simulations of proof systems for the propositional calculus, it is convenient to introduce several different measures of "size" for formulas and proofs.

The *length* of a formula A (denoted ℓA) is the length of some standard encoding of A over the alphabet {0,1}. Atoms and connectives are distinguished by binary labels. Note that for any standard encoding there is a constant c such that for any formula A containing m occurrences of atoms and n occurrences of connectives of arity no greater than k, $\ell A \leq c \cdot (m \cdot log(m) + n \cdot 2^k)$. (Remember that there are 2^{2^k} possible connectives of arity k.) The length of a set or sequence of formulas (such as a derivation) is the sum of the lengths of the constituent formulas.

The number of occurrences of atoms in formula A is denoted by $\ell^a A$, and the number of occurrences of connectives in A is denoted by $\ell^c A$. Both ℓ^a and ℓ^c can be applied to sets and sequences of formulas. If $\sigma = \frac{A_1,\ldots,A_n}{p_1,\ldots,p_n}$ is a substitution, then $\ell\sigma = \sum_{i=1}^{n} \ell A_i$, $\ell^a\sigma = \sum_{i=1}^{n} \ell^a A_i$, and $\ell^c\sigma = \sum_{i=1}^{n} \ell^c A_i$.

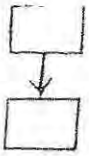
If D is a derivation in some proof system, then the *number of steps* of D (denoted $\ell^s D$) is the number of occurrences of formulas, sequents, clauses, *etc.* (as appropriate) in D. If A is a formula, sequent, clause, *etc.* in D or a hypothesis of D, then the *degree* of A in D (denoted $d(A)$) is the number of immediate descendants of A in D. The degree of a derivation is the maximum of the degrees of its constituent formulas, sequents, clauses, *etc.* (including hypotheses). If D has degree 1, then D is a *tree* derivation.

5.2. SIMULATION AND LOWER BOUND RESULTS

The simulation and lower bound results reported in this chapter are summarized in figure 5.2.i. Each solid box in the figure contains a single proof system or a collection of proof systems such that each system in the collection p-simulates every other system in the collection. Each dotted box contains a family of proof systems such that each adequate set of connectives gives rise to a different system. It is not known if all of the systems in a dotted box can simulate one another. The figure is arranged with the most powerful systems (*i.e.* those with potentially the shortest proofs) at the top, and weaker systems near the bottom.

The figure shows several systems that were not described in chapter 4. Extended Frege systems (box 2.) are defined in subsection 5.3.2. by combining ideas from Frege systems and extended resolution. Regular versions of a number of systems (boxes 2., 7., 9., 11., and 17.) are defined by analogy with regular resolution. Regular Gentzen systems (Boxes 9. and 11.) are discussed in paragraph 5.6.4.2. Extended tree resolution (box 3.) is merely extended resolution where the derivations must be trees. The final new system is Galil's system of enumeration dags (box 13.), which is described in paragraph 5.6.1.4.

A downward arrow connecting solid boxes in the figure

a. 

b.

indicates that every system in box a. p-simulates

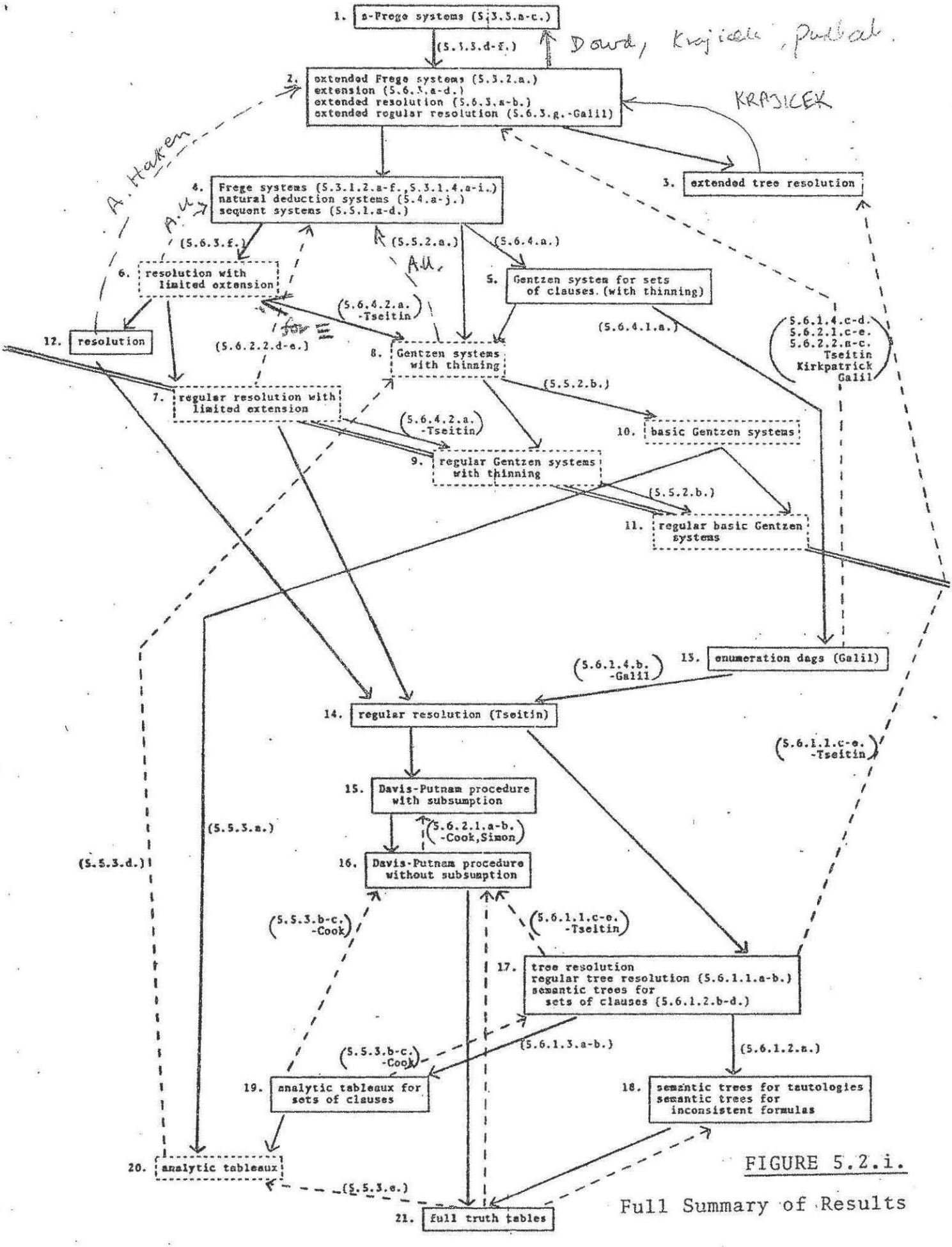
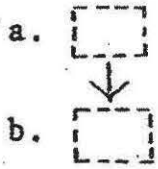
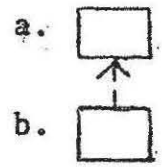


FIGURE 5.2.i.

Full Summary of Results

every system in box b. A downward arrow between two dotted boxes  indicates that every system in box a. p-simulates the system in box b. for the same set of connectives. The arrow in the figure from box 5. to box 8. indicates that the Gentzen system with thinning for sets of clauses p-simulates the Gentzen system with thinning for formulas in the connectives $\{\neg,\vee,\&\}$. The simulation relationship inside of boxes is always the same. Within solid boxes all systems p-simulate each other, while inside each dotted box, the only known simulations are the trivial ones: if $\kappa_1 \subseteq \kappa_2$, then the system for the connectives κ_1 p-simulates the system for the connectives κ_2.

Upward arrows in the figure indicate negative results about simulations. An upward dashed arrow  indicates that no system in box b. can directly simulate any system in box . The figure contains two upward arrows that need special explanation. The arrow from box 20. to box 8. indicates that for each adequate set of connectives κ, the system of analytic tableaux for κ cannot directly simulate the Gentzen system with thinning for κ. The upward arrow from box 7. to box 4. indicates that for certain sets of connectives κ (see paragraph 5.6.2.2. for details), the system of regular resolution with limited extension for formulas in the connectives κ cannot directly simulate any system in the equivalence class with Frege systems.

Further simulation relations can be deduced from figure 5.2.i. by using the fact that (p-)simulation is a transitive relation. For example, s-Frege systems p-simulate every other system shown, because there is a chain of downward arrows connecting box 1. to each of the other boxes in the figure. Any upward chain made by following dashed arrows in the forward direction and solid arrows in the backward direction and including at least one dashed arrow indicates that it is unlikely that the lower system simulates the upper one. For example, it is unlikely that any system of analytic tableaux (box 20.) can simulate resolution (box 12.), because if one could, then analytic tableaux for sets of clauses (box 19.) could simulate the Davis-Putnam procedure without subsumption (by a chain of simulation arrows through boxes 20., 12., 14., and 15.). But by theorems 5.5.3.b-c. this could not be a direct simulation, and thus it would be in a sense "unnatural". If there is neither a downward chain nor an upward chain containing at least one dashed arrow from system *A* to system *B*, then it is not known whether system *A* can simulate system *B*. Examples of this sort abound in figure 5.2.i. It is not known if regular resolution simulates resolution, nor if resolution simulates any Frege system. It is not even known if regular resolution can be simulated by the (seemingly much more primitive) Davis-Putnam procedure with subsumption, nor if the Davis-Putnam procedure (with or without subsumption) can simulate any of the systems of analytic tableaux.

The double solid line separates systems that are provably not polynomial-bounded (below the line) from systems that have not been shown not to be polynomial-bounded (above the line). The line goes through boxes 7., 9., and 11., indicating that some of these systems have been proven not to be polynomial-bounded, while others have not. The dividing line is drawn by theorem 5.6.2.2.d., which says that one of these systems is not polynomial-bounded if its set of connectives contains any two of $\{\neg, \equiv, \not\equiv\}$.

Proof systems and arrows in the figure are labelled with the number of the result where the indicated fact is proved. For example, the fact that any two Frege systems p-simulate each other (box 4.) is proved in subsection 5.3.1. The theorems leading to this result are lemmas and theorems 5.3.1.2.a. through 5.3.1.2.f. and 5.3.1.4.a. through 5.3.1.4.i. Where a result was known before, the originator of the result is credited. For example, the p-simulation of regular resolution (box 14.) by enumeration dags (box 13.) was first shown by Galil, and the result is reproduced here as theorem 5.6.1.4.b. When an arrow is unlabelled, the indicated result is trivial. For example, the systems in box 2. p-simulate the systems in box 4. because Frege systems are a special case of extended Frege systems. Similarly, the general system of any type p-simulates the regular system of the same type, because regular derivations are just a special form of general derivations.

The most interesting new results are those relating to boxes 1., 2., and 4. in figure 5.2.i. In subsection 5.3.1. it is proved that any two Frege systems p-simulate one another, no matter what connectives they use or what rules they have. Sections 5.4. and 5.5. show that natural deduction systems and sequent systems, both seemingly more powerful than Frege systems, are actually p-simulation equivalent to Frege systems and to each other. The idea of extended Frege systems is introduced in subsection 5.3.2., where it is shown that these systems p-simulate one another and all systems p-simulation equivalent to Frege systems. Then, in subsection 5.6.3. it is shown that extended Frege systems are p-simulation equivalent to the systems of extension and extended resolution. Subsection 5.3.3. discusses s-Frege systems, showing that these systems are at least as powerful (to within a polynomial) as any other system in figure 5.2.i.

The only new lower bound result is embodied in theorems 5.6.2.2.d. and 5.6.2.2.e., where Tseitin's lower bound for regular resolution is extended to certain systems of regular resolution with limited extension. Theorem 5.6.2.2.e. is particularly interesting, because it shows that Tseitin's techniques cannot be used for Frege systems and their allies.

Most of the other results shown in figure 5.2.i. are either already known or are more-or-less obvious. Theorem 5.6.4.1.a. is interesting, because it shows that, while resolution with limited extension is no less powerful than conventional cut-free Gentzen systems (theorem 5.6.4.2.a.),

a slight generalization of these systems (the Gentzen system for sets of clauses) is no less powerful than enumeration dags and regular resolution. This means that the two main families of systems in the neighbourhood of the double solid line, cut-free Gentzen systems and extension-free resolution-based systems, are interrelated, and probably have similar bounds on proof lengths.

5.3. FREGE SYSTEM SIMULATIONS

In this section various simulations between Frege systems are established. The most important results are in subsection 5.3.1., where it is shown that any two Frege systems can simulate each other. Subsection 5.3.2. introduces the idea of extended Frege systems, and shows how they simulate ordinary Frege systems and each other. The final subsection of this section is 5.3.3., where it is shown that s-Frege systems simulate extended Frege systems and each other.

5.3.1. Frege Systems without the Substitution Rule

Recall from subsection 4.2.1. that a Frege system is an implicationally complete system $F=\langle \kappa, \mathcal{R} \rangle$, where κ is an adequate set of connectives and $\mathcal{R}$ is a finite set of sound rules of inference in the connectives κ.

The proof that any two Frege systems simulate each other will proceed in a series of easy stages. Paragraph 5.3.1.1. introduces the notion of "direct translation", and discusses some of the useful properties of direct translations. Paragraph 5.3.1.2. proves in a series of stages that if there

are direct translations between the connectives of two Frege systems, then those systems simulate each other. Indirect translations and their properties are discussed in paragraph 5.3.1.3. Finally in paragraph 5.3.1.4. it is shown that two arbitrary Frege systems simulate each other.

5.3.1.1. Direct Translation

In order to compare Frege systems that operate on formulas with different sets of connectives, a way must be found to relate formulas with different connectives. At least three such methods are employed in this thesis, the first of which is called *direct translation*.

Let κ_1 and κ_2 be sets of connectives, and for i=1,2 let K_i be the set of formulas in the connectives $\kappa_i=\{*_i^1,\ldots,*_i^{k_i}\}$. For $1\le j\le k_i$, let $A_i^j=*_i^j(p_1,\ldots,p_n)$, where $n=n^{*_i^j}$. The formulas $\{A_i^j \mid 1\le j\le k_i\}$ are called the *primitive formulas* of κ_i. If $t:K_1\to K_2$ and $\sigma = \frac{B_1,\ldots,B_n}{q_1,\ldots,q_n}$ is a substitution in κ_1, then $t(\sigma) = \frac{t(B_1)\ldots t(B_n)}{q_1\cdots q_n}$ is a substitution in κ_2. The function $t:K_1\to K_2$ is called a *direct translation* from κ_1 to κ_2 if there exists a polynomial $p(n)$ such that for all formulas $B\in K_1$ and for all substitutions σ in κ_1,

1) $t(B)\sim B$,

2) $\ell t(B)\le p(\ell B)$, and

3) $t(B\sigma)=t(B)t(\sigma)$, provided that σ does not change the "distinguished" atom p_0. (The use of p_0 will become clear later.)

As an example, the first translations between $\{\neg,\vee\}$ and $\{\neg,\supset\}$ in paragraph 5.1.1.1. (not the ones that were *onto*) are examples of direct translations.

Property 3 places several restrictions on a direct translation t. The first of these is that for any atom p, $t(p)=p$. This is true because $p=p\frac{p}{p}$, so that by property 3, $t(p)=t(p\frac{p}{p})=t(p)\frac{t(p)}{p}$, which can only hold if either $t(p)=p$ or p does not occur in $t(p)$. Property 1 insures that this second alternative cannot occur. The second consequence of property 3 is that t is completely determined by $\{t(A_1^j)\,|\,1\leq j\leq k_1\}$. That is, if $t(A_1^j)=t'(A_1^j)$ for $1\leq j\leq k_1$, then for every $B\epsilon K_1$, $t(B)=t'(B)$. This fact is easily proved from property 3 by induction on the number of occurrences of connectives in B. Properties 2 and 3 also imply that for $1\leq j\leq k_1$ and for $1\leq i\leq n^{*A_1^j}$, p_i occurs at most once in $t(A_1^j)$. To see that this is so, let $\sigma = \frac{A_1^j}{p_i}$, and let $B = (\ldots(A_1^j\overbrace{\sigma)\ldots)\sigma}^{n \text{ times}}$. Then $\ell B = (n+1)\ell A_1^j - n\ell p_i$, since A_1^j contains one occurrence of p_i. By property 3, $t(B)=(\ldots(t(A_1^j)\overbrace{\sigma')\ldots)\sigma'}^{n \text{ times}}$, where $\sigma' = \frac{t(A_1^j)}{p_i}$. If p_i occurs r times in $t(A_1^j)$, then p_i

occurs r^{n+1} times in $t(B)$, so that $\ell t(B) \geq r^{n+1}$. But by property 2, $\ell t(B) \leq p(\ell B) \leq p((n+1)\ell A_1^j - n\ell p_i) \leq p((n+1)\cdot c)$, where $c = \ell A_1^j$. This gives $r^{n+1} \leq p((n+1)\cdot c)$, which must be true for fixed p, r, and c, and for arbitrarily large n. This can only be true if $r \leq 1$.

This last consequence of property 3 has two further consequences. The first is that if p is any atom in B, then the number of occurrences of p in $t(B)$ is no greater than the number of occurrences of p in B. This is proven by induction on the number of occurrences of connectives in B. The second is that if $c = \max_{1 \leq j \leq k_1} \ell t(A_1^j)$, then for any formula $B \in K_1$, $\ell t(B) \leq c \cdot \ell B$. This is also proven by induction on the substructure of B. Therefore, property 2 in the definition of direct translation could be strengthened without any loss of generality to: 2') there is a constant c such that $\forall B \in K_1$, $\ell t(B) \leq c \cdot \ell B$.

The composition of two direct translations is a direct translation. That is, if t_1 is a direct translation from κ_1 to κ_2, and t_2 is a direct tranlsation from κ_2 to κ_3, then t_3 defined by $t_3(B) = t_2(t_1(B))$ is a direct translation from κ_1 to κ_3. This is true because $t_3(B) = t_2(t_1(B)) \sim t_1(B) \sim B$, $\ell t_3(B) = \ell t_2(t_1(B)) \leq c_2 \cdot \ell t_1(B) \leq c_2 \cdot c_1 \cdot \ell B$, and $t_3(B\sigma) = t_2(t_1(B\sigma)) = t_2(t_1(B)t_1(\sigma)) = t_2(t_1(B))t_2(t_1(\sigma)) = t_3(B)t_3(\sigma)$.

There are several other properties of direct translations that will be used later. First is the obvious fact that if t is a direct translation from κ_1 to κ_2, and if $\kappa_1' \subseteq \kappa_1$, then the restriction of t to formulas in κ_1' is a direct translation from κ_1' to κ_2. Also, if t_1 is a direct translation from κ_1 to κ_3 and t_2 is a direct translation from κ_2 to κ_3, then t_1 and t_2 can be combined to give a direct translation from $\kappa_1 \cup \kappa_2$ to κ_3. Consequently, if there is a direct translation from κ_1 to κ_2, then there are direct translations both ways between $\kappa_1 \cup \kappa_2$ and κ_2.

Direct translations are very useful when they exist, but there is not always a direct translation from any one given set of connectives to any other. For example, there is no direct translation from $\{\neg, \vee, \equiv\}$ to $\{\neg, \vee\}$. The following development establishes one set of circumstances that guarantee the existence of a direct translation.

If κ is any adequate set of connectives (with no restriction on the arities of the connectives), then there exists a direct translation from $\kappa_0 = \{\mathsf{T}, \mathsf{F}, \neg, \vee, \subset, \supset, \&, |, \not\supset, \not\subset, \downarrow\}$ to κ. A careful and complete proof of this fact is rather tedious, so only a sketch will be given here. Since κ is adequate, there are a tautology T and an unsatisfiable formula F that can be expressed in terms of the connectives κ. Furthermore, T and F can be chosen so that the only atom in them is the "distinguished" atom p_0. These will serve as the direct translations of T and F respectively, and will also be used when needed to give constant truth values.

A truth function (or a connective) f is called *monotone* if $\tau_1 \leq \tau_2$ implies $f(\tau_1) \leq f(\tau_2)$, where $T \leq T$, $F \leq F$, $F \leq T$, and $\tau_1 \leq \tau_2$ if and only if $\forall j\ \tau_1(j) \leq \tau_2(j)$ ($\tau_1(j)$ is the j^{th} component of the n-tuple of truth values τ_1). It is easy to show that the composition of monotone functions is monotone, and that there exist truth functions that are not monotone. Then, since κ is adequate, κ must contain a non-monotone connective. From this connective and the formulas T and F, a direct translation for $\neg p$ can be constructed.

A truth function (or a connective) f is called *even* if f can be represented by a formula in the connectives $\{T, F, \neg, \equiv, \not\equiv\}$, and it is *odd* if it is not even. It is easy to show by induction on the representing formula that the number of T entries in the truth table for an even function is even. Note that the eight odd binary connectives are all in κ_0. Again, it can be shown that the composition of even functions is even, and that there are truth functions that are not even, so that κ must contain an odd connective $*$. It can then be shown that $*$ along with T and F can be used to give a direct translation of one of the eight odd binary connectives. Since each of the eight odd binary connectives can be represented in terms of each of the others plus $\neg$, direct translations for the other seven odd binary connectives can also be obtained.

Although the existence of direct translations from κ_0 to any adequate set of connectives κ is interesting for its own sake, there are several important consequences of this fact as well. The first of these is that if t is a direct translatio from κ_1 to κ_2 and if κ_2 is adequate, then there is a direct translation t' from κ_1 to κ_2 such that $\forall B \epsilon K_1$, $\forall p_i$ occurring in

B (except p_0), the number of occurrences of p_i in $t'(B)$ equals the number of occurrences of p_i in B, and $\ell t'(B) \geq \ell B$. If κ_1 contains some connective $*$ that does not depend on all of its arguments, then $t(*(p_1,\ldots,p_{n^*}))$ may not contain all of $\{p_1,\ldots,p_{n^*}\}$. The translation t' puts back those dropped atoms without changing the truth function represented. This is done by using translations of formulas in κ_0. For example, if $t(*(p_1,\ldots,p_{n^*}))$ contains all of the atoms of $\{p_1,\ldots,p_{n^*}\}$ except p_i, then $t'(*(p_1,\ldots,p_{n^*}))$ could be the direct translation of $(t(*(p_1,\ldots,p_{n^*}))\&(p_i \vee T))$. In this way it can be insured that no occurrences of atoms are lost and that lengths of formulas are never decreased by direct translations.

The final property of direct translations is that if κ_2 is adequate, and t is a direct translation from κ_1 to κ_2, then there is a direct translation t' from κ_1 to κ_2 that is one-to-one. The trick here is to use translations of formulas in κ_0 to attach a unique "tag" to the translation of each primitive formula of κ_1, so that the translation may be uniquely reversed. Suppose $T_1,\ldots,T_{k_1}$ are distinct tautologies in the connectives κ_2 built up using the distinguished atom p_0. Then for each $*_1^i \in \kappa_1$, let $t'(*_1^i(\vec{p}))$ be the direct translation (from $\kappa_0 \cup \kappa_2$ to κ_2) of $(t(*_1^i(\vec{p}))\&T_i)$. Since these "tags" are distinct, it can now be proven by induction on the substructure

of B_1 and B_2, that if $t'(B_1)=t'(B_2)$ then $B_1=B_2$. Note that this construction does not conflict with the previous one, so that if κ_2 is adequate and t is a direct translation from κ_1 to κ_2, it may be assumed without loss of generality that there is a constant c such that all of the following hold:

1) $t(B)\sim B$

2) $\ell B \leq \ell t(B) \leq c \cdot \ell B$

3) $t(B\sigma)=t(B)t(\sigma)$, provided σ does not substitute for p_0, and

4) $B' \neq B \Rightarrow t(B') \neq t(B)$.

5.3.1.2. Frege Systems and Direct Translations

In order for one Frege system to simulate another, a way must be found to extend the ideas of translation of formulas from the previous paragraph to the translation of inferences. The example in paragraph 5.1.1.1. showed how this can be done in one particular example, and this paragraph shows how it is done in general.

The method used here is a bit different from the method that was used to show simulations between Mendelson's and Shoenfield's systems. This method applies to any pair of Frege systems $F_1=\langle \kappa_1, \mathcal{R}_1 \rangle$ and $F_2=\langle \kappa_2, \mathcal{R}_2 \rangle$ for which there are direct translations t_1 from κ_1 to κ_2 and t_2 from κ_2 to κ_1. To show that F_2 simulates F_1, tautologies in κ_2 must be translated int κ_1 (*via* t_2), and derivations in system F_1 must be translated into derivations in system F_2 (*via* some function h) in such a way that for any tautology A in the connectives κ_2, and for any derivation D of $t_2(A)$ in system F_1, $h(D,A)$ is a

derivation of A in system F_2. The derivation $h(D,A)$ is constructed in two parts. The first is a translation of derivation D that gives a derivation of $t_1(t_2(A))$, and the second is a derivation of A from $t_1(t_2(A))$. The construction of the first part is described in lemma 5.3.1.2.a., and the second part is in lemmas 5.3.1.2.b.-d. These lemmas are tied together in theorem 5.3.1.2.e., the main result of this paragraph.

LEMMA 5.3.1.2.a.

If $I_1=<\kappa_1,\mathcal{R}_1>$ is an inference system, $F_2=<\kappa_2,\mathcal{R}_2>$ is a Frege system, and t is a direct translation from κ_1 to κ_2, then there are constants a_1 and a_2 such that whenever $\Gamma\vdash_{I_1}A$ *via* D, there is a derivation D' such that:

1) $t(\Gamma)\vdash_{F_2} t(A)$ *via* D',

2) $\ell^s D' \le a_1\cdot\ell^s D$, and

3) $\ell D' \le a_2\cdot(\ell D+\ell\Gamma)\cdot d(D)$.

Proof

Let $\mathcal{R}_1 = \{R_1=\Delta_1\rightarrow B_1,\ldots,R_k=\Delta_k\rightarrow B_k\}$. Since the rules of $\mathcal{R}_1$ are sound, $\Delta_i\models B_i$, and since $t(B)\sim B$, $t(\Delta_i)\models t(B_i)$, for $1\le i\le k$. Then, since F_2 is implicationally complete, there are derivations D^i such that $t(\Delta_i)\vdash_{F_2} t(B_i)$ *via* D^i, for $1\le i\le k$.

Let $a_1 = \max\limits_{1\le i\le k}\ell^s D^i$, $a_3 = \max\limits_{1\le i\le k}\ell D^i$, and $a_4 = \max\limits_{1\le i\le k}\ell^a D^i$.

Also, let a_5 be a constant such that $\ell t(B)\le a_5\cdot\ell B$. The required derivation D' is constructed from D by replacing each formula C in D by a derivation D_C, where $t(\Delta^C)\vdash_{F_2} t(C)$ *via* D_C, and Δ^C

is the set of formulas from which C is inferred in D. In particular, suppose that C is inferred from Δ^C by substitution σ^C in rule R_{i_C}. Then, $\Delta^C=\Delta_{i_C}\sigma^C$ and $C=B_{i_C}\sigma^C$, and $D_C=D^{i_C}t(\sigma^C)$ is a derivation of $t(B_{i_C})t(\sigma^C)=t(B_{i_C}\sigma^C)=t(C)$ from $t(\Delta_{i_C})t(\sigma^C)=t(\Delta_{i_C}\sigma^C)=t(\Delta^C)$. If D' is constructed in this way, then $t(\Gamma)\vdash_{F_2}t(A)$ *via* D'. Also, $\ell^S D'=\sum_{C\in D}\ell^S D_C$, and $\ell D'=\sum_{C\in D}\ell D_C$. But $\ell^S D_C=\ell^S D^{i_C}\leq a_1$, so that $\ell^S D'\leq\sum_{C\in D}a_1=a_1\cdot\ell^S D$. Since σ^C need not specify a substitution for any atom not in $\Delta_{i_C}\cup\{B_{i_C}\}$, $\ell\sigma^C\leq\ell\Delta^C+\ell C$. Then, $\ell D_C=\ell D^{i_C}t(\sigma^C)\leq\ell D^{i_C}+\ell^{a_D}{}^{i_C}\cdot\ell t(\sigma^C)\leq a_3+a_4\cdot a_5\cdot(\ell\Delta^C+\ell C)\leq\frac{a_2}{2}\cdot(\ell\Delta^C+\ell C)$, where $\frac{a_2}{2}=a_3+a_4\cdot a_5$. This gives $\ell D'=\sum_{C\in D}\ell D_C\leq\sum_{C\in D}\frac{a_2}{2}\cdot(\ell\Delta^C+\ell C)=\frac{a_2}{2}\cdot\sum_{C\in D\cup\Gamma}k^C\ell C$, where k^C is one greater than the number of formulas in D of which C is an immediate ancestor. Then, since $k^C\leq d(D)+1$,

$$\ell D'\leq\frac{a_2}{2}\cdot\sum_{C\in D\cup\Gamma}(d(D)+1)\cdot\ell C=\frac{a_2}{2}\cdot(d(D)+1)\cdot\sum_{C\in D\cup\Gamma}\ell C=$$

$$\frac{a_2}{2}\cdot(d(D)+1)\cdot(\ell D+\ell\Gamma)\leq a_2\cdot d(D)\cdot(\ell D+\ell\Gamma).\qquad\blacksquare 5.3.1.2.a.$$

As was pointed out in the example of paragraph 5.1.1.1., a result like lemma 5.3.1.2.a. is not enough to insure that system F_2 can simulate system I_1, because t need not be onto.

In that example, it was possible to make t onto, but in general this may not be possible. The following three lemmas give an alternative way of solving this problem. The method used is to "undo" the translation t by means of another translation and further inferences in system F_2. These further inferences are described in lemmas 5.3.1.2.b. and 5.3.1.2.c., and their use in conjunction with a translation from κ_2 to κ_1 is described in lemma 5.3.1.2.d.

LEMMA 5.3.1.2.b.

If κ is adequate and t' is a direct translation from κ to κ, then there are an inference system $I<\kappa,\mathcal{R}>$ and constants b_1 and b_2 such that for every formula A in the connectives κ there are derivations D_1 and D_2 such that:

1) $A \vdash_I t'(A)$ *via* D_1,

2) $t'(A) \vdash_I A$ *via* D_2, and for $i=1,2$,

3) $\ell^s D_i \leq b_1 \cdot \ell^c A$,

4) $\ell D_i \leq b_2 \cdot \ell^c A \cdot \ell A$, and

5) $d(D_i)=1$.

Proof

Since κ is adequate, there is a formula $\hat{E}(p,q)$ in the connectives κ such that $\hat{E}(p,q) \sim (p \equiv q)$. Let $b_3=\ell\hat{E}(p,q)$ $b_4=\ell^a\hat{E}(p,q)$, and let b_5 be the constant such that $\ell t'(A) \leq b_5 \cdot \ell A$. For each connective $*\epsilon\kappa$, let $A^*=*(p_1,\ldots,p_{n^*})$ (*e.g.* $A^\vee=(p_1 \vee p_2)$, $A^\equiv=(p_1 \equiv p_2)$, $A^\neg=\neg p_1$, *etc.*), and let $B^*=t'(A^*)$. $\mathcal{R}$ will contain the rules

1) $\rightarrow \hat{E}(p,p)$,

2) $\{\hat{E}(p,q),p\}\rightarrow q$,

3) $\{\hat{E}(p,q),q\}\rightarrow p$, and

4) for each connective $*$:
$\{\hat{E}(p_1,q_1),\ldots,\hat{E}(p_{n^*},q_{n^*})\}\rightarrow \hat{E}(A^*,B^*\frac{q_1\cdots q_{n^*}}{p_1\cdots p_{n^*}})$.

Since $\hat{E}$ represents equivalence and $t'(A)\sim A$, these rules are clearly sound, so that $I=<\kappa,\mathcal{R}>$ is an inference system. For any formula A in the connectives κ, let D^A be a derivation of $\hat{E}(A,t'(A))$ in the system I, defined inductively as follows: 1) if A is an atom p, then D^A is $\hat{E}(p,p)$, by rule 1, and 2) if A is $*(B_1,\ldots,B_{n^*})$, then D^A is $D^{B_1}\ldots D^{B_{n^*}}\hat{E}(A,t'(A))$, the last formula of which follows according to rule 4* under the substitution $\sigma = \frac{B_1\ldots B_{n^*}\, t'(B_1)\ldots t'(B_{n^*})}{p_1\cdots p_{n^*}\quad q_1\cdots q_{n^*}}$. Note that $\ell^s D^A=\ell^a A+\ell^c A$, $d(D^A)=1$, and for each formula $B=\hat{E}(C,t'(C))\in D^A$, $\ell B \le \ell\hat{E}(p,q)+\ell^a\hat{E}(p,q)\cdot(\ell C+\ell t'(C)) \le b_3+b_4(\ell A+b_5\ell A) \le b_6\ell A$, where $b_6 = b_3+b_4(1+b_5)$. Let k be the maximum arity of any connective in κ, so that $\ell^a A\le k\ell^c A$. This implies that $\ell^s D^A\le(k+1)\ell^c A$. Finally, let D_1 be $D^A t'(A)$ (which follows by rule 2) and let D_2 be $D^A A$ (which follows by rule 3). Then,

$A \vdash_I t'(A)$ *via* D_1, $t'(A) \vdash_I A$ *via* D_2, and for $i=1,2$,

$\ell^s D_i \le 1+\ell^s D^A \le 1+(k+1)\cdot\ell^c A \le b_1\cdot\ell^c A$, where $b_1=k+2$,

$\ell D_i \leq \ell^s D_i \cdot$(length of longest formula in D_i) $\leq b_1 \cdot \ell^c A \cdot b_6 \cdot \ell A$

$= b_2 \cdot \ell^c A \cdot \ell A$, where $b_2 = b_1 \cdot b_6$, and $d(D_i) = 1$. ⊠5.3.1.2.b.

LEMMA 5.3.1.2.c.

If $F = \langle \kappa, \mathcal{R} \rangle$ is a Frege system, and t' is a direct translation from κ to κ, then there are constants c_1 and c_2 such that for every formula A in the connectives κ there are derivations D_1' and D_2' such that:

1) $A \vdash_F t'(A)$ *via* D_1'

2) $t'(A) \vdash_F A$ *via* D_2', and for $i=1,2$

3) $\ell^s D_i' \leq c_1 \cdot \ell^c A$, and

4) $\ell D_i' \leq c_2 \cdot \ell^c A \cdot \ell A$.

Proof

Since κ is adequate, lemma 5.3.1.2.b. applies, so let I, b_1, b_2, A, D_1, and D_2 be as stated in that lemma. By lemma 5.3.1.2.a., with $I_1 = I$, $F_2 = F$, and t the identity function, there are derivations D_1' and D_2' such that:

1) $A \vdash_F t'(A)$ *via* D_1',

2) $t'(A) \vdash_F A$ *via* D_2', and for $i=1,2$,

3) $\ell^s D_i' \leq a_1 \cdot \ell^s D_i \leq a_1 \cdot b_1 \cdot \ell^c A \leq c_1 \cdot \ell^c A$, where $c_1 = a_1 \cdot b_1$, and

4) $\ell D_i' \leq a_2 \cdot (\ell D_i + \begin{cases} \ell A & \text{if } i=1 \\ \ell t'(A) & \text{if } i=2 \end{cases}) \cdot d(D_i)$

$\leq a_2 \cdot 2 \cdot \ell D_i \cdot 1$

$\leq 2 \cdot a_2 \cdot b_2 \cdot \ell^c A \cdot \ell A$

$\leq c_2 \cdot \ell^c A \cdot \ell A$, where $c_2 = 2 \cdot a_2 \cdot b_2$. ⊠5.3.1.2.c.

LEMMA 5.3.1.2.d.

If $I=\langle \kappa_1, \mathcal{R}_1 \rangle$ is an inference system, $F=\langle \kappa_2, \mathcal{R}_2 \rangle$ is a Frege system, t_1 is a direct translation from κ_1 to κ_2, and t_2 is a direct translation from κ_2 to κ_1, then there are constants e_1 and e_2 such that whenever $t_2(\Gamma) \vdash_I t_2(A_0)$ *via* D there is a derivation D'' such that:

1) $\Gamma \vdash_F A_0$ *via* D''

2) $\ell^s D'' \leq e_1 \cdot (\ell^s D + \ell^c \Gamma + \ell^c A_0)$, and

3) $\ell D'' \leq e_2 \cdot [(\ell D + \ell \Gamma) \cdot d(D) + \ell^c \Gamma \cdot \ell \Gamma + \ell^c A_0 \cdot \ell A_0]$.

Proof

Let e_3 be a constant such that $\ell t_2(A) \leq e_3 \cdot \ell A$. Then, by lemma 5.3.1.2.a., there is a derivation D', where

1) $t_1(t_2(\Gamma)) \vdash_F t_1(t_2(A_0))$ *via* D',

2) $\ell^s D' \leq a_1 \cdot \ell^s D$, and

3) $\ell D' \leq a_2 \cdot (\ell D + \ell t_2(\Gamma)) \cdot d(D) \leq a_2 \cdot e_3 \cdot (\ell D + \ell \Gamma) \cdot d(D)$.

Let $\Gamma = \{A_1, \ldots, A_k\}$. Then, since $t_1 \circ t_2$ is a direct translation from κ_2 to κ_2, by lemma 5.3.1.2.c. there are derivations $D_0, D_1, \ldots, D_k$ such that

1) $t_1(t_2(A_0)) \vdash_F A_0$ *via* D_0,

2) $A_i \vdash_F t_1(t_2(A_i))$ *via* D_i $(1 \leq i \leq k)$,

3) $\ell^s D_i \leq c_1 \cdot \ell^c A_i$ $(0 \leq i \leq k)$, *and*

4) $\ell D_i \leq c_2 \cdot \ell^c A_i \cdot \ell A_i$.

Then, if $D'' = D_1 \ldots D_k D' D_0$, $\Gamma \vdash_F A$ *via* D''. Also,

$$\ell^s D'' \leq c_1 \cdot \ell^c A_1 + \ldots + c_1 \cdot \ell^c A_k + a_1 \cdot \ell^s D + c_1 \cdot \ell^c A_0 \leq e_1 \cdot (\ell^c \Gamma + \ell^s D + \ell^c A_0),$$

where $e_1=c_1+a_1$, and

$\ell D'' \leq c_2 \cdot \ell^c A_1 \cdot \ell A_1 + \ldots + c_2 \cdot \ell^c A_k \cdot \ell A_k + a_2 \cdot e_3 \cdot (\ell D + \ell \Gamma) \cdot d(D) + c_2 \cdot \ell^c A_0 \cdot \ell A_0$

$\leq e_2 \cdot [(\ell D + \ell \Gamma) \cdot d(D) + \ell^c \Gamma \cdot \ell \Gamma + \ell^c A_0 \cdot \ell A_0]$, where $e_2=c_2+a_2 \cdot e_3$.

⊠5.3.1.2.d.

The groundwork has now been laid so that the main result of this section can be proved.

THEOREM 5.3.1.2.e.

If $F_1=<\kappa_1, \mathcal{R}_1>$ and $F_2=<\kappa_2, \mathcal{R}_2>$ are Frege systems, t_1 is a direct translation from κ_1 to κ_2, and t_2 is a direct translation from κ_2 to κ_1, then F_2 p-simulates F_1. Furthermore, if F_1 is polynomial-bounded by a polynomial of degree r, then F_2 is polynomial-bounded by a polynomial of degree no more than $2r$.

Proof

Let $g=t_2$, and let $h(D,A)=D''$ if $\vdash_{F_1} t_2(A)$ *via* D, (where D'' is the derivation described by lemma 5.3.1.2.d.), and if it is not true that $\vdash_{F_1} t_2(A)$ *via* D, then let $h(D,A)=0$.

By lemma 5.3.1.2.d. (with Γ empty), $\vdash_{F_2} A$ *via* $h(D,A)$, and $\ell h(D,A) \leq e_2 \cdot (\ell D + \ell A)^2$. Also, $F_2(h(D,A))=A$ whenever $F_1(D)=t_2(A)$. Thus, system F_2 simulates F_1. Also, note that $g \in \mathcal{PF}$, and a close look at the proof of lemmas 5.3.1.2.a-d. shows that $h \in \mathcal{PF}$ also. Therefore F_2 p-simulates F_1. ⊠5.3.1.2.e.

Theorem 5.3.1.2.e. leads immediately to the following corollary.

COROLLARY 5.3.1.2.f.

Let C be any class of adequate sets of connectives such that there exist direct translations between any two sets in C. Then, any two Frege systems with connective sets from C can p-simulate each other.

Examples of such classes C are:

1) all adequate subsets of $\kappa_0=\{T,F,\neg,\vee,\subset,\supset,\&,|,\not\supset,\not\subset,\downarrow\}$,
2) all adequate subsets of $\kappa_0\cup\{\equiv,\not\equiv\}$ that contain either $\equiv$ or $\not\equiv$, and
3) all adequate subsets of $\kappa_0\cup\kappa$ that include κ as a subset (where κ is any set of connectives).

5.3.1.3. Indirect Translation

Direct translations between certain pairs of adequate sets of connectives do not exist. For example, there is no direct translation from $\{\neg,\vee,\equiv\}$ to $\{\neg,\vee\}$. In particular, there is no formula in the connectives $\{\neg,\vee\}$ that both is equivalent to $(p\equiv q)$ and contains only one occurrence of each of p and q. Therefore, the requirements for direct translations are too strict, and one of them must be dropped. The first two requirements are essential, and dropping the third requires giving up a great deal of convenience in working with these translations.

Let κ_1 and κ_2 be sets of connectives, and let K_1 and K_2 be the corresponding sets of formulas. A function $t:K_1\to K_2$ is called an *indirect translation* from κ_1 to κ_2 if there is a polynomial $p(n)$ such that for all formulas $B\in K_1$,

1) $t(B) \sim B$, and

2) $\ell t(B) \leq p(\ell B)$.

Note: For the indirect translations considered in this thesis, the condition $t \in \mathcal{PF}$ is also satisfied.

All direct translations are also indirect translations, and it was Spira who first showed the existence of an indirect translation between sets of connectives where no direct translation is possible [Spira 1971]. Spira's method may be extended to show the existence of indirect translations from any set of connectives to any adequate set of connectives. But since direct translations exist from κ_0 to any adequate set of connectives, the indirect translations of interest are those translations from other sets of connectives to κ_0.

Let κ be an arbitrary set of connectives, let K be the set of all formulas in the connectives κ, and let k be the maximum arity of connectives in κ. Thus, K is a set of trees whose maximum branching degree is k. A particular indirect translation from κ to $\kappa_1=\{\mathsf{T},\mathsf{F},\neg,\vee,\&\}$ will be described here and used throughout this and the following paragraph.

Let A be any formula in K. If no atoms occur in A or if A contains one atom occurrence, then let $t(A)$ be either T, F, p, or $\neg p$, whichever is equivalent to A. If A contains $n>1$ occurrences of atoms, then let C be the subformula of A that contains nearest to $\frac{1}{2}n$ occurrences of atoms. (Ties are resolved according to some arbitrary set of rules such as: if A has a subformula of size $\frac{1}{2}n+x$ and another of size $\frac{1}{2}n-x$,

choose the one of size $\frac{1}{2}n+x$; and if A has several subformulas of size s, choose the leftmost.) The number of occurrences of atoms in C must be between $\frac{n}{k+1}$ and $\frac{k \cdot n}{k+1}$. This is true because if D is a subtree of A with more than $\frac{k \cdot n}{k+1}$ occurrences of atoms, then since the degree of the root of D is at most k, one of the subtrees of D must contain more than $\frac{n}{k+1}$ occurrences of atoms.

Let B be the formula with one occurrence of the new atom p such that $A=B\frac{C}{p}$. Then, not counting p, the number of occurrences of atoms in B is also between $\frac{n}{k+1}$ and $\frac{k \cdot n}{k+1}$. Note that $A \sim ((B\frac{I}{p} \& C) \vee (B\frac{E}{p} \& \neg C))$, and let $t(A)=((t(B\frac{I}{p}) \& t(C)) \vee (t(B\frac{E}{p}) \& \neg t(C))$.

It is clear from this construction that $t(A) \sim A$. To show that there is a polynomial p such that $\ell t(A) \leq p(\ell A)$, let $f(n)=\max\{\ell^a t(B) \mid \ell^a B \leq n\}$. Clearly, $f(1)=1$, and f is nondecreasing. For $n>1$, it is certainly true that $f(n) \leq 4f(\frac{k \cdot n}{k+1})$, which leads to the solution $f(n) \leq n^{\frac{2}{\log_2 \frac{k+1}{k}}} \leq n^{2k}$. This last inequality holds because of the following lemma.

LEMMA 5.3.1.3.a.

For integers $k \geq 2$, $(k+1)^k \geq 2k^k$.

Proof

By the binomial theorem,

$$(k+1)^k = \sum_{i=0}^{k} \binom{k}{i} k^i$$

$$= \binom{k}{k} k^k + \binom{k}{k-1} k^{k-1} + \sum_{i=0}^{k-2} \binom{k}{i} k^i$$

$$= 1 \cdot k^k + k \cdot k^{k-1} + \sum_{i=0}^{k-2} \binom{k}{i} k^i$$

$$= 2k^k + \sum_{i=0}^{k-2} \binom{k}{i} k^i$$

$$\geq 2k^k$$ ☒5.3.1.3.a.

From this it can be concluded that $\frac{k+1}{k} \geq 2^{\frac{1}{k}}$, so that $log_2 \frac{k+1}{k} \geq \frac{1}{k}$, which implies that $\frac{2}{log_2 \frac{k+1}{k}} \leq 2k$. For $k=2$, this gives an upper bound on $f(n)$ of n^4. Pratt [Pratt 1974] has shown by a more careful analysis that $f(n)$ is in fact $O(n^{log_3 10})$, or about $O(n^{2.095})$, which nearly equals Khrapchenko's lower bound of n^2 [Khrapchenko 1971]. Pratt's analysis is specific to the case $k=2$, but a similar analysis for $k>2$ would probably yield upper bounds nearer to n^k. Finally, since $\ell t(B)$ is bounded by some constant times $\ell^a t(B) \cdot \ell B$, the bound (admittedly very loose) $\ell t(B) \leq c \cdot (\ell B)^{2k+1}$ is obtained.

5.3.1.4. Frege Systems and Indirect Translations

Let $\kappa_0=\{T,F,\neg,\vee,\subset,\supset,\&,|,\not\supset,\not\subset,\downarrow\}$, $\kappa_1=\{T,F,\neg,\vee,\&\}$, let κ be an arbitrary adequate set of connectives, and let $\mathfrak{K}=\kappa\cup\kappa_1$. Let k be the maximum arity of connectives in $\mathfrak{K}$. By corollary 5.3.1.2.f. all Frege systems in the connectives $\mathfrak{K}$ and κ can p-simulate each other. Thus, in order to show that any two Frege systems can p-simulate each other, it is sufficient to show a Frege system with the connectives κ_1 and another system with the connectives $\mathfrak{K}$ that p-simulate each other.

Let $F=<\kappa,\mathcal{R}>$ be a Frege system, and let $\hat{F}=<\mathfrak{K},\hat{\mathcal{R}}>$ be the Frege system obtained from F by the addition of sufficient new rules to make $\hat{F}$ implicationally complete. See, for example, the proof of lemma 5.3.1.2.b. Let t be the indirect translation from $\mathfrak{K}$ to κ_1 as described in paragraph 5.3.1.3. A Frege system $F_1=<\kappa_1,\mathcal{R}_1>$ can be obtained from Shoenfield's system for $\{\neg,\vee\}$ with the addition of rules for $\{T,F,\&\}$ as in the proof of lemma 5.3.1.2.b. Also, let $F_0=<\kappa_0,\mathcal{R}_0>$ be an arbitrary Frege system in connectives κ_0. Finally, let $\hat{E}(p,q)$ and $\hat{I}(r,s)$ be formulas in the connectives κ_1 such that $\hat{E}(p,q)\sim(p\equiv q)$ and $\hat{I}(r,s)\sim(r\supset s)$. For example, $\hat{E}(p,q)$ could be $((p\&q)\vee(\neg p\&\neg q))$, and $\hat{I}(r,s)$ could be $(\neg r\vee s)$. The notation defined here will be used throughout paragraph 5.3.1.4.

Lemmas 5.3.1.4.a-d. will show how $\hat{F}$ p-simulates F_1.

LEMMA 5.3.1.4.a.

There are an inference system $I=<\mathfrak{K},\tilde{\mathcal{R}}>$ and constants a_1 and a_2 such that for every formula A in the connectives $\mathfrak{K}$, there are derivations D_1 and D_2 such that:

1) $A \vdash_I t(A)$ *via* D_1,

2) $t(A) \vdash_I A$ *via* D_2, and for $i=1,2$,

3) $\ell^s D_i \leq a_1 \cdot (\ell A)^{3k}$,

4) $\ell D_i \leq a_2 \cdot (\ell A)^{5k+1}$,

5) $d(D_i) = 1$.

Proof

Let $\tilde{\mathcal{R}}$ contain the following rules:

1) $\rightarrow \hat{E}(p,p)$,

2) $\hat{E}(p,q),\hat{E}(q,r) \rightarrow \hat{E}(p,r)$,

3) $\hat{E}(p,q) \rightarrow \hat{E}(\neg p,\neg q)$,

4) $\hat{E}(p,q),p \rightarrow q$,

5) $\hat{E}(p,q),q \rightarrow p$,

6) $\hat{E}(p,q),\hat{I}(p,r) \rightarrow \hat{I}(q,r)$,

7) $\rightarrow \hat{I}(p,\hat{E}(q,q))$,

8) $\rightarrow \hat{I}(p,\hat{E}(\mathsf{T},p))$,

9) $\rightarrow \hat{I}(\neg p,\hat{E}(\mathsf{F},p))$,

10) $\hat{I}(p,\hat{E}(q,r)),\hat{E}(q,s) \rightarrow \hat{I}(p,\hat{E}(s,r))$,

11) $\hat{I}(p,\hat{E}(q,r)),\hat{I}(\neg p,\hat{E}(s,r)) \rightarrow \hat{E}(r,((q\&p)\vee(s\&\neg p)))$,

and for each connective $*\in\mathcal{R}$, $\tilde{\mathcal{R}}$ contains the rules:

12) for each $1\leq i\leq n^*$, and for each $\vec{u}\in\{\mathsf{T},\mathsf{F}\}^{i-1}$ and $\vec{v}\in\{\mathsf{T},\mathsf{F}\}^{n-i}$,

$\rightarrow \hat{E}(*(\vec{u},p_i,\vec{v}),t(*(\vec{u},p_i,\vec{v})))$, and

13) $\hat{I}(p,\hat{E}(q_1,r_1)),\ldots,\hat{I}(p,\hat{E}(q_{n^*},r_{n^*})) \rightarrow \hat{I}(p,\hat{E}(*(\vec{q}),*(\vec{r})))$.

All of these rules are sound, so that I is in fact an inference system.

Derivations D_1 and D_2 both are built from a derivation D, where $\vdash_I \hat{E}(A,t(A))$ *via* D. In fact, each is only one step longer than D, D_1 using rule 4, and D_2 using rule 5. Derivation D is constructed recursively on the number of occurrences of atoms in A.

If A contains no atoms or one occurrence of an atom, then rules 1, 2, 3, and 12 can be used to derive $\hat{E}(A,t(A))$ in a number of steps bounded by twice the number of occurrences of connectives in A plus one.

If A contains more than one occurrence of an atom and $t(A)$ is $((t(B\frac{I}{p})\&t(C))\vee(t(B\frac{E}{p})\&\neg t(C))$, assume inductively that D^0, D^+, and D^- are derivations of $\hat{E}(C,t(C))$, $\hat{E}(B\frac{I}{p},t(B\frac{I}{p}))$, and $\hat{E}(B\frac{E}{p},t(B\frac{E}{p}))$, respectively. If there were derivations D^{++} of $\hat{I}(p,\hat{E}(B\frac{I}{p},B))$ and D^{--} of $\hat{I}(\neg p,\hat{E}(B\frac{E}{p},B))$, then $D^{++}\frac{C}{p}$ would be a derivation of $\hat{I}(C,\hat{E}(B\frac{I}{p},A))$, and $D^{--}\frac{C}{p}$ would be a derivation of $\hat{I}(\neg C,E(B\frac{E}{p},A))$. Applying rules 6 and 3 to each of these formulas plus $\hat{E}(C,t(C))$, gives $\hat{I}(t(C),\hat{E}(B\frac{I}{p},A))$ and $\hat{I}(\neg t(C),\hat{E}(B\frac{E}{p},A))$ in 3 steps. Two applications of rule 10 give $\hat{I}(t(C),\hat{E}(t(B\frac{I}{p}),A))$ and $\hat{I}(\neg t(C),\hat{E}(t(B\frac{E}{p}),A))$, and one application of rule 11 yields $\hat{E}(A,((t(B\frac{I}{p})\&t(C))\vee(t(B\frac{E}{p})\&\neg t(C))))=\hat{E}(A,t(A))$, as desired. The total number of steps in this derivation is six plus the numb

of steps in $D^0D^0D^+D^-D^{++}\frac{C}{p}D^{--}\frac{C}{p}$. Derivations D^0, D^+, and D^- are derived inductively, and derivations D^{++} and D^{--} are constructed, using rules 7, 8, 9, and 13, by induction on the number of occurrences of connectives in B, the total number of steps being bounded by some constant c times that number of connectives.

Let $f(n)$ be an upper bound on the number of steps in the derivation D over all formulas A of length no greater than n. Then $f(1)=1$, and for $n>1$, $f(n)\leq 6+4\cdot f(\frac{k\cdot n}{k+1})+2\cdot c\cdot n$. Using lemma 5.3.1.3.a., it can be shown that $f(n)\leq 2\cdot c\cdot n^{3k}$. Finally, note that each formula in D has length bounded by some constant d times $\ell t(A)$, so that $\ell^s D\leq 2\cdot c\cdot(\ell A)^{3k}$, and $\ell D\leq 2\cdot c\cdot d\cdot(\ell A)^{5k+1}$. Each formula in the entire derivation is used only once, so $d(D)=1$. ⊠5.3.1.4.a.

Lemma 5.3.1.4.a. leads immediately to the following lemma.

LEMMA 5.3.1.4.b.

There is a Frege system $\tilde{F}=<\mathcal{R},\ddot{\mathcal{R}}>$ and there are constants b_1 and b_2 such that whenever $t(\Delta)\vdash_{F_1} t(A)$ *via* D_1 there is a derivation D_2 such that: (Note - Δ is any set of formulas in $\hat{K}$.)

1) $\Delta\vdash_{\tilde{F}} A$ *via* D_2,

2) $\ell^s D_2 \leq b_1\cdot(\ell^s D_1+(\ell\Delta)^{3k}+(\ell A)^{3k})$, and

3) $\ell D_2 \leq b_2\cdot(\ell D_1+(\ell\Delta)^{5k+1}+(\ell A)^{5k+1})$.

Proof

Let $\ddot{\mathcal{R}}=\mathcal{R}_1 \cup \tilde{\mathcal{R}}$, where $\tilde{\mathcal{R}}$ is the set of rules from lemma 5.3.1.4.a. If $\Delta=\{B^1,\ldots,B^m\}$, then by lemma 5.3.1.4.a., there are derivations $D^1,\ldots,D^m$ such that for $1\le i\le m$

1) $B^i \vdash_{\tilde{F}} t(B^i)$ *via* D^i,

2) $\ell^s D^i \le a_1\cdot(\ell B^i)^{3k}$, and

3) $\ell D^i \le a_2\cdot(\ell B^i)^{5k+1}$.

Also, there is a derivation D^0 such that

$t(A) \vdash_{\tilde{F}} A$ *via* D^0, $\ell^s D^0 \le a_1\cdot(\ell A)^{3k}$, and $\ell D^0 \le a_2\cdot(\ell A)^{5k+1}$.

Since D_1 is already a derivation in the system $\tilde{F}$, the desired derivation D_2 is just $D^1\ldots D^m D_1 D^0$, and the lemma follows, with $b_1=a_1$ and $b_2=a_2$. ⊠5.3.1.4.b.

LEMMA 5.3.1.4.c.

System $\tilde{F}$ of lemma 5.3.1.4.b. p-simulates system F_1.

Proof

Let $g(A)=t(A)$, and let $h(D_1,A)=D_2$, if $\vdash_{F_1} t(A)$ *via* D_1, where D_2 is the derivation whose existence is guaranteed by lemma 5.3.1.4.b. (where Δ is empty); and if it is not true that $\vdash_{F_1} t(A)$ *via* D_1, then let $h(D_1,A)=0$. By lemma 5.3.1.4.b., $\ell h(D_1,A) \le b_2(\ell D_1+(\ell A)^{5k+1}) = q(\ell D_1,\ell A)$, and $\tilde{F}(h(D_1,A))=A$ whenever $F_1(D_1)=g(A)$. Also, $\ell g(A)\le c\cdot(\ell A)^{2k+1}$, and $g,h\in \mathcal{PF}$. Therefore $\tilde{F}$ p-simulates F_1. ⊠5.3.1.4.c.

Since p-simulation is transitive, lemma 5.3.1.4.c. and theorem 5.3.1.2.e. can be combined to give the following.

LEMMA 5.3.1.4.d.

Every Frege system F p-simulates F_1 and F_0.

That is, the adequate sets of connectives of arity no greater than two which do not include $\equiv$ or $\not\equiv$ form a "core", and any Frege system whose connectives are one of these "core" sets can be p-simulated by any other Frege system. What remains to be shown is that these "core" systems can p-simulate the others.

The proof that F_1 p-simulates $\hat{F}$ will be similar to the development in paragraph 5.3.1.2., but since there may be no direct translation from $\mathcal{R}$ to κ_1, some new arguments will be needed. In particular, the analog of lemma 5.3.1.2.a. will have to work when t is an indirect translation, rather than a direct one. The analog for indirect translation of lemma 5.3.1.2.b. has already been established by lemma 5.3.1.4.a. The proof of the analog of lemma 5.3.1.2.a. is rather involved, and will be built up by a series of stages.

LEMMA 5.3.1.4.e.

There is an inference system $I=<\kappa_1, \mathcal{R}'>$ and there are constants e_1 and e_2 such that for any formula A in the connectives $\mathcal{R}$ and for any subformula E of A and formula G (with one occurrence of q) such that $G\frac{E}{q}=A$, there is a derivation D such that

1) $\vdash_I \hat{E}(t(A),((t(G\frac{I}{q})\&t(E))\vee(t(G\frac{E}{q})\&\neg t(E))))$ *via* D,

2) $\ell^{s}D \leq e_1\cdot(\ell A)^{4k}$,

3) $\ell D \leq e_2 \cdot (\ell A)^{6k+1}$, and

4) $d(D)=1$.

Proof

The idea of the proof is to recursively build a derivation D of $\hat{E}(t(A),((t(G\frac{I}{q})\&t(E))\vee(t(G\frac{E}{q})\&\neg t(E))))$. Clearly, if A contains no more than a certain small finite number n_0 of occurrences of atoms, then if I contains a rule for each of the finite number of possible cases, the required derivation can consist of a single step.

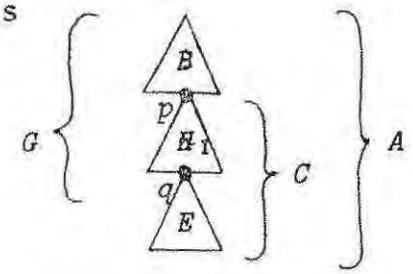
Let the number of occurrences of atoms in A be $n>n_0$, and let C be that subformula of A such that $A=B\frac{C}{p}$ and $t(A)=((t(B\frac{I}{p})\&t(C))\vee(t(B\frac{E}{p})\&\neg t(C)))$. If $C=E$ and $G=B\frac{q}{p}$, then the required derivation is the single-step instance of the rule $\hat{E}(p,p)$. Otherwise, there are three possible cases:

1) E is a subtree (subformula) of C, or

2) C is a subtree of E, or

3) C and E are disjoint subtrees of A.

In case 1), C can be written as $H_1\frac{E}{q}$, and G can be written as $B\frac{H_1}{p}$. Pictorially, this is



Let D^1, D^2, and D^3 be the following three derivations (derived recursively):

$\vdash_I \hat{E}(t(C),((t(H_1\frac{I}{q})\&t(E))\vee(t(H_1\frac{E}{q})\&\neg t(E))))$ *via* D^1,

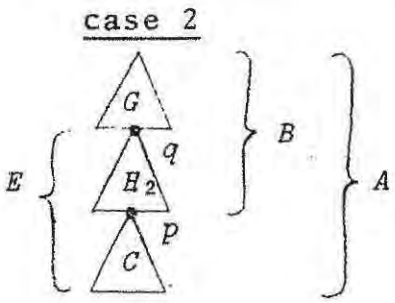
$\vdash_I \hat{E}(t(G\frac{I}{q}),((t(B\frac{I}{p})\&t(H_1\frac{I}{q}))\vee(t(B\frac{E}{p})\&\neg t(H_1\frac{I}{q}))))$ *via* D^2, and

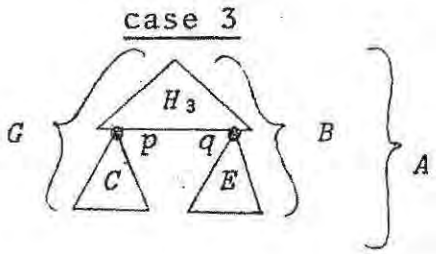
$\vdash_I \hat{E}(t(G\frac{E}{q}),((t(B\frac{I}{p})\&t(H_1\frac{E}{q}))\vee(t(B\frac{E}{p})\&\neg t(H_1\frac{E}{q}))))$ *via* D^3.

The desired conclusion of the derivation D is inferred from these three equivalences by an instance of the sound inference rule: R_1:

$\hat{E}(r,((v\&u)\vee(w\&\neg u))),\hat{E}(s,((p\&v)\vee(q\&\neg v))),\hat{E}(t,((p\&w)\vee(q\&\neg w)))$
$\rightarrow \hat{E}(((p\&r)\vee(q\&\neg r)),((s\&u)\vee(t\&\neg u)))$.

Cases 2 and 3 pictorially are:





Case 2 is similar to case 1, and gives rise to three recursively derived derivations, from which the conclusion of D can be inferred by an instance of rule R_2:

$\hat{E}(u,((x\&r)\vee(y\&\neg r))),\hat{E}(p,(s\&x)\vee(t\&\neg x))),\hat{E}(q,((s\&y)\vee(t\&\neg y)))$
$\rightarrow \hat{E}(((p\&r)\vee(q\&\neg r)),((s\&u)\vee(t\&\neg u)))$.

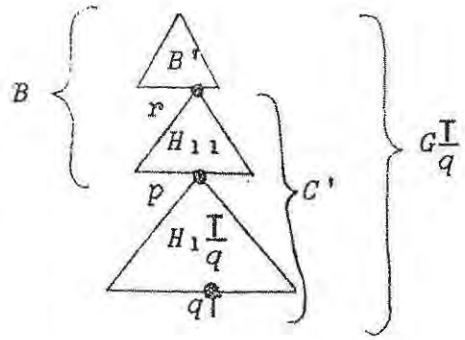
Case 3 gives rise to four recrusively derived derivations, from which the conclusion of D can be derived by an instance of rule R_3:

$\hat{E}(p,((v\&u)\vee(w\&\neg u))),\hat{E}(q,((x\&u)\vee(y\&\neg u))),\hat{E}(s,((v\&r)\vee(x\&\neg r))),$
$\hat{E}(t,((w\&r)\vee(y\&\neg r))) \rightarrow \hat{E}(((p\&r)\vee(q\&\neg r)),((s\&u)\vee(t\&\neg u)))$.

Therefore, the inference system I must contain rules R_1, R_2, and R_3, rules for each of the cases when $n \leq n_0$, and the rules $\rightarrow \hat{E}(p,p)$. Each formula in D is bounded in length by some constant $c \cdot \ell t(A)$, and $d(D) = 1$, by this construction.

To show a polynomial upper bound on the number of steps of D, it is necessary to consider two levels of recursion at once. Let $f(n) = \max\{\ell^s D \mid \ell^a A \leq n$, and D is constructed by this method$\}$. Then, f is nondecreasing, and if $n_0 = 1$, then $f(1) = 1$. For $n \geq 2$ it is necessary to consider a second level of recursion. This involves an analysis of a total of nine cases, one of which will be analyzed here as an example.

If case 1 holds, then $\ell^s D^1 \leq f(\frac{k \cdot n}{k+1})$, because the number of occurrences of atoms in C must be between $\frac{n}{k+1}$ and $\frac{k \cdot n}{k+1}$, by virtue of the fact that t is the indirect translation. The number of occurrences of atoms in $G\frac{\mathsf{T}}{q}$ and $G\frac{\mathsf{F}}{q}$, however, may be as large as n-1, so a second level of recursion must be considered for them. Since the formulas involved in derivations D^2 and D^3 are the same except for the interchanges of some T's and F's, they will have the same length. Consider D^2, and assume that case 1 again holds, so that $t(G\frac{\mathsf{T}}{q}) = ((t(B'\frac{\mathsf{T}}{r}) \& t(C')) \vee (t(B'\frac{\mathsf{F}}{r}) \& \neg t(C')))$, where $H_1\frac{\mathsf{T}}{q}$ is a subtree of C'. Pictorially, this is



Recursively, this calls for three derivations D^{21}, D^{22}, and D^{23}, where

$\vdash_I \acute{e}(t(C'),((t(H_{11}\frac{I}{p})\&t(H_1\frac{I}{q}))\vee(t(H_{11}\frac{E}{p})\&\neg t(H_1\frac{I}{q})))$ *via* D^{21},

$\vdash_I \hat{E}(t(B\frac{I}{p}),((t(B'\frac{I}{r})\&t(H_{11}\frac{I}{p}))\vee(t(B'\frac{E}{r})\&\neg t(H_{11}\frac{I}{p})))$ *via* D^{22}, and

$\vdash_I \hat{E}(t(B\frac{E}{p}),((t(B'\frac{I}{r})\&t(H_{11}\frac{E}{p}))\vee(t(B'\frac{E}{r})\&\neg t(H_{11}\frac{E}{p})))$ *via* D^{23}.

But now let n' be the number of occurrences of atoms in $G\frac{I}{q}$. Then, since C' is chosen by the t function, the number of occurrences of atoms in C' is between $\frac{n'}{k+1}$ and $\frac{k\cdot n'}{k+1}$, and since $n'<n$, this is less than $\frac{k\cdot n}{k+1}$. Similarly, the number of occurrences of atoms in $B\frac{I}{p}$ and $B\frac{E}{p}$ is between $\frac{n}{k+1}$ and $\frac{k\cdot n}{k+1}$, so that the number of steps in each of D^{21}, D^{22}, and D^{23} is no greater than $f(\frac{k\cdot n}{k+1})$. This implies that

$\ell^s D \leq 1+f(\frac{k\cdot n}{k+1})+2(1+3f(\frac{k\cdot n}{k+1})) = 3+7f(\frac{k\cdot n}{k+1})$, for this one case 1.1.

The analysis for the other eight cases is similar, the worst case being case 3.3 which gives

$$\ell^s D \leq 1+2f(\tfrac{k\cdot n}{k+1})+2(1+4f(\tfrac{k\cdot n}{k+1})) = 3+10f(\tfrac{k\cdot n}{k+1}).$$

Therefore, any function g satisfying

$$g(1)\geq 1, \text{ and}$$

$$g(n) \geq 3+10g(\tfrac{k}{k+1}n) \quad (n\geq 2)$$

gives an upper bound for $f(n)$. One such function is $g(n)=n^{4k}$, since $1^{4k}=1\geq 1$, and for $n\geq 2$,

$$\begin{aligned} 3+10\cdot(\tfrac{k}{k+1}n)^{4k} &= [\tfrac{3}{n^{4k}}+10\cdot(\tfrac{k}{k+1})^{4k}]\cdot n^{4k} \\ &\leq [\tfrac{3}{2^4}+10\cdot((\tfrac{k}{k+1})^k)^4]\cdot n^{4k} \\ &\leq [\tfrac{3}{16}+10\cdot(\tfrac{1}{2})^4]\cdot n^{4k} \qquad \text{(lemma 5.3.1.3.a.)} \\ &\leq [\tfrac{13}{16}]n^{4k} \\ &\leq n^{4k}. \end{aligned}$$

Therefore, $\ell^{\mathcal{S}}D\leq(\ell A)^{4k}$, so that $\ell D \leq c\cdot\ell t(A)\cdot\ell^{\mathcal{S}}D \leq c\cdot(\ell A)^{6k+1}$.

□5.3.1.4.e.

LEMMA 5.3.1.4.f.

Lemma 5.3.1.4.e. still holds, even if the inference system I is replaced by Frege system F_1.

Proof

This is an immediate consequence of lemma 5.3.1.2.a., where the direct translation from κ_1 to κ_1 is the identity function. □5.3.1.4.f.

Lemmas 5.3.1.4.e. and 5.3.1.4.f. form the crucial link in the simulation of system $\hat{P}$ by system F_1. Now the simulation may proceed by analogy with the proof of lemma 5.3.1.2.a., with indirect translation here playing the rôle that direct translation played there.

LEMMA 5.3.1.4.g.

There are constants g_1 and g_2 such that whenever $\Gamma \vdash_{\hat{R}} A$ *via* D there is a derivation D' such that:

1) $t(\Gamma) \vdash_{F_1} t(A)$ *via* D',

2) $\ell^s D' \leq g_1 \cdot (\ell\Gamma+\ell D)^{4k+1} \cdot d(D)$, and

3) $\ell D' \leq g_2 \cdot (\ell\Gamma+\ell D)^{6k+2} \cdot d(D)$.

Proof

The proof proceeds analogously to lemma 5.3.1.2.a. For each formula B in D, derivation D' has the formula $t(B)$. For each inference rule $R_i \in \hat{\mathcal{R}}$, where $R_i = \Delta_i \rightarrow B_i$, let D^i be a derivation in system F_1 such that $t(\Delta_i) \vdash_{F_1} t(B_i)$ *via* D^i. Now suppose that B is inferred from $\hat{\Delta}$ in D by substitution σ in rule R_i. Then, $B = B_i\sigma$, $\hat{\Delta} = \Delta_i\sigma$, and each formula in $\hat{\Delta}$ either is a hypothesis of D or appears in D before B. Let $D^B = D^i t(\sigma)$, so that $t(\Delta_i)t(\sigma) \vdash_{F_1} t(B_i)t(\sigma)$ *via* D^B. Unlike the case in lemma 5.3.1.2.a., insertion of D^B in place of B is not sufficient to make D' the desired derivation. This is because t is an indirect (rather than a direct) translation, so that $t(B_i)t(\sigma)$ is not necessarily the same formula as $t(B_i\sigma)$. In fact, what must be done is that $t(B_i\sigma)$ must be derived from $t(B_i)t(\sigma)$, and for each $C_j \in \Delta_i$, $t(C_j)t(\sigma)$ must be derived from $t(C_j\sigma)$.

This is where lemmas 5.3.1.4.e. and 5.3.1.4.f. come in. Remember that since $\hat{P}$ has only a finite number of finite rules of inference, B_i (and each $C_j \epsilon \Delta_i$) is one of a finite number of formulas for which these derivations must be constructed. Only the substitution σ is allowed to vary arbitrarily. To derive $t(B_i\sigma)$ from $t(B_i)t(\sigma)$, a derivation of $\hat{E}(t(B_i\sigma), t(B_i)t(\sigma))$ is constructed by recursion on the number of occurrences of atoms in B_i. Consider first the case where B_i contains no occurrences of atoms. In this case $B_i\sigma = B_i$ and $t(B_i)t(\sigma) = t(B_i) = t(B_i\sigma)$, so that the desired derivation is just an instance of the standard derivation of the tautology $\hat{E}(p,p)$.

Suppose that B_i contains at least one occurrence of the atom p, r is a new atom, and σ substitutes C for p. Then, there is a formula $\tilde{B}_i$ with one occurrence of r such that $B_i = \tilde{B}_i\frac{p}{r}$, and $B_i\sigma = \tilde{B}_i\sigma\frac{C}{r}$. By lemma 5.3.1.4.f., let D^0 be a derivation of $\hat{E}(t(B_i\sigma), ((t(\tilde{B}_i\sigma\frac{I}{r}) \& t(C)) \vee (t(\tilde{B}_i\sigma\frac{F}{r}) \& \neg t(C))))$. Since $\tilde{B}_i\frac{I}{r}$ and $\tilde{B}_i\frac{F}{r}$ contain one less occurrence of atoms than B_i, assume as inductive hypothesis that there exist derivations D^+ and D^- of $\hat{E}(t(\tilde{B}_i\frac{I}{r}\sigma), t(\tilde{B}_i\frac{I}{r})t(\sigma))$ and $\hat{E}(t(\tilde{B}_i\frac{F}{r}\sigma), t(\tilde{B}_i\frac{F}{r})t(\sigma))$, respectively. From these three equivalences (noting that $\tilde{B}_i\sigma\frac{I}{r} = \tilde{B}_i\frac{I}{r}\sigma$), $\hat{E}(t(B_i\sigma), ((t(\tilde{B}_i\frac{I}{r})t(\sigma) \& t(C)) \vee (t(\tilde{B}_i\frac{F}{r})t(\sigma) \& \neg t(C))))$ can be derived in a fixed number of steps. (In effect, equals

have been substituted for equals.) But this last equivalence is the same as $\hat{E}(t(B_i\sigma),((t(\tilde{B}_{ir}\underline{T})\&p)\vee(t(\tilde{B}_{ir}\underline{F})\&\neg p))t(\sigma))$. Applying lemma 5.3.1.4.f. again, let D'' be a derivation of $\hat{E}(t(B_i),((t(\tilde{B}_{ir}\underline{T})\&p)\vee(t(\tilde{B}_{ir}\underline{F})\&\neg p)))$, so that $D''t(\sigma)$ is a derivation of $\hat{E}(t(B_i)t(\sigma),((t(\tilde{B}_{ir}\underline{T})\&p)\vee(t(\tilde{B}_{ir}\underline{F})\&\neg p))t(\sigma))$. These last two equivalences lead to a derivation of $\hat{E}(t(B_i\sigma),t(B_i)t(\sigma))$ in a constant number of steps.

To analyze the length of this derivation, first observe that every formula in it is bounded in length by some constant c times $\ell t(B_i\sigma)$. Then, note that the number of steps in D'' is fixed because B_i is fixed, and that the number of recursion steps in derivations D^+ and D^- is also fixed, since at each level of recursion the formula into which σ is being substituted has one less occurrence of atoms than at the previous level, and again since B_i is fixed. Thus, the total number of steps in this entire derivation is bounded by some constant d times the number of steps in D^0. By lemma 5.3.1.4.f., this quantity is bounded above by $d\cdot e_1\cdot(\ell B_i\sigma)^{4k}$.

Putting this derivation together with similar derivations for $C_j\sigma$ for each $C_j\epsilon\Delta_i$, and with D^B, gives a derivation D_B of $t(B)=t(B_i\sigma)$ from $t(\Delta_i\sigma)=t(\hat{\Delta})$, such that there are constants (independent of σ) g_1 and g_2 such that $\ell^{\mathcal{E}}D_B \le g_1\cdot(\ell\hat{\Delta}+\ell B)^{4k}$ and $\ell D_B \le g_2\cdot(\ell\Delta+\ell B)^{6k+1}$. Putting together these derivations D_B

for each formula B in D gives the derivation D' such that

$t(\Gamma) \vdash_{F_1} t(A)$ *via* D',

$\ell^{s}D' \leq g_1 \cdot (\ell\Gamma+\ell D)^{4k+1} \cdot d(D)$, and

$\ell D' \leq g_2 \cdot (\ell\Gamma+\ell D)^{6k+2} \cdot d(D)$. ⊠5.3.1.4.g.

Lemmas 5.3.1.4.g. and 5.3.1.4.a. can be combined to give the following.

LEMMA 5.3.1.4.h.

Frege system F_1 p-simulates $\hat{F}$.

Proof

Since $\kappa_1 \subseteq \mathcal{R}$, g is the identity function. The function h is built using the constructions in lemmas 5.3.1.4.g. and 5.3.1.4.a. If $\vdash_{\hat{F}} A$ *via* D, then $h(D,A)$ is $D'D_2$, where D' is the derivation (in system F_1) of $t(A)$ constructed by lemma 5.3.1.4.g. and D_2 is the derivation of A from $t(A)$ constructed by lemma 5.3.1.4.a. It is a routine matter to verify that $g,h \in \mathcal{P}\mathcal{F}$ and that $F_1(h(D,A))=A$ whenever $\hat{F}(D)=g(A)$. ⊠5.3.1.4.h.

Combining this result with lemma 5.3.1.4.d. and the transitivity of p-simulation, the main theorem of this section is immediate.

THEOREM 5.3.1.4.i.

Any two Frege systems p-simulate each other.

5.3.2. Extended Frege Systems

Cook has suggested [Cook 1975b] that although the extension rule was introduced in conjunction with resolution, its use is really more natural with Frege systems. Let $F=\langle \kappa, \mathcal{R} \rangle$ be a Frege system, and let $\hat{E}(p,q)$ be any formula in the connectives κ such that $\hat{E}(p,q) \sim (p \equiv q)$. Then, the *extension rule* for F can be stated as follows: if p is any atom that does not appear in either the hypotheses or the conclusion of a derivation D, and if p does not appear in any of the previous formulas in D, and if A is any formula not containing p in the connectives κ, then the formula $\hat{E}(p,A)$ may be added to D. Stated another way, if $\Gamma \vdash_F B$ via D then $\Gamma \vdash_{eF} B$ via D, and if p does not occur in Γ nor A nor B then if $\Gamma, \hat{E}(p,A) \vdash_{eF} B$ via D then $\Gamma \vdash_{eF} B$ via $E(p,A)D$.

Clearly the extended Frege system eF is implicationally complete, since all of the derivations of F are also derivations of eF. To see that eF is consistent, suppose that $\Gamma, \hat{E}(p,A) \models B$. Since p has no occurrences in Γ, A, or B, it must also be true that $\Gamma, \hat{E}(\neg p,A) \models B$. But $\hat{E}(\neg p,A) \sim \neg \hat{E}(p,A)$, so it must be true that $\Gamma \models B$.

Since Frege systems are a special case of extended Frege systems, any extended Frege system p-simulates any Frege system. To see how extended Frege systems p-simulate one another, let $F_1=\langle \kappa_1, \mathcal{R}_1 \rangle$ and $F_2 \langle \kappa_2, \mathcal{R}_2 \rangle$ be two Frege systems, and let t be a translation (either direct or indirect, as appropriate) from κ_1 to κ_2. The techniques of subsection 5.3.1. can be used to show that the extension steps of a derivation can be simulated in a manner similar to the simulation of

inferences. To see how eF_2 simulates eF_1, suppose that $\Gamma \vdash_{eF_1} A$ *via* ΔD, and $\Delta = \hat{E}_1(p_1,A_1),\ldots,\hat{E}_1(p_n,A_n)$ are the extensions used in D, so that $\Gamma,\Delta \vdash_{F_1} A$ *via* D. By lemmas 5.3.1.2.a. and 5.3.1.4.g. there is a derivation D', where $t(\Gamma),t(\Delta) \vdash_{F_2} t(A)$ *via* D' and $\ell D' \leq p(\ell\Gamma+\ell\Delta+\ell D)$, for some polynomial p that depends only on F_1, F_2, and t. Let $\Delta' = \hat{E}_2(p_1,t(A_1)),\ldots,\hat{E}_2(p_n,t(A_n))$ be the extensions in system eF_2 corresponding to Δ. Applying lemma 5.3.1.4.e. if t is indirect (if t is direct, the argument is much simpler), it is not hard to show that for $1 \leq i \leq n$ there are polynomial-length-bounded derivations D_i such that $\hat{E}_2(p_i,t(A_i)) \vdash_{F_2} t(\hat{E}_1(p_i,A_i))$ *via* D_i. Combining these derivations with D' gives a derivation D'', where $t(\Gamma),\Delta' \vdash_{F_2} t(A)$ *via* D'' and $\ell D'' \leq q(\ell\Gamma+\ell\Delta+\ell D)$, for some fixed polynomial q. This leads to $t(\Gamma) \vdash_{eF_2} t(A)$ *via* $\Delta' D''$. Finally, the translation t can be "undone" by the application of lemmas 5.3.1.2.b. and/or 5.3.1.4.a., completing the proof of the main theorem of this subsection.

THEOREM 5.3.2.a.

If F_1 and F_2 are Frege systems, then the extended Frege systems eF_1 and eF_2 p-simulate each other.

It was reported in an earlier paper [Cook & Reckhow 1974] that Frege systems can simulate extended resolution (which, as will be shown in subsection 5.6.3., is in the same p-simulation class as extended Frege systems), but that claim was withdrawn in the corrections to that paper. The argument used in that

paper is still valid, although the conclusions drawn from that argument must be changed.

The argument was this. If $\Gamma \vdash_{eF} A$ *via* $\hat{E}(p,B)D$, and $\hat{E}(p,B)$ is the "outermost" extension in extended Frege derivation $\hat{E}(p,B)D$, then $\Gamma \vdash_{eF} A$ *via* $(\hat{E}(p,B)D)\frac{B}{p}$, since p does not occur in Γ or A. But $(\hat{E}(p,B)D)\frac{B}{p} = \hat{E}(B,B)D\frac{B}{p}$. Thus, if $\vdash_F \hat{E}(p,p)$ *via* D_0, then $\vdash_F \hat{E}(B,B)$ *via* $D_0\frac{B}{p}$, so that $\Gamma \vdash_{eF} A$ *via* $D_0\frac{B}{p}D\frac{B}{p} = (D_0D)\frac{B}{p}$. Note that the number of extensions in $(D_0D)\frac{B}{p}$ is one less than the number of extensions in $\hat{E}(p,B)D$, and that only a constant number of steps (the number of steps in D_0) have been added. Applying this construction iteratively until all extensions have been removed yields a derivation D' such that $\Gamma \vdash_F A$ *via* D' and $\ell^s D' \leq c \cdot \ell^s D$. In fact, if $\hat{E}(p_1,B_1),\ldots,\hat{E}(p_n,B_n)$ are the extensions of $\hat{E}(p,B)D$ from outermost to innermost, then D' is $D_0\frac{B_n}{p}\ldots D_0\frac{B_2}{p}D_0\frac{B_1}{p}(\ldots(D\frac{B_1}{p_1})\frac{B_2}{p_2}\ldots)\frac{B_n}{p_n}$.

In order to show that F simulates eF, however, it must be shown that there is a polynomial p such that $\ell D' \leq p(\ell D)$, and this is where the above argument fails. If for $1 \leq i \leq n$, p_{i+1} has two occurrences in B_i (which certainly is possible), then each occurrence of p_1 in D will be replaced in D' by a formula containing 2^n occurrences of the atom p_{n+1}. Since n may grow

linearly with $\ell^s D$, it must be concluded that $\ell D'$ need not be bounded by any polynomial in $\ell^s D'$ or ℓD.

The problem encountered here is similar to the problem from circuit theory of finding the smallest possible upper bound on the size of a formula that is equivalent to a given logical circuit. Equivalently, this is the problem of finding how much increase in circuit size is necessary to go from a fan-out two circuit to a fan-out one circuit for the same logical function. There is a clear equivalence between formulas and fan-out one circuits. To see the relationship between circuits with fan-out greater than one and formulas represented in terms of extensions, consider the extension $\hat{E}(p,B)$ and some formula A, containing several occurrences of p. The extension $\hat{E}(p,B)$ can be interpreted as giving the name p to the output of the circuit for B. A circuit for $A\frac{B}{p}$ can be constructed by connecting the p output of the circuit for B to each of the p inputs to the circuit for A. The size of this circuit is the sum of the sizes of the circuits for A and B.

Perhaps this argument shows that circuits are a more natural and more efficient way of representing logical functions than formulas are. If this is so, then extended Frege systems are perhaps more natural than ordinary Frege systems. But this will not be known for sure until after the fan-out one *vs* fan-out two question is answered.

5.3.3. Frege Systems with Substitution

Recall that an s-Frege system is a Frege system $F=\langle\kappa,\mathcal{R}\rangle$, to which has been added the substitution rule: if A is a formula in the connectives κ, and σ is a substitution in κ, then $A\sigma$ may be inferred from A. Also recall that the substitution rule is sound only if A is valid (or inconsistent), so that s-Frege systems do not admit derivations from hypotheses (unless all hypotheses are tautologies). Since Frege systems are a special case of s-Frege systems (for derivations without hypotheses), s-Frege systems p-simulate Frege systems.

In this section it will be shown first that any two s-Frege systems p-simulate one another, and then that any s-Frege system p-simulates any extended Frege system. It seems unlikely that extended Frege systems can simulate s-Frege systems. Arguments supporting this conjecture are found in [Cook 1975a].

Let $F_1=\langle\kappa_1,\mathcal{R}_1\rangle$ and $F_2=\langle\kappa_2,\mathcal{R}_2\rangle$ be Frege systems, and let sF_1 and sF_2 be the corresponding s-Frege systems. If there is a direct translation from κ_1 to κ_2, the arguments in the proof of lemma 5.3.1.2.a. go through unchanged if I_1 is replaced by sF_1 and F_2 is replaced by sF_2. The only addition needed is to note that if, in D, sF_1 infers $B\sigma$ from B, then sF_2 can infer $t(B)t(\sigma)$ from $t(B)$ in D'. Since t is a direct translation, $t(B)t(\sigma)=t(B\sigma)$, so the simulation can continue as usual. Combining this observation with lemmas 5.3.1.2.b. and 5.3.1.4.a. leads to the following.

THEOREM 5.3.3.a.

If $F_1=\langle\kappa_1,\mathcal{R}_1\rangle$ and $F_2=\langle\kappa_2,\mathcal{R}_2\rangle$ are Frege systems and sF_1 and sF_2 are the corresponding s-Frege systems, and if either there are direct translations both ways between κ_1 and κ_2 or $\kappa_1 \subseteq \{\mathsf{T},\mathsf{F},\neg,\vee,\subset,\supset,\&,|,\not\supset,\not\subset,+\}$, then sF_2 p-simulates sF_1.

The only difficulty with the above argument in the case where t is an indirect translation is that $t(B)t(\sigma)$ is not the same as $t(B\sigma)$. Thus, to show that any s-Frege system p-simulates any other, it suffices to show the existence of a "short" derivation of $\hat{E}(t(B\sigma),t(B)t(\sigma))$ in system F_2, where $\hat{E}(p,q) \sim (p\equiv q)$.

LEMMA 5.3.3.b.

If $\kappa_1=\{\mathsf{T},\mathsf{F},\neg,\vee,\&\}$, t is the indirect translation from κ to κ_1, k is the maximum arity of connectives in κ, and $\hat{E}(p,q) \sim (p\equiv q)$, then there is an inference system $I=\langle\kappa_1,\mathcal{R}\rangle$ and there are constants b_1 and b_2 such that for any formula B and substitution σ in the connectives κ there is a derivation D such that

1) $\vdash_I \hat{E}(t(B\sigma),t(B)t(\sigma))$ *via* D

2) $\ell^s D \leq b_1\cdot(\ell B\sigma)^{4k+1}$

3) $\ell D \leq b_2\cdot(\ell B\sigma)^{6k+2}$

4) $d(D)=1$.

Proof

The proof is based on a construction that recurses on the number of occurrences of atoms in B for which σ makes a substitution. If B has no occurrences of atoms for which σ makes a substitution, then $B\sigma=B$ and $t(B)t(\sigma)=t(B)$, so that

$t(B\sigma)=t(B)t(\sigma)$, and the required derivation is an instance of the rule $\rightarrow\hat{E}(p,p)$.

If B contains one or more occurrences of atoms, then let B' be the formula obtained from B by making every atom occurrence that σ substitutes for a new distinct atom. That is, B' is a formula with the atoms $p_1,\ldots,p_n$, each of which occurs exactly once in B', but not at all in B or σ, and there is a renaming $\sigma' = \frac{q_{i_1}q_{i_2}\cdots q_{i_n}}{p_1\,p_2\cdots p_n}$ such that $B'\sigma' = B$. If $\sigma = \frac{C_1\cdots C_m}{q_1\cdots q_m}$ and $\hat{\sigma} = \sigma'\sigma = \frac{C_{i_1}\cdots C_{i_n}}{p_1\cdots p_n}$, then $B\sigma=(B'\sigma')\sigma=B'(\sigma'\sigma)=B'\hat{\sigma}$. Note also, that since the method of computing $t(B)$ depends only on occurrences of atoms in a formula and not on whether some occurrences are the same atom, $t(B)=t(B')\sigma'$.

Let $\tilde{\sigma} = \frac{C_{i_1}\cdots C_{i_{n-1}}}{p_1\cdots p_{n-1}}$, $\sigma'' = \frac{q_{i_1}\cdots q_{i_{n-1}}}{p_1\cdots p_{n-1}}$, $C_{i_n}=C$, $q_{i_n}=q$, and $p_n=p$. Then, $B=(B'\sigma'')\frac{q}{p}$, $B\sigma=(B'\tilde{\sigma})\frac{C}{p}$, $t(B)=(t(B')\sigma'')\frac{q}{p}$, and $t(B)t(\sigma)=(t(B')\sigma')t(\sigma)=t(B')t(\hat{\sigma})=(t(B')t(\tilde{\sigma}))\frac{t(C)}{p}$. Since $\tilde{\sigma}$ substitutes for n-1 occurrences of atoms in B', assume as induction hypothesis that a derivation D_1 of $\hat{E}(t(B'\tilde{\sigma}),t(B')t(\tilde{\sigma}))$ has been constructed. Then, $D_1\frac{t(C)}{p}$ is a derivation of $\hat{E}(t(B'\tilde{\sigma})\frac{t(C)}{p},(t(B')t(\tilde{\sigma}))\frac{t(C)}{p})$, which is the same as $\hat{E}(t(B'\tilde{\sigma})\frac{t(C)}{p},t(B)t(\sigma))$. By lemma 5.3.1.4.e. there is a

derivation D_2 of $\hat{E}(t((B'\tilde{\sigma})\frac{C}{p}),((t((B'\tilde{\sigma})\frac{I}{p})\&t(C))\vee(t((B'\tilde{\sigma})\frac{F}{p})\&\neg t(C))))$, which is the same formula as $\hat{E}(t(B\sigma),((t((B'\tilde{\sigma})\frac{I}{p})\&p)\vee(t((B'\tilde{\sigma})\frac{F}{p})\&\neg p))\frac{t(C)}{p})$. Also, by lemma 5.3.1.4.e., there is a derivation D_3 of $\hat{E}(t(B'\tilde{\sigma}),((t((B'\tilde{\sigma})\frac{I}{p})\&p)\vee(t((B'\tilde{\sigma})\frac{F}{p})\&\neg p)))$, so that $D_3\frac{t(C)}{p}$ is a derivation of $\hat{E}(t(B'\tilde{\sigma})\frac{t(C)}{p},((t((B'\tilde{\sigma})\frac{I}{p})\&p)\vee(t((B'\tilde{\sigma})\frac{F}{p})\&\neg p))\frac{t(C)}{p})$. Finally, from the concluding formulas of $D_1\frac{t(C)}{p}$, D_2, and $D_3\frac{t(C)}{p}$, the formula $\hat{E}(t(B\sigma),t(B)t(\sigma))$ can be derived in one step by an instance of the rule $\hat{E}(p,q),\hat{E}(r,s),\hat{E}(p,s)\rightarrow\hat{E}(r,q)$.

To determine the total length of this derivation, first observe that each formula in it is bounded in length by some constant c times $\ell t(B\sigma)$. Then, $\ell^s D = 1+\ell^s D_1+\ell^s D_2+\ell^s D_3$, and by lemma 5.3.1.4.e., $\ell^s D_2+\ell^s D_3 \leq 2e_1(\ell B\sigma)^{4k}$. The number of levels of recursion to construct D_1 is bounded by the number of occurrences of atoms in B, so that the final bounds become $\ell^s D \leq \ell B(1+2e_1(\ell B\sigma)^{4k}) \leq b_1\cdot(\ell B\sigma)^{4k+1}$ and $\ell D \leq b_2\cdot(\ell B\sigma)^{6k+2}$.

■5.3.3.b.

This lemma can now be combined with the previous results to give the principal theorem.

THEOREM 5.3.3.c.

Any two s-Frege systems can p-simulate each other.

Now that it has been established that any two extended Frege systems p-simulate each other and any two s-Frege systems

p-simulate each other, to show that any s-Frege system p-simulates any extended Frege system it is sufficient to find one extended Frege system and one s-Frege system that p-simulates it.

LEMMA 5.3.3.d.

If κ is any adequate set of connectives, and $F_0=<\kappa,\mathcal{R}>$ is a Frege system, then there is a Frege system $F=<\kappa,\mathcal{R}\cup\mathcal{R}'>$ and there are constants d_1 and d_2 such that whenever $\vdash_{eF_0} A$ *via* D, there is a derivation D' such that

1) $\vdash_{sF} A$ *via* D',

2) $\ell^s D' \le d_1 \cdot \ell^s D$,

3) $\ell D' \le d_2 \cdot \ell^s D \cdot \ell D$, and

4) $d(D') \le \max(d(D),2)$.

Proof

The idea of this proof is to let $\mathcal{R}'$ expand to contain any rules that are useful to show how sF simulates eF_0. If $\vdash_{eF_0} A$ *via* D, this is equivalent to saying that there is a sequence of extensions $\Delta = \hat{E}(p_1,B_1),\ldots,\hat{E}(p_n,B_n)$ such that for $1\le i\le n$, $p_i \notin \{A\}\cup_{j=1}^{i}\{B_i\}$, $D=\Delta D_0$, and $\Delta \vdash_{F_0} A$ *via* D_0. Let $\hat{C}(p,q)$ and $\hat{I}(p,q)$ be formulas in the connectives κ such that $\hat{C}(p,q) \sim (p\&q)$ and $\hat{I}(p,q) \sim (p\supset q)$. Let

$$\Delta' = \hat{C}(\hat{C}(\ldots\hat{C}(\hat{C}(\hat{E}(p_1,B_1),\hat{E}(p_2,B_2)),\hat{E}(p_3,B_3))\ldots),\hat{E}(p_n,B_n)).$$

For each formula B in D_0, D' will contain the formula $\hat{I}(\Delta',B)$, and for each rule $R = A_1,\ldots,A_r\to G$ in $\mathcal{R}$, $\mathcal{R}'$ will contain the rules $\hat{I}(p,A_1),\ldots,\hat{I}(p,A_r) \to \hat{I}(p,G)$, so that formulas in this

section of D' are in one-to-one correspondence with formulas in D_0. What has been obtained so far is a derivation in F of $\hat{I}(\Delta',A)$ from $\hat{I}(\Delta',\hat{E}(p_1,B_1)),\ldots,\hat{I}(\Delta',\hat{E}(p_n,B_n))$ with the same number of steps as D_0. If this derivation is prefixed by derivations in F of $\hat{I}(\Delta',\hat{E}(p_1,B_1)),\ldots,\hat{I}(\Delta',\hat{E}(p_n,B_n))$, and followed by a derivation in sF of A from $\hat{I}(\Delta',A)$, then the desired derivation D' will be obtained.

To tackle the second problem first, let Δ'' be the formula such that $\Delta' = \hat{C}(\Delta'',\hat{E}(p_n,B_n))$ (*i.e.* Δ'' is the conjunction of the first n-1 extensions). Then, since $p_n \notin \{A\} \cup \bigcup_{j=1}^{n} \{B_j\}$, $\hat{I}(\Delta',A)\frac{\neg p_n}{p_n} = \hat{I}(\hat{C}(\Delta'',\hat{E}(p_n,B_n)),A)\frac{\neg p_n}{p_n} = \hat{I}(\hat{C}(\Delta'',\hat{E}(\neg p_n,B_n)),A)$. Now $\hat{E}(\neg p_n,B_n) \sim \neg\hat{E}(p_n,B_n)$, and $(p\&q)\supset C, (p\&\neg q)\supset C \models p\supset C$, so that the inference rule $\hat{I}(\hat{C}(p,\hat{E}(q,r)),s),\hat{I}(\hat{C}(p,\hat{E}(\neg q,r)),s) \rightarrow \hat{I}(p,s)$ is sound. Similarly, (for the case n=1), the rule $\hat{I}(\hat{E}(p,q),r),\hat{I}(\hat{E}(\neg p,q),r) \rightarrow r$ is sound. Thus, the formula $\hat{I}(\Delta'',A)$ is derived in two steps from the formula $\hat{I}(\Delta',A)$ in the system sF. Continuing in this way A is derived in $2n$ steps: n substitution steps, n-1 instances of the first rule, and one instance of the second.

The second problem is solved by the following lemma.

LEMMA 5.3.3.e.

If κ is an adequate set of connectives and $\hat{C}(p,q)$ and $\hat{I}(p,q)$ are formulas in the connectives κ such that $\hat{C}(p,q) \sim (p\&q)$ and $\hat{I}(p,q) \sim (p\supset q)$, then there is an inference

system $\mathcal{I}<\kappa,\tilde{\mathcal{R}}>$ and there are constants e_1 and e_2 such that if $\Delta_0=p_0$, and for $1\leq i\leq n$, $\Delta_i = C(\Delta_{i-1},p_i)$, then, for every $0\leq i\leq n$ there is a derivation D such that

1) $\vdash_{\mathcal{I}} \hat{I}(\Delta_n,p_i)$ *via* D,

2) $\ell^g D \leq e_1 \cdot n$,

3) $\ell D \leq e_2 \cdot n \cdot \ell\hat{I}(\Delta_n,p_i)$, and

4) $d(D)=1$.

Proof

Let $\hat{E}(p,q)$ be a formula in the connectives κ such that $\hat{E}(p,q) \sim (p\equiv q)$, as usual. The set of rules $\tilde{\mathcal{R}}$ will contain rules for manipulation of conjunctions and implications as necessary. To begin with, $\tilde{\mathcal{R}}$ contains the rules $\rightarrow\hat{I}(p,p)$ and $\hat{I}(\hat{C}(p,q),q)$, from which $\hat{I}(\Delta_i,p_i)$ can be derived in one step. The only other rule $\tilde{\mathcal{R}}$ needs is $\hat{I}(p,q) \rightarrow \hat{I}(\hat{C}(p,r),q)$. With this rule, $\hat{I}(\Delta_{j+1},p_i)$ can be derived from $\hat{I}(\Delta_j,p_i)$ in a single step. Thus, $n-i$ applications of this rule leads to the desired derivation D. ⊠5.3.3.e.

With lemma 5.3.3.e. completed, the proof of lemma 5.3.3.d. is now complete. ⊠5.3.3.d.

The groundwork has now been laid for the final theorem of this section on s-Frege systems.

THEOREM 5.3.3.f.

Any s-Frege system p-simulates any extended Frege system.

5.4. NATURAL DEDUCTION

Recall from subsection 4.2.2. the definition of natural deduction systems. In such systems, the deduction theorem (If $\Gamma, A \vdash B$, then $\Gamma \vdash A \supset B$.) can be used as a rule of inference. In most logic texts, the typical proof of this theorem for a Frege system shows how applications of the deduction theorem rule can be removed from a derivation, one at a time, until the resulting derivation has no applications of the rule. The problem with this method from a computational point of view is that each time an application of the deduction theorem rule is removed, the resulting derivation could be as long as about double the length of the original one. Thus, if there are many nested uses of the rule, the final derivation could be exponentially longer than the original. A better way of turning a natural deduction derivation into a Frege system derivation is given in the proof of lemma 5.4.d.

In order to facilitate the proof of that theorem, it will be useful to introduce some new tools, in the spirit of lemma 5.3.3.e. Let κ be any adequate set of connectives, and let $\hat{C}(p,q)$, $\hat{I}(p,q)$, and $\hat{E}(p,q)$ be the usual three formulas in the connectives κ such that $\hat{C}(p,q) \sim (p \& q)$, $\hat{I}(p,q) \sim (p \supset q)$, and $\hat{E}(p,q) \sim (p \equiv q)$. Note that since κ is adequate, there is a direct translation from $\{\&, \supset\}$ to κ, so that $\hat{C}(p,q)$ and $\hat{I}(p,q)$ each only need one occurrence of p and one occurrence of q. Let T be some fixed (short) tautology in the connectives κ, and define a sequence of conjunctive formulas $C_0, C_1, \ldots$ such that $C_0 = T$, and for $n \geq 1$, $C_n(p_1, \ldots, p_n) = \hat{C}(C_{n-1}(p_1, \ldots, p_{n-1}), p_n)$.

The following lemmas provide the means for manipulating these conjunctive formulas.

LEMMA 5.4.a.

There is an inference system $I_1=<\kappa,\mathcal{R}_1>$ and there are constants a_1, a_2, and a_3 such that if $n\geq 0$ and $\pi = <\pi_1,\ldots,\pi_n>$ is any permutation of $<1,\ldots,n>$, then there is a derivation D_n^π such that

1) $\vdash_{I_1} \hat{E}(C_n(p_1,\ldots,p_n),C_n(p_{\pi_1},\ldots,p_{\pi_n}))$ *via* D_n^π

2) $\ell^s D_n^\pi \leq a_1 \cdot n^2$,

3) $\ell D_n^\pi \leq a_2 \cdot n^2 \cdot \ell C_n(p_1,\ldots,p_n)$,

4) $d(D_n^\pi)=1$, and

5) no atom occurs more than a_3 times in any formula of D_n^π.

Proof

The idea of the proof is to let $\mathcal{R}_1$ contain rules that allow D_n^π to use an algorithm somewhat like the bubble sort to rearrange the atoms in a conjunction by a series of interchanges of adjacent atoms. The rules needed by $\mathcal{R}_1$ are (i) $\rightarrow\hat{E}(p,p)$, and (ii) $\hat{E}(p,q),\hat{E}(q,r) \rightarrow \hat{E}(p,r)$ for manipulating equivalences, and (iii) $\hat{E}(p,q) \rightarrow \hat{E}(\hat{C}(p,r),\hat{C}(q,r)$, and (iv) $\hat{E}(p,\hat{C}(q,r)) \rightarrow \hat{E}(\hat{C}(p,s),\hat{C}(\hat{C}(q,s),r))$ for manipulating conjunctions. It seems easiest to describe the derivation D_n^π by an example. It should be clear by analogy with the bubble sort algorithm how the derivation goes in general. For

notational simplicity in this example, assume that $\hat{C}(p,q)$ is $(p\&q)$ and $\hat{E}(p,q)$ is $(p\equiv q)$. Then, $C_n(p_1,\ldots,p_n)$ is $((\ldots((\top\&p_1)\&p_2)\&\ldots)\&p_n)$. The derivation of $\hat{E}(C_4(p,q,r,s),C_4(s,q,p,r))$ then goes as shown in figure 5.4.i.

The algorithm proceeds by a series of at most n-1 "passes" through $C_n(\vec{p})$, where on the i^{th} pass, $p_{\pi_{n+1-i}}$ is moved to its proper position by interchanges with adjacent atoms. Each pass takes at most n applications of rules (i),(iii), and (iv), and rule (ii) is used once at the end of each pass except the first. In the example, lines 1 and 2 are the first pass (r is put in place), lines 3, 4, 5, 6, and 7 are the second pass (p is put in place), and lines 8, 9, 10, 11, and 12 are the last pass. The total number of steps in the derivation is therefore no greater that $(n\text{-}1)(n\text{+}1)\text{-}1 = n^2\text{-}2$. Also, note that since $\hat{E}(p,q)$ is a fixed formula, the length of each formula in the derivation is bounded by some constant times $\ell C_n(p_1,\ldots,p_n)$, and no atom occurs more than some constant number of times in any formula in D_n^{π}. ⊠5.4.a.

Let $J=\langle j_1,\ldots,j_l\rangle$ and $K=\langle k_1,\ldots,k_m\rangle$ be two subsequences of $\langle 1,\ldots,n\rangle$, such that $J\cup K = \{1,\ldots,n\}$, and let K' be any permutation of K. Let $pJ=\langle p_{j_1},\ldots,p_{j_l}\rangle$, and similarly for pK and pK'. Also, let $\vec{p}=\langle p_1,\ldots,p_n\rangle$. The next lemma shows how a conjunction $C(\vec{p})$ can be broken up into the two subsequences $C(pJ)$ and $C(pK)$.

line number	formula	rule	source lines
1	$(((T\&p)\&q)\&r)\equiv(((T\&p)\&q)\&r)$	(i)	
2	$((((T\&p)\&q)\&r)\&s)\equiv((((T\&p)\&q)\&s)\&r)$	(iv)	1
3	$(T\&p)\equiv(T\&p)$	(i)	
4	$((T\&p)\&q)\equiv((T\&q)\&p)$	(iv)	3
5	$(((T\&p)\&q)\&s)\equiv(((T\&q)\&s)\&p)$	(iv)	4
6	$((((T\&p)\&q)\&s)\&r)\equiv((((T\&q)\&s)\&p)\&r)$	(iii)	5
7	$((((T\&p)\&q)\&r)\&s)\equiv((((T\&q)\&s)\&p)\&r)$	(ii)	2,6
8	$(T\&q)\equiv(T\&q)$	(i)	
9	$((T\&q)\&s)\equiv((T\&s)\&q)$	(iv)	8
10	$(((T\&q)\&s)\&p)\equiv(((T\&s)\&q)\&p)$	(iii)	9
11	$((((T\&q)\&s)\&p)\&r)\equiv((((T\&s)\&q)\&p)\&r)$	(iii)	10
12	$((((T\&p)\&q)\&r)\&s)\equiv((((T\&s)\&q)\&p)\&r)$	(ii)	7,11

FIGURE 5.4.i.

Derivation of $\hat{E}(C_4(p,q,r,s),C_4(s,q,p,r))$

LEMMA 5.4.b.

There is an inference system $I_2=\langle \kappa, \mathcal{R}_2\rangle$ and there are constants b_1, b_2, and b_3 such that whenever J and K are subsequences of $\langle 1,\ldots,n\rangle$ for which $J\cup K = \langle 1,\ldots,n\rangle$ there is a derivation D_n^{JK} such that

1) $\vdash_{I_2} \hat{E}(C_n(\vec{p}),\hat{C}(C_\ell(pJ),C_m(pK)))$ *via* D_n^{JK}

2) $\ell^s D_n^{JK} \le b_1 \cdot n$, (in fact, $\ell^s D_n^{JK} = n+1$)

3) $\ell D_n^{JK} \le b_2 \cdot n \cdot \ell C(\vec{p})$,

4) $d(D_n^{JK})=1$, and

5) no atom accurs more than b_3 times in any formula of D_n^{JK}.

Proof

The rules of $\mathcal{R}_2$ are (i) $\to \hat{E}(T,\hat{C}(T,T))$, (ii) $\hat{E}(p,\hat{C}(q,r)) \to \hat{E}(\hat{C}(p,s),\hat{C}(\hat{C}(q,s),r)$ (for the case $p_i \in J$, $p_i \notin K$), (iii) $\hat{E}(p,\hat{C}(q,r)) \to \hat{E}(\hat{C}(p,s),\hat{C}(q,\hat{C}(r,s)))$ (for the case $p_i \notin J$, $p_i \in K$), and (iv) $\hat{E}(p,\hat{C}(q,r)) \to \hat{E}(\hat{C}(p,s),\hat{C}(\hat{C}(q,s),\hat{C}(r,s)))$ (for the case $p_i \in J$, $p_i \in K$). A small example, with $J=\langle p,q,s\rangle$ and $K=\langle q,r\rangle$ will illustrate the construction of D_n^{JK}.

line number	formula	rule	source line
1	$T \equiv (T \& T)$	(i)	
2	$(T \& p) \equiv ((T \& p) \& T)$	(ii)	1
3	$((T \& p) \& q) \equiv (((T \& p) \& q) \& (T \& q))$	(iv)	2
4	$(((T \& p) \& q) \& r) \equiv$ $(((T \& p) \& q) \& ((T \& q) \& r))$	(iii)	3
5	$((((T \& p) \& q) \& r) \& s) \equiv$ $((((T \& p) \& q) \& s) \& ((T \& q) \& r))$	(ii)	4

This example should make it clear that the lemma is true. ⊠5.4.b.

Lemmas 5.4.a. and 5.4.b. can be combined to give the following result.

LEMMA 5.4.c.

If κ, $\hat{C}$, $\hat{E}$, T, and C_j are as defined above, then there is an inference system $I_3 = \langle \kappa, \mathcal{R}_3 \rangle$, and there are constants c_1, c_2, and c_3 such that whenever J, K, and K' are as defined above, there is a derivation $D_n^{JK'}$ such that

1) $\vdash_{I_3} \hat{E}(C_n(\vec{p}), \hat{C}(C_l(pJ), C_m(pK'))) \; via \; D_n^{JK'}$,

2) $\ell^B D_n^{JK'} \leq c_1 \cdot (n+m^2)$,

3) $\ell D_n^{JK'} \leq c_2 \cdot (n+m^2) \cdot \ell C_n(\vec{p})$,

4) $d(D_n^{JK'}) = 1$, and

5) no atom accurs more than c_3 times in any formula of $D_n^{JK'}$.

Proof

Let $\mathcal{K}_3 = \mathcal{R}_1 \cup \mathcal{R}_2 \cup \{\hat{E}(p,q), \hat{E}(r, \hat{C}(s,p)) \rightarrow \hat{E}(r, \hat{C}(s,q))\}$, and appeal to lemmas 5.4.a. and 5.4.b. ⊠5.4.c.

The machinery has now been established so that the main result of this section can be proven.

LEMMA 5.4.d.

If $N=\langle \kappa, \mathcal{R}_0 \rangle$ is a natural deduction system, then there is a Frege system $F=\langle \kappa, \mathcal{R} \rangle$ and there are constants d_1 and d_2 such that whenever $\vdash A_1,\ldots,\vdash A_k \vdash_N \vdash B$ *via* D_0, there is a derivation D such that

1) $A_1,\ldots,A_k \vdash_F B$ *via* D,

2) $\ell^{\mathcal{S}} D \leq d_1 \cdot \ell^{\mathcal{S}} D_0 \cdot \ell D_0$, and

3) $\ell D \leq d_2 \cdot \ell^{\mathcal{S}} D_0 \cdot (\ell D_0)^2 \cdot d(D_0)$.

Proof

The idea of this proof is similar to the proofs of lemmas 5.3.1.2.a. and 5.3.1.4.g., except that here each line $\Gamma \vdash G$ in D_0 is translated into the formula $\hat{I}(C_n(\Gamma'),G)$ in D, where Γ' is the permutation that places the formulas of Γ into lexicographic order. For each rule $R=\Gamma_1 \vdash B_1,\ldots,\Gamma_j \vdash B_j \rightarrow \Lambda \vdash H$ in $\mathcal{R}_0$, $\mathcal{R}$ contains the rule $\hat{I}(\hat{C}(p,C(\Gamma_1')),B_1),\ldots,\hat{I}(\hat{C}(p,C(\Gamma_j')),B_j) \rightarrow \hat{I}(\hat{C}(p,C(\Lambda')),H)$. The atom p is the one for which the environment $C(\Delta')$ is substituted when simulating an application of rule R with environment Δ and substitution σ. In addition, $\mathcal{R}$ contains all of the rules of $\mathcal{R}_3$ from lemma 5.4.c. and the rules $p \rightarrow \hat{I}(T,p)$; $\hat{I}(T,p) \rightarrow p$; $\hat{I}(p,q),\hat{E}(p,r) \rightarrow \hat{I}(r,q)$; and $\hat{I}(p,q),\hat{E}(r,p) \rightarrow \hat{I}(r,q)$. The first of these rules is used for deriving $\hat{I}(C_0,A_i)$ from A_i, and the second derives B from $\hat{I}(C_0,B)$. The other two are used in conjunction with the derivation of lemma 5.4.c. to obtain derivations D^+ and D^- of

an instance of $\hat{I}(\hat{C}(p,C(\Gamma_i')),B_i)$ and $\hat{I}(C(\Gamma'),G)$ from one another in the case where $\Gamma\vdash G = \Delta\Gamma_i\sigma\vdash B_i\sigma$ is an instance of $\Gamma_i\vdash B_i$. The particular instance of $\hat{I}(\hat{C}(p,C(\Gamma_i')),B_i)$ of interest in this case is $\hat{I}(\hat{C}(p,C(\Gamma_i')),B_i)\sigma\frac{C(\Delta')}{p} = \hat{I}(\hat{C}(C(\Delta'),C(\Gamma_i'\sigma)),B_i\sigma)$. Since $\Gamma=\Delta\Gamma_i\sigma$, Δ' is a subsequence of Γ' and $\Gamma_i'\sigma\cup\Delta = \Gamma$, so that under the substitution $\sigma' = \frac{\Delta'\Gamma_i'\sigma}{pJpK'}$, lemma 5.4.c. gives a derivation $D_n^{JK'}\sigma'$ of $\hat{E}(C(\Gamma'),\hat{C}(C(\Delta'),C(\Gamma_i'\sigma)))$. Note that in this case m is the number of formulas in Γ_i, and since Γ_i is fixed by rule R, which in turn is fixed by N, m is a constant that depends only on N. Also, note that n is the number of formulas in Γ, which is bounded above by $\ell(\Gamma\vdash G)$. This derivation $D_n^{JK'}\sigma'$ along with a single application of an instance of the appropriate one of the above two rules gives the desired derivation D^+ or D^-.

Finally, to construct the derivation D, each line $\Gamma\vdash G$ which is derived in D_0 by the instance $\Delta R\sigma$ of rule R, D_0 consists of j derivations D_i^+ corresponding to the hypothesis lines of R, followed by one application of the rule of F that simulates R, followed by a derivation D^- to put the conclusion into the form $\hat{I}(C(\Gamma'),G)$. This sub-derivation has a number of steps bounded by some constant c times $\ell(\Gamma\vdash G)$ (and $\ell(\Gamma\vdash G)\leq\ell D_0$). Therefore, there are constants d_1 and d_2 such that $\ell^S D \leq d_1\cdot\ell^S D_0\cdot\ell D_0$ and $\ell D \leq d_2\cdot\ell^S D_0\cdot(\ell D_0)^2\cdot d(D_0)$.

Finally, it must be noted that since N is implicationally complete, F is implicationally complete, and thus really is a Frege system. ☒5.4.d.

This theorem leads immediately to the following corollaries.

THEOREM 5.4.e.

If N is a natural deduction system, then there is a Frege system that p-simulates N.

COROLLARY 5.4.f.

Every Frege system p-simulates every natural deduction system.

Having established that natural deduction systems are no more powerful (to within a polynomial) than Frege systems, it would be reassuring to know that they are no less powerful.

LEMMA 5.4.g.

If κ, $\hat{C}$, $\hat{\Gamma}$, and C_n are defined as usual and $N=\langle\kappa,\mathcal{R}_0\rangle$ is a natural deduction system, then there is a Frege system $F=\langle\kappa,\mathcal{R}\rangle$ and there are constants g_1 and g_2 such that whenever $\hat{\Gamma}(C(\Gamma_1),A_1),\ldots,\hat{\Gamma}(C(\Gamma_k),A_k) \vdash_F \hat{\Gamma}(C(\Delta),B)$ *via* D, there is a derivation D_0 such that

1) $\Gamma_1\vdash A_1,\ldots,\Gamma_k\vdash A_k \vdash_N \Delta\vdash B$ *via* D_0,

2) $\ell^s D_0 \leq g_1\cdot\ell^s D\cdot\ell D$, and

3) $\ell D_0 \leq g_2\cdot\ell^s D\cdot(\ell D)^2\cdot d(D)$.

Proof

First, observe that theorem 5.4.d. guarantees the existence of some Frege system F with the connectives κ. The

proof of this theorem is very similar to the proof of theorem 5.3.1.2.a., except that here D_0 uses the line $\vdash G$ to simulate the formula G in D, Since N is implicationally complete, there is a derivation D^i which simulates each rule R^i in $\mathcal{R}$. Everything proceeds as with the proof of theorem 5.3.1.2.a., except that D_0 also needs to include derivations of $\vdash \hat{I}(\mathbb{C}(\Gamma_i),A_i)$ from $\Gamma_i \vdash A_i$ and of $\Delta \vdash B$ from $\vdash \hat{I}(\mathbb{C}(\Delta),B)$. To see how these derivations are constructed, consider the following derivation D' of $\Delta \vdash B$ from $\vdash \hat{I}(\mathbb{C}(\Delta),B)$. Let $\tilde{D}$ be a derivation of $p \vdash q$ from $\vdash \hat{I}(p,q)$, let $\hat{D}$ be a derivation of $\vdash p$ from $T \vdash p$, and let $\ddot{D}$ be a derivation of $p,q \vdash r$ from $\hat{C}(p,q) \vdash r$. Then, if Δ contains n formulas, D' can be constructed from $\tilde{D}\frac{\mathbb{C}(\Delta)B}{p\ \ q}$ followed by n instances of $\ddot{D}$ (Remember that an instance of a natural deduction derivation can include the specification of environment formulas), followed by $\Delta\hat{D}\frac{B}{p}$ (*i.e.* each line of $\hat{D}$ has Δ added to its environment). The number of lines of D' is proportional to n, and the length of each line is proportional to $\ell(\Delta \vdash B)$. Derivations of $\vdash \hat{I}(\mathbb{C}(\Gamma_i),A_i)$ from $\Gamma_i \vdash A_i$ are similar, except that $\tilde{D}$, $\hat{D}$, and $\ddot{D}$ are now changed so that their hypotheses and conclusions are interchanged. This completes the proof of theorem 5.4.g. ⊠5.4.g.

This theorem has as corollaries the last three results of this section.

THEOREM 5.4.h.

For every natural deduction system there is a Frege system that it p-simulates.

COROLLARY 5.4.i.

Every natural deduction system p-simulates every Frege system.

COROLLARY 5.4.j.

Any two natural deduction systems p-simulate each other.

The concept of s-natural deduction system was introduced in section 4.2.2., and a definition for extended natural deduction systems could be devised as well, but it is clear that the techniques of this section and sections 5.3.3. and 5.3.2. could be used to show that they lie in the same p-simulation equivalence classes as s-Frege systems and extended Frege systems, respectively.

5.5. SEQUENT AND TABLEAU SYSTEMS

Various types of tableau-type proof systems were introduced in section 4.2.3., including sequent systems, several variations of Gentzen systems, and Smullyan's analytic tableaux. This section presents some results concerning the complexities of these systems. Subsection 5.5.1. treats sequent systems, showing that they are p-simulation equivalent to Frege systems. Although Gentzen systems with cut are sequent systems, cut-free Gentzen systems are not implicationally complete, and thus are not sequent systems. These systems are discussed briefly in subsection 5.5.2., although further simulation results concerning them are deferred until subsection 5.6.4. Subsection 5.5.3. covers Smullyan's analytic tableaux, giving a simulation result and a lower bound result.

5.5.1. Sequent Systems *vs* Frege Systems

Recall from paragraph 4.2.3.1. the definition of sequent systems. Since the sequent $\Delta \vdash \Gamma$ is logically equivalent to the formula $\hat{\Upsilon}(\mathbb{C}(\Delta),\mathbb{D}(\Gamma))$ (where $\mathbb{D}$ is the analog of $\mathbb{C}$ for disjunctions), it is clear that the techniques of section 5.4. can be extended to show that sequent systems and Frege systems p-simulate one another. Thus, to avoid a lengthy and repetitious development, the following theorems are stated without proof.

THEOREM 5.5.1.a.

Every Frege system p-simulates every sequent system.

THEOREM 5.5.1.b.

Every sequent system p-simulates every Frege system.

COROLLARY 5.5.1.c.

Any two sequent systems p-simulate each other.

COROLLARY 5.5.1.d.

Any two systems from the class {Frege systems} $\cup$ {natural deduction systems} $\cup$ {sequent systems} p-simulate each other.

As with Frege systems and natural deduction systems, the concepts of extended sequent systems and sequent systems with substitution could be introduced, but it is clear that the techniques developed in this thesis could be used to show that they would be p-simulation equivalent to extended Frege systems and s-Frege systems, respectively.

5.5.2. Cut-free Systems

Recall from paragraph 4.2.3.1. the definitions of basic Gentzen system, Gentzen system with thinning, and Gentzen system with cut. For each adequate set of connectives κ, there is one system of each of these three types. Let G be the basic Gentzen system for the connectives κ, and let tG and cG be the corresponding systems with thinning and cut, respectively. Since any derivation of system G is also a derivation of systems tG and cG, these systems p-simulate G in a trivial way. Nearly as simply, cG p-simulates tG. Each application of the thinning rules is replaced by an instance of one of the cG derivations simulating that rule. For example, if $\Delta \vdash A,\Gamma$ is inferred from $\Delta \vdash \Gamma$ by the thinning introduction rule and Δ is non-empty, so that $\Delta=\Delta',B$, then $\Delta \vdash A,\Gamma$ is derived from $\Delta \vdash \Gamma$ by the following derivation.

1. $\Delta',B \vdash \Gamma$ (hypothesis)
2. $\Delta',B \vdash A,\Gamma,B$ (axiom)
3. $\Delta',B \vdash A,\Gamma$ (cut) (1,2)

Similar two-line derivations exist for thinning elimination and for the cases where Δ is empty (but Γ must then be non-empty).

Combining these trivial results with corollary 5.5.1.d. and the fact that cG is a sequent system leads to the following theorems.

THEOREM 5.5.2.a.

Every Frege system, natural deduction system, and sequent system p-simulates every basic Gentzen system and every Gentzen system with thinning.

THEOREM 5.5.2.b.

Every Gentzen system with thinning p-simulates the basic Gentzen system for the same connectives.

The existence of simulations in the reverse direction is doubtful. But a consequence of the definition of simulation is that if any verification system F is polynomial-bounded, then F simulates every verification system (for an infinite set). Thus, a proof that F_1 does not simulate F_2 implies that F_1 is not polynomial-bounded. Therefore, a proof that no such reverse-direction simulations exist would imply that basic Gentzen systems are not polynomial-bounded, a fact that (although conjectured) is not now known to be true.

The relationship between basic Gentzen systems (or Gentzen systems with thinning) with different sets of connectives is also unclear. If $\kappa_2 \subseteq \kappa_1$ and G_1 is the basic Gentzen system (or Gentzen system with thinning) for κ_1, and G_2 is the corresponding system for κ_2, then G_2 trivially p-simulates G_1. This is true because every rule of G_1 that applies to formulas in the connectives κ_2 is also a rule of G_2. Since G_1 is analytic, no G_1-derivation of a formula in κ_2 can contain any formula (or use any rule) with connectives not in κ_2.

If $\kappa_2 \not\subseteq \kappa_1$ but there is a direct translation $*$ from κ_1 to κ_2, then the usual techniques can be used to show that for any derivation of A in system G_1 there is a not-too-much-longer derivation of $t(A)$ in system G_2. But it is not clear how this helps to find short derivations of formulas B in the connectives κ_2 that are not the image under t of any formula A. The technique from paragraph 5.3.1.2. of deriving A from $t_1(t_2(A))$ will not work here, because cut-free Gentzen systems are analytic. For the case where there is only an indirect translation from κ_1 to κ_2, the relationship between G_1 and G_2 is totally unclear, because the analyticity of cut-free Gentzen systems renders all of the techniques of paragraph 5.3.1.4. useless. Thus, the only comparisons among cut-free Gentzen systems that can be stated with certainty are the following two theorems.

THEOREM 5.5.2.c.

If G is the basic Gentzen system for the connectives κ, then G p-simulates the basic Gentzen system for any $\kappa' \supseteq \kappa$.

THEOREM 5.5.2.d.

If G is the Gentzen system with thinning for the connectives κ, then G p-simulates the Gentzen system with thinning for any $\kappa' \supseteq \kappa$.

5.5.3. Analytic Tableaux

Smullyan's analytic tableaux were described in paragraph 4.2.3.2. Technically speaking, each adequate set of connectives gives rise to a different system of analytic tableaux. If S is the system of tableaux for κ, and G is the

basic Gentzen system for κ, then the construction described in paragraph 4.2.3.2. gives a way of turning a tableau for A into a basic Gentzen derivation of the sequent $A\vdash$. This method could be used equally well to obtain a derivation of $\vdash A$ from a tableau for $\neg A$. Since basic Gentzen systems are analytic, no sequent in any derivation of A can be longer than ℓA. Thus, the increase in size of a string representation of the tableau tree caused by replacing the formula at each node by a sequent is not too great, so that the following theorem is obtained.

THEOREM 5.5.3.a.

The basic Gentzen system for any adequate set of connectives $\kappa \supseteq \{\neg\}$ p-simulates the system of analytic tableaux for κ.

It is not clear whether tableaux simulate basic Gentzen systems, because basic Gentzen derivations need not be trees. Also, the relationship between tableau systems for different sets of connectives is similar to the situation for basic Gentzen systems.

But certain simulations involving analytic tableaux are known not to exist, because Cook has shown that the system of analytic tableaux for sets of clauses (which trivially p-simulates the system of tableaux for $\{\neg,\vee,\&\}$ is not polynomial-bounded [Cook 1973] [Cook & Reckhow 1974]. In his proof Cook constructs an infinite family of inconsistent sets of clauses $\{T_m\}$.

The sets of clauses T_m could be defined inductively as follows. The atoms of T_m are taken from the set $\{p_i \mid i \in \{0,1\}^*\}$. For $a \in \{0,1\}$, let S^a be the set of clauses obtained from the set of clauses S by replacing each atom p_i by the atom p_{ai}. For any literal ξ, let ξS be the set of clauses obtained from the set of clauses S by adding ξ to each clause. The inductive definition of T_m can now be stated as:

$$T_0 = \{\Box\}, \text{ and for } m \geq 0$$
$$T_{m+1} = pT_m^1 \cup \bar{p}T_m^0$$

Thus,

$T_1 = \{p, \bar{p}\}$

$T_2 = \{pp_1, p\bar{p}_1, \bar{p}p_0, \bar{p}\bar{p}_0\}$

$T_3 = \{pp_1p_{11}, pp_1\bar{p}_{11}, p\bar{p}_1p_{10}, p\bar{p}_1\bar{p}_{10}, \bar{p}p_0p_{01}, \bar{p}p_0\bar{p}_{01}, \bar{p}\bar{p}_0p_{00}, \bar{p}\bar{p}_0\bar{p}_{00}\}$

etc.

Note that T_m is a set of 2^m clauses, each of which contains m literals. The total number of distinct atoms in T_m is 2^m-1, and the total number of occurrences of literals in T_m is $m \cdot 2^m$. Thus there is a constant d such that $m \cdot 2^m \leq \ell T_m \leq d \cdot m^2 \cdot 2^m$.

In his proof Cook shows that there is a constant $c>0$ such that for every $m>0$, the smallest analytic tableau

for T_m has at least $2^{2^{cm}}$ nodes. The proof can easily be extended, by converting $\{T_m\}$ into formulas in $\{\neg,\vee,\&\}$ and using direct translations, to show the same lower bound for the system of analytic tableaux for any adequate set of connectives κ.

THEOREM 5.5.3.b. (Cook)

No system of analytic tableaux is polynomial-bounded.

Cook observed that for each m there is a resolution derivation of $\square$ from T_m of length proportional to ℓT_m, but a much stronger statement can be made. Those resolution derivations not only are regular, but they are trees and they can also be generated by the Davis-Putnam procedure without making use of the subsumption rule. The formulas $\{T_m\}$ also have semantic trees of size proportional to ℓT_m. This proves the following theorem.

THEOREM 5.5.3.c. (Cook)

The system of analytic tableaux for sets of clauses cannot directly simulate either the Davis-Putnam procedure without subsumption, the method of semantic trees, or tree resolution.

Somewhat less obvious is that if $\{T_m\}$ are converted into formulas in the right way, then there are derivations of $T_m\vdash$ in the appropriate Gentzen system with thinning of length proportional to $(\ell T_m)^3$. To see how these derivations are constructed, consider the case where $\kappa=\{\neg,\vee,\&\}$ and $m=3$. The formula for T_3 is a renaming of

$((pq)r)\&((pq)\bar{r})\&((p\bar{q})s)\&((p\bar{q})\bar{s})\&((\bar{p}t)u)\&((\bar{p}t)\bar{u})\&((\bar{p}\bar{t})v)\&((\bar{p}\bar{t})\bar{v})$, where AB is $A \vee B$, ξ is $\neg\xi$, and the parenthesization of the conjunction is irrelevant. The sequent $T_3\vdash$ can be derived in $7=2^3-1$ $\&$-elimination steps from $\Gamma\vdash$, where

$$\Gamma=\{((pq)r),((pq)\bar{r}),((p\bar{q})s),((p\bar{q})\bar{s}),((\bar{p}t)u),((\bar{p}t)\bar{u}),((\bar{p}\bar{t})v),((\bar{p}\bar{t})\bar{v})\}.$$

The derivation of $\Gamma\vdash$ is shown in figure 5.5.3.i. The sequent $\Delta\vdash$ is represented by Δ, and sequents with $\not\times$ above them are derived from an axiom by one application of the $\neg$-elimination rule. Sequents connected to their parent by a line labelled T are derived by thinning elimination, and all other sequents are derived by $\vee$-elimination. Observe that each $\vee$ in T_m is expanded by the $\vee$-elimination rule at most once, so that the number of steps in this derivation is proportional to ℓT_m. Notice also that no sequent is longer than $(\ell T_m)^2$, so that the total length of the derivation is proportional to $(\ell T_m)^3$.

The same methods can be used with any other adequate set of connectives. This completes a sketch of the proof of the following theorem.

THEOREM 5.5.3.d.

For any adequate set of connectives κ, the system of analytic tableaux for κ cannot directly simulate the Gentzen system for κ with thinning.

Although analytic tableaux can be exponentially worse than other systems for certain examples, the next theorem shows that for other examples another system is exponentially worse than analytic tableaux.

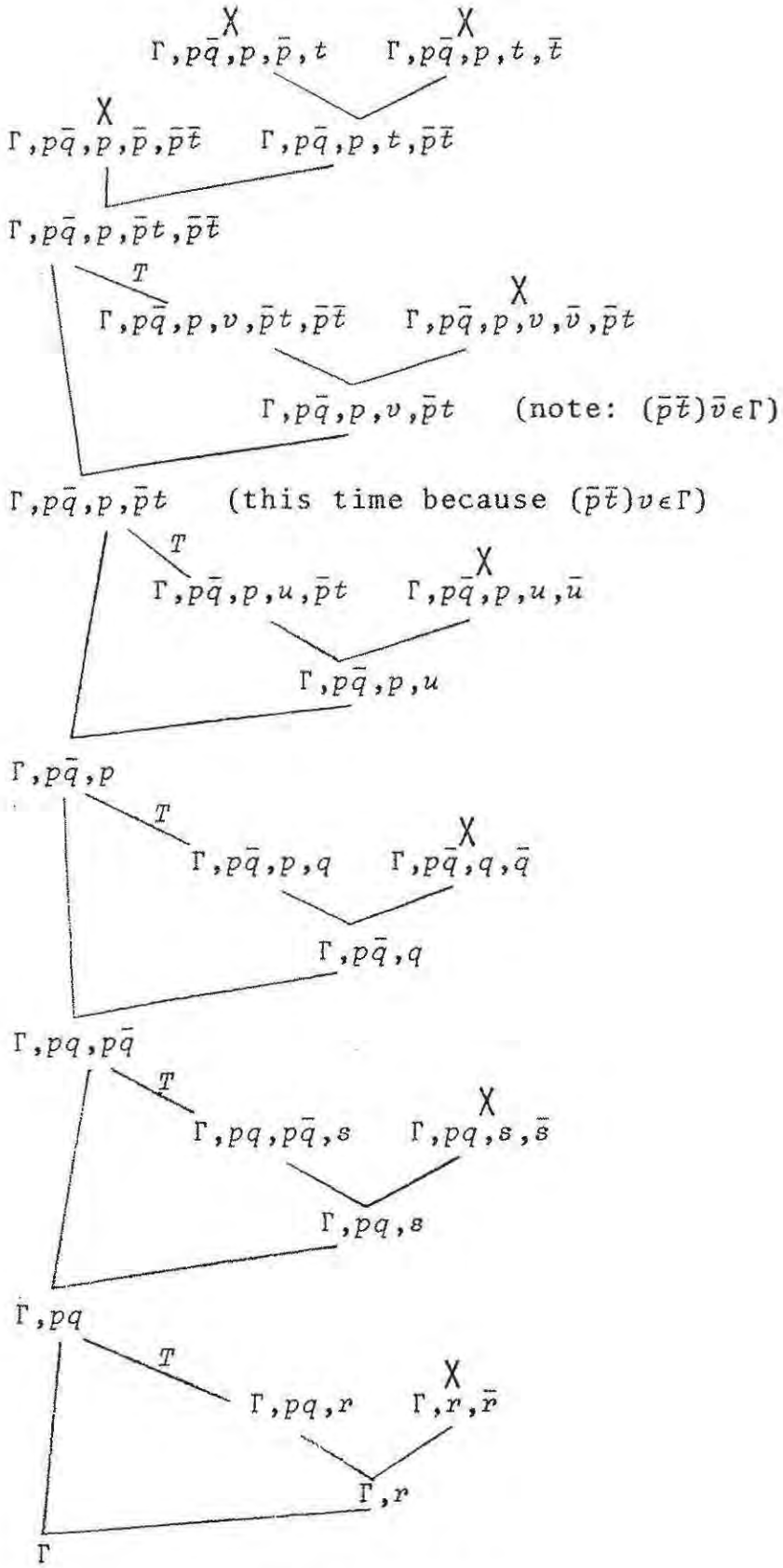


FIGURE 5.5.3.i.

Derivation of T_3 in Gentzen System with Thinning

THEOREM 5.5.3.e.

Full truth tables cannot directly simulate analytic tableaux.

Proof

For the case where $\kappa=\{\neg,\&\}$, consider the formula $((p\&\neg p)\&A)$, where ℓA is arbitrarily large and A contains about $\frac{\ell A}{log(\ell A)}$ different atoms. The full truth table for this formula has size exceeding $2^{\frac{\ell A}{log(\ell A)}}$, but there is an analytic tableau of size proportional to ℓA. Other sets of connectives can be handled equally easily. ⊠5.5.3.e.

Finally, it should be noted that although theorem 5.5.3.b. was derived independently, it does follow as a corollary of the results to come in subsection 5.6.1.

5.6. RESOLUTION

This section includes an assortment of simulation and lower bound results concerning the various forms of resolution and related systems. Subsection 5.6.1. discusses tree resolution, and shows how it relates to semantic trees, analytic tableaux, and Galil's enumeration trees. Regular resolution and the Davis-Putnam procedure are discussed in subsection 5.6.2., and subsection 5.6.3. covers the relationship between extended resolution and extended Frege systems. In the final subsection of this section, the cloudy area of systems just slightly more powerful than systems known not to be polynomial-bounded is investigated. Here the relationship between resolution and various cut-free Gentzen systems is studied in detail.

Throughout this section, the measure of the size of a resolution derivation will be taken to be the number of resolvents. This is reasonable, since by the resolution rule, no clause can have more than one literal more than its descendants. Therefore, if n is the number of resolvents in a resolution derivation of clause C, the total length of that derivation cannot be greater than some constant times $n \cdot (n+\ell C) \cdot log(n+\ell C)$, so that to within a polynomial, n is a good measure of the length of that derivation.

5.6.1. Tree Resolution

In this section the system of tree resolution is analyzed and compared with other systems. Paragraph 5.6.1.1. presents Tseitin's lower bound for tree resolution, and gives a negative result about simulation. Tree resolution is compared to the method of semantic trees in paragraph 5.6.1.2. and to the system of analytic tableaux for sets of clauses in paragraph 5.6.1.3. Paragraph 5.6.1.4. introduces Galil's generalization of semantic trees, called enumeration trees.

5.6.1.1. Tseitin's Lower Bound for Tree Resolution

A tree resolution derivation is a resolution derivation that is a tree. Any resolution derivation can be made into a tree resolution derivation by duplicating the sub-derivations of all clauses that are used more than once. This process can cause an exponential increase in the size of the derivation if the derivation contains long chains of clauses that are used more than once. But it is not at all clear how to show, given a set of clauses, that all short resolution derivations of $\square$ from that set of clauses must contain such long chains.

Tseitin has shown that sets of clauses exist where such long chains are necessary. One of Tseitin's lemmas will be useful later on in this subsection, so it will be given first. [Tseitin 1968]

LEMMA 5.6.1.1.a. (Tseitin)

Any tree resolution derivation of □ from the set S of clauses can be converted into a regular tree resolution derivation of □ from S with no more steps than the original.

COROLLARY 5.6.1.1.b.

Tree resolution and regular tree resolution p-simulate each other.

Tseitin's main theorem concerning tree resolution is the following.

THEOREM 5.6.1.1.c. (Tseitin)

There is an infinite family of inconsistent sets of clauses $\{S_i\}$ and there is a constant $c>0$ such that if n is the length of the shortest regular resolution derivation of □ from S_i and m is the length of the shortest extended tree resolution derivation (*i.e.* extended resolution derivation that is a tree) of □ from S_i, then every tree resolution derivation of □ from S_i contains more than $2^{c(log\ \max(m,n))^2}$ clauses.

In fact, careful analysis of Tseitin's proof shows that the shortest regular resolution derivation of □ from S_i can be generated by the Davis-Putnam procedure without subsumption. This leads to the following corollary.

COROLLARY 5.6.1.1.d.

Tree resolution cannot directly simulate extended tree resolution, regular resolution, or the Davis-Putnam procedure, with or without subsumption.

COROLLARY 5.6.1.1.e.

Tree resolution is not polynomial-bounded.

By lemma 5.1.2.a., any system that tree resolution can simulate must also not be polynomial-bounded.

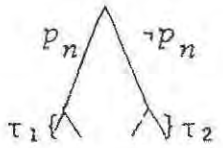
5.6.1.2. Tree Resolution *vs* Semantic Trees

Since there is a formula S' in $\{\neg,\vee,\&\}$ that is logically equivalent to any set of clauses S, any semantic tree for S' is also a semantic tree for S. Also, since $\neg S'$ is valid if and only if S' is inconsistent, any semantic tree that shows $\neg S'$ to be a tautology shows S' and S to be inconsistent. This proves the following theorem.

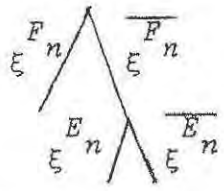
THEOREM 5.6.1.2.a.

The system of semantic trees for inconsistent sets of clauses p-simulates the system of semantic trees for inconsistent formulas and the system of semantic trees for tautologies.

The converse of this theorem may not be true. Consider the formula $F_n = (p_1 \equiv \ldots \equiv p_n) \vee \neg (p_1 \equiv \ldots \equiv p_n)$. It is not hard to see that every semantic tree for F_n must have 2^n leaves. By symmetry it may be assumed without loss of generality that the root of a semantic tree τ for F_n looks like



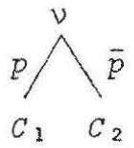
But now τ_1 must be a semantic tree for $(p_1 \equiv \ldots p_{n-1} \equiv T) \vee \neg(p_1 \equiv \ldots \equiv p_{n-1} \equiv T) \sim F_{n-1}$, and τ_2 must be a semantic tree for $(p_1 \equiv \ldots \equiv p_{n-1} \equiv F) \vee \neg(p_1 \equiv \ldots \equiv p_{n-1} \equiv F) \sim \neg(p_1 \equiv \ldots \equiv p_{n-1}) \vee \neg\neg(p_1 \equiv \ldots \equiv p_{n-1}) \sim F_{n-1}$. Thus, if the system of semantic trees for tautologies simulates the system of semantic trees for inconsistent sets of clauses, the function g from that simulation must take F_n into some set of clauses whose smallest semantic tree has at least $2^{c \cdot n}$ leaves, for some constant $c>0$. The obvious method of mapping F_n into $def(F_n) \cup \{\xi^{\overline{F_n}}\}$ will not work, because, if $E_n = (p_1 \equiv \ldots \equiv p_n)$, then $def(F_n)$ contains the clauses $\xi^{\overline{F_n}} \xi^{E_n} \xi^{\overline{E_n}}$, $\xi^{F_n} \xi^{\overline{E_n}}$, and $\xi^{F_n} \xi^{E_n}$, so that the semantic tree



is closed for $def(F_n) \cup \{\xi^{\overline{F_n}}\}$. Changing $def(F_n)$, so that each occurrence of a subformula has a different literal associated with it, might overcome this problem, but it is hard to imagine how any argument about simulation would not apply equally well to the usual $def(F_n)$.

To see how tree resolution relates to the system of semantic trees for sets of clauses, consider an inconsistent

set S of clauses and a semantic tree τ for S. Prune τ (by removing pairs of leaves) until no tree contained in τ is closed for S. Label each leaf τ with one of the clauses falsified by that branch. Consider the pair of leaves



and note that since τ has been pruned, it must be true that $p \epsilon C_1$ and $\bar{p} \epsilon C_2$. Since the same truth assignment that falsifies C_1 falsifies C_2 when the value of p is changed, C_1 and C_2 have a resolvent C_3. Label node ν with C_3, and note that if τ' is τ with the leaves labelled C_1 and C_2 removed, then τ' is a closed semantic tree for $S \cup \{C_3\}$. Prune τ' if necessary, and repeat this process until τ has been converted into a tree resolution derivation of $\square$ from S with no more resolvents than the number of nodes in τ. This completes the proof of the following theorem.

THEOREM 5.6.1.2.b.

Tree resolution p-simulates the system of semantic trees for sets of clauses.

COROLLARY 5.6.1.2.c.

No system of semantic trees is polynomial-bounded.

The converse of theorem 5.6.1.2.b. is also true.

THEOREM 5.6.1.2.d.

The system of semantic trees for sets of clauses p-simulates tree resolution.

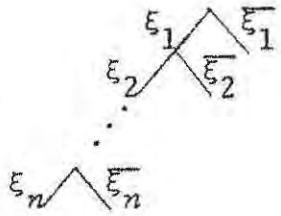
Proof

Let τ be a resolution tree (dag) for S, with the edges labelled as described at the beginning of subsection 4.2.4. By lemma 5.6.1.1.a. it may be assumed without loss of generality that τ is regular, so that no branch of τ contains a complementary pair of edge labels (nor more than one occurrence of any edge label). Note that the two edges coming into any node C of τ are labelled by a complementary pair of literals. Let b be a branch of τ terminating at clause $C \in S$, and let B be the set of literals labelling edges of b. Then, $C \subseteq B$, so that if $\bar{B}=\{\bar{\xi} \mid \xi \in B\}$, $\bar{B}$ falsifies C. Therefore, the tree τ' obtained from τ by deleting all node labels and complementing all edge labels is a closed semantic tree for S. ⊠5.6.1.2.d.

5.6.1.3. Tree Resolution *vs* Analytic Tableaux

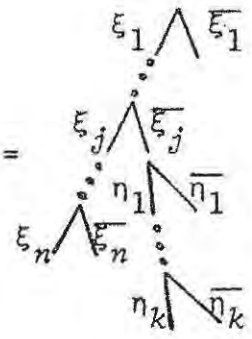
By theorems 5.6.1.2.b. and 5.6.1.2.d., the smallest tree resolution derivations of $\square$ from S are the same as the smallest closed semantic trees for S with only the labels changed. Let S be an inconsistent set of clauses, and let τ be a closed analytic tableau for S. A closed semantic tree τ' for S with at most twice as many nodes as τ can be constructed by a depth-first search of τ. First prune τ so that each branch of τ closes as soon as possible. Let ν_1 be a node of τ such that all of the sons of ν_1 are leaves. Then, if C_1 is the clause expanded at ν_1 and if $B_1=\langle \xi_1,\ldots,\xi_n \rangle$ is the sequence of node labels on the branch from the root of τ to ν_1 (with repetitions removed) and $\overline{B_1}=\{\overline{\xi_1},\ldots,\overline{\xi_n}\}$, then $C_1 \subseteq \overline{B_1}$ (because for each $\xi \in C_1$, $B_1 \cup \{\xi\}$ contains a complementary pair). Thus,

τ_1 =



is a closed semantic tree for $S_1=\{C_1,\{\xi_1\},\ldots,\{\xi_n\}\}$. Mark the path from the root of τ to ν_1 as "scanned". Let ν_1' be ν_1's father, and let C_1' be the clause expanded at ν_1'. Let ν_2 be any descendant of ν_1' (other than ν_1) whose sons are all leaves. If ν_1 is the only son of ν_1', then $C_1'=\{\xi_n\}$, so τ_1 is closed for $S_2=\{C_1,C_1',\{\xi_1\},\ldots,\{\xi_{n-1}\}\}$. In this case, let ν_1'' be ν_1'''s father, and try again to find a ν_2. Eventually either some ν_2 will be found, or it will be shown that for some $S_{n+1}\subseteq S$, τ_1 is closed for S_{n+1}. Let C_2 be the clause expanded at ν_2, and let $B_2=\langle\xi_1,\ldots,\xi_j,\eta_1,\ldots,\eta_k\rangle$ be the sequence of node labels from the root of τ to ν_2 (again, with repetitions removed). Note that $\overline{\xi_j}\in\{\eta_1,\ldots,\eta_k\}$, or the branch to ν_2 would have been pruned. Thus,

τ_2 =



is closed for $S_2=\{C_1,C_1',\ldots,C_1^{(n-j)},C_2,\{\xi_1\},\ldots,\{\xi_{j-1}\},\{\eta_1\},\ldots,\{\eta_k\}\}$. Mark the path in τ from the root to ν_2 as "scanned". Continue by

searching for the nearest ancestor of ν_2 whose sons have not already been scanned, and let ν_3 be a descendant of that ancestor with all leaves for sons. Note that at the i^{th} stage of this algorithm τ_i contains at most twice as many nodes as the part of τ that has been scanned. Also, if ν_j is any subtree of the scanned part of τ such that every descendant of ν_j with all leaves for sons has been scanned, and B is the set of node labels from the root of τ to ν_j, then τ_2 is closed for $S \cup \bar{B}$. Therefore, when every node of τ with all leaves for sons has been scanned, the semantic tree τ_j for that stage will be closed for S. This proves the following theorem.

THEOREM 5.6.1.3.a.

Tree resolution and the system of semantic trees for sets of clauses p-simulate the system of analytic tableaux for sets of clauses.

COROLLARY 5.6.1.3.b.

No system of analytic tableaux is polynomial-bounded.

The converse of theorem 5.6.1.3.a. is not true, by theorem 5.5.3.c., so that the systems of tree resolution and semantic trees for sets of clauses are strictly more powerful (to within a polynomial) than analytic tableaux.

5.6.1.4. Galil's Enumeration Trees

Galil has investigated an interesting generalization of semantic trees, called enumeration trees. Recall from paragraph 4.2.4.1. that $S \backslash p$ is the set of clauses obtained from S by deleting all clauses containing $\bar{p}$ and all occurrences

of p so that $S\backslash p \sim S\frac{E}{p}$. Note that if

$$\tau = \overset{p}{\underset{\tau_1}{\wedge}}\overset{\bar{p}}{\underset{\tau_2}{\wedge}}$$

is a closed semantic tree for S, then it is necessary and sufficient for τ_1 to be closed for $S\backslash\bar{p}$ and τ_2 to be closed for $S\backslash p$. Also, note that if τ is closed for $S'\subseteq S$, then τ is closed for S.

This leads to the following definition of an *enumeration tree*. If S is a set of clauses, then let $S\xi = \{C\xi \mid C\in S\}$. As a basis, the enumeration tree for $\square$ is $\square$. If τ_1 is an enumeration tree for $S_1 \cup S_1'$ and τ_2 is an enumeration tree for $S_2 \cup S_2'$, and p does not occur in S_1, S_1' S_2, or S_2', then

$$\tau = \overset{S}{\underset{\tau_1}{\wedge}}\underset{\tau_2}{}$$

is an enumeration tree for S if $(S_1 \cup S_1' p \cup S_2 \cup S_2' \bar{p}) \subseteq S$. By the arguments above, any closed semantic tree for S can be made into an enumeration tree for S by suitably labelling the nodes. Conversely, if τ_1' is a closed semantic tree for $S_1 \cup S_1'$ and τ_2' is a closed semantic tree for $S_2 \cup S_2'$, then

$$\tau' = \overset{\bar{p}}{\underset{\tau_1'}{\wedge}}\overset{p}{\underset{\tau_2'}{\wedge}}$$

is closed for $S\supseteq(S_1 \cup S_1' p \cup S_2 \cup S_2' p)$. This proves that enumeration trees are sound and complete, and also proves the following theorem. [Galil 1975]

198 missing.

THEOREM 5.6.1.4.a. (Galil)

Enumeration trees and semantic trees for sets of clauses p-simulate each other.

But the story does not end here, however, because while the information required to verify that a semantic tree is closed is distributed along the branches, the information required to verify that an enumeration tree τ is closed for S is contained in the node labels of the two sons of S. That is, if S' labels the roots of two subtrees of τ it is not necessary to check both of them; the existence of one enumeration tree for S' guarantees that all occurrences of S' are inconsistent. This leads to the idea of *enumeration dags* (Galil also calls these enumeration trees), which are obtained from enumeration trees by coalescing nodes with the same label and deleting extra subtrees. Since enumeration trees are a special case of enumeration dags, they are p-simulated by enumeration dags. Using corollary 5.6.1.1.d., theorem 5.6.1.2.d., and theorem 5.6.1.4.a., Galil disproves the converse with the following theorem.

THEOREM 5.6.1.4.b. (Galil)

The system of enumeration dags p-simulates regular resolution.

No proof

Anticipating a result from subsection 5.6.2., Galil has also extended Tseitin's proof that regular resolution is not polynomial-bounded to prove the following.

THEOREM 5.6.1.4.c. (Galil)

Enumeration dags cannot directly simulate extended regular resolution.

COROLLARY 5.6.1.4.d.

The system of enumeration dags (along with all systems enumeration dags simulate, including regular resolution) is not polynomial-bounded.

5.6.2. Regular Resolution

The system of regular resolution was introduced by Tseitin, and it is of interest primarily because something can be said about it that cannot (in the present state of knowledge) be said about resolution in general. Although it is difficult to see how non-regular resolution derivations could be much shorter than regular ones, it is equally difficult to see how to prove that no set of clauses has a non-regular derivation that is much shorter than its shortest regular derivation.

Paragraph 5.6.2.1. discusses the Davis-Putnam procedure, which is a special case of regular resolution. The effect of the subsumption rule on lengths of proofs, and Kirkpatrick's lower bound are both discussed. Paragraph 5.6.2.2. covers Tseitin's lower bound for regular resolution, and shows how Kirkpatrick's techniques can be applied to improve that bound. Finally, it is shown how the Tseitin-Kirkpatrick bound can be extended to cover certain systems of regular resolution with limited extension.

5.6.2.1. The Davis-Putnam Procedure

It is not hard to find examples of sets of clauses where the Davis-Putnam procedure produces tree proofs with one order of elimination of variables and exponentially long proofs with another order. Cook has given one such example [Cook 1972b], and his family $\{T_m\}$ [Cook 1973] is another.

A bit more subtle, but still not too difficult, are examples where the Davis-Putnam procedure without subsumption gives exponentially longer proofs than the Davis-Putnam procedure with the subsumption rule, no matter what variable elimination order is used. Such examples have been demonstrated by Cook [Cook 1971c] and Simon [Simon 1971], proving the following theorem and its corollary.

THEOREM 5.6.2.1.a. (Cook, Simon)

The Davis-Putnam procedure without subsumption cannot directly simulate the Davis-Putnam procedure with subsumption.

COROLLARY 5.6.2.1.b.

The Davis-Putnam procedure without subsumption is not polynomial-bounded.

It is much more difficult, however, to find examples of sets of clauses where the Davis-Putnam procedure with subsumption generates exponentially long proofs, no matter what variable elimination order is used. Although such a class of examples was given by Tseitin [Tseitin 1968], a similar class was discovered by Kirkpatrick [Kirkpatrick 1974], and Kirkpatrick's lower bound is slightly higher than Tseitin's.

what balls! Kirkpatrick adapted his stuff from Tseitin.

THEOREM 5.6.2.1.c. (Tseitin, Kirkpatrick)

There exists an infinite family of inconsistent sets of clauses $\{S_n\}$ and there are constants c_1 and c_2 such that if S_n contains n clauses, then every Davis-Putnam derivation of $\square$ from S_n (with subsumption) includes at least $2^{c_1 \cdot \frac{n}{log(n)}}$ clauses, while there is a regular extended resolution derivation of $\square$ from S_n consisting of no more than $c_2 \cdot n \cdot log(n)$ clauses.

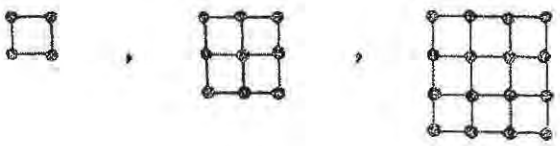
Proof Outline

Tseitin obtained the slightly more conservative bounds of $2^{c_1 \cdot \sqrt{n}}$ and $c_2 \cdot n^3$, respectively. Both Tseitin and Kirkpatrick developed their proofs by defining an inconsistent set of clauses $S(G)$ associated with any undirected graph G. They then showed a correspondence between steps of a derivation of $\square$ from $S(G)$ and certain operations on the graph G. They thus reduced the problem of finding sets of clauses with no short proofs to the problem of finding graphs with certain properties.

Given any undirected graph $G=(V,E)$, the set of clauses $S(G)$ is constructed as follows. For any set of literals $\{\xi_1,\ldots,\xi_n\}$, and for $\varepsilon\epsilon\mathbb{B}$, let $[\xi_1,\ldots,\xi_n]^{\varepsilon}$ denote the set of 2^{n-1} clauses that are equivalent to $\xi_1 \neq \ldots \neq \xi_n \equiv \varepsilon$. For example, $[p,q,r]^{T} = \{pqr, p\bar{q}\bar{r}, \bar{p}q\bar{r}, \bar{p}\bar{q}r\}$, and $[\bar{p},q,r,s]^{F} = \{pqr\bar{s}, pq\bar{r}s, p\bar{q}rs, \bar{p}qrs, p\bar{q}\bar{r}\bar{s}, \bar{p}q\bar{r}\bar{s}, \bar{p}\bar{q}rs, \bar{p}\bar{q}\bar{r}s\}$. If the edges of G are labelled with distinct literals, and $f:V\rightarrow\mathbb{B}$, then $S(G,f) = \bigcup_{v\epsilon V}[C_v]^{f(v)}$, where

C_v is the set of labels of edges incident with v in G. Tseitin shows that $S(G,f)$ is inconsistent if and only if $|\{v|f(v)=\top\}|$ is odd, and that if f and f' both map an odd number of vertices into $\top$ and if G' is the same graph as G with a (possibly) different labelling with distinct literals, then $S(G',f')$ is a renaming of $S(G,f)$. Thus, it may be assumed without loss of generality that the edges of G are labelled $p_1,\ldots,p_m$, that $f(v_1)=\top$ and $f(v_2)=\ldots=f(v_n)=\bot$, and that it is this labelling of the vertices and edges of G that gives $S(G)$. Notice that if $d(v)$ is the degree of vertex v in G. then the number of clauses in $S(G)$ is $\sum_{v\in V}(2^{d(v)-1})$.

Tseitin and Kirkpatrick both show (in somewhat different ways) that if any sequence of single-edge deletions that reduces G to the empty graph on n vertices must at some stage produce a connected graph $\hat{G}$, where there are at least m edges of G that are incident with at least one vertex of $\hat{G}$ but are not edges of $\hat{G}$, then any derivation of $\square$ from $S(G)$ must contain more than 2^m clauses. The difference between Tseitin's and Kirkpatrick's lower bounds arises from the fact that they used different graphs. Tseitin's graphs are two dimensional square grids:



while Kirkpatrick's graphs are modified n-cubes.

Kirkpatrick defined the n-cube B_n to be an n-dimensional cube viewed as a graph. That is, if the vertices of B_n are the 2^n n-tuples of 0's and 1's, then vertices $\vec{i}$ and $\vec{j}$ are adjacent in B_n if and only if $\vec{i}$ and $\vec{j}$ differ in exactly one position. Modified n-cubes D_n are constructed by replacing the vertices of B_n by cycles of n vertices. That is, $D_n=(V_n,E_n)$, where

$V_n = \{(\vec{i},r) \mid \vec{i}\in\{0,1\}^n, r\in\{0,\ldots,n-1\}\}$ and

$E_n = \{((\vec{i},r),(\vec{j},s)) \mid$ either $r=s$ and $\vec{i}$ and $\vec{j}$ differ in the r^{th} position and only in the r^{th} position, or $(r-s)\equiv 1 \pmod n$ and $\vec{i}=\vec{j}\}$.

Note that for $n\geq 3$ each vertex of D_n has degree three, and D_n has $n\cdot 2^n$ vertices. Kirkpatrick's lower bound is obtained by letting $S_{4\cdot n\cdot 2^n} = S(D_n)$. ☒5.6.2.1.c.

COROLLARY 5.6.2.1.d. (Tseitin, Kirkpatrick)

The Davis-Putnam procedure with subsumption cannot directly simulate regular extended resolution.

COROLLARY 5.6.2.1.e. (Tseitin, Kirkpatrick)

The Davis-Putnam procedure with subsumption (and any system simulated by the Davis-Putnam procedure with subsumption) is not polynomial-bounded.

5.6.2.2. Tseitin's Lower Bound for Regular Resolution

Tseitin never actually discussed the Davis-Putnam procedure in [Tseitin 1968], but part of his proof of the following theorem effectively shows that the shortest regular resolution derivation of $\square$ from $S(G)$ can be generated by the Davis-Putnam procedure (without subsumption). Thus, theorem 5.6.2.1.c. applies to regular resolution as well as the Davis-Putnam procedure. Galil has extended Tseitin's work even further, showing that theorem 5.6.2.1.c. also applies to enumeration dags [Galil 1975]. Finally, it is not hard to see (Galil has also made this observation.) that combining Tseitin's and Galil's proofs with Kirkpatrick's graphs gives the following theorem.

THEOREM 5.6.2.2.a. (Tseitin, Kirkpatrick, Galil)

There exists an infinite family of inconsistent sets of clauses $\{S_n\}$ and there are constants c_1, c_2, and c_3 such that if $\ell S_n = n$, and if the shortest derivations of S_n have lengths r in the system of regular resolution, d in the system of enumeration dags, and e in the system of regular extended resolution, then

$$r \geq 2^{c_1 \cdot \frac{n}{(log(n))^2}},$$

$$d \geq 2^{c_2 \cdot \frac{n}{(log(n))^2}}, \text{ and}$$

$$e \leq c_3 \cdot n \cdot log(n).$$

COROLLARY 5.6.2.2.b. (Tseitin, Kirkpatrick, Galil)

Neither regular resolution nor enumeration dags can directly simulate regular extended resolution.

COROLLARY 5.6.2.2.c. (Tseitin, Kirkpatrick, Galil)

Neither regular resolution nor enumeration dags is polynomial-bounded.

Recall from paragraph 4.2.4.2. the definitions of $cl(B)$, $def(B)$, resolution with limited extension, and extension derivations. Since the computation of $def(B)$ depends on the connectives in B, it is more proper to consider the methods of resolution with limited extension (or systems of extension) with two different sets of connectives as distinct proof systems.

The fact that the graphs D_n have degree three can be exploited to show the same lower bound for the system of regular resolution with limited extension for any set of connectives κ that contains both $\equiv$ and $\not\equiv$ or $\neg$ and either $\equiv$ or $\not\equiv$. First note that $[\xi^A \xi^B \xi^{A\equiv B}]^{\mathsf{T}}$ is the same set of clauses as $cl(A\equiv B)$, and that $cl(A\not\equiv B) = [\xi^A \xi^B \xi^{A\not\equiv B}]^{\mathsf{F}}$. Let v be the vertex of D_n that gets labelled T, and let α be the label of one edge (u,v) incident with v. Let τ be a directed spanning tree of D_n such that the root of τ is v and the only edge incident with v in τ is α. For example, the darkened lines in figure 5.6.2.2.i. give such a spanning tree τ for D_3. Note that $S(D_n) = [\alpha,\beta,\gamma]^{\mathsf{T}} \cup \bigcup_{\substack{w\neq v \\ w\in V}} [\alpha',\beta',\gamma']^{\mathsf{F}}$, where α', β',

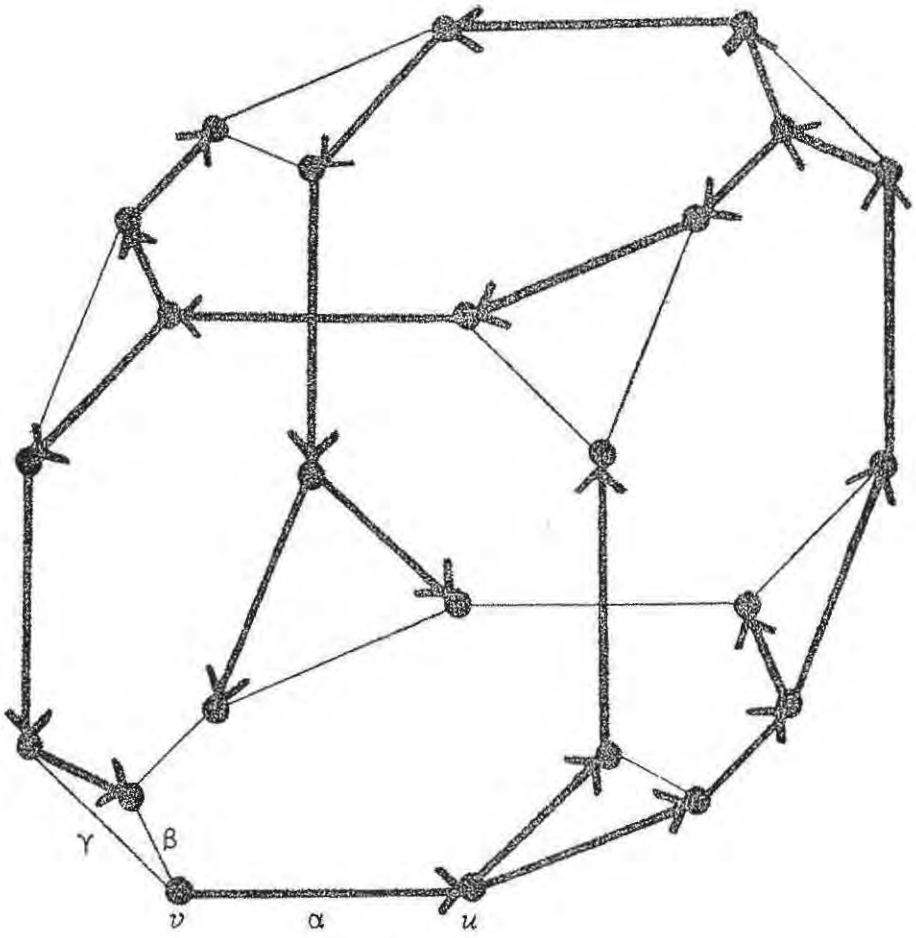


FIGURE 5.6.2.2.i.

Directed Spanning Tree for the Modified n-cube D_3

γ' are the edges incident with w and α,β,γ are the edges incident with v. For edges $\xi \in E$, define $F(\xi)$ inductively as follows:

1) if $\xi \notin \tau$, then $F(\xi)=\xi$,

2) if $\xi \in \tau$, then $F(\xi) = (F(\xi_1) \not\equiv F(\xi_2))$, where ξ_1, ξ_2 are the edges of D_n incident with the head of ξ (*i.e.* the end of ξ farthest from the root of τ).

Finally, let $F_n = (F(\alpha) \equiv (\beta \equiv \gamma))$, and note that F_n is unsatisfiable. Then F_n is a formula in $\{\xi \mid \xi$ labels an edge of $D_n - \tau\}$. If the literal ξ is associated with the formula $F(\xi)$ for $\xi \in \tau$, and p is associated with $\beta \equiv \gamma$ and q is associated with F_n, then

$def(F_n) = [\beta,\gamma,p]^{T} \cup [\alpha,p,q]^{T} \cup_{\substack{u \xi v \\ u \not\geq v}} [\alpha',\beta',\gamma']^{F}$. Therefore,

$def(F_n) = (S(D_n) - [\alpha,\beta,\gamma]^{T}) \cup [\beta,\gamma,p]^{T} \cup [\alpha,p\ q]^{T}$. Let $\hat{D}$ be a regular derivation of $\square$ from $S = (def(F_n) \cup \{q\})$. (Note: $def(F_n) \cup \{q\}$ is $def(\neg F_n) \cup \{\xi^{\overline{\neg F_n}}\}$ and $\neg F_n$ is a tautology.) Let $\hat{D}_1$ be the derivation obtained from $\hat{D}$ by replacing S by $S \backslash \bar{q}$ and dropping all resolutions on q. Thus, $\hat{D}_1$ is a regular derivation no longer than $\hat{D}$ of $\square$ from

$(S(D_n) - [\alpha,\beta,\gamma]^{T}) \cup [\beta,\gamma,p]^{T} \cup \{\alpha\bar{p}, \bar{\alpha}p\}$. If any clause $\alpha\bar{p}C$ or $\bar{\alpha}pC'$ appears in $\hat{D}_1$, then $\hat{D}_1$ may be shortened by replacing $\alpha\bar{p}C$ by $\alpha\bar{p}$ (or replacing $\bar{\alpha}pC'$ by $\bar{\alpha}p$) and deleting some resolutions, if necessary. After deleting all occurrences of such clauses, a derivation $\hat{D}_2$ is obtained, still no longer than $\hat{D}$, and such that no clause in the derivation contains $\alpha\bar{p}$ or $\bar{\alpha}p$ (except

these initial clauses themselves). Replacing every occurrence in $\hat{D}_2$ of p by α and $\bar{p}$ by $\bar{\alpha}$ gives a derivation $\hat{D}_3$ (again some resolvents may have to be discarded) of $\square$ from $(S(D_n)-[\alpha,\beta,\gamma]^{\top})\cup[\beta,\gamma,\alpha]^{\top}\cup\{\alpha\bar{\alpha},\alpha\bar{\alpha}\} = S(D_n)\cup\{\alpha\bar{\alpha}\}$. All uses of $\alpha\bar{\alpha}$ can clearly be eliminated without changing the conclusion of the derivation, so that a derivation $\hat{D}_4$ of $\square$ from $S(D_n)$ is obtained, where $\hat{D}_4$ has no more resolvents than $\hat{D}$. Thus, any short derivation of $\neg F_n$ by regular resolution with limited extension gives a shorter regular resolution derivation of $\square$ from $S(D_n)$. Note that F_n could easily be modified to be a formula in $\{\neg,\equiv\}$ or$\{\neg,\not\equiv\}$. Combined with the proof of theorem 5.6.2.1.c., this gives the following theorem.

THEOREM 5.6.2.2.d.

The system of regular resolution with limited extension for any set of connectives containing $\{\equiv,\not\equiv\}$, $\{\neg,\equiv\}$, or $\{\neg,\not\equiv\}$ cannot directly simulate regular extended resolution, and is thus not polynomial-bounded.

It is interesting to note that, since F_n is just a large formula in $\equiv$ and $\not\equiv$, the techniques of lemmas 5.3.3.e., 5.4.a., 5.4.b., and 5.4.c. for manipulating conjunctions can be adapted to manipulating iterated formulas in $\equiv$ and $\not\equiv$, to give an inference system I that allows derivations of $\neg F_n$ of length $p(\ell F_n)$, for some polynomial p. Thus, the following theorem is obtained.

THEOREM 5.6.2.2.e.

The system of regular resolution with limited extension for any κ containing $\{\equiv,\not\equiv\}$, $\{\neg,\equiv\}$, or $\{\neg,\not\equiv\}$ cannot simulate any Frege system.

5.6.3. Extended Resolution

The method of extended resolution and the family of systems of extension for each set of connectives κ were defined in paragraph 4.2.4.2. Since an extension derivation of A is an extended resolution derivation of ξ^A from $def(A)$, to show that extension (for any set of connectives κ) p-simulates extended resolution, it suffices to observe that $def(A)$ can be computed in polynomial time, and that an extended resolution derivation of $\square$ from $def(A) \cup \{\overline{\xi^A}\}$ can be converted to a derivation of ξ^A merely by ignoring all resolutions involving the clause $\overline{\xi^A}$.

THEOREM 5.6.3.a.

The system of extension (for any set of connectives κ) p-simulates extended resolution.

The converse of this theorem is also true.

THEOREM 5.6.3.b.

Extended resolution p-simulates the system of extension for $\{\neg, \vee, \&\}$.

Proof

Let S be any unsatisfiable set of clauses, and let S' be the formula in $\{\neg, \vee, \&\}$ constructed equivalent to S in the obvious way. Thus, S' is unsatisfiable, so $\neg S'$ is a tautology. Let D be an extension derivation of $\neg S'$. That is, $\vdash_e \neg S'$ *via* D, so that $def(\neg S') \vdash_{er} \xi^{\neg S'}$ *via* D. Note that $def(\neg S') = def(S')$ and $\xi^{\neg S'} = \overline{\xi^{S'}}$, and that $def(S')$ can be obtained by extension of the set of clauses S. Thus, if there is a derivation D' such that $S \cup def(S') \cup \{\overline{\xi^{S'}}\} \vdash_r \square$ *via* D', then $S \vdash_{er} \square$ *via* $def(S')DD'$.

Derivation D' can be constructed in a very straightforward manner. Let $C_i = \xi_1 \ldots \xi_m$ be any clause in S, and let $C_i' = ((\ldots(\xi_1 \vee \xi_2) \vee \ldots) \vee \xi_m)$. Then $def(C_i') \subseteq def(S')$, where $def(C_i') = \bigcup_{j=2}^{m} \{\overline{\alpha_j}\alpha_{j-1}\xi_j, \alpha_j\overline{\alpha_{j-1}}, \alpha_j\overline{\xi_j}\}$, $\alpha_1 = \xi_1$, and $\alpha_m = \gamma_i$. A resolution derivation D_i of γ_i from $\{C_i\} \cup def(C_i')$ can be constructed in $2(m-1)$ steps. If $m=1$, then $\gamma_i = \alpha_m = \alpha_1 = \xi_1 = C_i$. Otherwise, for $j=2,\ldots,m$, resolve $\alpha_{j-1}\xi_j \ldots \xi_m$ with $\alpha_j\overline{\alpha_{j-1}}$ to get $\alpha_j\xi_j \ldots \xi_m$ and then with $\alpha_j\overline{\xi_j}$ to get $\alpha_j\xi_{j+1} \ldots \xi_m$. The result is a $2(m-1)$ step derivation D_i of $\{\gamma_i\}(=\{\alpha_m\})$ from $\{C_i\} \cup def(C_i')$. Now suppose $S=\{C_1,\ldots,C_n\}$, so that $S'=((\ldots(C_1'\&C_2')\&\ldots)\&C_n')$. Then $def(S') = \bigcup_{i=1}^{n} def(C_i') \cup \bigcup_{j=2}^{n} \{\overline{\beta_j}\beta_{j-1}, \overline{\beta_j}\gamma_j, \beta_j\overline{\beta_{j-1}}\overline{\gamma_j}\}$, where $\beta_1=\gamma_1$ and $\beta_n=\xi^{S'}$. A $2(n-1)$ step derivation D_0 of $\{\xi^{S'}\}$ from $\{\{\gamma_1\},\ldots,\{\gamma_n\}\} \cup def(S')$ can be constructed as follows. If $n=1$, then $\xi^{S'}=\beta_n=\beta_1=\gamma_1$. Otherwise, for $j=2,\ldots,n$, resolve β_{j-1} with $\beta_j\overline{\beta_{j-1}}\overline{\gamma_j}$ to obtain $\beta_j\overline{\gamma_j}$, and then resolve this with γ_j to obtain β_j. Finally, deriving $\square$ from $\{\xi^{S'}, \overline{\xi^{S'}}\}$ in the one-step derivation $\hat{D}$ leads to the derivation $D' = D_1D_2 \ldots D_nD_0\hat{D}$ of $\square$ from $S \cup def(S') \cup \{\overline{\xi^{S'}}\}$. The simple observation that $\ell D'$ is proportional to $\ell def(S')$ completes the proof. ⊠5.6.3.b.

The relationship between systems of extension for different sets of connectives is best shown indirectly, *via* extended Frege systems. It can be shown that the system of extension for any set of connectives κ and any extended Frege system for κ p-simulate each other. Then the fact that any two extended Frege systems p-simulate each other can be used to show that all systems of extension and the system of extended resolution are in the same p-simulation equivalence class with extended Frege systems.

THEOREM 5.6.3.c.

If κ is any adequate set of connectives and $F=<\kappa,\mathcal{R}>$ is a Frege system, then the system of extension for κ p-simulates the extended Frege system eF.

Proof

The basic outline of this proof was suggested by a paragraph in [Tseitin 1968], where Tseitin briefly indicates how extended resolution simulates Gentzen systems. In fact, it was that paragraph that stimulated this entire line of research into the comparative complexities of proof systems.

Let A be any tautology in the connectives κ, and let D be any derivation such that $\vdash_{eF} A$ *via* D, so that $\Gamma \vdash_F A$ *via* D', $D=\Gamma D'$, and $\Gamma = \{\hat{E}(p_1,B_1),\ldots,\hat{E}(p_n,B_n)\}$ where p_i does not appear in $\{A \cup \bigcup_{j=1}^{i-1} \hat{E}(p_j,B_j) \cup B_i\}$. For $1 \le i \le n$ associate the literal p_i with the formula B_i, and let $def(D) = \bigcup_{\text{all formulas } B \text{ in } D} def(B)$, noting that $def(A) \subseteq def(D)$,

and that $def(D)$ can be obtained from $def(A)$ by a sequence of extensions.

In particular, note that for $1 \leq i \leq n$, $def(\hat{E}(p_i,B_i)) \subseteq def(D)$, and $def(\hat{E}(p_i,p_i)) \subseteq def(\hat{E}(p_i,B_i))$. Since $\hat{E}(p,p)$ is a tautology, there is a resolution derivation D, where $def(\hat{E}(p,p)) \vdash_r \xi^{\hat{E}(p,p)}$ *via* $D^{\hat{E}}$. (Extension will not be required, because $(def(\hat{E}(p,p)) \cup \{\xi^{\overline{\hat{E}(p,p)}}\}) \models \square$ but $def(\hat{E}(p,p))$ is consistent.) Then for $1 \leq i \leq n$ there are renamings σ_i such that $def(\hat{E}(p_i,B_i)) \vdash_r \xi^{\hat{E}(p_i,B_i)}$ *via* $D^{\hat{E}}\sigma_i = D_i$. Note that D_i contains the same number of steps as $D^{\hat{E}}$.

For each rule $R=\{A_1,\ldots,A_k\} \rightarrow B$ in $\mathcal{R}$, let $def(R) = \bigcup_{i=1}^{k} def(A_i) \cup def(B)$, and let D^R be a resolution derivation of ξ^B from $def(R) \cup \{\xi^{A_1}\} \cup \ldots \cup \{\xi^{A_n}\}$. (Again, extension is not needed. The required derivation can be obtained by deleting all resolutions involving $\overline{\xi^B}$ from a resolution derivation of $\square$ from $def(R) \cup \{\xi^{A_1}\} \cup \ldots \cup \{\xi^{A_n}\} \cup \{\overline{\xi^B}\}$.) Note that for any formula A and any substitution σ there is a renaming σ' such that $def(A)\sigma' \subseteq def(A\sigma)$. Because of this property of def, if B' is inferred from $A_1',\ldots,A_n'$ by rule R, and if $\Gamma = \bigcup_{i=1}^{n} def(A_i') \cup def(B')$, then there is a renaming σ' such that $\Gamma \cup \{\xi^{A_1'}\} \cup \ldots \{\xi^{A_n'}\} \vdash_r \xi^{B'}$ *via* $D^R\sigma'$.

It should now be clear how to construct a resolution derivation D'' of ξ^A from $def(D)$. It begins with derivations $D_1,\ldots,D_n$ of the single-literal clauses $\xi^{\hat{E}(p_1,B_1)},\ldots,\xi^{\hat{E}(p_n,B_n)}$. Then, for every formula B' in D', D'' contains the single-literal clause $\xi^{B'}$, preceded by the appropriate renaming of the derivation corresponding to the rule by which B' was derived in D'. The total number of steps in D'' is bounded above by some constant times $(n+\ell^s D')$, and each step contains a bounded number of literals. This gives the required extension derivation of A. ☒5.6.3.c.

The converse of this theorem has an equally straightforward proof.

THEOREM 5.6.3.d.

If κ is an adequate set of connectives, then there is a Frege system $F=<\kappa,\mathcal{R}>$ such that the extended Frege system at p-simulates the system of extension for κ.

Proof

The proof of this theorem is simplified by an appeal to the following rather neat result from [Galil 1975].

LEMMA 5.6.3.e. (Galil)

There is a constant e such that if m distinct atoms appear in D, if $S \vdash_{er} C$ *via* D, and no clause of $S\cup\{C\}$ contains more than j literals, then there is an extended resolution derivation D' such that

1) $S \vdash_{er} C$ *via* D',

2) no clause of D' contains more than j literals, and

3) $\ell^s D' \leq e \cdot m \cdot \ell^s D$.

Now let A be a formula in the connectives κ, and let D be an extension derivation of A, so that $def(A) \vdash_{er} \xi^A$ *via* D, and $(def(\Gamma) \cup def(A)) \vdash_r \xi^A$ *via* D', where Γ is the set of formulas introduced by extension. If k is the maximum arity of connectives in κ, then no clause in $(def(\Gamma) \cup def(A))$ contains more than $k+1$ literals. Then, by lemma 5.6.3.e., it may be assumed without loss of generality that no clause of D' has more than $k+1$ literals. For $1 \leq i \leq k+1$, let $\vee_i(p_1,\ldots,p_i)$ be a formula in the connectives κ such that $\vee_i(p_1,\ldots,p_i) \sim ((\ldots(p_1 \vee p_2) \vee \ldots) \vee p_i)$. Since $k+1$ is finite, it is allowable for $\mathcal{R}$ to contain rules that simulate all possibilities of resolutions involving clauses with no more than $k+1$ literals. For example, if C_1, C_2, and C_3 are clauses of lengths $i_1, i_2, i_3 \leq k+1$ and C_3 is the resolvent of C_1 and C_2, then $\mathcal{R}$ will contain the rule $\vee_{i_1}(C_1), \vee_{i_2}(C_2) \rightarrow \vee_{i_3}(C_3)$ (or a renaming of it).

If C is a clause of length i, then let $\tilde{C} = \vee_i(C)$, and if S is a set of clauses, then let $\tilde{S} = \{\tilde{C} \mid C \in S\}$. Then, since $\vee_1(\xi) = \xi$, if $S = (def(\Gamma) \cup def(A))$, then $\tilde{S} \vdash_F \xi^A$ *via* $\tilde{D}'$. Thus, to get a derivation of A in the system eF, it will suffice to find a series of extensions Δ such that $\Delta \vdash_F \tilde{S}$ and $\Delta, \xi^A \vdash_F A$. For

this let $\Delta = \bigcup_{B \epsilon sub(\Gamma \cup \{A\})} \hat{E}(\xi^B, *(\xi^{C_1}, \ldots, \xi^{C_i}))$, where $B = *(C_1, \ldots, C_i)$. Then, since $cl(B) \sim \hat{E}(\xi^B, *(\xi^{C_1}, \ldots, \xi^{C_i}))$, each formula $\tilde{C}$ corresponding to a clause C of $cl(B)$ can be derived in system F from $\hat{E}(\xi^B, *(\xi^{C_1}, \ldots, \xi^{C_i}))$. Notice that this derivation is a renaming of a fixed derivation that depends only on $*$.

All that remains is to show how to derive A from $\Delta \cup \{\xi^A\}$. If F contains the rules $\hat{E}(p_1, q_1), \ldots, E(p_i, q_i) \rightarrow \hat{E}(*(\vec{p}), *(\vec{q}))$, for each connective $* \epsilon \kappa$, plus the rules $\hat{E}(p,p)$ and $\hat{E}(p,q), p \rightarrow q$, it is not hard to see how to construct a short derivation of $\hat{E}(\xi^A, A)$ from Δ^A, where Δ^A is the sequence of extensions defining A. Since $\Delta^A \subseteq \Delta$, this completes the proof. ⊠5.6.3.d.

A minor modification of the proof of theorem 5.6.3.d. leads to the following theorem.

THEOREM 5.6.3.f.

If κ is any adequate set of connectives, then there is a Frege system $F = \langle \kappa, \mathcal{R} \rangle$ that p-simulates the system of resolution with limited extension for κ.

Proof Sketch

The proof of this theorem is similar to the proof of theorem 5.6.3.d. The clause $C = \xi^{A_1} \ldots \xi^{A_n}$ is simulated by the formula $\tilde{C} = \bigvee_n(A_1, \ldots, A_n)$. Since the system of resolution with *limited* extension is not allowed to do any extensions

beyond $def(A)$, theorem 5.6.3.e. does not apply, so that D' may contain arbitrarily long clauses. Thus, $\mathcal{R}$ cannot contain enough rules to directly simulate all of the possible resolutions in one step. Instead $\mathcal{R}$ contains rules for reordering disjunctions analogous to those of lemma 5.4.a. and a single resolution rule: $V_2(p,q),V_2(\neg p,r) \rightarrow V_2(q,r)$. This gives a derivation in system F of A from $\tilde{\tilde{S}}$. But each formula $\tilde{C}$ of $\tilde{S}$ is a tautology whose derivation is an instance of the derivation for the principal connective of the formula being defined by C.

Thus, any extension derivation D of A leads to a derivation D' of A in the system F. However, if D contains extensions that define literals corresponding to exponentially long formulas, D' will be exponentially long. Thus, all that can be said with certainty is that if D' contains no extensions defining literals equivalent to formulas that are not subformulas of A, then $\ell D'$ is bounded by a polynomial in ℓD and ℓA. ☒5.6.3.f.

The results of this subsection are rounded out with the inclusion of the following very interesting theorem from [Galil 1975].

THEOREM 5.6.3.g. (Galil)

Extended regular resolution p-simulates extended resolution.

5.6.4. Resolution *vs* Cut-Free Gentzen Systems

The "weakest" systems that have not been shown not to be polynomial-bounded are (non-regular) resolution, systems of (non-regular) resolution with limited extension, and cut-free Gentzen systems. All of these systems are p-simulated by any system in the equivalence class including Frege systems, natural deduction systems, and sequent systems. Since the computation of $def(A)$ can be carried out in polynomial time, the p-simulation of (regular or non-regular) resolution by (regular or non-regular, respectively) resolution with limited extension is trivial. The p-simulation of resolution with limited extension by Frege systems was established by theorem 5.6.3.f. The p-simulation of basic Gentzen systems by Gentzen systems with thinning is trivial (theorem 5.5.2.b.), and the p-simulation of Gentzen systems with thinning by Gentzen systems with cut is established by theorem 5.5.2.a. Theorem 5.5.2.a. could be combined with established techniques for reordering conjunctions and disjunctions to show that Gentzen systems with cut p-simulate the Gentzen system for sets of clauses.

THEOREM 5.6.4.a.

The Gentzen system with cut for $\{\neg,\vee,\&\}$ p-simulates the Gentzen system for sets of clauses.

(Note: The only real difference between the Gentzen system for sets of clauses and the Gentzen system with thinning for $\{\neg,\vee,\&\}$ (when applied to sets of clauses) is that the order of disjunctions does not matter in the former system. That is, when both systems are applied to sets of clauses, the

Gentzen system for sets of clauses is merely a minor generalization of the Gentzen system with thinning for $\{\neg,\vee,\&\}$, and thus trivially p-simulates that system.)

In the other direction, the systems under consideration in this subsection are all simple generalizations of systems that have been shown not to be polynomial-bounded (regular resolution, regular resolution with limited extension for $\kappa\supseteq\{\neg,\equiv\}$, analytic tableaux, *etc.*). It therefore seems worthwhile to investigate these systems in as much detail as possible, with the hope of establishing relationships that will allow these systems to be added to the list of systems that are provably not polynomial-bounded. Although there is hope that the proof techniques of Tseitin (as extended by Kirkpatrick and Galil) could be used to give non-polynomial lower bounds for these systems, theorem 5.6.2.2.e. tends to indicate that these techniques cannot be extended to Frege systems.

The relationships between different cut-free Gentzen systems were discussed in subsection 5.5.2., and the relationships between various resolution-based systems have been discussed earlier in this section. This subsection covers the relationship between certain cut-free Gentzen systems and certain resolution-based systems. In paragraph 5.6.4.1. it is shown that the Gentzen system for sets of clauses p-simulates enumeration dags, while paragraph 5.6.4.2. shows that each Gentzen system with thinning is p-simulated by the system of resolution with limited extension for the same connectives.

5.6.4.1. The Gentzen System for Sets of Clauses

Recall the definition of the Gentzen system (with thinning) for sets of clauses from paragraph 4.2.3.1. and the definition of Galil's enumeration dags from paragraph 5.6.1.4. The purpose of this paragraph is to prove the following theorem.

THEOREM 5.6.4.1.a.

The Gentzen system with thinning for sets of clauses p-simulates the system enumeration dags.

Proof

Let G be the Gentzen system with thinning for sets of clauses. Since no rule of G introduces any formula into the consequent side of any sequent, every sequent in every G-derivation has an empty consequent. Thus, there is no confusion if the sequent $\Delta\vdash$ is denoted simply by Δ. It is convenient to identify the formula $\bigvee(\xi_1,\ldots,\xi_m)$ with the clause $C=\{\xi_1,\ldots,\xi_m\}$, and the conjunction $\&(C_1,\ldots,C_n)$ with the set of clauses $S=\{C_1,\ldots,C_n\}$. Note that n-1 $\&$-elimination steps are sufficient to derive the sequent $\&(C_1,\ldots,C_n)$ from the sequent $\{C_1,\ldots,C_n\}$. Since G is analytic, any derivation of the sequent $S=\{C_1,\ldots,C_n\}$ must consist entirely of sequents that are sets of clauses. The remaining rules that apply are the axiom rule: $\rightarrow\xi,\bar{\xi},S'$, the thinning elimination rule: $S'\rightarrow S',C$, and the $\vee$-elimination rule: S',C_1 , $S',C_2 \rightarrow S',C_1C_2$.

Let S be an inconsistent set of clauses, and let D be an enumeration dag for S. A derivation D' of S in the system G can be constructed from D in a very straightforward way.

If S' is a set of clauses labelling some node of D, then D' will contain the sequent S', along with a derivation in system G of S' from the sequents corresponding to the parents of S' in D.

Assume that the parents of S' are $S_1 \cup S_1'$ and $S_2 \cup S_2'$, and that $(S_1 \cup S_1' \xi \cup S_2 \cup S_2' \bar{\xi}) \subseteq S'$. If $S_1 \cup S_1' = S_2 \cup S_2' = \{\Box\}$, then $\{\xi, \bar{\xi}\} \subseteq S'$ (or $\Box \in S'$, in which case D could have been pruned), so that the sequent S' is an axiom in system G.

If one parent of S' is $\{\Box\}$, then $(S_1 \cup S_1' \xi \cup \{\bar{\xi}\}) \subseteq S'$. Assume that the sequent $S_1 \cup S_1'$ has already been derived, and $S_1' = \{C_1, \ldots, C_k\}$. If $k=0$, then S' can be derived from S_1 by a series of thinning steps. Otherwise, for $1 \leq i \leq k$, let S^i be the axiom $\{S_1, C_1\xi, \ldots, C_{i-1}\xi, C_{i+1}, \ldots, C_k, \xi, \bar{\xi}\}$. Then, derive $S_1, C_1, \ldots, C_k, \xi$ by a single thinning step, and for $1 \leq i \leq k$, derive $S_1, C_1\xi, \ldots, C_i\xi, C_{i+1}, \ldots, C_k, \xi$ from $S_1, C_1\xi, \ldots, C_{i-1}\xi, C_i, \ldots, C_k, \xi$ and S^i by a single v-elimination step. The result of this last step is the sequent $S_1, S_1'\xi, \xi$, from which S' can be derived by thinning.

Finally, assume that both parents of S' have been derived, and that $S_1' = \{C_1, \ldots, C_k\}$, and $S_2' = \{B_1, \ldots, B_j\}$. If $k=0$, then S' can be derived from S_1 by thinning, and if $j=0$, S' can be derived by thinning from S_2. Otherwise, S' is derived from S_1, S_1' and S_2, S_2' by the derivation shown in figure 5.6.4.1.i.

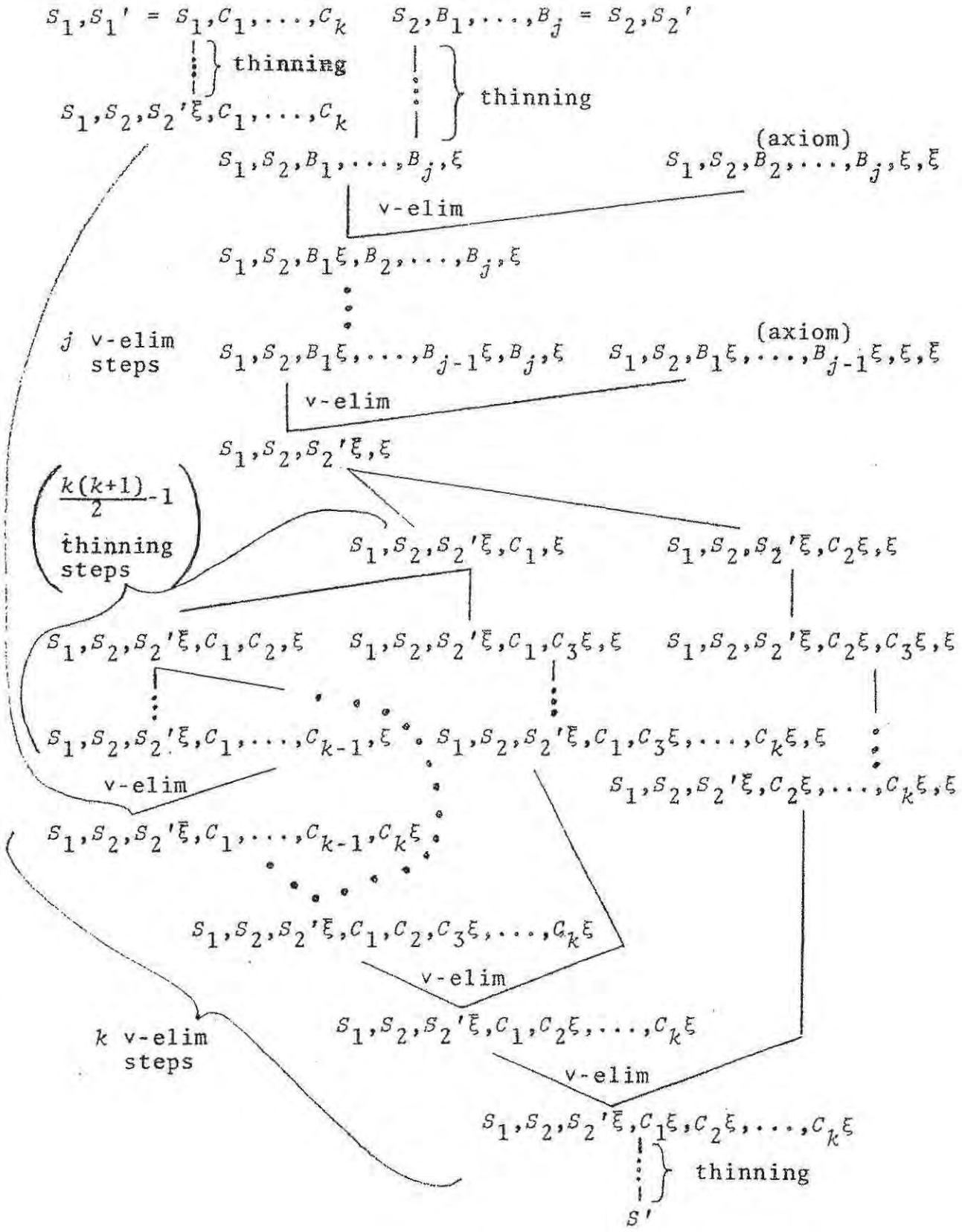


FIGURE 5.6.4.1.i.

Simulation of Enumeration Dags by the Gentzen System with Thinning for Sets of Clauses

In any case, the number of steps in the derivation of S' from its parents is bounded by a polynomial (of degree two) in the number of clauses in S'. Then, since the number of clauses in S' is not less than the number of clauses in each of its parents, no sequent in the entire derivation contains more clauses than S. Therefore, the total length of derivation D' is bounded by $c \cdot \ell^8 D \cdot (\ell S)^2$, where c is a constant.

☒5.6.4.1.a.

Since the Gentzen system for sets of clauses is such a minor generalization of Gentzen systems for $\{\neg, \vee, \&\}$, it seems unlikely that it could ever be shown that Gentzen systems (with thinning, at least) cannot simulate regular resolution.
?? Depends on the connectives, surely. False (probably) for ≡.

5.6.4.2. Resolution with Limited Extension

In paragraph 5.6.2.2. it was shown that certain systems of regular resolution with limited extension are not polynomial-bounded. This paragraph presents a proof (suggested in [Tseitin 1968]) that for any adequate set of connectives κ the system of resolution with limited extension for κ p-simulates the Gentzen system with thinning for κ. This suggests that a Gentzen derivation should be called *regular* if its simulating derivation in the system of resolution with limited extension is regular. It will turn out that a Gentzen derivation D is non-regular if there is a formula A and a sequence of sequents $S_1, \ldots, S_k$ such that for $1 \leq i \leq k-1$, S_i is a parent of S_{i+1} in D, A appears (either as an antecedent formula or as a consequent formula) in S_1 and S_k, but for some $1 < i < k$,

A does not appear (as an antecedent or consequent) in the sequent S_i (although A must appear as a subformula). It is not hard to see that regular Gentzen systems are complete, but a corollary of the proof of the following theorem is that for κ containing $\{\neg,\equiv\}$, $\{\neg,\not\equiv\}$, or $\{\equiv,\not\equiv\}$, the regular Gentzen system for κ with thinning is not polynomial-bounded.

THEOREM 5.6.4.2.a. (Tseitin)

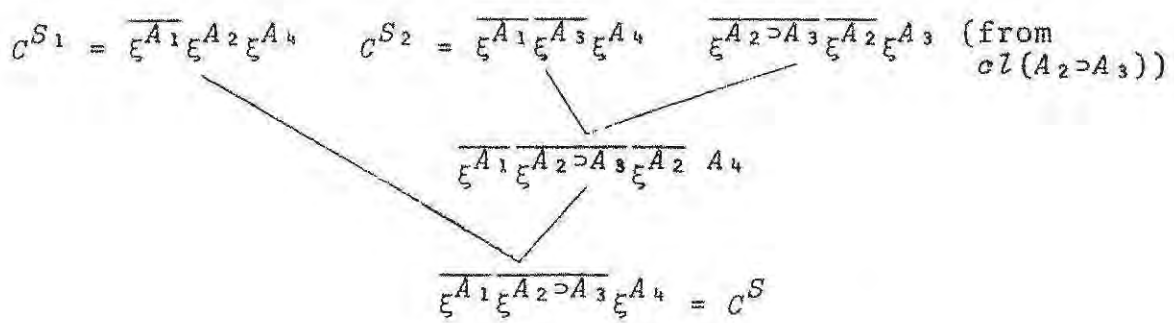
If κ is any adequate set of connectives, then the system of (regular) resolution with limited extension for κ p-simulates the (regular) Gentzen system with thinning for κ.

Proof

The main idea behind this proof comes from [Tseitin 1968]. That is to represent the sequent $S = A_1,\ldots,A_m \vdash B_1,\ldots,B_n$ by the clause $C^S = \overline{\xi^{A_1}},\ldots,\overline{\xi^{A_m}},\xi^{B_1},\ldots,\xi^{B_n}$. Note that because the Gentzen systems with thinning are analytic, if B appears in any sequent of a derivation of $\vdash A$, then B is a subformula of A, so that ξ^B appears in $def(A)$. Then, if S is inferred from $S_1,\ldots,S_k$ by applying the $*$-introduction or $*$-elimination rule to some formula B in S, then C^S can be derived from $C^{S_1},\ldots,C^{S_k}$ and $cl(B)$ using a number of resolutions that depends only on $*$. For example, if $S = A_1,A_2\supset A_3\vdash A_4$ is derived from $S_1 = A_1\vdash A_2,A_4$ and $S_2 = A_1,A_3\vdash A_4$ by the $\supset$-elimination rule, then

$cl(A_2\supset A_3) = \{\overline{\xi^{A_2\supset A_3}}\,\overline{\xi^{A_2}}\,\xi^{A_3},\ \xi^{A_2\supset A_3}\xi^{A_2},\ \xi^{A_2\supset A_3}\overline{\xi^{A_3}}\}$,

$C^S = \overline{\xi^{A_1}}\,\overline{\xi^{A_2\supset A_3}}\,\xi^{A_4}$, $C^{S_1} = \overline{\xi^{A_1}}\,\xi^{A_2}\,\xi^{A_4}$, and $C^{S_2} = \overline{\xi^{A_1}}\,\overline{\xi^{A_3}}\,\xi^{A_4}$. This leads to the following two-step derivation.



Similar cases hold for all other introduction and elimination rules, a consequence of the fact that

$$cl(*(A_1,\ldots,A_k)) \sim (\xi^{*(A_1,\ldots,A_k)} \equiv *(\xi^{A_1},\ldots,\xi^{A_k})).$$

Note that if S is obtained from S' by thinning, then $C^{S'} \subseteq C^S$. Also note that if $C_1' \subseteq C_1, \ldots, C_n' \subseteq C_n$ and $\{C_1,\ldots,C_n\} \vdash_r C$ *via* D, then there are a clause $C' \subseteq C$ and a derivation D' such that $\{C_1',\ldots,C_n'\} \vdash_r C'$ *via* D' and $\ell D' \leq \ell D$. This fact is easily proven by induction on the number of steps in D. Putting these two facts together, if for $1 \leq i \leq k$, S_i is derived from S_i' by thinning, $C_i \subseteq C^{S_i'}$, and S is derived from $S_1,\ldots,S_k$ by $*$-elimination or introduction, then there is a clause $C \subseteq C^S$ that can be derived from $C_1,\ldots,C_k$ in no more steps than the derivation of C^S from $C^{S_1},\ldots,C^{S_k}$.

All that remains is to show how to derive $C \subseteq C^S$ when one or more of the parents of sequent S is an axiom. Consider again the example of the $\supset$-elimination rule. Suppose, for example, that $A_1 = A_2$ and $A_3 = A_4$, so that both S_1 and S_2 are

axioms. Then, $C^S = \overline{\xi^{A_2}}\,\overline{\xi^{A_2 \supset A_3}}\,\xi^{A_3}$, which is one of the clauses in $cl(A_2 \supset A_3)$. In fact, if all of the parents of C^S are axioms when S is obtained by applying some introduction or elimination rule to the formula B in S, then it will always be the case that some clause of $cl(B)$ subsumes C^S. For the case where some but not all of the parents of S are axioms, consider the $\supset$-elimination example again, this time with $A_1 = A_2$, but $A_3 \neq A_4$. Then, $C^S = \overline{\xi^{A_2}}\,\overline{\xi^{A_2 \supset A_3}}\,\xi^{A_4}$ can be derived in one step from $C^{S_2} = \overline{\xi^{A_2}}\,\overline{\xi^{A_3}}\,\xi^{A_4}$ and $\overline{\xi^{A_2 \supset A_3}}\,\overline{\xi^{A_2}}\,\xi^{A_3}$, one of the clauses from $cl(A_2 \supset A_3)$. Note that in all cases, the only clause from $cl(A_2 \supset A_3)$ that was used was $\overline{\xi^{A_2 \supset A_3}}\,\overline{\xi^{A_2}}\,\xi^{A_3}$, the clause that says $(\xi^{A_2 \supset A_3} \supset (\xi^{A_2} \supset \xi^{A_3}))$. The other two clauses say $((\xi^{A_2} \supset \xi^{A_3}) \supset \xi^{A_2 \supset A_3})$, and they are used in simulations of the $\supset$-introduction rule.

If D is a derivation of the sequent $\vdash A$ in the Gentzen system with thinning (for the connectives in A), then the result of this construction will be a resolution derivation D' of some clause $C \subseteq \{\xi^A\}$ from $def(A)$, such that the number of steps in D' is bounded by some constant c times the number of steps in D. Since $def(A)$ is consistent, $C \neq \square$, so $C = \xi^A$.

∎5.6.4.2.a.

Note that in this proof, the cut rule is easily simulated by a single resolution step. But since the cut rule is not analytic, the simulating resolution derivation will have to use clauses from $cl(B)$ for some formulas B that are

not subformulas of A. This is Tseitin's proof that extension p-simulates Gentzen systems with cut. Combined with theorems 5.6.3.d. and 5.5.1.b., this gives an alternate proof that every extended Frege system p-simulates every Frege system.

Also, note that this simulation can be turned around, using the same association between clauses and sequents. Every use of the resolution rule can be simulated by a single application of the cut rule. Combined with theorem 5.5.1.a., this gives an alternate proof of theorem 5.6.3.f., that Frege systems p-simulate resolution with limited extension.

5.7. DISCUSSION

The results of this chapter have already been summarized in figure 5.2.i. What has been accomplished in this chapter is that a potentially infinite collection of proof systems has been reduced to 21 categories. Most of the work of this reduction came in subsection 5.3.1., where the tools were developed to show that all Frege systems p-simulate each other.

In terms of the $\mathcal{P}$ vs $\mathcal{NP}$ question, the most interesting part of figure 5.2.i. is the part above the double solid line. The systems below the line are no longer of interest with respect to this question, because it has been proven that they cannot be used to show that $\mathcal{P} = \mathcal{NP}$ (or even that $\mathcal{NP}$ is closed under complements). The systems just above the line (boxes 5.-12. in figure 5.2.i.) are of interest

because they are all closely related to systems below the line. Thus, there is hope that the techniques of Tseitin (as extended by Kirkpatrick and Galil) can be elaborated upon to give non-polynomial lower bounds for these systems.

The remaining three boxes at the top of figure 5.2.i. present interesting problems. (Extended tree resolution was defined by Tseitin only so that the statement of theorem 5.6.1.1.c. would be as strong as possible.) By theorem 5.6.2.2.e., the techniques of Tseitin, Kirkpatrick, and Galil cannot be used to show a non-polynomial lower bound for Frege systems and their allies. Such a lower bound (if it exists) awaits the discovery of an entirely new proof technique; perhaps even some form of diagonalization will be needed.

The relationship between Frege systems and systems with extension was discussed in subsection 5.3.2. A proof that Frege systems cannot simulate extended Frege systems would have as a corollary (in addition to the conclusion that Frege systems would not be polynomial-bounded) that fan-out one circuits cannot "simulate" (in a polynomial number of gates) fan-out two circuits.

why?

Finally, the top box in the figure, containing s-Frege systems, is the most interesting of all, because each s-Frege system p-simulates every other system shown in the figure. A non-polynomial lower bound for any s-Frege system would imply that no system shown in figure 5.2.i. is a polynomial-bounded verification system. This would provide strong circumstantial

evidence to support the speculation that $\mathcal{NP}$ is not closed under complements. Conversely, if there is any system in figure 5.2.i. that can be used to show that $\mathcal{NP}$ is closed under complements (*i.e.* if one of them is a polynomial-bounded verification system), then every s-Frege system is polynomial-bounded. Furthermore, it would not be hard to imagine that a proof that some system is polynomial-bounded could be extended to give a polynomial-time decision procedure, thus proving that $\mathcal{P} = \mathcal{NP}$.

6. Conclusion

This thesis is only a first step in the classification of propositional proof systems according to their complexity. Section 6.1. summarizes the results that have been obtained, and discusses their significance. Section 6.2. follows with a discussion of the old questions that remain unanswered and the new questions that have been raised as a result of this thesis. The thesis closes with section 6.3., which discusses suggestions for further research into the $\mathcal{P}$ vs $\mathcal{NP}$ question and proof system complexity.

6.1. SUMMARY OF RESULTS AND DISCUSSION

The question of whether or not $\mathcal{P}$ equals $\mathcal{NP}$ was raised over four years ago [Cook 1971b]. In those four years a great deal has been said about the problem, but its solution still seems remote. This thesis has attacked the problem from a new angle, and although the $\mathcal{P}$ vs $\mathcal{NP}$ problem has not been solved, some new insights have been gained. The significance of the question of whether or not $\mathcal{NP}$ is closed under complement is now more fully understood, and the concept of polynomial-bounded verification systems gives an intuitive and useful characterization of the class $\mathcal{NP}$. Finally, the idea of simulation of one verification system by another, as formalized in subsection 5.1.2., gives a way of comparing the complexities of verification systems analogous to the ways various resource-bounded reducibility notions allow complexity comparisons between languages.

Formalized logical proof systems have been around for at least 96 years [Frege 1879], but in all that time, little has been done to compare the efficiencies of different systems. The concepts of verification system and simulation developed in this thesis provide a formal framework within which proof systems can be compared with respect to lengths of proofs. These methods have been applied to large classes of systems for the propositional calculus, and it has been found that interesting statements can be made. These simulation results are summarized in figure 5.2.i.

The most interesting new simulations are those related to the top three boxes in figure 5.2.i. In the top box is the family of s-Frege systems. Since any two s-Frege systems simulate one another, no s-Frege system is more than algebraically less powerful than any other, and since s-Frege systems simulate all of the other systems investigated in this thesis, they are the most powerful (to within a polynomial) of the systems studied here.

The next box in figure 5.2.i. contains systems that are all relatively new. The new concept of extended Frege systems was introduced in subsection 5.3.2., and these systems were shown to be p-simulation equivalent to Tseitin's system of extended resolution. This makes Tseitin's observation that extended resolution simulates Gentzen systems (and thus Frege systems, as well) a trivial consequence of the fact that Frege systems are a special case of extended Frege systems.

The third box in figure 5.2.i. (box 4) contains Frege systems, natural deduction systems, and sequent systems. The terms "natural deduction" and "sequent system" have been used informally in the literature before, but in this thesis, these terms (along with the term "Frege system") are given precise meanings, and it is shown how nearly all published propositional proof systems (except the resolution-based ones) fit into one of these three categories. A large portion of the work in chapter 5. is devoted to showing that any system from one of these three classes p-simulates any other such system. The proof methods used are general, and apply to an infinite class of proof systems. They show that no new powerful connectives or inference rules (of the forms allowed by these systems), no matter how complicated or "clever", can increase the power of these systems by more than a polynomial.

Figure 5.2.i. also summarizes the results that tie the proof system simulations of this thesis to the question of the closure of $\mathcal{NP}$ under complements - the non-polynomial lower bounds. Although most of the lower bound results shown in the figure were known, the one new result (theorems 5.6.2.2.d-e.) is particularly interesting. First, it extends the results of Tseitin, Kirkpatrick, and Galil, by showing that a new family of proof systems (all systems of regular resolution with limited extension for formulas in any set of connectives containing two of $\{\neg,\equiv,\not\equiv\}$) is not polynomial-bounded. Second, this result shows that no obvious extension of Tseitin's techniques can be used to give a non-polynomial lower bound for Frege systems. Therefore, a proof that systems in any of

the top three boxes in figure 5.2.i. are not polynomial-bounded awaits the discovery of entirely new proof techniques.

6.2. OPEN QUESTIONS

This thesis has attacked the question of proof system complexity in a systematic way, and a number of questions about proof system complexity have been answered. A great many unanswered questions can be found by referring to figure 5.2.i. The cases where the relationship between two proof systems is most fully known with respect to the "simulates" relation are either when both systems lie within the same solid box in the figure (*i.e.* the two systems simulate one another) or when one system is known to simulate a second system, but the second is known not to directly simulate the first. This second alternative is indicated in the figure by the existence of a chain of downward arrows from the first system to the second and a second chain from the first system to the second following downward arrows and including at least one upward dashed arrow (followed backwards). Although no such pairs are known, there could also be two systems such that neither can simulate the other.

The most important open questions with respect to the $\mathcal{P}$ *vs* $\mathcal{NP}$ question are those connected with systems above the double solid line in figure 5.2.i. Is regularity a significant restriction on the form of derivations, or can regular resolution simulate non-regular resolution? Even if it cannot in general, can regular resolution simulate non-regular

resolution on the sets of clauses arising from Tseitin's (or Kirkpatrick's) graphs? If it can, then general resolution would be proven not to be polynomial-bounded. Can it ever really help to apply the limited extension operation to a formula that is already in conjunctive normal form, or can resolution simulate resolution with limited extension?

Although a few relationships between cut-free Gentzen systems and resolution-based systems have been derived, the full details of the relationship between these families of systems remain unknown. It is not even known if two cut-free Gentzen systems with different sets of connectives can simulate each other. It would seem unlikely that any of these cut-free or extension-free systems could simulate Frege systems, but theorem 5.6.2.2.e. now provides only a partial answer to this question.

As it was pointed out in subsection 5.3.2., the relationship between extended Frege systems and Frege systems without extension is closely tied to the relationship between fan-out two circuits and fan-out one circuits. This emphasizes the fact that both are important open questions in complexity theory.

The position of s-Frege systems at the top of figure 5.2.i. makes them the focus of a number of open (and probably very difficult) questions. Are they polynomial-bounded? Can any other proof system simulate them? What is there about the substitution rule that makes these systems so powerful?

Although their relevance to the $\mathcal{P}$ *vs* $\mathcal{NP}$ question has been ruled out, the systems below the double solid line in figure 5.2.i. also give rise to some interesting open questions. For instance, all of the examples that have been found for showing a non-polynomial lower bound for the Davis-Putnam procedure with subsumption are also non-polynomial for regular resolution. Is this necessarily the case (*i.e.* can the Davis-Putnam procedure with subsumption simulate regular resolution), or can examples be found for which regular resolution allows polynomial-length proofs while the Davis-Putnam procedure with subsumption does not? The relationships between the Davis-Putnam procedure (with or without subsumption) and two other systems are also of interest. Can the Davis-Putnam procedure simulate either tree resolution or analytic tableaux? If the answer to either question could be shown to be "no", then an incomparable pair of proof systems will have been found (with respect to the "directly simulates" relation).

Hajos calculus

6.3. SUGGESTIONS FOR FURTHER RESEARCH

The concept of verification systems is not specific to propositional calculus, nor even to logical theories in general. It would be interesting to see if reasonable verification systems could be devised for the complements of various problems from $\mathcal{NP}$, such as non-isomorphic graph pairs, non-isomorphic subgraph pairs, non-three-colourable graphs, *etc*. It would then be instructive to try to find simulations between these systems and the various systems for tautologies and unsatisfiable sets of clauses discussed in this thesis.

Perhaps such verification systems and simulations could lead to a more intuitive understanding of why the complements of $\mathcal{NP}$-complete languages are difficult to recognize.

A great many open questions were discussed in section 6.2., but nothing was said about how any of them might be solved. Any of the questions relating to proof systems that have already been shown not to be polynomial-bounded might be interesting, but each question is likely to require its own special proof technique. For example, to show that the Davis-Putnam procedure without subsumption cannot directly simulate the system of analytic tableaux for sets of clauses, one must find a family of sets of clauses for which tableaux are polynomial-bounded but Davis-Putnam derivations without subsumption are not. The same family could not be used to show that the Davis-Putnam procedure with subsumption cannot directly simulate regular resolution or that tableaux cannot simulate full truth tables. In a similar vein, it would be interesting to relate the various specialized resolution strategies (linear resolution, Method I, *etc.*) to the systems in figure 5.2.i., but it is unlikely that any of them (except possibly Method I) would be found to be as powerful as regular resolution.

Of the open questions that might have a bearing on the $\mathcal{P}$ *vs* $\mathcal{NP}$ question, the ones with a reasonable chance of being answered by known techniques are those relating to systems in figure 5.2.i. between the double solid line and the box containing Frege systems. There are several ways one might try to go about extending Tseitin's non-polynomial lower bound

to non-regular proofs. One way would be to give a general construction showing how regular resolution (or enumeration dags) could simulate non-regular resolution. Although there is no intuition to suggest how irregularities can make proofs shorter, it is equally difficult to see how to systematically remove irregularities from a proof without increasing the proof size by a potentially exponential amount. A more restricted approach that would still give a non-polynomial lower bound for general resolution would be to apply Tseitin's theorem to resolution more directly. One way to do this would be to show that regular resolution simulates non-regular resolution just for Tseitin's (or Kirkpatrick's) graph-derived examples. A second approach to try would be to analyze the machinery developed within Tseitin's proof and to use that machinery in new ways to show that the regularity restriction in Tseitin's proof is not needed.

The relationship between resolution and resolution with limited extension should be investigated more carefully, and an attempt should be made to extend theorems 5.6.2.2.d-e. to all (regular) resolution with limited extension. The relationship between enumeration dags and the Gentzen system for sets of clauses is somewhat analogous to the relationship between the Davis-Putnam procedure and regular resolution. In both cases one "step" in the first system corresponds to a number of "steps" in the second. Perhaps similar ideas could be used in the two cases to show that the first system can (or cannot) simulate the second.

In order to prove non-polynomial lower bounds for Frege systems, extended Frege systems, and s-Frege systems, a whole new approach is needed. The simulations involving these systems show that they are all very flexible, and it seems doubtful that any combinatorial argument (such as in Tseitin's proof for regular resolution) could be used for these systems. Perhaps one of these systems is powerful enough that somehow it could be used to "talk about" polynomial-time-bounded computations in such a way that diagonal arguments could be used. See [Cook 1975a] for an interesting start in this direction.

BIBLIOGRAPHY

Note: The numbers on the first line of an entry indicate the pages where that reference is referred to.

[Anderson & Bledsoe 1970] 81
A linear format for resolution with merging and a new technique for establishing completeness, by R. Anderson and W. W. Bledsoe. *J. ACM*, vol. 17, no. 3 (July 1970), pp. 525-534.

[Bauer *et al.* 1973] 87
A note on disjunctive form tautologies, by M. Bauer, D. Brand, M. Fischer, A. Meyer, and M. Paterson. *SIGACT News*, vol. 5, no. 2 (April 1973), pp. 17-20.

[Bennett 1962] 19,22
On Spectra, by James Hallam Bennett. Doctoral dissertation, Princeton University, May 1962.

[Cobham 1965] 17,36
The intrinsic computational difficulty of functions, by Alan Cobham. *Proc. of the 1964 International Congress on Logic, Methodology, and Philosophy of Science*, Yehasura Bar-Hillel (Ed.), North-Holland, Amsterdam, 1965, pp. 24-30.

[Cook 1971a] 20
Characterizations of pushdown machines in terms of time-bounded computers, by Stephen A. Cook. *J. ACM.*, vol. 18, no. 1 (Jan. 1971), pp. 4-18.

[Cook 1971b] 2,20,25,30,40,229
The complexity of theorem-proving procedures, by Stephen A. Cook. *Proc. Third ACM Symp. on Theory of Computing*, Shaker Heights, Ohio, May 3-5 1971, pp. 151-158.

[Cook 1971c] 6,200
Examples for the Davis-Putnam procedure, by Stephen A. Cook. Unpublished manuscript, June 1971.

[Cook 1972a] 18
Linear time simulation of deterministic two-way pushdown automata, by Stephen A. Cook. *Proc. of IFIP Congress 71, Information Processing 71, Vol. I, Foundations and Systems*, C. V. Frieman (Ed.), North-Holland, Amsterdam, 1972, pp. 75-80.

[Cook 1972b] 6,81,200
Class notes from CSC 2409S, Logic and Mechanical Proof Procedures, by Stephen A. Cook. Dept. of Computer Sci., Univ. of Toronto, Spring 1972.

[Cook 1973] 7,183,200
An exponential example for analytic tableaux (draft), by Stephen A. Cook. Unpublished manuscript, March 1973.

[Cook 1975a] 161,237
Feasibly constructive proofs and the propositional calculus, preliminary version, by Stephen A. Cook. *Proc. Seventh ACM Symp. on Theory of Computing*, Albuquerque, New Mexico, May 5-7 1975, pp. 83-97.

[Cook 1975b] 157
Presentation in Albuquerque Conference, by Stephen A. Cook. May 5-7 1975.

[Cook & Reckhow 1972] 18
Diagonal theorems for random access machines, by Stephen A. Cook and Robert A. Reckhow. *Tech. Rep. No. 42*, Dept. of Computer Sci., Univ. of Toronto, June 1972.

[Cook & Reckhow 1973] 18
Time-bounded random access machines, by Stephen A. Cook and Robert A. Reckhow. *J. Computer and System Sciences*, vol. 7, no. 4 (Aug. 1973), pp. 354-375.

[Cook & Reckhow 1974] 7,12,30,158,183
On the lengths of proofs in the propositional calculus, preliminary version, by Stephen Cook and Robert Reckhow. *Proc. Sixth ACM Symp. on Theory of Computing*, Seattle, Washington, April 30-May2 1974, pp. 135-148. Corrections in *SIGACT News*, vol. 6, no. 3 (July 1974), pp. 15-22.

[Davis & Putnam 1960] 5,78
A computing procedure for quantification theory, by Martin Davis and Hilary Putnam. *J. ACM*, vol. 7 (1960), pp. 201-215.

[Dunham & North 1962] 76
Theorem testing by computer, by B. Dunham and J. H. North. *Proc. of the Symp. on Math. Theory of Automata*, Jerome Fox (Ed.), 1962, pp. 173-177.

[Earley 1970] 18
An efficient context-free parsing algorithm. *Comm. ACM*, vol. 13, no. 2 (Feb. 1970), pp. 94-102.

[Edmonds 1965] 18
Paths, trees, and flowers, by Jack Edmonds. *Canadian J. Math.*, vol. 17 (1965), pp. 449-467.

[Fitch 1952] 64
Symbolic Logic, by F. B. Fitch. Ronald Press Co., New York, 1952.

[Frege 1879] 60,63,230
Begriffsschrift, eine der arithmetischen nachgebildete Formelsprache des reinen Denkens, by G. Frege. Halle, 1879.

[Galil 1975] 4,7,12,197,204,213,216
The complexity of resolution procedures for theorem proving in the propositional calculus, by Zvi Galil. *Tech. Rep. TR 75-239*, Dept. of Computer Sci., Cornell Univ., May 1975, 112 pages.

[Garey *et al.* 1974] 41
Some simplified NP-complete problems, by M. R. Garey, D. S. Johnson, and L. Stockmeyer. *Proc. Sixth ACM Symp. on Theory of Computing*, Seattle, Washington, April 30-May 2 1974, pp. 47-63.

[Hartmanis 1971] 18
Computational complexity of random access stored program machines, by J. Hartmanis. *Math. Systems Theory*, vol. 5, no. 3 (1971), pp. 232-245.

[Hartmanis & Stearns 1965] 17,31
On the computational complexity of algorithms, by J. Hartmanis and R. E. Stearns. *Trans. Amer. Math. Soc.*, vol. 117 (1965), pp. 285-306.

[Hennie & Stearns 1966] 17
Two-tape simulation of multitape Turing machines, by F. C. Hennie and R. E. Stearns. *J. ACM*, vol. 13, no. 4 (Oct. 1966), pp. 533-546.

[Hilbert & Ackermann 1950] 63
Principles of Mathematical Logic, by D. Hilbert and W. Ackermann. Chelsea Pub. Co., New York, 1950.

[Hopcroft & Ullman 1969] 17,30
Formal Languages and their Relation to Automata, by John E. Hopcroft and Jeffrey D. Ullman. Addison-Wesley, Reading, Mass., 1969.

[Karp 1973] 23,30,40,41
Reducibility among combinatorial problems, by Richard M. Karp. *Complexity of Computer Computations*, R. E. Miller and J. W. Thatcher (Eds.), Plenum Press, New York, 1973, pp. 85-103.

[Krapchenko 1971] 141
Complexity of the realization of a linear function in the class of π-circuits, by V. M. Khrapchenko. *Math. Notes of the Acad. of Sci. of the USSR*, 1971, pp. 21-23.

[Kirkpatrick 1974] 7,200
Topics in the complexity of combinatorial algorithms, by David G. Kirkpatrick. Doctoral thesis, *Tech. Rep. No. 74*, Dept. of Computer Sci., Univ. of Toronto, December 1974.

[Kleene 1952] 63
Introduction to Metamathematics, by S. C. Kleene. D. Van Nostrand, Reinhole, Inc., Princeton, New Jersey, 1952.

[Kleene 1967] 4,5,13,63,64,68,70,73,89
Mathematical Logic, by S. C. Kleene. Wiley, New York, 1967.

[Knuth 1974a] 30
A terminological proposal, by D. E. Knuth. *SIGACT News*, vol. 6, no. 1 (Jan. 1974), pp. 12-18.

[Knuth 1974b] 30
Postscript about NP-hard problems, by D. E. Knuth. *SIGACT News*, vol. 6, no. 2 (April 1974), pp. 15-16.

[Kowalski & Hayes 1969] 4,90
Semantic trees in automatic theorem proving, by R. Kowalski and P. Hayes. *Machine Intelligence, Vol. 4*, B. Meltzer and D. Michie (Eds.), New York, 1969, pp. 87-101.

[Kuroda 1964] 1
Classes of languages and linear-bounded automata, by S. Y. Kuroda. *Inf. and Control*, vol. 7, no. 2, pp. 207-223.

[Ladner *et al.* 1974] 30,38
Comparison of polynomial-time reducibilities, by Richard Ladner, Nancy Lynch, and Alan Selman. *Proc. Sixth ACM Symp. on Theory of Computing*, Seattle, Washington, April 30-May 2 1974, pp. 110-121.

[Loveland 1970] 6,81
A linear format for resolution, by D. Loveland. *Proc. of the IRIA 1968 Symp. on Automatic Demonstration*, Lecture Notes on Mathematics no. 125, Springer-Verlag New York, Inc., New York, 1970.

[Mendelson 1964] 4,63,96
Introduction to Mathematical Logic, by Elliott Mendelson. D. Van Nostrand, Reinhold, Inc., Princeton, New Jersey, 1964.

[Meyer & Stockmeyer 1972] 25,38,43
The equivalence problem for regular expressions with squaring requires exponential space, by A. R. Meyer and L. J. Stockmeyer. *Proc. 13th Symp. on Switching & Automata Theory*, IEEE Computer Soc., October 1972, pp. 125-129.

[Pratt 1974] 141
The effect of basis on size of Boolean expressions, by V. R. Pratt. Extended abstract, Massachusetts Institute of Technology, November 1974.

[Pratt 1975] 25,42
Every prime has a succinct certificate, by V. R. Pratt To appear in *SIAM J. on Computing*.

[Quine 1955] 76
A way to simplify truth functions, by W. V. Quine. *Amer. Math. Monthly*, vol. 62 (1955), pp. 627-631.

[Robinson 1965a] 5,76
A machine-oriented logic based on the resolution principle, by J. A. Robinson. *J. ACM*, vol. 12 (1965), pp. 23-41.

[Robinson 1965b] 6,81
Automatic deduction with hyper-resolution, by J. A. Robinson. *International J. of Computer Math.*, vol. 1 (1965), pp. 227-234.

[Robinson 1968] 4,90
The generalized resolution principle, by J. A. Robinson. *Machine Intelligence, Vol. 3*, D. Michie (Ed.), American Elsevier, New York, 1968, pp. 77-94.

[Rogers 1967] 23,26
Theory of Recursive Functions and Effective Computability, by Hartley Rogers, Jr. McGraw-Hill, New York, 1967.

[Sethi 1973] 41
Complete register allocation problems, by Ravi Sethi. *Proc. Fifth ACM Symp. on Theory of Computing*, Austin Texas, April 30-May 2 1973, pp. 182-195.

[Shepherdson & Sturgis 1963] 18
Computability of recursive functions, by J. C. Shepherdson, and H. E. Sturgis. *J. ACM*, vol. 10, no. 2 (April 1963), pp. 217-255.

[Shoenfield 1967] 63,97
Mathematical Logic, by Joseph R. Shoenfield. Addison-Wesley, Reading, Mass., 1967.

[SIGACT 1973] 24
Proc. Fifth ACM Symp. on Theory of Computing, Austin, Texas, April 30-May 2 1973.

[SIGACT 1974] 24
Proc. Sixth ACM Symp. on Theory of Computing, Seattle, Washington, April 30-May 2 1974.

[Simon 1971] 200
On the time required by the Davis-Putnam tautology recognition algorithm, by Imre Simon. *Res. Rep. CSRR-2050*, Dept. of Applied Analysis and Computer Sci., Univ. of Waterloo, June 1971.

[Slagle 1967] 81
Automatic theorem-proving with renamable and semantic resolution, by J. Slagle. *J. ACM*, vol. 14, no. 4 (Oct. 1967), pp. 687-697.

[Smullyan 1968] 5,68,73
First Order Logic, by Raymond M. Smullyan. Springer-Verlag, New York, 1968.

[Spira 1971] 139
On time hardware complexity tradeoffs for Boolean functions, by P. M. Spira. *Fourth Hawaii International Symp. on Systems Sci.*, 1971, pp. 525-527.

[SWAT 1972] 24
Proc. 13th Symp. on Switching & Automata Theory, IEEE Computer Soc., October 1972.

[SWAT 1973] 24
Proc. 14th Symp. on Switching & Automata Theory, IEEE Computer Soc., October 1973.

[Thomason 1970] 5,64
Symbolic Logic: An Introduction, by Richard H. Thomason. Macmillan, Toronto, 1970.

[Tseitin 1968] 6,10,12,79,80,85,87,107,190,200,204,211,222
On the complexity of derivation in propositional calculus, by G. S. Tseitin. *Studies in Constructive Math. and Math. Logic, Part II*, A. O. Slisenko (Ed.), 1968, pp. 115-125.

[Wos *et al.* 1964] 5,81
The unit preference strategy in theorem proving, by L. Wos, D, Carson, and G. Robinson. *Proc. of the AFIPS Fall Joint Computer Conf.*, vol. 26 (1964), pp. 616-621.

[Wos *et al.* 1965] 5,81
Efficiency and completeness of the set of support strategy in theorem-proving, by L. Wos, G. Robinson, and D. Carson. *J. ACM*, vol. 12, no. 4 (Oct. 1965), pp. 536-541.