STUDIES IN THE COMPLETENESS AND EFFICIENCY OF THEOREM-PROVING BY RESOLUTION

by

Robert Kowalski

Ph.D. thesis

University of Edinburgh

April 1970

ABSTRACT

Inference systems $\mathcal{T}$ and search strategies Σ for $\mathcal{T}$ are distinguished from proof procedures $\mathcal{P} = (\mathcal{T}, \Sigma)$. The completeness of procedures is studied by studying separately the completeness of inference systems and of search strategies. Completeness proofs for resolution systems are obtained by the construction of semantic trees. These systems include minimal α-restricted binary resolution, minimal α-restricted M-clash resolution and maximal pseudo-clash resolution. Certain refinements of hyper-resolution systems with equality axioms are shown to be complete and equivalent to refinements of the paramodulation method for dealing with equality.

The completeness and efficiency of search strategies for theorem-proving problems is studied in sufficient generality to include the case of search strategies for path-search problems in graphs. The notion of theorem-proving problem is defined abstractly so as to be dual to that of and/or tree. Special attention is given to resolution problems and to search strategies which generate simpler before more complex proofs.

For efficiency, a proof procedure $(\mathcal{T}, \Sigma)$ requires an efficient search strategy Σ as well as an inference system $\mathcal{T}$ which admits both simple proofs and relatively few redundant and irrelevant derivations. The theory of efficient proof procedures outlined here is applied to proving the increased efficiency of the usual method for deleting tautologies and subsumed clauses. Counter-examples are exhibited for both the completeness and efficiency of alternative methods for deleting subsumed clauses.

The efficiency of resolution procedures is improved by replacing the single operation of resolving a clash by the two aperations of generating factors of clauses and of resolving a clash of factors. Several factoring methods are investigated for completeness. Of these the m-factoring method is shown to be always more efficient than the Wos-Robinson method.

ACKNOWLEDGEMENTS

This research has been supported by an I.B.M. Fellowship administered by Imperial College and more recently by the Science Research Council.

I am indebted to my supervisor Dr. Bernard Meltzer for valuable encouragement and guidance and to my colleague Mr. Pat Hayes for many hours of rewarding collaboration and criticism. Thanks are also due for helpful discussions with my colleagues Miss Isobel Smith and Mr. Donald Kuehner as well as with Dr. Cordell Green, Dr. Ira Pohl, Professor J.A. Robinson and Dr. Larry Wos.

I am grateful to the many people who have shared the burden of typing this thesis.

To my wife Danusia and to my daughters Dania and Tania I am especially grateful for their patience and sacrifice.

TABLE OF CONTENTS

Chapter 0

The subject of this thesis is the completeness and efficiency of various theorem-proving methods. These methods apply primarily to resolution inference systems [39] and are investigated by means of theoretical, rather than experimental, studies. The theoretical methodology of these studies implies that they are oriented mainly toward automatic, rather than interactive, theorem-proving. Relationships between completeness and efficiency are remarked upon throughout the body of this thesis and are explored more thoroughly in this preliminary chapter.

The theorem-proving methods investigated in this thesis include deletion rules, factoring restrictions and minimality, α-ordering and M-clash restrictions. Chapters 1 and 2 concentrate respectively on the syntax and semantics of resolution systems. In chapter 3, restrictions on the paramodulation method for dealing with equality [38] are studied and related, for efficiency and completeness, to the hyper-resolution method using equality axioms [20]. The completeness and efficiency of search strategies for theorem-proving problems are investigated in chapter 4. Parts of chapters 2, 3 and 4 have already been reported in [17], [20] and [21] respectively.

The major function of this introductory chapter is to outline and defend a theory of efficiency for automatic theorem-proving. This theory incorporates conclusions formulated after the investigations of chapters 1-4 and is intended to provide a framework within which these investigations can be evaluated. For this latter reason we have chosen to place this chapter at the beginning, rather than at the end, of this thesis.

Section 0.1 introduces and discusses the significance of a fundamental distinction between inference systems $\mathcal{T}$, search strategies Σ for $\mathcal{T}$ and proof procedures $\mathcal{P} = (\mathcal{T}, \Sigma)$. Relationships between the efficiency of proof procedures and properties of inference systems are investigated in section 0.2. Further investigations, in 0.3, relate the efficiency of proof procedures to the completeness of inference systems and search strategies. An earlier version of a part of this chapter was reported and discussed in a panel discussion at the Fourth Annual Systems Symposium [19].

0.1 Proof Procedures, Inference Systems and Search Strategies.

A fundamental distinction, basic to the study of efficiency, is that between a system of axioms and inference rules $\mathcal{T}$ and a proof procedure $\mathcal{P} = (\mathcal{T}, \Sigma)$ for obtaining proofs admissible for $\mathcal{T}$ by means of a search strategy Σ. In the case of resolution proof procedures, $\mathcal{T}$ is a function of input sets of clauses S_0. Thus $\mathcal{T} = \mathcal{T}(S_0)$ consists of the set of clauses S_0 together with resolution and possibly factoring rules. We also write $\mathcal{J} = \mathcal{J}(S_0)$ when S_0 is the set of axioms of a proof procedure $(\mathcal{J}, \Sigma)$ which derives theorems directly from axioms. Thus in general, an inference system $\mathcal{J}(S_0)$ consists of an initial set of sentences S_0 together with inference rules Γ which can be applied to construct derivations from S_0. (The set S_0 may be fixed, when it consists of a given set of axioms, or may be a free variable, when it stands for a set of axioms supplemented by different special hypotheses and, possibly, by negations of theorems to be proved.) Derivations constructible from sentences in S_0 by means of the rules Γ are said to be admissible for $\mathcal{J}$. The set S^* of all sentences derivable from S_0 is called the search space determined by $\mathcal{J}(S_0)$. A search strategy Σ for $\mathcal{J}$ is an algorithm for generating derivations admissible for $\mathcal{J}$ in order to eventually generate a proof of a given theorem. Thus Σ induces an ordering of occurences of sentences from S^* defined by the sequence in which derivations of these sentences are generated by Σ. We distinguish between an admissible derivation $\mathcal{D}$ of a sentence C and the set of sentences generated by Σ before obtaining a first proof of C. $\mathcal{D}$ contains only sentences necessary for proving C whereas Σ will almost always generate, before proving C, proofs of sentences irrelevant to a first proof of C. Search strategies for resolution systems include level saturation, unit preference [53], fewest components [50] and diagonal search (chapter 4).

Although we restrict attention to proof procedures of the form (Γ, Σ), it should be noted that not all proof procedures can be analysed as consisting of inference systems Γ and search strategies Σ for generating proofs forward from axioms (or input sets of clauses) to theorems (or $\square$). In general it is necessary to consider procedures $\hat{\mathcal{P}} = (\hat{\Gamma}, \hat{\Sigma})$ which generate proofs backwards from theorems to axioms of $\hat{\Gamma}$ by means of a search strategy $\hat{\Sigma}$. The system $\hat{\Gamma}$ is dual to an inference system in the sense that its operations are the inverse of inference rules Γ. The search space $\hat{S}^*$ determined by $\hat{\Gamma}$ consists of all sentences which can be used to derive the given theorem and is structured in the form of an and/or tree [49]. Beth, Kleene [18] and other researchers have observed that semantic tableau procedures obtain proofs constructible by means of Gentzen-type axioms and inference rules Γ. The semantic tableau method consists of a search algorithm $\hat{\Sigma}$ for the search space $\hat{S}^*$ determined by a system $\hat{\Gamma}$ dual to Γ. It is interesting to note that Beth's original procedure $\hat{\mathcal{P}}$ employed an incomplete $\hat{\Sigma}$ which resulted in the incompleteness of $\hat{\mathcal{P}}$ [29]. The Geometry Theorem-Proving Machine [9] is an incomplete procedure of the form $(\hat{\Gamma}, \hat{\Sigma})$ employing incomplete $\hat{\Gamma}$.

Given a system Γ or $\hat{\Gamma}$ it is often possible to construct a corresponding dual system. A system $\hat{\mathcal{T}}$, dual to a resolution system $\mathcal{T}$, can be constructed by including in the search space $\hat{S}^*$ for $\hat{\mathcal{T}}$ all clauses which can occur in resolution derivations of the null clause. $\hat{S}^*$ defined in this way is the set of all clauses constructible from a potentially infinite set of variables and from the predicate and function symbols occurring in S_0. For the system $\hat{\mathcal{T}}$ of Slagle's program for symbolic integration [49], an inference system $\mathcal{T}(S_0)$, dual to $\hat{\mathcal{T}}$, can be constructed by defining S_0 to be the set of integration formulae of some integration table and by defining Γ to be a set of rules, inverse to those of $\hat{\mathcal{T}}$, for constructing new formulae from existing ones. The search spaces S^* and $\hat{S}^*$ for a system and its dual need not be identical. For the resolution systems $\mathcal{T}$ and $\hat{\mathcal{T}}$ above, $S^* \subset \hat{S}^*$, whereas for the symbolic integration systems $\mathcal{T}$ and $\hat{\mathcal{T}}$, $\hat{S}^* \subset S^*$. ($X \subset Y$ if X is properly contained in Y.)

The notions of and/or tree problem (for systems $\hat{\mathcal{T}}$) and theorem-proving problem (for systems $\mathcal{T}$, chapter 4) are dual to one another and both generalise the tree (or graph) problem [8] of finding a path between initial and terminal nodes. Given a system $\mathcal{T}$ or $\hat{\mathcal{T}}$, having constructed a system dual to the one given, it is possible to construct search strategies for the combined search space $S^* \cup \hat{S}^*$.

Such strategies have been studied for the tree problem and are referred to as bi-directional search. Many of these methods, including the cardinality comparison method of [32], extend to the more general situation. It is interesting to note that when the cardinality comparison method is applied to the resolution or integration systems $\mathcal{T}$ and $\hat{\mathcal{T}}$ above, it avoids generating objects in $\hat{S}^* - S^*$, for the resolution example, and in $S^* - \hat{S}^*$, for the integration example.

The remainder of this thesis is concerned explicitly with proof procedures of the form $(\mathcal{T}, \Sigma)$. Despite this restriction, most of the remarks in this chapter apply equally to procedures $(\hat{\mathcal{T}}, \hat{\Sigma})$ as well as to bi-directional procedures more generally.

Proof procedures $(\mathcal{T}, \Sigma)$ can usually be analysed in more than one way as inference system and search strategy. Set of support resolution can be treated as either a restricted inference rule determining a restricted search space or as a restricted search strategy for an unrestricted resolution rule. More generally, restrictions on derivations generated by $\wp$ can often be incorporated into the definition of either $\mathcal{T}$ or Σ. The significance of an appropriate analysis $(\mathcal{T}, \Sigma)$ of $\wp$ is related to the distinct notions of completeness which can be formulated for $\mathcal{T}$, Σ and $\wp$.

An inference system $\mathcal{T}(S_0)$ is complete for a set of sentences $\mathfrak{S}$ if, whenever S_0 implies a sentence $C \in \mathfrak{S}$, then there exists a derivation of C from S_0 which is admissible for $\mathcal{T}$. $\mathcal{T}(S_0)$ is refutation complete for $\mathfrak{S}$ if, whenever $S_0 \in \mathfrak{S}$ implies a contradiction then there exists an admissible derivation from S_0 of an effectively recognizable contradiction (e.g. $\Box$). The existence of admissible derivations and therefore the completeness or refutation completeness of inference systems $\mathcal{T}$ is independent of search strategies for $\mathcal{T}$. A search strategy Σ for $\mathcal{T}$ is complete for $\mathcal{T}$ if Σ will eventually generate all derivations admissible for $\mathcal{T}$ (assuming that Σ can continue generating derivations after obtaining a first proof of a desired theorem). Σ may be complete or incomplete independently of the completeness of $\mathcal{T}$. In particular $\mathcal{T}$ may be complete for $\mathfrak{S}$ but Σ may be incomplete if Σ will not generate some derivation admissible for $\mathcal{T}$. On the other hand, Σ may be complete when $\mathcal{T}$ is incomplete, by virtue of exhaustively generating all derivations admissible for $\mathcal{T}$. A proof procedure $\mathcal{P} = (\mathcal{T}(S_0), \Sigma)$ is complete for $\mathfrak{S}$ (refutation complete for $\mathfrak{S}$) if whenever S_0 implies C $\in \mathfrak{S}$ ($S_0 \in \mathfrak{S}$ and C some effectively recognizable contradiction) then Σ eventually generates an admissible derivation of C from S_0. Thus $\mathcal{P}$ can be complete (or refutation complete) for $\mathfrak{S}$ even when Σ

is incomplete for $\mathcal{T}$: for example, when $\mathfrak{S}$ is the set of all sets of clauses, $\mathcal{P} = (\mathcal{T}, \Sigma)$ is set of support resolution, $\mathcal{T}$ is unrestricted resolution and Σ generates all and only those derivations admissible for $\mathcal{T}$ which are compatible with the set of support restriction. However $\mathcal{P}$ is incomplete for $\mathfrak{S}$ if $\mathcal{T}$ is incomplete for $\mathfrak{S}$ and, equivalently, $\mathcal{T}$ is complete for $\mathfrak{S}$ if $\mathcal{P}$ is. The set $\mathfrak{S}$ is usually the set $\mathfrak{S}^*$ of all sentences constructible in the language of $\mathcal{T}$. Situations where $\mathfrak{S} \subset \mathfrak{S}^*$ occur in the case of decision procedures which are incomplete for $\mathfrak{S}^*$ but complete for the decidable subset $\mathfrak{S}$. More generally all proof procedures incomplete for $\mathfrak{S}^*$ are complete for some proper subset $\mathfrak{S} \subset \mathfrak{S}^*$ (e.g. $\mathfrak{S} = \emptyset$). Unless stated otherwise, the set $\mathfrak{S}$ relative to which inference systems and proof procedures are evaluated both for completeness and for efficiency is taken to be the set for which $\mathcal{P}$ is expected to prove theorems. (More detailed discussion related to this topic is contained in the first part of section 0.3.) For the most part all remarks concerning completeness apply equally to refutation completeness. Unless stated otherwise, the term "completeness" is used to refer to both kinds of completeness.

It is interesting to note that the original completeness proofs for unrestricted resolution [39], hyper-resolution [40], clash resolution [42] and AM-clash resolution [51] are all stated directly for proof procedures $(\mathcal{T}, \Sigma)$ where Σ can be interpreted as a complete level saturation search for $\mathcal{T}$. These completeness proofs imply the completeness of the corresponding resolution system $\mathcal{T}$ and also of resolution procedures $(\mathcal{T}, \Sigma')$ for any complete Σ' for $\mathcal{T}$. The original completeness proofs for set of support [54], resolution with merging [2] and linear resolution [23], [24] are stated directly for inference systems $\mathcal{T}$. All completeness theorems in this thesis are stated explicitly either for inference systems or for search strategies.

When analysing a procedure $\mathcal{P}$ for an inference system $\mathcal{T}$ and search strategy Σ it is convenient to have $\mathcal{T}$ incorporate the logical restrictions of $\mathcal{P}$ and to have Σ incorporate its heuristic restrictions. Suppose that a procedure $\mathcal{P} = (\mathcal{T}, \Sigma)$ is complete with incomplete Σ and suppose that there exists an equivalent procedure $(\mathcal{T}', \Sigma') = (\mathcal{T}, \Sigma)$ such that Σ' is complete for $\mathcal{T}'$. The heuristic restrictions of $\mathcal{P}$ incorporated in Σ are transferred to logical restrictions in $\mathcal{T}'$. In the following discussions we shall assume that proof procedures are analysed in a way which minimizes their heuristic restrictions. This convention implies that restrictions such as set of support, P_1-resolution, etc. are incorporated in the inference system of resolution procedures. In general whenever $\mathcal{P} = (\mathcal{J}, \Sigma) = (\mathcal{J}', \Sigma')$ and $S'^* \subset S^*$ then $\mathcal{P} = (\mathcal{J}', \Sigma')$ is considered to be the preferred analysis of $\mathcal{P}$. If Σ' is complete for $\mathcal{J}'$ then the heuristics incorporated in Σ' are restricted to imposing an ordering on the sequence in which admissible derivations are generated by Σ'.

The distinction between inference systems $\mathcal{J}$ and proof procedures $\mathcal{P} = (\mathcal{J}, \Sigma)$ induces an additional distinction between measures of simplicity (or complexity) of derivations admissible for $\mathcal{J}$ and measures of ease (or difficulty) of obtaining such derivations. A related distinction between notions of complexity and difficulty can be observed in the context of informal mathematics. Informally proved theorems almost always have more than one proof (derivation), some of which are simpler than others. In particular it is not uncommon for early proofs of theorems to be more complex than later proofs. Indeed an important part of mathematical activity is concerned with just this simplification of complex proofs. It is not difficult to construct precise measures of complexity for informally obtained proofs. What is wanted is that such measures be compatible with intuitive notions of complexity. The number of distinct sentences occurring in a given derivation provides a measure of complexity which is approximately satisfactory in this respect. A more appropriate measure might be the total number of distinct symbols occurring within the given derivation or perhaps some combined measure giving different weights to numbers of sentences and numbers of symbols. Measures formulated for informal derivations can be applied to derivations admissible for formally defined inference systems. For inference systems (such as resolution) which admit derivations $\mathcal{D}$ of tree-like structure, the largest number of sentences occurring in any one branch of $\mathcal{D}$ (level of $\mathcal{D}$) has often been treated as a measure of the complexity of $\mathcal{D}$. The preceding and subsequent remarks suggest that a more appropriate measure might involve the total number of distinct sentences or symbols occurring in $\mathcal{D}$. In any case, for the remainder of this thesis it suffices for the most part to assume only that complexity of derivations is defined in such a way that no derivation is ever simpler than any of its subderivations. In this connexion we note that contractions and semi-contractions $\mathcal{D}'$ of derivations $\mathcal{D}$ (section 1.10) tend to be simpler (and never significantly more complicated) than $\mathcal{D}$.

The difficulty of informally obtaining a proof of a given theorem coincides with the total effort needed to obtain a first proof and includes work done on unsuccessful attempts. This effort can be quantified in a variety of ways: in particular, by the total amount of time expended or by the total number of sentences (or symbols) constructed before obtaining a first proof. Similar measures of difficulty can be applied to theorems proved by formally defined proof procedures. As a first approximation it is convenient to identify this difficulty with the total number of occurrences of sentences (or derivations) generated before a first proof. Compared with measuring difficulty in terms of time, this measure has the advantage of allowing comparisons of difficulty to be made among proof procedures and informal theorem-provers independently of the computer implementation of proof procedures.

For the first proof of a given theorem, whether obtained formally or informally, measures of difficulty can be applied to measure efficiency. More specifically we shall regard a proof method $\mathcal{P}_1$ as more efficient than a method $\mathcal{P}_2$ for proving a given theorem when the number of derivations generated by $\mathcal{P}_1$ before obtaining a first proof is less than the number generated by $\mathcal{P}_2$. This measure of efficiency allows comparison of proof procedures relative to a given theorem, it does not provide a direct means of evaluating for efficiency a single proof procedure which is intended to obtain proofs of theorems within some set of sentences $\mathcal{S} \subseteq \mathcal{S}^*$. For this purpose we shall assume that some informal proof method $\mathcal{P}^*$ is given and assumed to be an ideal to which all formal proof procedures are compared for efficiency within $\mathfrak{S}$. Thus, in particular, when we require of $\mathcal{P}$ that formal difficulties coincide with informal difficulties, this requirement can be interpreted as imposing a norm that for all theorems in $\mathfrak{S}$ difficulties are equal both for $\mathcal{P}$ and $\mathcal{P}^*$, or more liberally that for all theorems in $\mathfrak{S}$ difficulties for $\mathcal{P}$ and $\mathcal{P}^*$ differ by at most some given ϵ (ϵ may be allowed to depend upon n, the difficulty of proving the given theorem by means of $\mathcal{P}^*$), or still more liberally that average differences in difficulties for theorems in $\mathfrak{S}$ are no greater than ϵ (where ϵ may depend upon n). Although none of these precise formulations admits an effective test for determining whether $\mathcal{P}$ meets the desired requirement, they serve the important function of clarifying the intended interpretation of the more imprecise formulation. We intend to identify the requirement that a proof procedure $\mathcal{P}$ be efficient with the requirement that difficulties of proofs of theorems in $\mathfrak{S}$ obtained by $\mathcal{P}$ coincide with those of proofs obtained by $\mathcal{P}^*$. We intend further that this latter requirement be interpreted in the most liberal sense. Various objections to the identification of our requirement with that of efficiency can be countered by elaborating upon the choice of the informal method $\mathcal{P}^*$ or by liberalising the tolerance function ϵ. (We assume that $\mathcal{P}^*$ is never less efficient than any formal method $\mathcal{P}$.

For, in particular, $\mathcal{P}^*$ can be assumed to be intelligent enough to be capable of employing the methods of $\mathcal{P}$. Recall too that difficulties are measured in terms of the number of alternative possible subproofs examined before finding a first proof - and not in terms of time.) In any case we do not intend so much to define an absolute standard of efficiency as much as we intend to explicate in useful form the intuitive notion which we interpret as being relative to variable standards of human performance. The value of this explication depends upon its utility for founding the theory of efficiency presented in this chapter.

As in the case of informal methods of proof, the efficiency of a proof procedure $\mathcal{P} = (\mathcal{T}, \Sigma)$ is related to the complexity of proofs admissible for $\mathcal{T}$. In particular, if for a given theorem $\mathcal{T}$ admits no proofs containing fewer than n sentences, then n is a lower bound on the difficulty of proving the theorem by means of $\mathcal{P}$. It has been common to confuse complexities of simplest proofs admissible for inference systems with the efficiency of proof procedures. This identification of simplicity with efficiency is a mistaken one since, for both formal and informal methods, not only may simple proofs be difficult to obtain but complex proofs may sometimes be easier to find than simpler ones. Similarly mistaken is the identification of efficiency with the degree to which the ratio of the number of sentences occurring in a first proof to the number of sentences generated before finding that proof approaches unity. Relative to this measure a proof procedure is most efficient when it generates only sentences occurring in a single first proof - independently of the complexity of that proof which may be so great that it contains far more sentences than is tolerable by comparison with informal methods. Relationships between the complexities of proofs and the efficiency of proof procedures ($\mathcal{T}$, Σ) depend upon several factors including the numbers and kinds of redundant and irrelevant derivations admitted by $\mathcal{T}$ and the efficiency of the search strategy Σ for $\mathcal{T}$. Before investigating in section 0.2 properties of inference systems which are relevant to the efficiency of proof procedures, we conclude this section with several remarks concerning search strategies.

Whereas proof procedures admit a notion of efficiency, no such notion applies to inference systems in the absence of search strategies. In contrast, the efficiency of a search strategy Σ for an inference system $\mathcal{T}$ can be studied independently of the efficiency of ($\mathcal{T}$, Σ). For a given $\mathcal{T}$, a strategy Σ_1 is more efficient than Σ_2 when Σ_1 generates fewer derivations than does Σ_2 before the generation of a first proof. A proof procedure $\mathcal{P}$ = ($\mathcal{T}$, Σ) can be hopelessly inefficient even when

Σ is most efficient for $\mathcal{T}$. Such a situation arises in the example of the preceding paragraph where $\mathcal{T}$ admits proofs of only intolerable complexity and Σ generates no sentences not occurring in the first and simplest proof of the given theorem. Although efficient search strategies cannot guarantee efficient proof procedures, efficient $(\mathcal{T}, \Sigma_1)$ can be rendered intolerably inefficient by employing, instead of Σ_1, an inefficient search strategy Σ_2. In a worst case, Σ_2 might be an incomplete search strategy which, generating a potential infinity of irrelevant derivations, delays forever the generation of proofs. Σ_2 might be complete but delay the generation of a first proof beyond some limit of tolerable difficulty. In any case the goal of constructing efficient proof procedures can be met only by the development of efficient search strategies. Since formally defined theorem-proving problems generalise the path-finding problem for graphs, it is reasonable to expect that methods employed to increase the efficienty of graph searching can be extended to methods for theorem-proving. These methods include the use of learning, analogy, induction and other heuristic techniques studied in the field of artificial intelligence. The diagonal and upwards diagonal search strategies of chapter 4 are intended to provide a theoretically sound framework for the extension of heuristic methods to theorem-proving problems.

Experience gained through research in artificial intelligence suggests that the efficiency of search strategies can be improved by simulating methods employed by intelligent beings. In the case of theorem-proving the simulation of intelligent methods suggests that search strategies should aim at generating simpler before more complex proofs while generating non-proofs in a selective order based upon intelligent estimates of their relevance to a simplest admissible proof. The suggestion that search strategies should attempt to generate simpler before more complicated proofs may be a controversial one. It is put forward here for three reasons: (1) The convention for analysing proof procedures in a way which minimizes their heuristic restriction implies that simple proofs which are not first generated by an efficient ($\mathcal{T}$, Σ) will tend to be inadmissible for $\mathcal{T}$; (2) within constraints imposed by logical considerations affecting efficiency, all else being equal, mathematicians seek to find simpler before more complicated proofs; (3) most importantly, proofs of increased efficiency for alterations to inference systems require the assumption that Σ generates, before all other proofs, the simplest proof admissible for $\mathcal{T}$. This third point will be elaborated in section 0.2.

It is interesting to note that certain inference-related rules can be defined only in the context of search strategies. Deletion of variants and, more generally, of subsumed clauses is an important example in the case of resolution procedures: it is impossible to state that subsumed clauses do not occur within a refutation $\mathcal{D}$ of an initial set of clauses S_0 without referring to the subsuming clauses which themselves need not occur either in $\mathcal{D}$ or S_0. Both the completeness and efficiency of deleting subsuming clauses depends upon the sequence in which search strategies generate resolvents of clashes. Completeness of deletion rules $\mathcal{R}$ for procedures $\mathcal{P}$ is relative to the completeness of $\mathcal{P}$. $\mathcal{R}$ is complete relative to $\mathcal{P}$ if $\mathcal{P}$, employing $\mathcal{R}$, generates a proof of a theorem whenever $\mathcal{P}$, without $\mathcal{R}$, generates a proof of the same theorem. The completeness of deleting subsumed clauses has been proved relative to procedures $(\mathcal{T}, \Sigma)$ where Σ is level saturation and $\mathcal{T}$ is unrestricted binary resolution [39] or AM-clash resolution [48]. Our own proof [17] fails because no regard is taken of the dependency of deletion rules upon search strategies. Completeness of the usual deletion rule for subsumed clauses is proved relative to $(\mathcal{T}, \Sigma)$ for most resolution systems $\mathcal{T}$ and search strategies Σ in section 1.11, where counter-examples are also exhibited for the relative completeness and efficiency of alternative formulations of this same deletion rule. In this connexion we note that we have been unable to prove increased efficiency except for the case of deleting newly generated subsumed clauses. The relative completeness and increased efficiency of deleting tautologies is a simpler but not entirely trivial matter (section 1.12). For both deleting tautologies and subsumed clauses, proofs of efficiency are extracted from proofs of relative completeness and require the assumption that Σ generates simpler before more complex refutations.

The preceding remarks have attempted to indicate some of the more important relationships between the efficiency of proof procedures and the efficiency of search strategies. It is hoped that the distinction of inference system from search strategy will help to resolve some of the controversy concerning the use of 'complete' versus 'heuristic' methods in theorem-proving [41]. More specifically the development of efficient proof procedures can be served by a division of labour between logical studies of inference systems and studies of search strategies by the methods of operations research and artificial intelligence. These separate studies need to be co-ordinated by means of a theory which seeks to relate properties of inference systems and search strategies to properties of efficient proof procedures.

We have assumed that the fundamental property which needs to be required of proof procedures is that the difficulties of formally generated first proofs should tend toward those of informally obtained first proofs. We shall argue that a useful set of sufficient conditions for $\mathcal{P} = (\mathcal{T}, \Sigma)$ to approach this goal is that

(1) the complexities of simplest admissible formal proofs should be related to the complexities of informal proofs first constructed for theorems,

(2) $\mathcal{T}$ should restrict as much as possible the admissibility of both redundant derivations and derivations irrelevant to a simplest proof,

(3) Σ should aim at generating simpler before more complicated admissible proofs, and

(4) Σ should generate derivations in a selective order determined by intelligent estimates of their relevance to a simplest proof.

These four conditions have already been alluded to in preceding discussions. Conditions (1), (2) and (3) are further elaborated upon in section 0.2 and condition (4) is discussed in 0.3.

0.2 Refinements and the Elimination of Redundant and Irrelevant Inferences.

In this section we compare for efficiency procedures

($\mathcal{T}'$, Σ) and ($\mathcal{T}$, Σ) where $\mathcal{T}'$ is obtained from $\mathcal{T}$ either by imposing restrictions on the inference rules of $\mathcal{T}$ or by omitting axioms from the axiom set S_0 of $\mathcal{T}$. Following Luckham [24], $\mathcal{T}'$ is said to be a refinement of $\mathcal{T}$ when $S'^* \subset S^*$ where S'^* and S^* are the search spaces determined by $\mathcal{T}'$ and $\mathcal{T}$ respectively. By individually comparing for efficiency procedures ($\mathcal{T}_1$, Σ) and ($\mathcal{T}_2$, Σ) with ($\mathcal{T}$, Σ), where $\mathcal{T}_1$ and $\mathcal{T}_2$ are refinements of $\mathcal{T}$, we can obtain indirect comparisons of efficiency for ($\mathcal{T}_1$, Σ) and ($\mathcal{T}_2$, Σ). Furthermore, by extracting from criteria for refinements, we obtain criteria for single inference systems to admit extension to efficient proof procedures. We shall argue that if $\mathcal{T}'$ is a refinement of $\mathcal{T}$ and if Σ generates simpler before more complex proofs then ($\mathcal{T}'$, Σ) is more efficient than ($\mathcal{T}$, Σ) if the simplest proof admissible for $\mathcal{T}$ is also admissible for $\mathcal{T}'$; ($\mathcal{T}$, Σ) is more efficient than ($\mathcal{T}'$, Σ) if $\mathcal{T}$ admits simpler proofs than $\mathcal{T}'$ without admitting inordinately many redundant and irrelevant derivations not admitted by $\mathcal{T}'$.

If $\mathcal{T}'$ is a refinement of $\mathcal{T}$ then either $\mathcal{T}'$ eliminates redundant derivations admissible for $\mathcal{T}$ or (provided $\mathcal{T}'$ does not eliminate all proofs of a theorem) $\mathcal{T}'$ eliminates derivations irrelevant to a proof of the given theorem. The most obvious kind of redundancy exists when a system $\mathcal{T}$ admits distinct derivations of the same sentence C (or of variants C and C' when $\mathcal{T}$ is a resolution system). For resolution systems $\mathcal{T}$, another kind of redundance exists when $\mathcal{T}$ admits both a derivation $\mathcal{D}$ of a clause C and a second derivation $\mathcal{D}'$ of a clause C' which subsumes C. Other relationships between derivations $\mathcal{D}$ and $\mathcal{D}'$ can be attributed to redundancy. A precise characterization of these relationships is not necessary for present purposes. An irrelevant derivation is one which, for reasons other than redundancy, is not necessary for the construction of a proof. Redundant and irrelevant derivations may be eliminated either by restrictions which prohibit their generation or by deletion after generation. The second method is related to the first because deletion prohibits the generation of derivations constructible from deleted derivations.

The method of eliminating redundancies, which, as we shall observe below, need not always contribute to efficiency, is the principal method employed in this thesis for studying the improvement of inference systems. We shall argue that the potential improvement of eliminating redundancies and irrelevancies is related not only to the numbers of derivations eliminated but also to the complexity of the simplest proofs retained. In this connexion it is worth noting that systems which represent sentences as sets of clauses omit redundancies caused in other systems by explicit rules (or axioms) for double negation, for commutativity, associativity and idempotency of conjunction and disjunction, for renaming bound variables, vacuous quantification and for interchanging adjacent quantifiers of the same type. These redundancies are omitted without the expense of complicating proofs.

The use of explicit operations for factoring clauses saves partial results obtained when attempting to resolve clauses. The method of marked factoring (section 1.6) eliminates without complicating derivations, redundancies allowed by the Wos-Robinson method [53]. The method of m-factoring (section 4.7) achieves similar results while also providing an effective means for implementing merging restrictions [2]. A restricted version of marked factoring (nucleus clauses unfactored, section 4.6) reduces further redundancies with some attendant complication of derivations. (It is interesting to note that this method sometimes eliminates all refutations which lift ground refutations.) The method of section 2.9 for the unique decomposition of hyper-resolution clashes can be interpreted either as a means for eliminating redundancies from P_1-resolution or as a method for implementing hyper-resolution while saving intermediate results. Under neither of these interpretations does this method complicate derivations.

For most resolution systems the retention of tautologies only introduces redundances and complicates derivations (section 1.12). On the other hand, retention of variants and subsumed clauses generates redundancy - but sometimes simplifies derivations (section 1.11). Minimality restrictions (section 1.13), which can be imposed on α-restricted binary derivations (2.6) and on M-clash derivations (2.7), both simplify derivations and eliminate many redundancies. M-clash restrictions complicate derivations; additional complication is caused by the α-restrictions on M-clash derivations (2.7). Chapter 3 establishes an equivalence between a refinement of the paramodulation method for dealing with equality and the hyper-resolution method using equality axioms. This equivalence implies both equivalent numbers of redundant and irrelevant derivations and also equivalent complexities of proofs for the two systems. For both systems, initial trivialization of inequalities (3.4) restricts redundancies and retains simplest refutations.

Almost certainly the most significant contribution to the elimination of redundant inferences has been the Prawitz method for restricting the instantiation of matrix clauses over the Herbrand universe [34]. This method, now incorporated in other Herbrand procedures [35], [22], [14] (by means of the unification or matching algorithm). improves efficiency by eliminating redundancies without complicating proofs. In the case of resolution systems, most general unification eliminates redundancies by omitting infinitely many ground derivations lifted by single general derivations. The Prawitz method also eliminates irrelevant derivations in a manner similar to that of the purity principle [39]. Clauses which cannot occur in proofs, because they contain literals which cannot mate with other literals in the initial set of clauses, are inhibited from generating irrelevant derivations. (In the pre-Prawitz Gilmore method [10] such clauses would not be distinguished from other clauses and would potentially need to be instantiated in all possible ways over the Herbrand universe.) Methods similar to the Prawitz method have been conjectured but not verified for the predicate calculus with equality [5], [28], [43], [38].

M-clash resolution eliminates both redundant and irrelevant derivations. On the other hand, linear resolution ([23] and [24]) eliminates redundancies but no irrelevancies, since, as shown by Loveland, for any clause C derivable by unrestricted resolution there is a linear derivation $\mathcal{D}'$ of a clause C' which subsumes C. The linear derivation $\mathcal{D}'$ is no more complicated than the derivation $\mathcal{D}$ of C in the sense that it contains no greater number of applications of resolution. However $\mathcal{D}'$ can be much more complicated than $\mathcal{D}$ if complexity is measured by resolution level.

M-clash resolution eliminates irrelevancies because only clauses false in the interpretation M can be derived by the M-clash resolution rule. That M-clash resolution eliminates no irrelevant derivations other than those of clauses true in M is a consequence of the deduction completeness theorem 2.5.1.

The elimination of redundant and irrelevant derivations does not, by itself, guarrantee efficient proof procedures. Indeed it is even possible for a complete inference system $\mathscr{I}$, which admits neither redundant nor irrelevant derivations, to be incapable of extension to a procedure ($\mathscr{I}$, Σ) which proves informally easy theorems without great formal difficulty. Such an inference system $\mathscr{I}$ would admit proofs of only great formal complexity. More generally if $\mathscr{I}'$ is a refinement of $\mathscr{I}$ then ($\mathscr{I}'$, Σ) may be less efficient than ($\mathscr{I}$, Σ) if $\mathscr{I}'$ does not admit the first proof obtained by Σ which is admissible for $\mathscr{I}$. Under the assumption that Σ generates simpler before more complex proofs, $\mathscr{P}' = (\mathscr{I}', \Sigma)$ is more efficient than $\mathscr{P} = (\mathscr{I}, \Sigma)$ (or no less efficient) when $\mathscr{I}'$ admits the simplest proof admitted by $\mathscr{I}$ (assuming also that the order in which Σ generates derivations admissible for $\mathscr{I}'$ coincides with the order in which Σ generates derivations admissible for $\mathscr{I}$, restricted to derivations admissible for $\mathscr{I}'$). Under these assumptions, $\mathscr{P}'$ generates the same first proof $\mathcal{D}$ generated by $\mathcal{P}$; before generating $\mathcal{D}$, $\mathcal{P}$ generates all the derivations generated by $\mathcal{P}'$ and those derivations generated by $\mathcal{P}$ and not by $\mathcal{P}'$ are redundancies and irrelevancies not admitted by $\mathcal{I}'$. If Σ generates more complex before simpler proofs then $\mathcal{P}'$ may be more efficient than $\mathcal{P}$ even when $\mathcal{I}'$ eliminates simplest proofs and very few other derivations. Such combinations of search strategies and refinements yielding more efficient proof procedures are pathological and do not seem to fit into any comprehensive theory of efficient proof procedures. For this and other reasons mentioned already in section 0.1, we shall compare inference systems relative to the assumption that they are incorporated in proof procedures with search strategies which generate simpler before more complex proofs.

$(\mathcal{I}', \Sigma)$ can be more efficient than $(\mathcal{I}, \Sigma)$ even when $\mathcal{I}'$ eliminates simplest proofs provided that $\mathcal{I}'$ eliminates sufficiently many redundancies and irrelevancies. The more $\mathcal{I}'$ eliminates unnecessary derivations the greater $\mathcal{I}'$ may complicate simplest proofs while still improving efficiency. Suppose for example that $\mathcal{I}'$ is a refinement of $\mathcal{I}$ and that Σ is a level saturation search strategy for $\mathcal{I}$ and $\mathcal{I}'$. Suppose that, for a given initial set of sentences S_0, $\mathcal{I}$ and $\mathcal{I}'$ admit respectively $d(n)$ and $d'(n)$ derivations of level less than or equal to n. Then, for each n, $d'(n) \leq d(n)$ and $r(n) = \dfrac{d'(n)}{d\,(n)}$ is the fraction of derivations of level less than or equal to n admitted by $\mathcal{T}$ which are also admitted by $\mathcal{T}'$. If N and N' are the least levels of proofs from S_0 admissible for $\mathcal{T}$ and $\mathcal{T}'$ respectively, then $\mathcal{P}'$ is more, less or equally efficient to $\mathcal{P}$ depending on whether $d'(N') < d(N)$, $d'(N') > d(N)$ or $d'(N') = d(N)$ (assuming for simplicity, that Σ generates all derivations of a given level before generating a proof of that level). Thus $(\mathcal{T}', \Sigma)$ is more efficient than $(\mathcal{T}, \Sigma)$ if $N = N'$ or if $N' > N$ but $\mathcal{T}'$ omits sufficiently many derivations for $d'(N') < d(N)$. For classes of initial sets of sentences S_0, estimates of the function r (as a function of n and S_0) can sometimes be obtained by comparing derivations admissible for $\mathcal{T}$ and $\mathcal{T}'$. Other investigations can be made to estimate either d or d' and bounds on the difference between N and N' (as a function of S_0 and of the theorem to be proved) can often be extracted from completeness proofs for $\mathcal{T}'$ relative to $\mathcal{T}$. Similar studies can be done for other notions of complexity when Σ is a saturation search by degree of complexity. The functions d, d' and r and N' as a function of N vary widely with various properties of initial sets S_0 and of theorems provable from S_0. For this reason calculations of these functions may be impossible in all but either worst or best cases or cases which can be considered typical for some class of theorems.

Despite the great difficulties of obtaining, for wide classes of theorems, precise comparisons of the potential efficiency of refinements, certain important principles emerge quite clearly. If $\mathcal{I}'$ refines $\mathcal{I}$ and Σ generates simpler before more complex proofs, then the greater the number of derivations eliminated and the simpler the proofs admitted by $\mathcal{I}'$, the more efficient is $(\mathcal{I}', \Sigma)$ than $(\mathcal{I}, \Sigma)$. Both extended set of support in resolution [55] and the employment of lemmas in model elimination [22] extend inference systems, simplifying proofs and introducing redundancies and irrelevancies. Both extensions are motivated by the use of lemmas and previously proved theorems to increase the efficiency of proving theorems in informal mathematics.

It has been suggested that the efficiency of proof procedures can be improved by increasing the power of inference systems [27], [44]. This notion can be quantified by identifying the power of $\mathcal{I}$ for a given theorem with the degree of complexity of the simplest proof admitted by $\mathcal{I}$. Thus a system $\mathcal{I}$ is more powerful than $\mathcal{I}'$ when the simplest proof admitted by $\mathcal{I}$ is simpler than the simplest proof admitted by $\mathcal{I}'$ for the same theorem. In particular $\mathcal{I}$ is never less powerful than $\mathcal{I}'$ if $\mathcal{I}$ extends $\mathcal{I}'$. Gödel's results on lengths of proofs [11] show that many proofs can be greatly simplified by applying rules within a system of higher-order logic. Among resolution systems, unrestricted resolution admits the simplest proofs and is therefore most powerful, although not necessarily uniquely so.

Just as refinements often over-complicate proofs, extensions often introduce too many redundancies and irrelevancies. The problem of admitting too many derivations is especially acute for higher-order logic and first-order logic with axiom schemata. Gould's negative results [12] show that there is no algorithm for eliminating in higher-order logic the kind of irrelevancies eliminated by the unification algorithm in first-order logic. Axiom schemata in first-order logic become axioms in second-order logic. For this reason Gould's results are not very surprising since extension of the unification algorithm to higher-order logic would imply very strong restrictions on the instantiation of axiom schemata in first-order theories. Darlington's f-matching method [5] provides just such an extension of the unification algorithm to the restricted instantiation of axiom schema. For the schema of substitutivity for equality (which can already be restricted to a finite number of instances), the completeness of f-matching is equivalent to that of the paramodulation system conjectured to be complete by Robinson and Wos [38]. For the axiom schema of induction in number theory, f-matching may fail to provide instances which are necessary to prove even easy theorems.

The difficulties encountered by various attempts to inhibit the generation of irrelevancies by logical restrictions on inference systems suggests that a plateau has been reached for improving efficiency by eliminating irrelevant derivations within complete inference systems. Further progress for improving efficiency may be possible by employing incomplete inference systems. This possibility will be discussed in section 0.3. It should be remarked first that at least two research programmes can be formulated for increasing the efficiency of existing proof procedures without sacrificing the completeness of inference systems. The first programme involves the simulation in search strategies of intelligent informal methods for finding proofs. The second programme is that of constructing refinements of inference systems with the explicit goal of eliminating as many redundancies as possible while still retaining simplest proofs. The first proposal has already been discussed in the preceding section and will be examined further in section 0.3 in connexion with discussions pertaining to the completeness of search strategies. With regard to the second proposal, it should be remarked that existing refinements of inference systems (e.g. resolution) admit inordinately large numbers of redundant derivations. Unlike irrelevancies, redundancies can be recognized during the course of searching for proofs. It might be hoped that these redundancies can be recognized and eliminated before rather than after their generation.

It seems reasonable to extract, from criteria for refinements and extensions, criteria for single inference systems $\mathcal{T}$ to admit extension to efficient proof procedures $(\mathcal{T}, \Sigma)$. These criteria include requirements that $\mathcal{T}$ admit simple proofs and few redundancies and irrelevancies. For formal methods to compete with informal methods in restricting the generation of redundant and irrelevant inferences, it seems unlikely that first obtained formal proofs can be much simpler than those first obtained by informal methods. On the other hand, if formal complexities are much greater than informal complexities then formal difficulties will tend to be greater than informal difficulties. For these reasons it seems desirable that formal complexities of proofs should approximate those of informally obtained first proofs of the same theorem.

We have already remarked that informally obtained first proofs of theorems are often more complex than later proofs. For an ideally efficient proof procedure $(\mathcal{T}, \Sigma)$, assuming that Σ is complete for $\mathcal{T}$ and generates simplest admissible proofs before more complex proofs, the preceding remarks imply that the first proof generated by $(\mathcal{T}, \Sigma)$

is likely to be more complex than the simplest proof theoretically possible for a given theorem. This not only suggests the possibility of improving efficiency by the appropriate choice of refinements for proving given theorems but also suggests the merit of methods for obtaining simplest proofs of theorems after generating the more complex and more efficiently obtained first proofs admissible for refinements. A simple program for the simplification of complex proofs can be outlined for resolution inference systems:

Suppose that $\mathcal{R}'$ is a refinement of the unrestricted resolution system $\mathcal{R}$. Suppose that Σ generates simpler before more complex refutations and that Φ and Φ' are the simplest refutations, of an unsatisfiable set of clauses S_0, admissible for $\mathcal{R}$ and $\mathcal{R}'$ respectively. Assume that $(\mathcal{R}', \Sigma)$ is more efficient than $(\mathcal{R}, \Sigma)$ for refuting S_0 and that Φ' is the first obtained refutation of S_0. With few exceptions the following method will construct Φ (or an equivalently simple refutation of S_0) from Φ', generally with much less difficulty than would be involved in obtaining Φ directly by $(\mathcal{R}, \Sigma)$. Although Φ' may not lift a ground refutation, it can be verified that it is easy to construct both a ground refutation Φ_0 and a contraction Φ_1 of Φ' which is a refutation of S_0 and lifts Φ_0. (Φ_0 and Φ_1 can be constructed simultaneously from Φ' by applying methods similar to those applied to prove the contraction theorem 1.10.2 and to apply the minimality restriction (section 1.13). In the notation of 1.13, $\mathcal{D}_0 = \tilde{\mathcal{D}}_1$.) The set S_0' of clauses which occur at tips of $\mathcal{D}_0$ constitute a truth-functionally unsatisfiable set of instances of clauses in S_0. Σ applied to $\mathcal{T}(S_0')$ will generate a simplest refutation $\mathcal{D}_2$ of S_0'. $\mathcal{D}_2$ can be lifted to obtain a refutation $\mathcal{D}_3$ of S_0. Generally $\mathcal{D}_3$ will be a simpler proof than $\mathcal{D}'$ and either will be identical to $\mathcal{D}$ or will be of a complexity equivalent to that of $\mathcal{D}$. Similar methods can be profitably applied to the simplification of proofs in more general contexts.

0.3 Completeness of Proof Procedures.

Before examining relationships between completeness and efficiency it is necessary to recall that both completeness and efficiency are evaluated relative to the set of sentences $\mathfrak{S}$ within which a proof procedure $\mathcal{P}$ is expected to prove theorems. This explicit identification of the set $\mathfrak{S}$ is necessary in order to avoid, when undesired, evaluation relative either to the possibly larger set $\mathfrak{S}^*$ of all sentences or alternatively to the set $\mathfrak{S}^0$, the largest set for which $\mathcal{P}$ is complete. For any proof procedure $\mathcal{P}$ such a set $\mathfrak{S}^0$ always exists and may in extreme cases equal either $\emptyset$ or more likely $\mathfrak{S}^*$. $\mathfrak{S}^0$ may be properly contained in $\mathfrak{S}$, in which case $\mathcal{P}$ is potentially required to prove theorems in $\mathfrak{S} - \mathfrak{S}^0$ which are unprovable for $\mathcal{P}$; $\mathfrak{S}^o$ may be identical to $\mathfrak{S}$; or $\mathfrak{S}$ may be properly contained in $\mathfrak{S}^o$, in which case $\mathcal{P}$, although theoretically capable of proving theorems in $\mathfrak{S}^o - \mathfrak{S}$ is not required to do so, possibly because $\mathcal{P}$ is known to be inefficient for theorems in $\mathfrak{S}^o - \mathfrak{S}$ or because sentences in $\mathfrak{S}^o - \mathfrak{S}$ do not arise in some principal intended application of $\mathcal{P}$. In any case, for a given proof procedure the set $\mathfrak{S}^o$ need not in general be effectively recognizable (i.e. recursive). In contrast it is important to require that sentences in $\mathfrak{S}$ be distinguished from sentences in $\mathfrak{S}^* - \mathfrak{S}$ before a proof is attempted by $\mathcal{P}$. In particular it is not adequate to specify that $\mathcal{P}$ is expected to prove, for instance, only easy or only difficult theorems, if no effective and efficient recognition algorithm exists and is employed for distinguishing such possible theorems. Without further qualification, it will be implicitly assumed in the remainder of this section that proof procedures $\mathcal{P}$ are evaluated for completeness (and efficiency) relative to the set $\mathfrak{S}$ for which $\mathcal{P}$ is expected to prove theorems. It will be assumed that sentences in $\mathfrak{S}^* - \mathfrak{S}$ are effectively and efficiently distinguishable from sentences in $\mathfrak{S}$. Because of these assumptions, decision procedures for decidable subsets $\hat{\mathfrak{S}}$ of $\mathfrak{S}^*$ are evaluated as complete when they are intended to prove theorems in $\hat{\mathfrak{S}}$ and as incomplete if they are expected to prove theorems in $\mathfrak{S}^*$. For the same reasons, procedures $\mathcal{P}$ complete for $\mathfrak{S}^*$ will be investigated for relationships between completeness and efficiency, not necessarily relative to $\mathfrak{S}^*$, but relative to the set $\mathfrak{S}$ to which $\mathcal{P}$ is expected to be applied.

All proof procedures, complete or incomplete, are limited in practice by an upper bound on the amount of effort available for generating a proof of a given theorem. Failure to obtain a proof by a complete procedure $\mathcal{P}$ within such finite limitations implies that the alleged theorem either is not valid or is valid but too difficult to be proved with the limited amount of effort available. Similar failure by an incomplete $\mathcal{P}$ implies, as a third possibility, that the alleged theorem is valid but unprovable by $\mathcal{P}$ even with unlimited effort. For all practical purposes it is only this third possibility which distinguishes incomplete from complete proof procedures. (Indeed the existence of this possibility provides an operational definition of incompleteness which coincides in extension with the definition of the first paragraph of this section.) We shall attempt to determine whether the existence of this third possibility justifies evaluating complete procedures as always superior to incomplete procedures.

Of all proof procedures we require only that formal difficulties tend toward the informal difficulties of first proving theorems. (The degree to which a proof procedure approaches this goal can be evaluated independently of its completeness or incompleteness. Indeed it is wholly upon this basis that we intend to base our evaluation of the significance of incompleteness for efficiency.) Thus when a best (i.e. most efficient) proof procedure fails to obtain a proof of a given theorem within given limitations on the amount of effort available it can be inferred that the theorem is too difficult to be proved by any good proof procedure within the same limitations. It is important to notice that in this sense an incomplete procedure $\mathcal{P}'$ can be superior to a complete procedure $\mathcal{P}$. $\mathcal{P}$ may fail to prove, even with considerable but limited effort, theorems which are easy to prove informally with less difficulty, in particular, than that unsuccessfully expended by $\mathcal{P}$. In contrast, $\mathcal{P}'$, because of its incompleteness, may be incapable of proving only informally difficult theorems which are in any case too difficult to be proved by any efficient proof procedure within the bounds on effort available. Thus what matters for efficiency is not necessarily the frequency with which an incomplete procedure $\mathcal{P}'$ is expected to prove theorems theoretically unprovable for $\mathcal{P}'$ - but, more significantly, the frequency with which $\mathcal{P}'$ is expected (and unable) to prove theorems informally provable with less effort than that unsuccessfully expended by $\mathcal{P}'$. More generally a complete or incomplete procedure fails to be satisfactory only when it fails to prove with a given bounded amount of effort a theorem which is informally provable with comparable effort.

The longer that inference systems and proof procedures (such as those of [5], [28], [38] and [43]) are conjectured, but not proved, to be complete, the less significant for efficiency is the possibility of their incompleteness. The increased suitability of these systems and procedures is due not only to the increased likelihood of their completeness but more importantly to the increased likelihood that in the case of incompleteness, only informally difficult theorems are formally unprovable. Since successive attempts to disprove completeness will tend to eliminate simpler before more complex counter-examples, continued failure of these attempts increases the likelihood that, in the event of incompleteness, only complex counter-examples exist. Increasingly complex counter-examples correspond to increasingly more difficult theorems, and therefore continuing failure to disprove completeness decreases the probability that easy theorems are unprovable. This decreased probability increases, in turn, the suitability of the given inference systems and proof procedures for automatic theorem-proving. It is an interesting possibility that more information may be available about the suitability of proof procedures which are conjectured but not proved to be complete than is available for proof procedures which are definitely known to be either complete or incomplete. None the less we shall argue that proof procedures $\mathcal{P} = (\mathcal{T}, \leq)$ employing complete $\mathcal{T}$ are often at an advantage compared to procedures employing incomplete inference systems. This advantage is that completeness proofs for inference systems $\mathcal{T}$ often yield information relevant to the efficiency of procedures $(\mathcal{T}, \Sigma)$, namely that complexities of simplest admissible proofs relate to the complexities of informally obtained first proofs.

It has already been noted that incomplete procedures, because of their incompleteness, are able to eliminate more irrelevancies than can be eliminated by complete procedures. Almost certainly it is only this possibility of eliminating greater numbers of irrelevant derivations which can account for an absolute preference for incomplete proof procedures. Indeed this reason accounts for the fact that decision and semi-decision procedures, complete for sets of sentences $\mathfrak{S} \subset \mathfrak{S}^*$ but incomplete for $\mathfrak{S}^*$, can be more efficient than procedures complete for $\mathfrak{S}^*$ when they are applied to proving theorems in $\mathfrak{S}$. In particular the incompleteness of resolution procedures, for deriving logical consequences from sets of clauses, is a property which contributes to their efficiency for obtaining refutations of unsatisfiable sets of clauses. (Bounds on the incompleteness of resolution systems, relevant to efficiency, are established by Theorem 2.5.1.) The possible advantages of incomplete proof procedures are apparent when these procedures are applied to obtaining proofs of theorems which they are able to prove. The disadvantages of incomplete procedures arise when they are applied to proving theorems which they are incapable of proving.

In general it is to the disadvantage of incomplete proof procedures that usually little or no information is available concerning the extent or character of their incompleteness. In particular no such information has been reported for the interactive theorem-proving programs of the Applied Logic Corporation [14]. Certainly what should be required of incomplete procedures is that only very few if any easy theorems should be unprovable. Norton notes that this requirement fails to be satisfied by his incomplete proof procedure for proving theorems in group theory [31].

We have already remarked, in the preceding section, that completeness proofs for refinements $\mathcal{T}'$ of inference systems $\mathcal{T}$ often provide information about the comparative efficiency of proof procedures $(\mathcal{T}, \Sigma)$ and $(\mathcal{T}', \Sigma)$. This information is easiest to obtain when completeness proofs for $\mathcal{T}'$ relative to $\mathcal{T}$ proceed by transforming proofs $\mathcal{D}$ admissible for $\mathcal{T}$ into proofs $\mathcal{D}'$ admissible for $\mathcal{T}'$. Comparison of the complexities of $\mathcal{D}$ and $\mathcal{D}'$

can be applied, by the method outlined in section 0.2, to comparison of the efficiencies of $\mathcal{P}$ and $\mathcal{P}'$. Similar but more limited information concerning efficiency can sometimes be extracted from completeness proofs (for inference systems) which proceed directly by semantic arguments. In particular the application to completeness proofs of semantic tree construction exhibits a relationship between the complexity of resolution proofs and the complexity of a certain kind of semantic argument for establishing the same theorem. More generally, completeness proofs for inference systems which can be interpreted as employing rules for Herbrand instantiation of matrix clauses (e.g. Gilmore [10], Prawitz [34], [35], and Loveland [22] and Robinson resolution systems) indicate a relationship between complexities of simplest formal proofs and notions of complexity, invariant for these systems, based upon the number and truth-functional complexity of the fewest ground instances of matrix clauses necessary to reduce the proof of a given theorem to the proof of a corresponding theorem in propositional logic. That complexities of simplest proofs correspond closely to complexities of informally obtained proofs does not by itself imply that formal difficulties correspond to the informal difficulties of obtaining first proofs of theorems. For this stronger correspondence, it is necessary in addition that the inference system $\mathcal{J}$ admits few redundant and irrelevant derivations while the search strategy Σ finds simpler before more complex proofs, generating derivations in a discriminating order of relevance to a simplest proof. It seems that the problem of investigating inference systems $\mathcal{T}$ for redundant and irrelcvant derivations is no more difficult for incomplete $\mathcal{T}$ than it is for complete $\mathcal{T}$. In contrast, the problem of relating formal to informal complexities of first proofs seems to be an easier one for complete $\mathcal{T}$. We shall argue that complete search strategies Σ are likely to be more suitable thah incomplete Σ for application to inference systems in efficient procf procedures.

We recall that proof procedures $\mathcal{P}$ are analysed as consisting of inference systems $\mathcal{T}$ and search strategies Σ ; the logical restrictions of $\mathcal{P}$ are incorporated in $\mathcal{T}$, heuristic restrictions are incorporated in Σ and restrictions which are ambiguously logical or heuristic are treated as logical restrictions and incorporated in $\mathcal{T}$. Relative to these conventions, we argue the case for complete search strategies Σ against that for incomplete Σ. Since arguments for incomplete Σ seem to be based primarily on the paradigm of intelligent human behaviour as applied to finding proofs of theorems, we limit our arguments to those based on this same paradigm. We note that the case for complete search strategies can also be be interpreted more generally as one for complete proof procedures.

Characteristic of intelligent informal theorem-proving is the high degree of selectivity employed in exploring possibilities for proving theorems. This selectivity seems to suggest that informal search strategies so restrict the generation of derivations that they must almost certainly be incomplete. Contradicting this conclusion is the unlikelihood that an intelligent theorem-prover would completely eliminate, on purely heuristic grounds, a logically possible subproof of a given alleged theorem. This unlikelihood suggests that informal search strategies (and also proof procedures) are complete. The apparent contradiction can be reconciled by interpreting the selectivity of informal search strategies positively, as employing highly discriminating but not incomplete heuristics for ordering logically possible subproofs with respect to their expected relevance to a desired simplest proof, instead of negatively, as eliminating beyond reconsideration possible but unlikely subproofs of the alleged theorem. The heuristic for deleting clauses, which contain function symbols nested to a degree exceeding a given fixed bound [1], [53], is an application of the negative interpretation of selectivity. A corresponding application of the positive interpretation is a heuristic which would give preference among clauses of otherwise equal merit, to clauses containing less function nesting over clauses containing greater nesting, without completely eliminating the latter clauses. (It is interesting to note that implementation of the positively interpreted heuristic improves the efficiency of diagonal search - assuming that complexity of derivations $\mathcal{D}$ is a monotonic increasing function of the number of symbols occurring in $\mathcal{D}$.) Similarly, search strategies employing only the unit section of unit preference search [53] apply the negative interpretation of selectivity, whereas diagonal search strategies employing expected length of clause as a heuristic function (section 4.3) apply a positive interpretation. In general complete search strategies, employing positive criteria for discriminating between possible subproofs, simulate intelligent search methods more faithfully than incomplete strategies, employing negative criteria for rejecting candidate subproofs. Assuming that efficient search strategies are essential for the efficiency of proof procedures and that simulation of intelligent informal methods is indispensable for the efficiency of search strategies, it follows that complete search strategies are more likely than incomplete strategies to serve the goals of efficient automatic theorem-proving.

0.4 Conclusion

In this chapter we have investigated various notions and assumptions relevant to the efficiency of automatic proof procedures. In particular, we have argued for the utility of formulating distinctions between inference system, search strategy and proof procedure, distinctions between complexity and difficulty and assumptions relating formal and informal methods of proof. We have attempted to indicate that these distinctions and assumptions can be usefully applied within a theory of efficiency to

(1) outline formal methods of evaluating refinements, extensions and single proof procedures for efficiency,

(2) reconcile apparently conflicting intuitions regarding efficiency (e.g. concerning complete vs. heuristic methods),

(3) distinguish intuitions on the basis of their being compatible with, incompatible with or logically implied by the theory and

(4) suggest practicable programmes of research for improving the efficiency of proof procedures.

It is hoped that additional evidence for the utility of this theory will be provided by the investigations of chapters 1-4.

Chapter 1

This chapter is concerned with the syntax of resolution systems. Sections 1.1 - 1.5 examine the syntax of expressions, substitutions, unification, clash resolution and clash restriction. In section 1.6 factoring and resolution of factors are introduced as methods for improving the efficiency of implementing resolution rules. Derivations are treated as labelled trees (section 1.8) and useful properties of trees are stated in 1.7. In section 1.9 the trace of a search strategy is defined and is used in turn to define the efficiency of proof procedures and the completeness of deletion rules. These notions are applied in 1.11 and 1.12 to an investigation of the completeness and efficiency of rules for deleting subsumed clauses and tautologies.

The contraction theorem (section 1.10) isolates and formalizes a useful method for constructing and transforming derivations. It is applied in chapter 2 to construct derivations from semantic trees and in chapter 3 to permute hyper-resolution derivations. The contraction theorem generalises the lifting lemma and indicates how deletion of subsumed clauses can simplify derivations.

In section 1.13 a strong restriction on derivations is incorporated in the minimality condition. The preservation of minimality conditions under contractions implies that minimality is compatible with deletion of subsumed clauses. This same property is used in chapter 2 to prove the existence of minimal α-restricted binary proofs and minimal M-clash proofs.

1.1 Expressions.

We assume familiarity with the basic concepts of resolution theory as can be found, for instance, in Robinson's review article [42] . The following definitions are intended therefore primarily to establish the terminology and notation used in the sequel.

Atomic formulae A are referred to as atoms. Literals L are atoms A or their negations $\bar{A}$; in the first case L is said to be positive, in the second case negative. If L is a literal then by $|L|$ we denote the atom A such that $L = A$ or $L = \bar{A}$. If L is negative we identify $\bar{L}$ with the atom $|L|$.

A clause C is a set of literals. If $C = \{L_1, \ldots, L_n\}$ is a set of literals then by $\bar{C}$ we denote, as in [3] , the set $\{\bar{L}_1, \ldots, \bar{L}_n\}$. It is convenient to follow the convention of [2] letting $\overset{\circ}{\cup}$ denote disjoint union. Thus $C \overset{\circ}{\cup} D$ is defined only when $C \cap D = \emptyset$ and then $C \overset{\circ}{\cup} D = C \cup D$. If a clause C contains no elements then we denote C by $\Box$ and C is called the null clause. C is a tautology if for some literal L, both $L, \bar{L} \in C$. A clause C is positive (negative) if all its literals are positive (negative), otherwise C is non-positive (non-negative). We recall that a clause is interpreted as the universal closure of the disjunction of its literals.

Function letters may have no argument places, in which case they are individual constants. An expression is a term, literal, clause or set of clauses. If an expression contains no variables then it is called a ground expression.

We note that the representation of sentences by sets of clauses is an important factor contributing to the efficiency of resolution systems. Rules for commutativity, associativity, and idempotency of disjunction and conjunction, for interchanging adjacent quantifiers of similar kind and for deleting vacuous quantifiers are all unnecessary. The elimination of these rules contributes to efficiency by shortening derivations and by reducing the number of sentences derivable from a given set of sentences.

1.2 Substitutions.

A substitution σ is a set of substitution components t_i/x_i where t_i is a term and x_i is a variable (as in [39] t_i is not x_i). If $\sigma = \{t_1/x_1, \ldots, t_n/x_n\}$ then the variables x_i and terms t_i (for $1 \leq i \leq n$) are called respectively the variables and terms of σ. If the terms of σ are ground terms then σ is called a ground substitution. For any expression X and substitution σ, the expression $X\sigma$ is well-defined and is the result of applying σ to X. $X\sigma$ is called an instance of X. If C and D are clauses then C subsumes D, if $C\sigma \subseteq D$ for some substitution σ. The following properties of substitutions are well known:

(1) Given substitutions σ_1 and σ_2 their

composition $(\sigma_1 \sigma_2)$ is always well-defined and is such that $X(\sigma_1 \sigma_2) = (X\sigma_1)\sigma_2$.

(2) Composition of substitutions is associative, i.e. $((\sigma_1 \sigma_2)\sigma_3) = (\sigma_1(\sigma_2 \sigma_3))$ for all $\sigma_1, \sigma_2, \sigma_3$. We may therefore omit parentheses for compounded substitutions in the usual way.

(3) The empty substitution $\varepsilon = \emptyset$ is an identity for composition, i.e. $\sigma\varepsilon = \varepsilon\sigma = \sigma$ for all σ.

(4) $X\sigma = X$ if the variables of σ do not occur in X.

If C_1 and C_2 are clauses and $C_1\sigma_1 = C_2$, $C_2\sigma_2 = C_1$ for some σ_1 and σ_2 then C_1 and C_2 are variants (variants differ only by a systematic renaming of variables). Under the usual interpretation of clauses variants are logically equivalent. A set of clauses $S = \{C_1, \ldots, C_n\}$ is standardised if no two distinct C_i and C_j share common variables. Every set of clauses S is logically equivalent to a standardised set S' where S' may be obtained from S by appropriately replacing clauses in S by variants. Resolution conventions for standardising sets of clauses eliminate the usual rules for renaming bound variables.

1.3 Unification.

A set of expressions E is unifiable if $E\sigma$ is a singleton for some σ ; σ is called a unifier of E. A family $\mathcal{E} = \{E_1, \ldots, E_n\}$ of sets of expressions is simultaneously unifiable, if $E_i\sigma$ is a singleton for each i and some σ . A <u>most general unifier</u> (m.g.u.) of a set of expressions E is a unifier θ of E such that if σ also unifies E then $\sigma = \theta\lambda$ for some λ . If E is unifiable then such an m.g.u. θ of E exists; moreover we may insist, as we do in the sequel, that the variables of θ and the variables occurring in terms of θ all occur in E (see [39]). Similarly a <u>most general simultaneous unifier</u> (m.g.s.u.) of a family $\mathcal{E}$ is a simultaneous unifier θ of $\mathcal{E}$ such that if σ simultaneously unifies $\mathcal{E}$ then $\sigma = \theta\lambda$ for some λ . If $\mathcal{E}$ is simultaneously unifiable then such an m.g.s.u. θ exists and may be restricted as for the case of m.g.u.s above. Notice that θ is an m.g.s.u. of $\mathcal{E} = \{E\}$ if any only if θ is an m.g.u. of E. It follows that we may restrict attention when desirable to statements regarding families $\mathcal{E}$ and their simultaneous unifiers and m.g.s.u.s . We shall often refer to simultaneous unifiers more simply as unifiers.

Algorithms for obtaining m.g.u.s and m.g.s.us of unifiable families are given in [39] and [43] . The refinement theorem below and its corollaries formalise many of the intuitive properties expected of m.g.s.u.s Among the implications of corollaries 1.3.3 and 1.2.4 is that the problem of computing arbitrary m.g.s.u.s can be reduced to that of consecutively finding m.g.u.s of sets containing just two expressions. Corollary 1.3.5 is used as a lemma in verifying that resolvents of clashes can be obtained as resolvents of factors.

Let $\mathcal{E}$ and $\mathcal{E}'$ be families of sets of expressions. $\mathcal{E}'$ is a <u>refinement</u> of $\mathcal{E}$ if for every $E' \in \mathcal{E}'$ there is an $E \in \mathcal{E}$ such that $E' \subseteq E$. Notice that if $\mathcal{E}$ is unifiable then so is $\mathcal{E}'$ (e.g. any unifier of $\mathcal{E}$ unifies $\mathcal{E}'$).

<u>Lemma 1.3.1.</u> Given $\mathcal{E}$ unifiable and $\mathcal{E}'$ a refinement of $\mathcal{E}$, let θ_1 and θ_2 be m.g.s.u.s of $\mathcal{E}'$ and $\mathcal{E}\theta_1$ respectively then $\theta_1\theta_2$ is an m.g.s.u. of $\mathcal{E}$.

<u>Proof.</u> $\theta_1\theta_2$ unifies $\mathcal{E}$ since θ_2 unifies $\mathcal{E}\theta_1$ and $(\mathcal{E}\theta_1)\theta_2 = \mathcal{E}\theta_1\theta_2$. If σ unifies $\mathcal{E}$ then σ unifies $\mathcal{E}'$ and $\sigma = \theta_1\lambda_1$ for some λ_1. Moreover λ_1 unifies $\mathcal{E}\theta_1$ since $(\mathcal{E}\theta_1)\lambda_1 = \mathcal{E}\sigma$. So $\lambda_1 = \theta_2\lambda_2$ for some λ_2. But then $\sigma = \theta_1(\theta_2\lambda_2) = (\theta_1\theta_2)\lambda_2$.

<u>Theorem 1.32.</u> (Refinement Theorem). Let $\mathcal{E}$ be unifiable and let $\mathcal{E}_1, \ldots, \mathcal{E}_n$ be refinements of $\mathcal{E}$. Then $\theta_1 \ldots \theta_n\theta$ is an m.g.s.u. of $\mathcal{E}$ where

θ_i is an m.g.s.u of $\mathcal{E}_i\theta_0 \ldots \theta_{i-1}$ $(\theta_0 = \varepsilon)$,

and θ is an m.g.s.u. of $\mathcal{E}\theta_1 \ldots \theta_n$

<u>Proof.</u> It suffices to show that for all k, $0 \leq k \leq n$,

$\theta_0 \ldots \theta_k\theta'$ is an m.g.s.u. of $\mathcal{E}$ where

θ' is an m.g.s.u. of $\mathcal{E}\theta_1 \ldots \theta_k$.

For $k = 0$ this is trivial. Assume by way of induction hypothesis that the above is true for some $k < n$. By the preceding Lemma 1.3.1, since $\mathcal{E}_{k+1}\theta_0 \ldots \theta_k$ is a refinement of

$\mathcal{E}\theta_0\ldots\theta_k$, we have that

$\theta_{k+1}\theta''$ is an m.g.s.u. of $\mathcal{E}\theta_0\ldots\theta_k$ where

θ'' is an m.g.s.u. of $\mathcal{E}\theta_0\ldots\theta_{k+1}$.

Thus letting $\theta_{k+1}\theta''$ be the m.g.s.u. θ' of the induction hpothesis,

$\theta_0\ldots\theta_k\theta_{k+1}\theta''$ is an m.g.s.u. of $\mathcal{E}$ where

θ'' is an m.g.s.u. of $\mathcal{E}\theta_0\ldots\theta_k\theta_{k+1}$

<u>Corollary 1.3.3.</u> Let $\mathcal{E} = \{E_1,\ldots,E_n\}$ be unifiable. Then $\theta_1\ldots\theta_n$ is an m.g.s.u. of $\mathcal{E}$ where θ_i is an m.g.u. of $E_i\theta_0\ldots\theta_{i-1}$ ($\theta_0 = \varepsilon$).

<u>Proof.</u> Let $\mathcal{E}_i = \{E_i\}$. Then $\mathcal{E}_i$ is a refinement of $\mathcal{E}$ and θ_i is an m.g.s.u. of $\mathcal{E}_i\theta_0\ldots\theta_{i-1}$. So $\theta_1\ldots\theta_n\theta'$ is an m.g.s.u. of $\mathcal{E}$ where θ' is an m.g.s.u. of $\mathcal{E}\theta_1\ldots\theta_n$. But each $E_i\theta_1\ldots\theta_n$ is a singleton, so $\mathcal{E}\theta_1\ldots\theta_n$ is unified and θ' may be taken to be ε.

<u>Corollary 1.3.4.</u> Let $E = \{X_1,\ldots,X_n\}$ be unifiable where $X_1,\ldots,X_n$ are expressions. Then

$\theta_2\ldots\theta_n$ is an m.g.u. of E where

θ_i is an m.g.u. of $\{X_1,X_i\}\theta_1\ldots\theta_{i-1}$

($\theta_1 = \varepsilon$).

<u>Proof.</u> $\mathcal{E}_i = \{\{X_1,X_i\}\}$ is a refinement of $\mathcal{E} = \{E\}$ for $2 \leq i \leq n$. So

$\theta_2\ldots\theta_n\theta'$ is an m.g.s.u. of $\mathcal{E}$ where

θ' is an m.g.s.u. of $\mathcal{E}\theta_2\ldots\theta_n$.

But $\mathcal{E}\theta_2\ldots\theta_n$ is already unified, so we may let $\theta' = \varepsilon$.

Corollary 1.3.5. Let $\mathcal{E}$ be a unifiable family and let $\mathcal{E}_1, \ldots, \mathcal{E}_n$ be refinements of $\mathcal{E}$ which share no variables. Then

$\theta_1 \ldots \theta_n \theta'$ is an m.g.s.u. of $\mathcal{E}$ where

θ_i is an m.g.s.u. of $\mathcal{E}_i$ and

θ' is an m.g.s.u. of $\mathcal{E}\theta_1 \ldots \theta_n$

Proof. It suffices to show that

$\mathcal{E}_i = \mathcal{E}_i \theta_0 \ldots \theta_{i-1}$ where $\theta_0 = \epsilon$.

But since, for $i \neq j$, $\mathcal{E}_i$ and $\mathcal{E}_j$ share no variables $\mathcal{E}_i = \mathcal{E}_i \theta_j$.

Note that corollary 1.3.3 is essentially the simultaneous unification theorem of Andrews' [2] and that Hart's Theorem [15] is essentially Corollary 1.3.3 stated for $n = 2$.

1.4 Clashes

A standardised set of clauses $\mathcal{C} = \{A_1, \ldots, A_n, B\}$ is a clash if for $1 \leq i \leq n$

$A_i = E_i \overset{\circ}{\cup} A_{0i}$, $E_i \neq \emptyset$,

$B = F_1 \overset{\circ}{\cup} \ldots \overset{\circ}{\cup} F_n \overset{\bullet}{\cup} B_0$, $F_i \neq \emptyset$ and

$\mathcal{E} = \{E_1 \cup \overline{F}_1, \ldots, E_n \cup \overline{F}_n\}$ is unifiable with m.g.s.u. θ .

The clause

$$C = (A_{01} \cup \ldots \cup A_{0n} \cup B_0)\theta$$

is the resolvent of $\mathcal{C}$. The clauses in $\mathcal{C}$ are the parents of C. B is called the nucleus and the clauses A_i are called the satellites of $\mathcal{C}$. The sets of literals E_i in A_i and

$F_1 \dot{\cup} \ldots \dot{\cup} F_n$ in B are called the <u>literals resolved upon</u> in $\mathcal{C}$. Literals L $\in E_i$ and K $\in F_i$ are said to <u>mate</u> in $\mathcal{C}$. θ is called an m.g.s.u. of $\mathcal{C}$. If n = 1 then the distinction between the nucleus and satellite of $\mathcal{C}$ is ambiguous and $\mathcal{C}$ and its resolvent C are said to be binary. Note that $\mathcal{C}$ may contain variants of the same clause.

The definition of clash given above coincides with Brown's definition [3] and differs from the definition of latent clash given by Robinson in [42]

1.5 <u>Clash Restriction.</u>

Robinson includes in the definition of latent clash restrictions similar to the restriction below. This restriction is not included in the definition of clash above since in section 3.3 we investigate the completeness of a resolution rule for clashes which are not necessarily restricted.

Let $\mathcal{C} = \{E_1 \dot{\cup} D_1, \ldots, E_m \dot{\cup} D_m\}$ be a clash with m.g.s.u. θ where E_i is the non-empty set of literals in $E_i \dot{\cup} D_i$ resolved upon in $\mathcal{C}$. Then $C = (D_1 \cup \ldots \cup D_m)\theta$ is the resolvent of $\mathcal{C}$. $\mathcal{C}$ is restricted if

L $\in E_i\theta$ implies L $\notin$ C.

The motivation for introducing the notion of clash restriction is twofold:

(1) If $\mathcal{C}$ is restricted then the resolvent C of $\mathcal{C}$ can be obtained from $\mathcal{C}$ by a sequence of binary

resolutions.

(2) The sequence of binary resolvents of (1) contains no tautologies if neither C nor any of its parents in $\mathcal{C}$ is a tautology.

If $\mathcal{C}$ is not restricted then either (1) or (2) may fail to hold. For example if $\mathcal{C} = \{A_1, A_2, B\}$ where $A_1 = \{L_1, \overline{L}_2\}$ $A_2 = \{L_2, \overline{L}_1\}$ and $B = \{\overline{L}_1, \overline{L}_2\}$ then C=B cannot be obtained from $\mathcal{C}$ by any sequence of binary resolutions. If $\mathcal{C} = \{A_1, A_2, B\}$ where $A_1 = \{L_1, L_2\}$, $A_2 = \{L_2, L_1\}$ and $B = \{\overline{L}_1, \overline{L}_2\}$ then $C = A_1 = A_2$ can be obtained from $\mathcal{C}$ by a sequence of binary resolutions, but not without introducing tautologous resolvents.

The importance of (1) and (2) stems from the desirability of replacing clashes by sequences of binary resolutions. This point is taken up again in sections 2.8 and 2.9.

1.6 Factoring.

It is often convenient to regard as two consecutive operations the single operation of resolving a clash $\mathcal{C}$: first each clause in $\mathcal{C}$ is replaced by a factor and then the resulting set of factors $\mathcal{C}'$ is resolved in such a way as to obtain the resolvent C of $\mathcal{C}$. The principal motivation for considering factoring is to increase the efficiency of searching for refutations.

Several notions of factoring are possible and are studied in greater detail in Chapter 4. The following definitions are sufficiently general for present purposes and are equivalent, by the refinement theorem and its corollaries, to the definitions given by Wos and Robinson in [53].

If $C = C_1 \dot{\cup} \ldots \dot{\cup} C_n$ is a clause and $\mathcal{E} = \{C_1, \ldots, C_i\}$, $i \leq n$, then $\mathcal{E}$ is called a partition of C and a complete partition of C if $i = n$ (i.e. if $C = \cup \mathcal{E}$). Let $\mathcal{E}$ be a unifiable partition of C with m.g.s.u. θ then $C\theta$ is a factor of C. Resolution of factors is defined as for clauses in general with the restriction however that the sets of literals E_i and F_i resolved upon (in the notation of the definition given in section 1.4) are singletons. In other words a standardised set of factors $\mathcal{C} = \{A_1, \ldots, A_n, B\}$ is a clash if for $1 \leq i \leq n$

$$A_i = \{L_i\} \dot{\cup} A_{0i},$$
$$B = \{K_1, \ldots, K_n\} \dot{\cup} B_0 \text{ and}$$
$$\mathcal{E} = \{\{L_1, \bar{K}_1\}, \ldots, \{L_n, \bar{K}_n\}\} \quad \text{is unifiable.}$$

Then $C = (A_{01} \cup \ldots \cup A_{0n} \cup B_0)\theta$ is the resolvent of $\mathcal{C}$ where θ is an m.g.s.u. of $\mathcal{C}$.

The following more restrictive notion of factoring is equivalent to that introduced by Hayes and Kowalski in [17] . Let C be a clause and let $\mathcal{E} = \{C_1, \ldots, C_m\}$ be a unifiable partition of C with m.g.s.u. θ . The pair $D = (\cup(\mathcal{E}\theta), C\theta)$ is called a marked factor of C. The set $\cup(\mathcal{E}\theta)$ is called the set of distinguished literals of D. It will usually be the case that we identify the marked factor D with its second element $C\theta$. A marked factor cannot be factored. Resolution of a set $\mathcal{C}$ of marked factors is defined only for the case where the literals resolved upon in $\mathcal{C}$ are all and only the distinguished literals of factors in $\mathcal{C}$. A marked factor is a satellite factor, if its set of distinguished literals is a singleton. Thus satellites of a clash whose elements are marked factors are satellite factors. A marked factor $(\cup(\mathcal{E}\theta), C\theta)$ is an i-factor (idem-factor) of a clause C if $\mathcal{E}$ is already unified (i.e. if $\theta = \varepsilon$ and $\cup(\mathcal{E})\theta \subseteq C$). If a clause C contains n distinct literals then it has 2^n distinct i-factors and n distinct satellite i-factors.

The following theorem justifies replacing the operation of clash resolution by the two operations of generating factors and of resolving clashes of factors.

Theorem 1.6.1. Let $\mathcal{C} = \{A_1,\ldots,A_n,B\}$ be a clash with resolvent C where for $1 \leq i \leq n$

$$A_i = E_i \dot{\cup} A_{0i}, \quad E_i \neq \emptyset,$$

$$B = F_1 \dot{\cup} \ldots F_n \dot{\cup} B_0, \quad F_i \neq \emptyset,$$

$$\mathcal{E} = \{E_1 \cup \overline{F}_1, \ldots, E_n \cup \overline{F}_n\} \text{ and}$$

$$C = (A_{01} \cup \ldots \cup A_{0n} \cup B_0)\theta \text{ where } \theta \text{ is an m.g.s.u. of } \mathcal{E}$$

then $\mathcal{C}' = \{A_1{}', \ldots, A'_n, B'\}$ is a clash of marked factors with resolvent C where for $1 \leq i \leq n$

$$A_i{}' = (E_i\theta_i, A_i\theta_i)$$

$$B' = ((F_1 \dot{\cup} \ldots \dot{\cup} F_n)\theta_{n+1}, B\theta_{n+1})$$

$$\mathcal{E}_i = \{E_i\}, \quad \mathcal{E}_{n+1} = \{\overline{F}_1, \ldots, \overline{F}_n\} \text{ and}$$

θ_j is an m.g.s.u. of $\mathcal{E}_j$ for $1 \leq j \leq n + 1$.

$\mathcal{C}'$ is restricted if $\mathcal{C}$ is.

<u>Proof.</u> Because $\mathcal{C}$ is standardised all of the $\mathcal{E}_j$, $1 \leq j \leq n+1$, are refinements of $\mathcal{E}$ which share no variables. By Corollary 1.3.5 of the refinement theorem the m.g.s.u. θ of $\mathcal{E}$ may be taken to be $\theta_1 \ldots \theta_n \theta'$ where θ' is an m.g.s.u. of $\mathcal{E}\theta_1 \ldots \theta_{n+1}$.

For $1 \leq i \leq n$, let

$$\{L_i\} = E_i\theta_i \quad , \quad \{K_i\} = F_i\theta_{n+1} \quad \text{and}$$

$$\mathcal{E}' = \{\{L_1, \bar{K}_1\}, \ldots, \{L_n, \bar{K}_n\}\}$$

Then $\mathcal{E}' = \mathcal{E}\theta_1 \ldots \theta_{n+1}$ and θ' is an m.g.s.u. of $\mathcal{E}'$.

Therefore $\mathcal{C}'$ is a clash and its resolvent is

$$C' = (A_{01}\theta_1 \cup \ldots \cup A_{0n}\theta_n \cup B_0\theta_{n+1})\theta'$$

$$= (A_{01}\theta_1 \ldots \theta_{n+1} \cup \ldots \cup A_{0n}\theta_1 \ldots \theta_{n+1} \cup B_0\theta_1 \ldots \theta_{n+1})\theta'$$

$$= (A_{01} \cup \ldots \cup A_{0n} \cup B_0)\theta_1 \ldots \theta_{n+1}\theta'$$

$$= C$$

Suppose that $\mathcal{C}$ is restricted and that $\mathcal{C}'$ is not. Then for some L' resolved upon in $\mathcal{C}'$, $L'\theta' \in C$. But $L' = L_i$ or $L' = K_i$ for some i. Therefore for some L resolved upon in $\mathcal{C}$, $L' = L\theta_j = L\theta_1 \ldots \theta_{n+1}$ for some j, $1 \leq j \leq n+1$, and $L\theta = L'\theta' \in C$. It follows that $\mathcal{C}$ is not restricted, contrary to the assumption.

The refinement theorem and its corollaries suggest various ways of computing an m.g.s.u. θ of a clash $\mathcal{C}$. In particular the computation of θ can be reduced first to the computation of the m.g.s.u.s $\theta_1, \ldots, \theta_{n+1}$ and θ'

of the theorem above. This particular reduction is an attractive one because each θ_i can be computed independently. It does not seem unreasonable to assume therefore that the effort involved in resolving a clash $\mathcal{C}$ is equal to the effort involved in generating and resolving the corresponding set of marked factors $\mathcal{C}'$. In searching for refutations it is usual for variants of the same clause to occur in several different clashes. By storing the factors generated in resolving a clash it is not necessary to recompute them when they occur in other clashes. Thus by a suitable implementation of factoring it is possible both to simplify the programming and to increase the efficiency of clash resolution.

1.7 Trees.

A tree is a pair (T,s) where T is a non-empty set of elements called nodes and s is a function $s: T \rightarrow T$ such that:

(1) $s^{-1}(N)$ is finite for all $N \in T$ (i.e. (T,s) is finitely branching.)

(2) $s(N_0) = N_0$ for exactly one element $N_0 \in T$ called the root of (T, s) and denoted by $r(T)$.

(3) Define $s^0(N)=N$ and $s^{n+1}(N)= s(s^n(N))$, Then for all $N \in T$ there is an $n \geq 0$ such that $s^n(N) = r(T)$. (i.e. well-foundedness: if $X \neq \emptyset$ is a subset of T then there exists an $N \in X$

such that $N = r(T)$ or $s(N) \notin X$).

(4) $s^n(N) = N$ if and only if $n = 0$ or $N = r(T)$ (i.e. T contains no loops).

If $N \in T$ and $s^{-1}(N) = \emptyset$ then N is a tip of (T,s), otherwise N is an interior node of (T,s). When, as is usually the case, there is no possibility of confusion we supress reference to the function s and refer to T itself as though it were the tree (T,s). It is sometimes convenient to think of trees as growing upwards. Thus $r(T)$ lies above no nodes in T and tips of T lie below no nodes in T.

A branch of a tree T is a set $\mathcal{B} \subseteq T$ such that

(1) $r(T) \in \mathcal{B}$

(2) $N \in \mathcal{B}$ implies $s(N) \in \mathcal{B}$ and

(3) $N \in \mathcal{B}$ implies $s^{-1}(N) \cap \mathcal{B}$ contains at most one element.

A branch $\mathcal{B}$ is complete if

(3') $N \in \mathcal{B}$ implies $s^{-1}(N) \cap \mathcal{B}$ contains exactly one element unless $s^{-1}(N) = \emptyset$.

Notice that any node $N \in T$ determines a branch $\mathcal{B}$ of T, $\mathcal{B} = \{ s^n(N) : n=0,1,\dots \}$. Every branch of T contains at most one tip.

Given a tree T and node $N \in T$ let T_N be the smallest subset of T such that

(1) $N \in T_N$ and

(2) $N' \in T_N$ implies that $s^{-1}(N') \subseteq T_N$.

Thus T_N consists of the node N together with all nodes of T lying above N. T_N has a natural interpretation as a subtree, (T_N, s') of T where $s'(N)=N$ and $s'(N')=s(N')$ for $N' \neq N$. . T_N is called the <u>subtree of T rooted in N.</u> Notice that $T_{r(T)}=T$.

Given a tree T a <u>cut through T</u> is a non-empty set $X \subseteq T$ such that $X \cap \mathcal{B}$ is a singleton for every complete branch $\mathcal{B} \subseteq T$, i.e. X contains exactly one node from every complete branch $\mathcal{B}$ of T. If X is a cut through T then X determines a subtree $(T/X, s')$ where $T/X = \{ s^n(N) : N \in X, n=1,2,... \}$ and s' is the restriction of s to T/X. Note the following properties of cuts:

<u>1.7.1.</u> If $X=\{r(T)\}$ then X is a cut and $T/X = r(T)$

<u>1.7.2.</u> If T is finite and $X=\{ N : N \in T \text{ and } s'(N)=\emptyset \}$ then X is a cut and $T/X = T$.

<u>1.7.3.</u> X is the set of tips of T/X.

<u>1.7.4.</u> (König's Lemma). If X is a cut through T then T/X is finite.

<u>Proof:</u> Each complete branch $\mathcal{B}$ in T/X is finite since each such $\mathcal{B}$ contains a tip in T/X, i.e. the unique element of $\mathcal{B} \cap X$. Suppose T/X is infinite then we can construct an infinite complete branch $\mathcal{B}_0$ of T/X as follows:

$r(T/X) = r(T) \in \mathcal{B}_0$. If $N \in \mathcal{B}_0$ then the subtree of T/X rooted in N is infinite. Since $s^{-1}(N)$ is finite the subtree of T/X

rooted in some $N' \in s^{-1}(N)$ is infinite.
Let $N' \in \mathcal{B}_0$. Then $\mathcal{B}_0$ is infinite and contains no tip.

<u>1.7.5</u> If X is a cut through T and $X \neq \{r(T)\}$ then $s^{-1}(N) \subseteq X$ for some $N \in T$.

<u>Proof:</u> Suppose for every $N \in T$ there is an $N' \in s^{-1}(N)$ such that $N' \in X$. Then construct a complete branch $\mathcal{B}_0$ of T such that $\mathcal{B}_0 \cap X = \emptyset$ as follows: $r(T) \in \mathcal{B}_0$. If $N \in \mathcal{B}$ and $N' \in s^{-1}(N)$, $N' \notin X$, then $N' \in \mathcal{B}_0$. Then $X \cap \mathcal{B}_0 = \emptyset$ and therefore X is not a cut through T.

<u>1.8 Derivations.</u>

Let T be a tree and c a function defined on the nodes of T having clauses as values. For $X \subseteq T$ define $c(X) = \{ c(N) : N \in X \}$. A pair $\mathcal{D} = (T,c)$ is a <u>derivation</u> relative to given logically valid inference rules, if for all interior $N \in T$, c(N) can be obtained from $c(s^{-1}(N))$ by a single application of one of the given inference rules. If X is the set of tips of T, if $c(X) \subseteq S$ and if $C=c(r(T))$ then $\mathcal{D}$ is a <u>derivation of C from S.</u> If $C = \Box$ then $\mathcal{D}$ is a <u>refutation of S.</u> We also say that $\mathcal{D}$ is a derivation from S (or refutation of S) when $\mathcal{D}$ is a derivation from a set S' (refutation of S') and S' consists of variants of clauses in S. (This convention is necessitated by the decision to consider as clashes only standardised sets of clauses). If X is a cut through T and $c(X) \subseteq S$ then $\mathcal{D}' = (T/X, c')$, where c' is c restricted to T/X, is a derivation from S of $C = c(r(T)) = c'(r(T/X))$ and S logically implies C; if $C = \square$ then S is unsatisfiable. In order to simplify notation we usually write $\mathcal{D}' = (T/X, c)$ instead of $\mathcal{D} = (T/X, c')$. Similarly for $N \in T$ we denote the derivation $\mathcal{D}_N = (T_N, c')$, where c' is c restricted to T_N by writing $\mathcal{D}_N = (T_N, c)$.

Until section 2.8 we use the term "derivation" to refer to clash derivation. $\mathcal{D} = (T, c)$ is a clash derivation if for all interior $N \in T$, $c(s^{-1}(N))$ is a clash and $c(N)$ is its resolvent. Given such a derivation $\mathcal{D} = (T, c)$ and N interior to T, $c(s^{-1}(N))$ is said to be the clash at N. If $N' \in s^{-1}(N)$ then the subset of $c(N')$ of literals resolved upon in $c(s^{-1}(N))$ is called the set of literals resolved upon at N'. If $c(N')$ is a satellite of the clash $c(s^{-1}(N))$ then N' is called a satellite node of $\mathcal{D}$. Similarly if $c(N')$ is nucleus of $c(s^{-1}(N))$, N' is called a nucleus node of $\mathcal{D}$. If $N \in T$ then an occurrence of $L \in c(N)$ descends from the same occurrence of $L \in c(N)$; If $N' \in s^{-1}(N)$, if θ is the m.g.s.u. of the clash at N, if $L'\theta = L \in c(N)$ for $L' \in c(N')$ not resolved upon at N and if the occurrence of L' in $c(N')$ descends from an occurrence of L" in $c(N")$, then the occurrence of L in $c(N)$ descends from the occurrence of L" in $c(N")$.

1.9 Search Strategies.

We distinguish between complete inference systems and complete proof procedures. A refutation complete inference system is a set of effective inference rules which when applied to an unsatisfiable set of clauses S_0 yields a refutation of S_0. The refutation completeness of a resolution rule $\mathcal{R}$ can be formulated as an assertion that for any unsatisfiable set S_0 there exists a refutation $\mathcal{D}$ such that each resolvent of a clash in $\mathcal{D}$ is obtainable by an application of $\mathcal{R}$. A refutation complete proof procedure is semi-effective method for eventually obtaining a refutation of a set of clauses S_0 when S_0 is unsatisfiable. Thus a proof procedure consists both of an inference system and of a search strategy for obtaining refutations within the system of inference rules.

The usual statements of completeness for resolution systems implicitly assert the completeness of a particular class of resolution proof procedures. It is easy to invent British Museum methods for searching resolution refutations. Such methods might, for instance, enumerate all resolution derivations rejecting those which were not refutations of a given input set S_0 continuing until a first such refutation were found. At any given time only one derivation might be under consideration. Such search strategies would be extremely inefficient and much of the efficiency of resolution derives from the efficiency of the search strategies associated with it.

We shall say that an arbitrary set of clauses $\mathcal{C}$ is a clash if some standardised set $\mathcal{C}'$ of variants of clauses in $\mathcal{C}$ is a clash. The resolvent of $\mathcal{C}$ is identical to the resolvent of $\mathcal{C}'$. Given a set of clauses S and a resolution rule $\mathcal{R}$ let $\mathcal{R}(S)$ be the set of all resolvents C of clashes $\mathcal{C} \subseteq S$, where $\mathcal{C}$ is an admissible set of premises for application of the rule $\mathcal{R}$. For a given input set of clauses S_0 let $\mathcal{R}^0(S_0) = S_0$ and for $n > 0$

$$\mathcal{R}^n(S_0) = \{ C \ : \ C \in \mathcal{R}\left(\bigcup_{i=0}^{n-1} \mathcal{R}^i(S_0) \right)$$

and $C \notin \bigcup_{i=0}^{n-1} \mathcal{R}^i(S_0) \}$.

thus $C \in \mathcal{R}^n(S_0)$ if and only if there is a derivation $\mathcal{D}$ of C from S_0 such that each application of resolution in $\mathcal{D}$ is an application of $\mathcal{R}$ and such that $n + 1$ is the number of distinct nodes in the longest complete branch of $\mathcal{D}$. The set $\bigcup_{i=0}^{\infty} \mathcal{R}^i(S_0)$ is called the search space for S_0.

Given a set of clauses S_0 a resolution procedure (resolution rule $\mathcal{R}$ plus search strategy) generates a sequence of clauses $(C_1, \dots, C_n, \dots)$ from the search space for S_0. This procedure either terminates without generating the null clause, terminates when some first $C_n = \square$ or does not terminate and no $C_n = \square$. In all of these cases we may imagine that the procedure continues until all of the clauses in $\bigcup_{i=0}^{\infty} \mathcal{R}^i(S_0)$ have been generated. The resulting (finite or infinite) sequence $(C_1, \dots, C_n, \dots)$ is called the trace for S_0 of the given proof procedure. Necessary conditions that a sequence $(C_1, \ldots, C_n, \ldots)$ be a trace for a set of clauses S_0 of a proof procedure $\mathcal{P}$, consisting of a resolution rule $\mathcal{R}$ and search strategy, are that

(1) for every C_n, $n \geq 1$, either $C_n \in S_0$ or C_n is the resolvent of a clash $\mathcal{C} = \{C_{n_1}, \ldots, C_{n_m}\}$ such that $C_n \in \mathcal{R}^1(\mathcal{C})$ and $n_i < n$ for all i, $1 \leq i \leq m$, and

(2) if $\mathcal{C} = \{C_{n_1}, \ldots, C_{n_m}\}$ is a clash with resolvent $C \in \mathcal{R}^1(\mathcal{C})$ then $C = C_n$ for some $n > \max \{n_1, \ldots, n_m\}$ (provided that no C_{n_i} has been deleted).

A search strategy is a <u>depth saturation strategy</u> if for every trace $(C_1, \ldots, C_n, \ldots)$

$C_i \in \mathcal{R}^n(S_0)$, $C_j \in \mathcal{R}^m(S_0)$ and $i < j$ imply $n \leq m$.

Depth saturation has the appealing defining property of generating simpler derivations before more complicated ones. What is more desirable is that simpler refutations be generated before more complicated ones and that derivations which can predictably contribute to simpler refutations be generated before derivations which can predictably contribute only to more complicated refutations. This last property partially defines the family of diagonal search strategies studied in chapter 4.

If a resolution procedure includes deletion rules (e.g. deletion of variants, subsumed clauses, tautologies, etc.)

then we include in the trace Υ for any set S_0 all clauses rejected by the deletion rules but include no other clauses obtainable by derivations from such clauses after their deletion. Thus if $(C_1, \ldots, C_n, \ldots)$ is a trace Υ and the clause C_i is deleted immediately after the generation of the clause C_j then $i \leq j$ and at most only the clauses $C_i, C_{i+1}, \ldots, C_j$ in Υ are obtainable by derivations containing C_i. This convention allows us to treat the number $n - 1$ of clauses occurring before the first $C_n = \Box$ in the trace $(C_1, \ldots, C_n, \ldots)$ of an unsatisfiable set S_0 as a <u>measure of the difficulty</u> of refuting S_0 by the given proof procedure. This measure is a first approximation which does not take into account, for instance, the effort involved in testing for the applicability of the deletion rules themselves. It might be agreed, that, given a program which implements a proof procedure, a more appropriate measure would be the total time taken to refute S_0. Such a measure would however more accurately quantify the effort expended by the program than it would the effort expended by the proof procedure itself. In fact, given such a program, an ideal measure would be the total cost involved in refuting S_0 (including charges for use of a computer, and for writing and maintaining the program). In any case it is important to note that the difficulty of refuting an unsatisfiable set S_0 is completely independent of the complexity of a refutation of S_0. The complexity of a derivation $\mathcal{D}=(T, c)$ can be measured entirely in terms of intrinsic properties of $\mathcal{D}$ (i.e. the number of nodes in T, the length of the longest complete branch of T, etc.), whereas the difficulty of refuting a set of clauses S_0 has to be measured in terms of the total effort expended to obtain a first refutation of S_0. The purpose of developing more efficient theorem-proving methods can be met only by reducing the difficulty involved in refuting unsatisfiable sets of clauses. Thus much of the research in automatic theorem-proving, involved in reducing the complexity of derivations, is unrelated to the principal goal of theorem-proving research.

A deletion rule is compatible with a proof procedure $\mathcal{P}$ (complete relative to $\mathcal{P}$) if whenever $\mathcal{T}_1$ is the trace for some S_0 of $\mathcal{P}$, $\mathcal{T}_2$ is the trace obtained by applying the deletion rule to clauses in $\mathcal{T}_1$ and some C_n in $\mathcal{T}_1$ is $\square$, then some $C'_{n'}$ in $\mathcal{T}_2$ is $\square$. A deletion rule may be complete yet fail to be efficient if $n' > n$. If C_n and $C'_{n'}$ are the first occurrences of $\square$ in $\mathcal{T}_1$ and $\mathcal{T}_2$ respectively then a sufficient condition for the deletion rule to increase the efficiency of refuting S_0 (ignoring the effort involved in applying the deletion rule) is for n' to be less than n. In sections 1.11 and 1.12 we investigate the completeness and efficiency of deletion of subsumed clauses and tautologies.

1.10 Contractions.

The lifting theorem asserts that given a derivation $\mathcal{D} = (T,c)$ and given for every tip $N \in T$ a clause A_N which has $c(N)$ as an instance then there exists an isomorphic derivation

$\mathcal{D}' = (T,c')$ from $S' = \{A_N : N \in T$ is a tip$\}$ such that if S' is standardised then $c(N)$ is an instance of $c'(N)$ for all $N \in T$ and $c'(N) = A_N$ for N a tip of T. The contraction theorem is obtained by generalising the lifting theorem, allowing A_N to subsume $c(N)$ when $N \in T$ is a tip. The resulting derivation $\mathcal{D}'$ from S' of a clause which subsumes $c(r(T))$ is then a contraction of $\mathcal{D}$. The contraction theorem yields the lifting theorem as a special case and in its more general form is used for applications later in chapter 1 as well as in chapters 2 and 3. We note that our generalisation of the lifting lemma was motivated in part by Brown's generalisation in [3] .

A set of clauses $\mathcal{C}'$ subsumes another set of clauses $\mathcal{C}$ if for some substitution σ and every $A \in \mathcal{C}$ there is an unique $A' \in \mathcal{C}'$ such that $A'\sigma \subseteq A$. We also require that for every $A' \in \mathcal{C}'$ there be an unique $A \in \mathcal{C}$ such that $A'\sigma \subseteq A$. Thus σ induces a 1-1 correspondence between clauses $A' \in \mathcal{C}'$ and $A \in \mathcal{C}$ such that $A'\sigma \subseteq A$.

Let $\mathcal{C}$ and $\mathcal{C}'$ be clashes. Then $\mathcal{C}'$ covers $\mathcal{C}$ if

(1) $\mathcal{C}'$ subsumes some subset of $\mathcal{C}$ (let σ be such that $A'\sigma \subseteq A$ for corresponding $A' \in \mathcal{C}'$ and $A \in \mathcal{C}$),

(2) $A' \in \mathcal{C}'$ is a satellite (or nucleus) of $\mathcal{C}'$ if and only if $A'\sigma$ is a satellite (nucleus) of $\mathcal{C}$,

(3) the resolvent of $\mathcal{C}'$ subsumes the resolvent of $\mathcal{C}$,

(4) $\mathcal{C}'$ is restricted if $\mathcal{C}$ is and

(5) if $A' \in \mathcal{C}'$ then $L \in A'$ is resolved upon in $\mathcal{C}'$ if and only if $L\sigma \subseteq A'\sigma$ is resolved upon in $\mathcal{C}$.

$\mathcal{C}'$ <u>weakly covers</u> $\mathcal{C}$ if (1), (3) - (5) above hold for $\mathcal{C}$ and $\mathcal{C}'$. The notion of covering here is only weakly related to Sibert's notion defined in [48]. The lemma below is the local version of the contraction theorem and is used in section 1.11 to study the subsumption strategy.

<u>Lemma 1.10.1.</u> Let $\mathcal{C}$ be a clash with resolvent C and let $\mathcal{C}'$ subsume $\mathcal{C}$. Then either

(1) some $C' \in \mathcal{C}'$ subsumes C or

(2) some subset $\mathcal{C}''$ of $\mathcal{C}'$ is a clash and $\mathcal{C}''$ covers $\mathcal{C}$.

If $\mathcal{C}$ is a set of instances of clauses in $\mathcal{C}'$, then $\mathcal{C}'' = \mathcal{C}'$ and C is an instance of the resolvent of $\mathcal{C}'$.

<u>Proof.</u> Let $\mathcal{C} = \{A_1, \ldots, A_n, B\}$ and $\mathcal{C}' = \{A_1', \ldots, A_n', B'\}$. Let σ be such that $A_i'\sigma \subseteq A_i$ and $B'\sigma \subseteq B$. For $1 \leq i \leq m$ let

$$A_i = E_i \mathbin{\dot\cup} A_{0i}, \quad E_i \neq \emptyset,$$

$$B = F_1 \mathbin{\dot\cup} \ldots \mathbin{\dot\cup} F_m \mathbin{\dot\cup} B_0, \quad F_i \neq \emptyset, \text{ where}$$

$$\mathcal{E} = \{E_1 \cup \bar{F}_1, \ldots, E_m \cup \bar{F}_m\} \text{ and}$$

$$C = (A_{01} \cup \ldots \cup A_{0m} \cup B_0)\theta \text{ where } \theta \text{ is an}$$

m.g.s.u. of $\mathcal{E}$.

<u>Case (1).</u> If $A_i'\sigma \subseteq A_{0i}$ for some i then $A_i'\sigma\theta \subseteq C$ so A_i' subsumes C. Similarly if $B'\sigma \subseteq B$ then B' subsumes C.

<u>Case (2).</u> If case (1) does not apply then $B'\sigma \cap (F_1 \dot{\cup} \ldots \dot{\cup} F_m) \neq \emptyset$. Assume that $B'\sigma \cap F_i \neq \emptyset$ for $1 \leq i \leq m' \leq m$ (by rearranging subscripts if necessary). For $1 \leq i \leq m'$ let

$$A_i' = E_i' \dot{\cup} A_{0i}' \ , \ E_i' \neq \emptyset ,$$
$$B' = F_1' \dot{\cup} \ \ldots \dot{\cup} F_m' \ \dot{\cup} \ B_0' \ , F_i' \neq \emptyset ,$$
$$\mathcal{E}' = \{E'_1 \cup \bar{F}'_1 \ , \ldots , E'_{m'} \cup \bar{F}'_{m'}\} \quad \text{where}$$
$$E_i'\sigma \subseteq E_i \ , \quad F_i'\sigma \subseteq F_i \ ,$$
$$A_{0i}'\sigma \subseteq A_{0i} \ \text{and} \ B_0'' \subseteq B_0 .$$

Notice that $\sigma\theta$ unifies $\mathcal{E}'$. Let θ' be an m.g.s.u. of $\mathcal{E}'$ then $\sigma\theta = \theta'\lambda$ for some λ. The resolvent of $\mathcal{C}'' = \{A_1' , \ldots , A_{m'}' , B'\} \subseteq \mathcal{C}'$ is

$$C' = (A_{01}' \cup \ldots \cup A_{0m'}' \cup B_0'\} \theta' \quad .$$

C' subsumes C since $C'\lambda \subseteq C$ (because

$$A_{0i}'\theta'\lambda = A_{0i}'\sigma\theta \subseteq A_{0i}\theta \subseteq C \ \text{and}$$
$$B_0'\theta'\lambda = B_0'\sigma\theta \subseteq B_0\theta \subseteq C).$$

That a literal L is resolved upon in $\mathcal{C}''$ if and only if L is resolved upon in $\mathcal{C}$ follows from the fact that

$$E_i'\sigma \subseteq E_i \ \text{and} \ F_i'\sigma \subseteq F_i .$$

Suppose $\mathcal{C}''$ is not restricted. Then for some i either

$$E_i'\theta' \subseteq C' \ \text{or} \ F_i'\theta' \subseteq C'.$$

But then

$$E_i'\theta'\lambda = E_i'\sigma\theta \subseteq C'\lambda = C \ \text{and} \ E_i\theta \subseteq C \ \text{or}$$

$$F_i'\theta'\lambda \;= F_i'\sigma\theta \subseteq C'\lambda = C \text{ and } F_i\theta \subseteq C$$

and $\mathcal{C}$ is not restricted. It follows that $\mathcal{C}''$ covers $\mathcal{C}$.
In case each A_i is an instance of A_i' and B is an instance of B' then case (a) does not apply, $m' = m$; so $\mathcal{C}'' = \mathcal{C}'$ and since all inclusions become equalities $C'\lambda = C$, i.e. C is an instance of C'.
Let $\mathcal{D} = (T,c)$ and $\mathcal{D}' = (T',c')$ be derivations. We define the notion, $\mathcal{D}'$ contracts $\mathcal{D}$ (also $\mathcal{D}'$ is a contraction of $\mathcal{D}$), by induction on the number of n of nodes in T :

(1) If $n = 1$ and $T = \{N_0\}$, then $T' = \{N'_0\}$ and $c'(N_0')$ subsumes $c(N_0)$.

(2) If $n > 1$ then let $N_0 = r(T)$, $s^{-1}(N_0) = \{N_1,\dots,N_m\}$, $\mathcal{C} = c(s^{-1}(N_0))$ and $\mathcal{D}_i = (T_{N_i},c)$, $1 \leq i \leq m$.

One of (a) and (b) holds.

(a) $\mathcal{D}'$ contracts some $\mathcal{D}_i$ and $c'(N_0')$ subsumes $c(N_0)$, where $N_0' = r(T')$.

(b) Let $N_0' = r(T')$, then $s^{-1}(N_0') \neq \emptyset$. Let
$s^{-1}(N_0') = \{N_1',\dots, N'_{m'}\}$,
$\mathcal{C}' = c'(s^{-1}(N_0')$ and
$\mathcal{D}_i' = (T'_{N'_i},c')$, $1 \leq i \leq m'$. Then
$\mathcal{D}_i'$ contracts $\mathcal{D}_i$ for all i, $1 \leq i \leq m' \leq m$
(after rearranging subscripts if necessary)
and $\mathcal{C}'$ covers $\mathcal{C}$.

Thus if $\mathcal{D}$ is a derivation of a clause C from clauses S, if $\mathcal{D}'$ is a derivation of C' from S' and if $\mathcal{C}'$ contracts $\mathcal{C}$, then C' subsumes C and each clause in S' subsumes a clause in S (provided $S' = \{c'(N') : N' \in T' \text{ is a tip}\}$). Associated with every contraction $\mathcal{D}' = (T'\ c')$ of a derivation $\mathcal{D} = (T, c)$ is a 1 - 1 mapping $\Psi : T' \to T$ such that for every $N' \in T'$ the clash $c'(s^{-1}(N'))$ covers the clash $c(s^{-1}(N))$. The mapping Ψ is defined (using the notation in the definition of contraction) by induction on the number n of nodes in T:

(1) If $n = 1$ then $\Psi(N_0') = N_0$.

(2) If $n > 1$ then

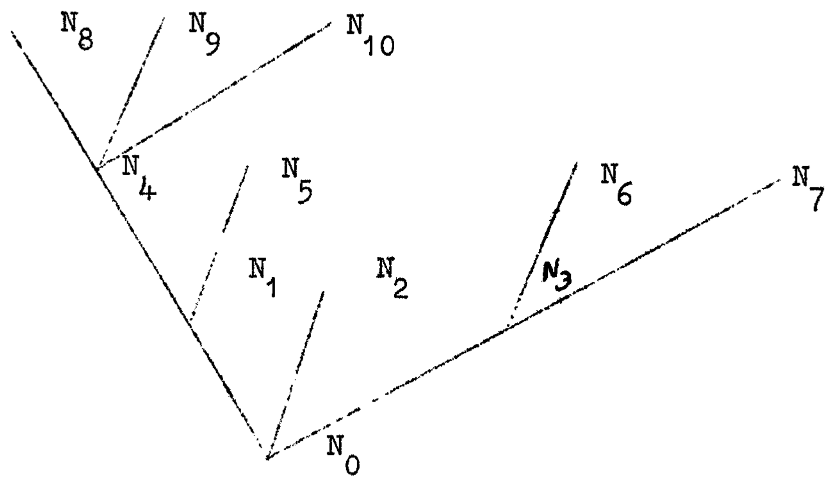
(a) If $\mathcal{D}'$ contracts some $\mathcal{D}_i$ with associated mapping Ψ_i then Ψ_i is also associated with the contraction $\mathcal{D}'$ of $\mathcal{D}$, otherwise

(b) If Ψ_i is associated with the contraction $\mathcal{D}_i'$ of $\mathcal{D}_i$ for $1 \leq i \leq m'$, then

$$\Psi(N_0') = N_0 \text{ and}$$
$$\Psi(N') = \Psi_i(N') \text{ for } N' \in T' \cap T'_{N_i'}.$$

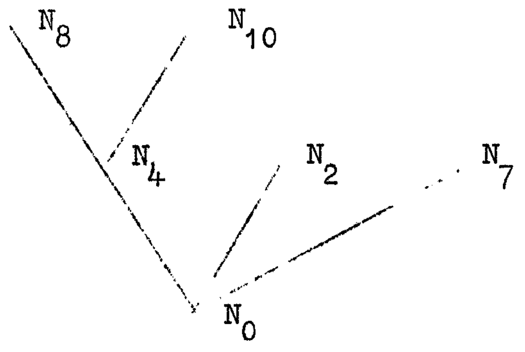
Examples.

(1) Let $\mathcal{D} = (T, c)$ where $T = \{N_0, \ldots, N_{10}\}$ and where the function s on T is defined by the diagram below and c is defined by the following equations.



$c(N_0) = \{\bar{P}(b)\}$	$c(N_6) = \{Q(a)\}$
$c(N_1) = \{\bar{P}(a), \bar{P}(b)\}$	$c(N_7) = \{\bar{Q}(a), P(a), \bar{Q}(b)\}$
$c(N_2) = \{Q(y), \bar{P}(y)\}$	$c(N_8) = \{P(b)\}$
$c(N_3) = \{P(a), \bar{Q}(b)\}$	$c(N_9) = \{Q(a), Q(b)\}$
$c(N_4) = \{Q(b), \bar{P}(a)\}$	$c(N_{10}) = \{\bar{P}(b), \bar{Q}(x), \bar{P}(x)\}$
$c(N_5) = \{\bar{Q}(b), \bar{P}(b)\}$	

Let $\mathfrak{D}' = (T', c')$ where $T' = \{N_0, N_2, N_4, N_7, N_8, N_{10}\}$ and where s on T' is defined by the diagram below and c' is defined by the following equations

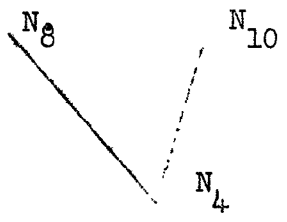


$c'(N_0) = \{\bar{P}(b)\}$	$c'(N_7) = \{P(z), \bar{Q}(b)\}$
$c'(N_2) = \{Q(y), \bar{P}(y)\}$	$c'(N_8) = \{P(u)\}$
$c'(N_4) = \{\bar{P}(v)\}$	$c'(N_{10}) = \{\bar{P}(b), \bar{P}(v)\}$

Then $\mathcal{D}'$ contracts $\mathcal{D}$ and $\psi(N_i) = N_i$ for all $N_i \in T'$ where ψ is the mapping associated with the contraction.

(2) Let $\mathcal{D}$, $\mathcal{D}'$ and ψ be defined as in example (1).

Let $\mathcal{D}'' = (T'', c'')$ where $T'' = \{N_4, N_8, N_{10}\}$ and where s on T" is defined by the following diagram and c" by the following equations.



$c''(N_4) = \square$

$c''(N_8) = \{P(u)\}$

$c''(N_{0}) = \{\bar{P}(w)\}$

then $\mathcal{D}''$ is a contraction of both $\mathcal{D}$ and $\mathcal{D}'$.

The associated mapping ψ' is defined by $\psi'(N_i) = N_i$ for all $N_i \in T''$, both for the contraction of $\mathcal{D}$ by $\mathcal{D}''$ and of $\mathcal{D}'$ by $\mathcal{D}''$.

<u>Theorem 1.10.2</u> (contraction theorem). Let $\mathcal{D} = (T, c)$ and for every tip N let A_N be a clause which subsumes c(N). Let $S = \{A_N : N \in T \text{ is a tip}\}$ be standardised. Then there exists a contraction $\mathcal{D}' = (T', c')$ of $\mathcal{D}$ which is a derivation from S. If each c(N) is an instance of A_N, when N is a tip, then $T' = T$ and $\mathcal{D}'$ lifts $\mathcal{D}$.

<u>Proof</u> (by induction on the number n of nodes in T).

If $n = 1$ then $T = N_0$. Let $T' = T$ and $c'(N_0) = A_N$.

Suppose that $n > 1$ and that the theorem holds for derivation trees containing fewer than n nodes, let $N_0 = r(T)$ and $s^{-1}(N_0) = \{N_1, \ldots, N_m\}$. Let $\mathcal{D}_i = (T_{N_i}, c)$. Since each T_{N_i}

contains fewer than n nodes, by the induction hypothesis, there exist contractions $\mathcal{D}_i' = (T_i', c_i')$ of $\mathcal{D}_i$. Each $\mathcal{D}_i'$ is a derivation from S and since S is standardised no clause in $\mathcal{D}_i'$ shares variables with any clause in $\mathcal{D}_j'$ for $i \neq j$.
Let $N_i' = r\ (T_i')$, $1 \leq i \leq m$, let

$$\mathcal{C} = \{c\ (N_1), \dots, c\ (N_m)\} \quad \text{and}$$
$$\mathcal{C}' = \{c_1'\ (N_1), \dots, c'_m(N'_m)\} \ .$$

$\mathcal{C}$ is a clash with resolvent $c\ (N_0)$ and $c_i'(N_i')$ subsumes $c\ (N_i)$ for each i, $1 \leq i \leq m$. $\mathcal{C}'$ is standardised.
By the preceding lemma either

(1) some $c_i'\ (N'_i)$ subsumes $c\ (N_0)$ or

(2) some $\mathcal{C}'' \subseteq \mathcal{C}'$ is a clash with resolvent C' and $\mathcal{C}''$ covers $\mathcal{C}$. Let $\mathcal{C}'' = \{c_1'(N_1'), \dots, c_m'\ (N'_m\ )\}$.

<u>Case (1)</u> Let $\mathcal{D}' = \mathcal{D}_i'$. Then $\mathcal{D}'$ is the desired contraction of $\mathcal{D}$.

<u>Case (2)</u> Let $\mathcal{D}' = (T', c')$ be defined as follows:

$T' = \{N_0'\} \ \dot{\cup}\ T_1' \ \dot{\cup} \dots \dot{\cup}\ T'_{m'}$,

$s^{-1}\ (N_0') = \{N_1', \dots, N'_{m'}\}$,

$c'\ (N_0') = C'$ and

$c'\ (N) = c_i'\ (N)$ for $N \in T' \cap T_i'$.

Then $\mathcal{D}'$ is the desired contraction of $\mathcal{D}$. In case each $c(N)$ is an instance of A_N for each tip N then by induction hypothesis each $T_i' = T_{N_i}$ and $\mathcal{D}_i'$ lifts $\mathcal{D}_i$. Therefore case (1) does not apply and $\mathcal{C}'' = \mathcal{C}'$. If we let $N_0' = N_0$ then $T' = T$, $c\ (N)$ is an instance of $c'(N)$ for each N and $\mathcal{D}'$ lifts $\mathcal{D}$.

The fact that the contraction lemma provides information about the completeness and efficiency of subsumption suggests that a similar theorem might serve the same purpose for deletion of tautologies. In section 1.12 we show that for any derivation $\mathcal{D}$ from clauses S, where $\mathcal{D}$ possibly contains tautologies, there exists a derivation $\mathcal{D}'$ from S where $\mathcal{D}'$ contains no tautologies and $\mathcal{D}'$ is a semi-contraction of $\mathcal{D}$. The definition of semi-contraction is obtained by replacing the condition that $\mathcal{C}'$ covers $\mathcal{C}$ (in (2b) of the definition of contraction) by the weaker condition that $\mathcal{C}'$ weakly covers $\mathcal{C}$. Thus every contraction is a semi-contraction but not conversely. Associated with every semi-contraction $\mathcal{D}'$ of a derivation $\mathcal{D}$ is a mapping ψ defined as for contractions.

In order to apply the generalised version of the lifting lemma and to obtain information about the completeness of deleting subsumed clauses and tautologies we need to examine some of the properties preserved under contractions and semi-contractions. We note that if $\mathcal{D}' = (T', c')$ is a semi-contraction (contraction) of a derivation $\mathcal{D} = (T, c)$ then

1.10.3. $\mathcal{D}'$ is a refutation if $\mathcal{D}$ is,

1.10.4. $\mathcal{D}'$ is binary if $\mathcal{D}$ is,

1.10.5. for all $N \in T', c'(N)$ is not a tautology if $c(\psi(N))$ is not, where ψ is the mapping associated with the contraction $\mathcal{D}'$ of $\mathcal{D}$ (thus $\mathcal{D}'$ contains no more tautologies than $\mathcal{D}$) and

1.10.6. if $\mathcal{D}''$ is a semi-contraction (contraction) of $\mathcal{D}'$ then $\mathcal{D}''$ is a semi-contraction (contraction) of $\mathcal{D}$.

The following properties are noted in the sequel:

1.10.7. $\mathcal{D}'$ is minimal if $\mathcal{D}$ is (Theorem 1.13.2).

1.10.8. If $\mathcal{D}'$ contracts $\mathcal{D}$ and the clash at $\psi(N)$ is an M-clash then the clash at N is an M-clash as well (remark proceding Theorem 2.4.1).

1.10.9. If N is inferior to T', $\mathcal{C}' = c'\ (s^{-1}(N)\ )$, $\mathcal{C} = c\ (s^{-1}\ (\ \psi(N)\ )$ and $A' \in \mathcal{C}'$ subsumes $A \in \mathcal{C}$ then A' is α- restricted if A is (remark proceding Theorem 2.6.1).

1.11 Deletion of Subsumed Clauses.

Strategies for deleting variants and subsumed clauses would seem to be promising first candidates for establishing rigorous proofs of efficiency in theorem-proving. Our attempts to obtain such results have uncovered unexpected problems not only for efficiency but for completeness as well. In particular our proof for the completeness of subsumption in [17] is not to the point, while Sibert's proof [48] applies only to a very inefficient version of depth saturation search. In fact counterexample (1) below shows that a certain strategy for deleting variants is incomplete for P_1 - resolution.

In the counterexample below we make use of the following simple depth saturation strategy Σ which is defined only for binary resolution rules $\mathcal{R}$. Σ is defined by specifying the trace $\Upsilon = (C_1, \ldots, C_n, \ldots)$ for an initial set of clauses S_0 of the proof procedure determined by Σ and a given binary resolution rule $\mathcal{R}$:

(1) Let $C_1, \ldots, C_m$ be distinct clauses in S_0 where $S_0 = \{C_1, \ldots, C_m\}$.

(2) Let $p_0 = 1$ and $q_0 = 2$. Suppose that p_i and q_i are defined but that p_{i+1} and q_{i+1} are not. Suppose that $C_1, \ldots, C_n$ are defined and that C_{n+1} is not. Let $\mathcal{C} = \{C_{pi}, C_{qi}\}$. If $\mathcal{R}^1(\mathcal{C}) \neq \emptyset$ then let $C_{n+1}, \ldots, C_{n+k}$ be distinct clauses in $\mathcal{R}^1(\mathcal{C})$ where $\mathcal{R}^1(\mathcal{C}) = \{C_{n+1}, \ldots, C_{n+k}\}$.

(a) If $p_i + 1 = q_i$ then let $p_{i+1} = 1$ and $q_{i+1} = q_i + 1$.

(b) Otherwise, let $p_{i+1} = p_i + 1$ and $q_{i+1} = q_i$.

<u>Examples.</u> The following examples are used in establishing counterexamples 1-3 below.

(1) Let the initial set of clauses S_0 be $\{C_1, \ldots, C_4\}$ where $C_1 = \{P(a,b)\}$, $C_2 = \{\bar{P}(x,y), P(f(x), y)\}$, $C_3 = \{\bar{P}(x,y), Q(y)\}$ and $C_4 = \{\bar{Q}(b)\}$. Let $\mathcal{R}$ be P_1 - resolution. Let $\Upsilon = (C_1, \ldots, C_n, \ldots)$ be the trace for S_0 of $\mathcal{R}$ and Σ. Then

$C_5 = \{P(f(a), b)\}$ is the resolvent of $\{C_1, C_2\}$,

$C_6 = \{Q(b)\}$ of $\{C_1, C_3\}$,

$C_7 = \{P(f(f(a)), b)\}$ of $\{C_2, C_5\}$,

$C_8 = \{Q(b)\}$ of $\{C_3, C_5\}$ and

$C_9 = \Box$ of $\{C_4, C_6\}$.

It is easy to verify that for all $n \geq 2$,

$$C_{3n+1} = \{P(f^n(a), b)\} ,$$
$$C_{3n+2} = \{Q(b)\} \text{ and}$$
$$C_{3n+3} = \Box$$

(2) Let S_0 be $\{C_1, \ldots, C_5\}$ where $C_1 = \{G(y), P(y)\}$, $C_2 = \{\bar{G}(f(x))\}$, $C_3 = \{P(f(a))\}$, $C_4 = \{\bar{P}(f(b)), P(a)\}$ and $C_5 = \{\bar{P}(f(a))\}$. Let $\mathcal{R}$ be binary resolution and let $\mathcal{T} = (C_1, \ldots, C_n, \ldots)$ be the trace for S_0 of $\mathcal{R}$ and Σ. Then

$C_6 = \{P(f(x))\}$ is the resolvent of $\{C_1, C_4\}$,

$C_7 = \{G(f(b)), P(a)\}$ of $\{C_1, C_4\}$,

$C_8 = \{G(f(a))\}$ of $\{C_1, C_5\}$,

$C_9 = \Box$ of $\{C_3, C_5\}$,

$C_{10} = \{P(a)\}$ of $\{C_4, C_6\}$,

$C_{11} = \Box$ of $\{C_5, C_6\}$,

$C_{12} = \{P(a)\}$ of $\{C_2, C_7\}$ and

$C_{13} = \Box$ of $\{C_1, C_8\}$.

C_n is undefined for $n \geq 14$.

Subsumption is <u>admissible</u> for a resolution rule $\mathcal{R}$ if whenever $\mathcal{C}$ is a clash with resolvent $C \in \mathcal{R}^1(\mathcal{C})$, $\mathcal{C}'$ a clash with resolvent C' and $\mathcal{C}'$ covers $\mathcal{C}$ then $C \in \mathcal{R}^1(\mathcal{C}')$. In particular subsumption is admissible for $\mathcal{R}$ if $\mathcal{R}$ is preserved under contractions (i.e. if whenever $\mathcal{D}'$ contracts $\mathcal{D}$ and every application of resolution in $\mathcal{D}$ is an application of $\mathcal{R}$ then every application of resolution in $\mathcal{D}'$ is an application of $\mathcal{R}$).

Theorem 1.11.1 states that if subsumption is admissible for $\mathcal{R}$ then simple deletion of subsumed clauses (defined below) is complete relative to $\mathcal{P}$ and to any search strategy Σ for $\mathcal{R}$.

Let $\mathcal{T}_1$ be the trace for a set of clauses S_0 of a proof procedure $\mathcal{P}$. We define the trace $\mathcal{T}_2$ for S_0 of $\mathcal{P}$ with a given deletion rule by specifying which clauses in $\mathcal{T}_1$ are generated in $\mathcal{T}_2$ and which of these clauses in $\mathcal{T}_2$ are deleted in $\mathcal{T}_2$. The order of clauses in $\mathcal{T}_2$ is the order inherited from $\mathcal{T}_1$. Thus if the n-th clause C_n in $\mathcal{T}_1$ is generated in $\mathcal{T}_2$ then C_n is the n'-th clause in $\mathcal{T}_2$ where $n' \leq n$ and $n-n'$ is the number of clauses generated in $\mathcal{T}_1$ before C_n but not generated in $\mathcal{T}_2$ before C_n.

Let $\mathcal{T}_1 = (C_1, \ldots, C_n, \ldots)$ be the trace for a set of clauses S_0 of a proof procedure $\mathcal{P}$. The corresponding trace $\mathcal{T}_2$ of $\mathcal{P}$ with <u>simple deletion of</u> subsumed clauses is defined inductively :

(1) If C_n in $\mathcal{T}_1$ is in S_0 then C_n is generated in $\mathcal{T}_2$.

(2) If C_n in $\mathcal{T}_1$ is the resolvent of a clash $\mathcal{C} = \{C_{n_1}, \ldots, C_{n_m}\}$, $n_i < n$, then C_n is generated in $\mathcal{T}_2$ if and only if each C_{n_i} is generated and not yet deleted in $\mathcal{T}_2$.

(3) If C_n is generated in $\mathcal{T}_2$ then

(a) if C_n is subsumed by some C_i, $i < n$, generated and yet undeleted in $\mathcal{T}_2$ then C_n is deleted,

(b) if C_n properly subsumes some C_i, $i < n$, generated in $\mathcal{T}_2$ then C_i is deleted. (C properly subsumes D if C subsumes D but D does not subsume C).

Counterexample 1 provides an example of a well-defined strategy for deleting subsumed clauses. This strategy is incomplete as is the strategy which is derived from it by limitation to the deletion of variants.

Counterexample (1). Let deletion of subsumed clauses be defined by replacing conditions (3a) and (3b) in the definition of simple deletion by (3a') and (3b') below.

(3a') If C_n is properly subsumed by some $C_i, i<n$, generated and yet undeleted in $\mathcal{T}_2$ then C_n is deleted.

(3b') If C_n subsumes some C_i, $i<n$, generated in $\mathcal{T}_2$ then C_i is deleted.

That this deletion strategy is not complete can be verified by taking the S_0, $\mathfrak{R}$ and Σ of example (1). (Note that subsumption is admissible for $\mathfrak{R}$.) S_0 is unsatisfiable and $\square$ is the 9-th clause generated in $\mathcal{T}_1$. Applying the deletion rule defined above we obtain the trace $\mathcal{T}_2 = (C'_1,\ldots, C'_n,\ldots)$.

For $n\leq 8$, $C'_n = C_n$ and C'_6 is deleted when C'_8 is generated.

For $n\geq 8$, when n is even, $C'_n = \{Q(b)\}$ and C'_{n-2} is deleted when C'_n is generated.

For $n\geq 9$, when n is odd, $C'_n = \{P(f^{\frac{n+3}{2}}(a),b)\}$ and is never deleted.

Thus no C'_n in $\mathcal{T}_2$ is the null clause.

<u>Theorem 1.11.1.</u> Let subsumption be admissible for a given resolution rule $\mathcal{R}$. Then subsumption is complete relative to $\mathcal{R}$ and any search strategy Σ for $\mathcal{R}$.

<u>Proof.</u> Let $\mathcal{T}_1$ be the trace for a set of clauses S_0 of $\mathcal{R}$ and Σ. Let $\mathcal{T}_2$ be the corresponding trace with simple deletion of subsumed clauses. It suffices to show that for every C_n in $\mathcal{T}_1$ there is a clause $C_{n'}$ generated in $\mathcal{T}_2$ which is never deleted in $\mathcal{T}_2$ and such that $C_{n'}$ subsumes C_n. ($C_{n'}$ is never deleted in $\mathcal{T}_2$ if $C_{n'}$ is not deleted after the generation of C_m in $\mathcal{T}_2$ for all $m \geq n'$.

We observe that there exists no infinite sequence of clauses $C_{n_1}, \ldots, C_{n_i}, \ldots$ such that $C_{n_{i+1}}$ properly subsumes C_{n_i}. From this observation it follows that for every clause C_m generated in $\mathcal{T}_2$ there exists a clause $C_{m'}$ generated and never deleted in $\mathcal{T}_2$ which subsumes C_m. The proof now procedes by induction on n, the index in $\mathcal{T}_1$ of the clause C_n. If $n = 1$ then C_1 is generated in $\mathcal{T}_2$ and is subsumed by some $C_{1'}$ generated and never deleted in $\mathcal{T}_2$.

Suppose that $n \geq 1$ and that every C_i, $i < n$, is subsumed by some $C_{i'}$ generated and never deleted in $\mathcal{T}_2$. If C_n is not a resolvent then C_n is generated in $\mathcal{T}_2$ and is subsumed by some $C_{n'}$ generated and never deleted in $\mathcal{T}_2$. If C_n is the resolvent of $\mathcal{C} = \{C_{n_1}, \ldots, C_{n_m}\}$, $n_i < n$, let $\mathcal{C}' = \{C_{n'_1}, \ldots, C_{n'_m}\}$ where each $C_{n'_i}$ subsumes C_{n_i} and is generated but never deleted in $\mathcal{T}_2$. Then $\mathcal{C}'$ subsumes $\mathcal{C}$. By the contraction lemma either some $C_{n'}$ subsumes C_n or some $\mathcal{C}'' \subseteq \mathcal{C}'$ covers $\mathcal{C}$. In the first case we are through. In the second case, by the admissibility of subsumption and the completeness of the trace $\mathcal{T}_2$, the resolvent of $\mathcal{C}''$ is generated in $\mathcal{T}_2$ and is some $C_{n'}$. $C_{n'}$ subsumes C_n and some never deleted $C_{n''}$ generated in $\mathcal{T}_2$ subsumes $C_{n'}$ and therefore subsumes C_n.

As can be seen by examining the proof of theorem 1.11.1 simple deletion of subsumed clauses need not be efficient, even ignoring the effort involved in applying the deletion rule itself. Counterexample (2) shows how this deletion rule can hurt efficiency by delaying the generation of the first null clause.

<u>Counterexample (2).</u> Take the S_0, $\mathcal{R}$ and Σ of example (2). Then the trace $\mathcal{T}_1$ for S_0 of $\mathcal{R}$ and Σ is the trace $\mathcal{T}$ of example (2). S_0 is unsatisfiable and the first instance of $\square$ in $\mathcal{T}_1$ is C_9. If $\mathcal{T}_2 = (C_1', \ldots, C'_n, \ldots)$ is the trace for S_0 of $\mathcal{R}$ and Σ with simple deletion of subsumed clauses, then the first instance of $\square$ in $\mathcal{T}_2$ is C'_{10}. More particularly:

For $n \leq 8$, $C'_n = C_n$ and C'_3 is deleted when

C'_8 is generated.

C_9 is not generated in $\mathcal{T}_2$ since C'_3 has been deleted

and therefore $\{C'_3, C'_5\}$ is not resolved in $\mathcal{T}_2$.

$C'_9 = \{P(a)\}$, the resolvent of $\{C'_4, C'_6\}$ and

C'_4 and C'_7 are deleted when C'_9 is generated.

C'_{10} is the resolvent of $\{C'_5, C'_6\}$.

Counterexample (2) suggests that it might be possible to remedy the inefficiency of simple deletion by replacing deleted clauses by the clauses which subsume them. In other words if the search algorithm would generate the resolvent C of the clash $\mathcal{C} = \{C_{n_1}, \dots, C_{n_m}\}$ but certain C_{n_i} are deleted and subsumed by undeleted $C_{n'_i}$, then examine the set $\mathcal{C}' = \{C_{n'_1}, \dots, C_{n'_m}\}$ and if some $\mathcal{C}'' \subseteq \mathcal{C}'$ is a clash then generate its resolvent C' in place of C. Admittedly this procedure is quite difficult to define precisely for arbitrary search strategies. But for the case of simple depth saturation there is no problem. However counterexample (3) shows that even in this case efficiency cannot be guaranteed since the replacement procedure may lead to the premature generation of resolvents.

Counterexample (3). Let Σ be simple depth saturation. Then Σ' (Σ with the strategy of replacing subsumed clauses) is defined by (1) and (2) in the definition of Σ and by (3) below.

(3) Suppose that C_n has just been generated.

(a) If C_n is subsumed by some undeleted C_i, $i < n$, then delete C_n.

(b) If C_n properly subsumes some undeleted C_i, $i < n$, then replace C_i by C_n (i.e. let C_i assume the new value C_n).

It is easy to verify that Σ' is complete with resolution rules $\mathcal{R}$ which admit subsumption. The reader will note that redundancies are introduced by condition (3b) since a clause C_n may now occur in several positions C_i and therefore the resolvent of the same clash may be generated more than once. These redundancies can be eliminated without losing completeness by modifying (3b). However even with such a modification the counterexample below continues to hold since no such redundancies are actually introduced in this example by applying Σ' unmodified.

Let S_0 and $\mathcal{R}$ be the S_0 and $\mathcal{R}$ of example (2) and let $\mathcal{T}_2 = (C_1', \ldots, C'_n, \ldots)$ be the trace for S_0 of $\mathcal{R}$ and Σ'. Then C'_{10} is the first instance of $\Box$ in $\mathcal{T}_2$ whereas C_9 is the first instance of $\Box$ in $\mathcal{T}_1$.

For $n \leq 7$, $C'_n = C_n$ but C_3 assumes the new value $C_6 = \{P(f(x))\}$ when C_6 is generated.

$C'_8 = \{P(a)\}$, the resolvent of $\{C_3, C_4\}$, which was not a clash in $\mathcal{T}_1$. C'_4 and C'_7 assume the new value C'_8.

$C'_9 = C_8$, the resolvent of $\{C'_1, C'_5\}$.

$C'_{10} = \Box$, the resolvent of $\{C'_3, C'_5\}$.

Suppose that subsumption is admissible for a resolution rule $\mathcal{R}$. Let Σ be a search strategy for $\mathcal{R}$ and let $\mathcal{T} = (C_1, \ldots, C_n, \ldots)$ be the trace for a set of clauses S_0 of $\mathcal{R}$ and Σ. We say that subsumption is monotonic in $\mathcal{T}$ if whenever a clause C_n in $\mathcal{T}$ is the resolvent of a clash $\mathcal{C} = \{C_{n_1}, \ldots, C_{n_m}\}$ and whenever $\mathcal{C}' = \{C_{n'_1}, \ldots C_{n'_{m'}}\}$ covers $\mathcal{C}$ where $C_{n'_i}$ subsumes C_{n_i} and $n'_i \leq n_i$ then if $C_{n'}$ is the resolvent of $\mathcal{C}'$ then $n' \leq n$. If Σ is simple depth saturation and if subsumption is admissible for a binary resolution rule $\mathcal{R}$, then subsumption is monotonic in any trace Υ of $\mathcal{R}$ and Σ. If $\mathcal{R}$ admits subsumption and Σ is a diagonal search strategy for $\mathcal{R}$, then subsumption is monotonic in any trace Υ of $\mathcal{R}$ and Σ with a possible exception for the case of clashes $\mathcal{C}$ and $\mathcal{C}''$ (as above) where a clause $C_{n'_i}$ subsuming C_{n_i} contains more literals than C_{n_i}.

Counterexamples (2) and (3) show that monotonicity of subsumption does not guarantee efficiency either for simple deletion or replacement deletion of subsumed clauses. Theorem 1.11.2 implies that monotonicity of subsumption is a sufficient condition for the efficiency of deleting newly generated subsumed clauses. This strategy includes as special case the ordinary strategy for deleting variants.

Let $\Upsilon_1 = (C_1, \dots, C_n, \dots)$ be the trace for a set of clauses S_0 of a proof procedure $\mathcal{P}$. The trace Υ_2 of $\mathcal{P}$ with the <u>deletion of newly generated subsumed clauses</u> is defined inductively:

(1) If C_n in Υ_1 is in S_0 then C_n is generated in Υ_2.

(2) If C_n in Υ_1 is a resolvent of the clash $\mathcal{C} = \{C_{n_1}, \dots, C_{n_m}\}$, $n_i < n$, then C_n is generated in Υ_2 if and only if each C_{n_i} is generated and undeleted in Υ_2.

(3) If C_n is generated in Υ_2 then C_n is deleted if and only if C_n is subsumed by some C_i, $i < n$.

<u>Theorem 1.11.2.</u> Given a proof procedure $\wp$ and an unsatisfiable set of clauses S_0, let $\mathcal{T}_1$ be the trace for S_0 of $\wp$ and let $\mathcal{T}_2$ be the trace for S_0 of $\wp$ with deletion of newly generated subsumed clauses. If subsumption is monotonic in $\mathcal{T}_1$ and if C_n is the first instance of $\square$ in $\mathcal{T}_1$, then some $C'_{n'} = \square$ in $\mathcal{T}_2$ and $n' \leq n$.

<u>Proof.</u> We show by induction that for all $n \geq 1$ there is an $n' \leq n$ such that $C'_{n'}$ in $\mathcal{T}_2$ is undeleted and subsumes C_n in $\mathcal{T}_1$.

If $n = 1$ then $C_1 \in S_0$ and $C_1' \in S_0$ are identical, C_1' is undeleted and subsumes C_1. Suppose that for a given $n > 1$ each C_i, $i < n$, is subsumed by an undeleted $C'_{i'}, i' \leq i$. If $C_n \in S_0$ then C_n is generated in $\mathcal{T}_2$ and is some $C'_{n'}$ in $\mathcal{T}_2$ where $n' \leq n$. If $C'_{n'}$ is deleted then some undeleted C'_i for $i < n'$ subsumes $C'_{n'}$. But then C'_i subsumes C_n and $i < n$. If C_n is the resolvent of $\mathcal{C} = \{C_{n_1}, \ldots, C_{n_m}\}$, $n_i < n$, then $\mathcal{C}' = \{C'_{n'_1}, \ldots, C'_{n'_m}\}$ subsumes $\mathcal{C}$ where $C'_{n'_i}$ is undeleted and subsumes C_{n_i} and $n'_i \leq n_i$. But then by the contraction lemma either some $C'_{n'_i}$ subsumes C_n or some $\mathcal{C}'' \subseteq \mathcal{C}'$ covers $\mathcal{C}$. In the second case the resolvent $C'_{n'}$ of $\mathcal{C}''$ subsumes C_n and $n' \leq n$. If $C'_{n'}$ is deleted in $\mathcal{T}_2$ then some C'_i subsumes $C'_{n'}$ and C_n where $i < n' \leq n$.

Theorems 1.11.1 and 1.11.2 and counterexamples (1)-(3) do not constitute a thorough analysis of deletion rules for subsumed clauses. A more satisfactory analysis would probably involve comparing the number of clauses omitted by the deletion rules with the number of new clauses introduced before the first rules with the number of new clauses introduced before the first instance of $\square$. It is quite possible for deletion to delay the generation of $\square$ and yet compensate by omitting the generation of more clauses than are introduced by this delay. It might be hoped that such an approach would also be applicable to other more difficult problems of efficiency in theorem-proving.

1.12 Deletion of Tautologies.

If $\mathcal{T}_1 = (C_1, \ldots, C_n, \ldots)$ is the trace for S_0 of a proof procedure $\mathcal{P}$ then the ordinary rule for deleting tautologies can be defined by specifying which clauses C_n in $\mathcal{T}_1$ are generated and which of these are deleted in the corresponding trace $\mathcal{T}_2$ of $\mathcal{P}$ with deletion of tautologies.

(1) If C_n in $\mathcal{T}_1$ is in S_0 then C_n is generated in $\mathcal{T}_2$.

(2) If C_n in $\mathcal{T}_1$ is the resolvent of $\mathcal{C} = \{C_{n_1}, \ldots, C_{n_m}\}$, $n_i < n$, then C_n is generated in $\mathcal{T}_2$ if and only if each C_{n_i} is generated and undeleted in $\mathcal{T}_2$.

(3) If C_n is generated in $\mathcal{T}_2$ then C_n is deleted in $\mathcal{T}_2$ if and only if C_n is a tautology.

Theorem 1.12.2 implies that if $\mathcal{R}$ is any resolution rule preserved under semi-contractions then deletion of tautologies is compatible with $\mathcal{R}$ and any search strategy Σ for $\mathcal{R}$. Equivalently deletion of tautologies is compatible with $\mathcal{R}$ and Σ if whenever $\mathcal{C}$ is a clash with resolvent $C \in \mathcal{R}^1(\mathcal{C})$, $\mathcal{C}'$ is a clash with resolvent C' and $\mathcal{C}'$ weakly covers $\mathcal{C}$, then

$C' \in \mathcal{R}^1(\mathcal{C}')$.

Suppose that a resolution rule $\mathcal{R}$ is preserved under semi-contractions. Let $\mathcal{R}$ be the proof procedure determined by $\mathcal{R}$ and a search strategy Σ for $\mathcal{R}$ and let $\mathcal{T}=(C_1,\dots,C_n,\dots)$ be a trace of $\mathcal{P}$. Then weak covers are <u>monotonic</u> in $\mathcal{T}$ if whenever a clause C_n in $\mathcal{T}$ is the resolvent of a clash $\mathcal{C} = \{C_{n_1},\dots,C_{n_m}\}$ and whenever $\mathcal{C}' = \{C_{n'_1},\dots,C_{n'_{m'}}\}$ weakly covers $\mathcal{C}$ where $C_{n'_i}$ subsumes C_{n_i} and $n'_i \leq n_i$ then if $C_{n'}$ is the resolvent of $\mathcal{C}'$ then $n' \leq n$. Theorem 1.12.3 implies that monotonicity of weak covers is a sufficient condition for the efficiency of deleting tautologies.

<u>Lemma 1.12.1.</u> Let $\mathcal{C}$ be a clash with non-tautologous resolvent C and let $D \in \mathcal{C}$ be a tautology. Then either

(a) some $C' \in \mathcal{C}$ subsumes C or

(b) some subset $\mathcal{C}' \subset \mathcal{C}$, $D \notin \mathcal{C}'$, is a clash with resolvent C' subsuming C and $\mathcal{C}'$ weakly covers $\mathcal{C}$.

<u>Proof.</u> Let $D = \{L, \bar{L}\} \mathbin{\mathring{\cup}} D_0$. Let $\mathcal{C}=\{A_1,\dots,A_n,B\}$ where $A_i = E_i \mathbin{\mathring{\cup}} A_{0i}$, $B = F_1 \mathbin{\mathring{\cup}} \dots \mathbin{\mathring{\cup}} F_n \mathbin{\mathring{\cup}} B_0$, $E_i \neq \emptyset$, $F_i \neq \emptyset$ and $C = (A_{01} \cup \dots \cup A_{0n} \cup B_0)\,\theta$ where θ is an m.g.s.u. of $\mathcal{E} = \{E_1 \cup \bar{F}_1,\dots, E_n \cup \bar{F}_n\}$. Since C is not a tautology at least one of the literals L or $\bar{L}$ is resolved upon in D. There are three cases to consider.

<u>Case (a).</u> D is the nucleus B of $\mathcal{C}$ and only one of L or $\bar{L}$ is resolved upon in D. We may assume that L is resolved upon and that $L \in F_1$. Then A_1 subsumes C. For since

$E_1\theta = \bar{F}_1\theta = \{\bar{L}\theta\}$ and since $\bar{L} \in B_0$, $\bar{L}\theta \in B_0\theta$ and $E_1\theta \subseteq B_0\theta$. So $A_1\theta \subseteq B_0\theta \cup A_{01}\theta \subseteq C$.

<u>Case (b1).</u> D is the nucleus B of $\mathcal{C}$ and both L and $\bar{L}$ are resolved upon in D. We may assume that $L \in F_1$ and $\bar{L} \in F_2$. Let $\mathcal{C}' = \{A_1, A_2\}$ and $\mathcal{E}' = \{E_1 \cup \bar{E}_2\}$. θ unifies $\mathcal{E}'$ because

$$E_1\theta = \bar{F}_1\theta = \{\bar{L}\theta\} \quad \text{and}$$
$$\bar{E}_2\theta = F_2\theta = \{\bar{L}\theta\}.$$

Let θ' be an m.g.s.u. of $\mathcal{E}'$ and let $\theta = \theta'\theta''$. The resolvent of $\mathcal{C}'$ is

$$C' = (A_{01} \cup A_{02})\,\theta' \text{ and}$$
$$C'\theta'' = (A_{01} \cup A_{02})\,\theta \subseteq C.$$

So C' subsumes C. Suppose $\mathcal{C}$ is restricted and $\mathcal{C}'$ is not. Then either $L\theta'$ or $\bar{L}\theta'$ is in C'. But then $L\theta$ or $\bar{L}\theta$ is in $C'\theta'' \subseteq C$ and $\mathcal{C}$ is not restricted.

<u>Case (b2).</u> D is a satellite of $\mathcal{C}$. Suppose that L is resolved upon in D and that D is A_1. Then $L \in E_1$. Let $\mathcal{C}' = \mathcal{C} - \{D\}$ and $\mathcal{E}' = \mathcal{E} - \{E_1 \cup \bar{F}_1\}$. Then θ unifies $\mathcal{E}'$. Let θ' be an m.g.s.u. of $\mathcal{E}'$ and let $\theta = \theta'\theta''$. The resolvent of $\mathcal{C}'$ is

$$C' = (F_1 \cup A_{02} \cup \dots \cup A_{0n} \cup B_0)\,\theta'.$$

C' subsumes C since

$$E_1\theta = \bar{F}_1\theta = \{L\theta\} \text{ and } F_1\theta = \{\bar{L}\theta\} \subseteq A_{01}\theta, \text{ so}$$
$$C'\theta'' = F_1\theta \cup (A_{02} \cup \dots \cup A_{0n} \cup B_0)\theta$$
$$\subseteq A_{01}\theta \cup (A_{02} \cup \dots \cup A_{0n} \cup B_0)\theta = C.$$

$\mathcal{C}$ is not restricted since $\bar{L}\theta \in F_1\theta$ and $\bar{L}\theta \in C$.

Theorem 1.12.2. Given a derivation $\mathfrak{D} = (T,c)$ from S of a non-tautology $c(r(T))$ there exists a derivation $\mathfrak{D}' = (T', c')$ from S of a clause which subsumes $c(r(T))$. $\mathfrak{D}'$ is a semi-contraction of $\mathfrak{D}$ and $\mathfrak{D}'$ contains no tautologies.

Proof (by induction on the number n of tautologies in $\mathfrak{D}$). If $n = 0$ then take $\mathfrak{D}' = \mathfrak{D}$. Otherwise $n > 0$ and we assume that the theorem holds for any derivation containing fewer than n tautologies. Let $\mathcal{C}$ be a clash in $\mathfrak{D}$ containing at least one tautology D, i.e. $\mathcal{C} = c(s^{-1}(N_0))$ for some $N_0 \in T$ and $D = c(N')$ for some $N' \in s^{-1}(N_0)$. Choose N_0 such that $c(N_0)$ is not a tautology. By the preceding lemma either

case(1) some $C' = c(N_j) \in \mathcal{C}$ subsumes $c(N_0)$, or

case (2) some $\mathcal{C}' \subset \mathcal{C}$, $D \notin \mathcal{C}'$, is a clash, and $\mathcal{C}'$ weakly covers $\mathcal{C}$.

In either case let T_0 be the subtree of T obtained by ignoring all of T lying above N_0 (i.e. $T_0 = (T - T_{N_0}) \dot{\cup} \{N_0\}$) and let $\mathfrak{D}_0 = (T_0, c)$. Associate with every tip $N \in T_0$ a clause A_N which subsumes $c(N)$: $A_{N_0} = C'$ and $A_N = c(N)$ for $N \neq N_0$. By the contraction theorem we obtain a contraction $\mathfrak{D}_0' = (T_0', c_0')$ of $\mathfrak{D}_0$. Let $N_0' \in T_0'$ be the tip corresponding to N_0 (i.e. $N_0' = \Psi^{-1}(N_0)$ where Ψ is the mapping associated with the contraction $\mathfrak{D}_0'$ of $\mathfrak{D}_0$). Then $c_0'(N_0') = A_{N_0} = C'$.
In case (1) let $\mathfrak{D}'' = (T'', c'')$ be obtained by identifying N_0' in T_0' with N_j in T_{N_j} ($c'_0(N_0') = c(N_j) = C'$). In case (2) let $\mathfrak{D}''$ be obtained by grafting the derivation trees T_N

to the node N_0' in T_0' where $N \in s^{-1}(N_0)$ and $c(N) \in \mathcal{C}'$. More precisely let

$$T'' = T_0' \cup \{N : N \in T_{N_i} \text{ and } c(N_i) \in \mathcal{C}'\}$$
$$c''(N) = c(N) \text{ for } N \in T_{N_i}$$
$$c''(N) = c_0'(N) \text{ for } N \in T'_0$$
$$s^{-1}(N_0') = \{N_i : c(N_i) \in \mathcal{C}'\} .$$

In both cases, we obtain a semi-contraction $\mathfrak{D}''$ of $\mathfrak{D}$. $\mathfrak{D}''$ is a derivation from S and $\mathfrak{D}''$ contains fewer than n tautologies (by 1.10.3). By the induction hypothesis there exists a semi-contraction $\mathfrak{D}'$ of $\mathfrak{D}''$ such that $\mathfrak{D}'$ contains no tautologies and is a derivation from S of a clause subsuming $c(r(T))$. By the transitivity of semi-contractions $\mathfrak{D}'$ is the desired semi-contraction of $\mathfrak{D}$.

Theorem 1.12.3. Given a proof procedure $\mathcal{P}$ and an unsatisfiable set of clauses S_0, let $\mathcal{T}_1$ be the trace for S_0 of $\mathcal{P}$ and let $\mathcal{T}_2$ be the trace for S_0 of $\mathcal{P}$ with deletion of tautologies. If weak covers are monotonic in $\mathcal{T}_1$ and if C_n is the first instance of $\Box$ in $\mathcal{T}_1$, then some $C'_{n'} = \Box$ in $\mathcal{T}_2$ and $n' \leq n$.

Proof. The proof is similar to that of 1.11.2. We show that for all $n \geq 1$ there is an $n' \leq n$ such that if C_n in $\mathcal{T}_1$ is not a tautology then $C'_{n'}$ undeleted in $\mathcal{T}_2$ subsumes C_n.

If $n = 1$ then $C_1 \in S_0$ and $C'_1 \in S_0$ are identical. C_1' subsumes C_1 and is undeleted in $\mathcal{T}_2$ if C_1 is not a tautology. Suppose that for a given $n > 1$ each non-tautologous C_i, $i \leq n$, is subsumed by $C'_{i'}$, $i' \leq i$, undeleted in $\mathcal{T}_2$. If $C_n \in S_0$ then

C_n is generated in $\mathcal{T}_2$ and is some $C'_{n'}$ in $\mathcal{T}_2$ where $n' \leq n$. $C'_{n'}$ subsumes C_n and is undeleted if C_n is not a tautology. If C_n is not a tautology and is the resolvent of $\mathcal{C} = \{C_{n_1}, \ldots, C_{n_m}\}$, $n_i < n$, then either some non-tautologous C_{n_i} subsumes C_n or some $\mathcal{C}' \subseteq \mathcal{C}$ weakly covers $\mathcal{C}$ and $\mathcal{C}'$ contains no tautologies. In the first case some undeleted $C'_{n'_i}$ subsumes C_{n_i} and also C_n where $n'_i \leq n_i < n$. In the second case $\mathcal{C}'' = \{C'_{n'_1}, \ldots, C'_{n'_{m'}}\}$, where $C'_{n'_i}$ subsumes C_{n_i} and $n'_i \leq n_i$, subsumes $\mathcal{C}'$ and each $C'_{n'_i} \in \mathcal{C}''$ is undeleted in $\mathcal{T}_2$. If $C'_{n'}$ is the resolvent of $\mathcal{C}'$ then $n' \leq n$ and either some $C'_{n'_i}$ subsumes $C'_{n'}$ and C_n where $n'_i < n' \leq n$ or some subset $\mathcal{C}'''$ of $\mathcal{C}''$ covers $\mathcal{C}''$ and therefore weakly covers $\mathcal{C}$ and therefore the resolvent of $\mathcal{C}'''$ undeleted in $\mathcal{T}_2$ is some $C'_{n''}$ in $\mathcal{T}_2$ where $n'' \leq n$ since each $n'_i \leq n_i$ for $C'_{n_i} \in \mathcal{C}'''$.

1.13 Minimal Derivations.

In sections 1.9-1.12 we adopted the convention of calling an arbitrary set of clauses $\mathcal{C}$ a clash if some standardised set $\mathcal{C}'$ of variants of clauses in $\mathcal{C}$ is a clash. In this section it is convenient to revert to the more restrictive definition of clash, reserving the term for standardised sets of clauses. We introduce the notion of a ground clash $\mathcal{C}$ which is like a clash of ground clauses except that in this case we allow that $\mathcal{C}$ contains variables and is not standardised. More precisely, $\mathcal{C}$ is a <u>ground clash</u> if $\mathcal{C}$ is of the form $\{A_1, \ldots, A_n, B\}$ where

$A_1 = \{L_1\} \dot{\cup} A_{01}, \ldots, A_n = \{L_n\} \dot{\cup} A_{0n}$

$B = \{\bar{L}_1, \ldots, \bar{L}_n\} \dot{\cup} B_0.$

The resolvent of $\mathcal{C}$ is $C = A_{01} \cup \ldots \cup A_{0n} \cup B_0$.

Notice that given an arbitrary clash $\mathcal{C}$ with m.g.s.u. θ and resolvent C, the set of clauses $\mathcal{C}\theta$ is a ground clash with resolvent C provided that for no $A \in \mathcal{C}$ and no $L, L' \in A$, where L is resolved upon in $\mathcal{C}$ and L' is not, does $L\theta = L'\theta$. Thus in particular $\mathcal{C}\theta$ is a ground clash if $\mathcal{C}$ is restricted. A derivation $\mathcal{D} = (T,c)$ is a ground derivation if, for every interior $N \in T$, $c(s^{-1}(N))$ is a ground clash with resolvent c(N). Thus every derivation from a set of ground clauses is a ground derivation but not conversely.

Given a derivation $\mathcal{D} = (T,c)$, let the pair $\tilde{\mathcal{D}} = (T,\tilde{c})$ be defined by letting

$\tilde{c}(r(T)) = c(r(T))$ and, for $N \neq r(T)$,

$\tilde{c}(N) = c(N)\,\theta_1 \ldots \theta_n$ where θ_i is the m.g.s.u. of the clash at $s^i(N)$ and where $s^n(N) = r(T), s^{n-1}(T) \neq r(T)$.

If $\tilde{\mathcal{D}}$ is a derivation then it is a ground derivation lifted by $\mathcal{D}$. However $\tilde{\mathcal{D}}$ may not be a derivation even if every clash in $\mathcal{D}$ is restricted (witness Andrews' counterexample [2]). Theorem 1.13.1 implies that a necessary and sufficient condition for $\tilde{\mathcal{D}}$ to be a derivation is that $\mathcal{D}$ contract some ground derivation $\mathcal{D}'$.

A derivation $\mathcal{D} = (T,c)$ is standardised if, for all $N, N' \in T$ such that $T_N \cap T_{N'} = \emptyset$, c(N) and c(N') share no variables. A derivation $\mathcal{D}$ may fail to be standardised even though each clash in $\mathfrak{D}$ is standardised (since literals resolved upon in disjoint subderivations of $\mathfrak{D}$ may contain common variables). It is easy to verify that if $\mathfrak{D} = (T,c)$ is a derivation (but not a ground derivation) then the derivation $\mathfrak{D}' = (T,c')$ obtained by applying the contraction theorem to $\mathfrak{D}$ and the set $S' = \{A_N : A_N$ is a variant of $c(N)$, $N \in T$ a tip$\}$, where S' is standardised, is standardised and equivalent to $\mathfrak{D}$ in the sense that $c'(N)$ is a variant of $c(N)$ for all $N \in T$.

<u>Theorem 1.13.1.</u> If $\mathfrak{D} = (T,c)$ is standardised and contracts (or semi-contracts) a ground derivation $\mathfrak{D}' = (T',c')$ with associated mapping ψ, then $\tilde{\mathfrak{D}}$ is a ground derivation and contracts (semi-contracts) $\mathfrak{D}'$ with mapping ψ. For some λ and for all $N \in T$,

$$\tilde{c}(N)\lambda \subseteq c'(\psi(N)).$$

<u>Proof</u> (by induction on the number n of nodes in T'). We prove the theorem for the case where $\mathfrak{D}$ contracts $\mathfrak{D}'$. The proof is identical when $\mathfrak{D}$ is a semi-contraction of $\mathfrak{D}'$. If n=1 then, for some N_0 and N_0', $T' = \{N_0'\}$, $T = \{N_0\}$ and $\psi(N_0) = N_0'$. $\tilde{\mathfrak{D}} = \mathfrak{D}$ is a ground derivation and since $c(N_0)$ subsumes $c'(N_0')$, $\tilde{c}(N_0)\lambda \subseteq c'(\psi(N_0))$ for some λ.

Assume that $n>1$ and that the theorem holds for any derivation contracting a ground derivation which contains fewer than n nodes. Let $N_0' = r(T')$, $N_0 = r(T)$, $s^{-1}(N_0') = \{N_0', \ldots, N'_{m'}\}$ and $\mathfrak{D}'_i = (T'_{N'_i}, c')$, $1 \leq i \leq m'$. If

$s^{-1}(N_0) \neq \emptyset$, let $s^{-1}(N_0) = \{N_1,\ldots,N_m\}$ and $\mathcal{D}_i = (T_{N_i}, c)$.

Suppose that $\mathcal{D}$ contracts some $\mathcal{D}_i{}'$ with mapping ψ and that $c(N_0)$ subsumes $c'(N_0)$. Since $T'_{N'_1}$ contains fewer than n nodes, $\mathcal{D}$ is a ground derivation, contracts $\mathcal{D}_i{}'$ with mapping ψ and, for all $N \in T$, $\tilde{c}(N)\lambda \subseteq c'(\psi(N))$ for some λ . Since $\tilde{c}(N_0) = c(N_0)$ subsumes $c'(N_0{}')$, $\tilde{\mathcal{D}}$ contracts $\mathcal{D}'$ with mapping ψ .

If $\mathcal{D}$ contracts no $\mathcal{D}_i{}'$ then $s^{-1}(N_0) \neq \emptyset$, $m \leq m'$, $\mathcal{D}_i$ contracts $\mathcal{D}_i^{!}$ with associated mapping ψ_i (ψ_i is the restriction of ψ to T_{N_i}) , $\mathcal{C} = c(s^{-1}(N_0))$ covers $\mathcal{C}' = c'(s^{-1}(N_0{}'))$ and $\psi(N_0) = N_0{}'$. Let θ be the m.g.s.u. of $\mathcal{C}$. By induction hypothesis each $\tilde{\mathcal{D}}_i = (T_{N_i}, \tilde{c}_i)$, where $\tilde{c}_i(N)\theta = \tilde{c}(N)$ for $N \in T_{N_i}$, is a ground derivation which contracts $\mathcal{D}_i{}'$ with mapping ψ_i and, for some λ_i and all $N \in T_{N_i}$, $\tilde{c}_i(N)\lambda_i \subseteq c'(\psi(N))$. Let $\sigma = \lambda_1 \ldots \lambda_m$. Since $\mathcal{C}$ covers $\mathcal{C}'$, $\mathcal{D}$ is standardised and $\mathcal{C}'$ is a ground clash, σ unifies $\mathcal{C}$ and therefore $\sigma = \theta\lambda$ for some λ . But then $\tilde{c}(N)\lambda \subseteq c'(\psi(N))$ for all $N \in T$.

$\tilde{\mathcal{D}}$ is a ground derivation which contracts $\mathcal{D}'$ with associated mapping ψ if for every $N \in T$, $N \neq r(T)$, $\tilde{c}(s^{-1}(N))$ is a clash which covers $c'(s^{-1}(\psi(N)))$. But in general whenever a clash $\mathcal{C}$ covers a clash $\mathcal{C}'$ with associated substitution σ then $\mathcal{C}$ is a clash which covers $\mathcal{C}'$ with associated substitution λ when θ and λ are such that $\sigma = \theta\lambda$. But this property clearly holds for the clashes $\mathcal{C}$ and $\mathcal{C}'$ at N_0 and $\psi(N_0)$ as well as for the clash $\tilde{c}_i(s^{-1}(N))$ and

$c'(s^{-1}(\Psi(N)))$ when $N \in T_i$. Therefore $\tilde{\mathcal{D}}$ is a ground derivation and contracts $\mathcal{D}'$ with mapping Ψ.

A notion similar to that of minimal derivation was introduced by Loveland for the case of binary ground derivations in order to prove the existence of linear refutations containing no tautologies [52]. The existence of various kinds of minimal derivations and refutations is proved in Chapter 2 by using Theorem 1.13.2 below. Implementation of the minimality restriction serves several functions: it provides a method for effectively applying the clash restriction, rejects derivations which do not lift ground derivations and tends to retain only the simpler of equivalent derivations. This last property can be stated precisely for the case of a minimal refutation $\mathcal{D}$ of a set S by saying that the number of distinct nodes in the longest branch of $\mathcal{D}$ is no greater than the minimal number of distinct atoms in any set S' of ground instances of clauses in S. Clearly the retention of only the simpler of equivalent derivations is important for efficiency.

A ground derivation $\mathcal{D} = (T,c)$ is <u>minimal</u> if for no $N \in T$, $N' \in T$ lying above N, $L' \in c(N')$ resolved upon at N' and $L \in c(N)$ does $|L'| = |L|$. An arbitrary derivation $\mathcal{D} = (T,c)$ is <u>minimal</u> if for no $N \in T$, $N' \in T$ lying above N, $L' \in c(N')$ resolved upon at N' and $L \in c(N)$ does $|L'\theta_1 \ldots \theta_n| = |L\theta_{k+1} \ldots \theta_n|$ where θ_i is the m.g.s.u. of the clash at $s^i(N')$, where $N = s^k(N')$, $s^n(N) = r(T)$

and $s^{n-1}(N) \neq r(T)$

It is easy to verify that a derivation $\mathfrak{D}$ is minimal if and only if $\tilde{\mathfrak{D}}$ is.

The following is a simple, if not most efficient, method for implementing the minimality condition:

(1) Associate with every derivation $\mathfrak{D} = (T,c)$ of a clause C the history $\mathfrak{D}^* = (T,a)$ of literals resolved upon in $\tilde{\mathfrak{D}}$, i.e.

(a) if $T = \{N_0\}$ then $a(N_0) = \emptyset$,

(b) if $N_0 = r(T)$, $s^{-1}(N_0) = \{N_1,\dots,N_m\}$, $\mathfrak{D}^*_i = (T_{N_i}, a_i)$ is associated with $\mathfrak{D}_i = (T_{N_i}, c)$, E_i is the set of literals resolved upon at N_i and θ is the m.g.s.u. of the clash at N_0,

then $\mathfrak{D}^* = (T,a)$ where

$a(N_0) = \emptyset$, $a(N_i) = E_i\theta$ and

$a(N) = a_i(N)\theta$ for $N \in T_{N_i} - \{N_i\}$.

(2) Reject, as incompatible with the minimality condition, a clause C obtained by a derivation $\mathfrak{D}$ with associated history $\mathfrak{D}^* = (T,a)$ if either

(a) for some $L \in C$, $N' \in T$ and $L' \in a(N')$,

$$| L | = | L' |$$ or

(b) for some $N \in T$, $N' \in T_{N'}$, $L \in a(N)$ and $L' \in a(N')$

$$| L | = | L' |.$$

Notice that condition (2a) generalises the clash restriction.

Theorem 1.13.2 below allows us to infer that a derivation $\mathcal{D}'$ is minimal if it lifts a minimal ground derivation $\mathcal{D}$.

<u>Theorem 1.13.2.</u> If $\mathcal{D}' = (T', c')$ is a semi-contraction of $\mathcal{D} = (T, c)$ and if $\mathcal{D}$ is minimal then $\mathcal{D}'$ is minimal,

<u>Proof.</u> Let ψ be associated with the semi-contraction $\mathcal{D}'$ of $\mathcal{D}$. $\tilde{\mathcal{D}}$ is a minimal ground derivation. It is easily verified that $\mathcal{D}'$ is a semi-contraction of $\tilde{\mathcal{D}}$ with mapping ψ. By 1.13.1, $\tilde{\mathcal{D}}'$ is a ground derivation, contracts $\tilde{\mathcal{D}}$ with mapping ψ and, for some λ and all $N \in T'$, $\tilde{c}'(N)\lambda \subseteq \tilde{c}(\psi(N))$).

It suffices to show that $\tilde{\mathcal{D}}'$ is minimal. If $\tilde{\mathcal{D}}'$ is not minimal then there exist $N, N' \in T$, N' lying above N, $L' \in \tilde{c}'(N')$ resolved upon at N' in $\tilde{\mathcal{D}}'$ and $L \in \tilde{c}'(N)$ such that $|L| = |L'|$. But then $\psi(N')$ lies above $\psi(N)$ in T, $L'\lambda$ is resolved upon at $\psi(N')$ in $\tilde{\mathcal{D}}$, $L\lambda \in \tilde{c}(\psi(N))$ and $|L\lambda| = |L'\lambda|$, contradicting the minimality of $\tilde{\mathcal{D}}$.

Theorem 1.13.2 ensures the compatibility of deletion of tautologies and of simple deletion of subsumed clauses with proof procedures implementing minimality and a resolution rule $\mathcal{R}$ which is preserved under semi-contractions, in the case of tautologies, and contractions, in the case of subsumed clauses. However it is necessary to modify the rule for simple deletion of subsumed clauses in the following way : Let $\mathcal{T} = (C_1, \dots, C_n, \dots)$ be a trace for a proof procedure implementing minimality and simple deletion of subsumed clauses. Suppose that C_n has just been generated. If C_n properly subsumes some

C_i, $i < n$, then C_i is deleted. Let the history $\mathcal{O}^*_n$, associated with the derivation of C_n, assume the new value (T,a) where $T = \{N_0\}$ and $a(N_0) = \emptyset$. Similarly if some C_i, $i < n$, subsumes C_n then C_n is deleted and the history $\mathcal{O}^*_i$, associated with the derivation of C_i, assumes the new value (T,a) where $T = \{N_0\}$ and $a(N_0) = \emptyset$.

CHAPTER 2.

Chapter 2 is concerned primarily with the application of semantic tree constructions to obtain completeness theorems for resolution inference systems (see [43] and [17]). These applications are limited to the first order logic without equality. With the exception of section 2.5 most of the results of 2.2 - 2.7 were obtained in collaboration with P.J. Hayes and were reported in [17] . Section 2.5 establishes the deduction completeness theorem proved by Slagle, Chang and Lee in [52]. A somewhat weaker theorem was proved independently by the author and was presented in [20]. The completeness theorems of 2.3 - 2.6 improve those reported in [51], [17] and [52] by imposing the minimality restriction on derivations. In section 2.8 we investigate clash-like sequences of binary resolutions (pseudo-clashes) which are then applied in 2.9 to establish the completeness of a modification of P_1-deduction (reported in [17]) which is more efficient than either P_1-deduction or hyper-resolution. Section 2.10 establishes the completeness of maximal pseudo-clash resolution. The analogous theorem fails for maximal clash resolution.

2.1 Herbrand Interpretations.

We recall that the intended interpretation of a clause is the universal closure of the disjunction of its elements. Sets of clauses are interpreted as conjunctions of their elements. We assume acquaintance with the fact that a set of clauses is satisfiable if and only if a corresponding set of clauses is satisfiable. A readable introduction to the necessary preliminaries is Davis' [7] . This section is concerned with establishing the definitions and propositions necessary to reduce the study of the semantics of sets of clauses to the study of Herbrand interpretations.

Given a set of clauses S, the Herbrand universe of S, H(S), is the set of all ground terms constructible from the function letters which occur in S (augmented by a single constant if S contains no constants). The Herbrand base of S, $\hat{H}(S)$, is the set of all ground instances over H(S) of all atoms which occur in clauses of S, i.e.

$$\hat{H}(S) = \{ |L|\sigma : L \in C \in S, \sigma = \{t_1/x_1, \ldots, t_n/x_n\},$$
$$t_i \in H(S) \text{ and } |L|\sigma \text{ is a ground atom} \}.$$

(In the sequel, when a set of clauses S has been fixed and $C\sigma$ is said to be a ground instance of $C \in S$, it will be understood that the terms t_i of σ all belong to H(S). Note that if S is a finite set of ground clauses then $\hat{H}(S)$ is finite although H(S) may be infinite.

If K is a set of ground atoms then a set of literals $\mathcal{a}$ is an assignment to K if

(1) $L \in \mathcal{Q}$ implies $|L| \in K$, and

(2) $L \in \mathcal{Q}$ implies $\bar{L} \notin \mathcal{Q}$.

An assignment $\mathcal{Q}$ to K is <u>complete</u> if

(3) $L \in K$ implies $L \in \mathcal{Q}$ or $\bar{L} \in \mathcal{Q}$.

Given a set of clauses S a complete assignment $\mathcal{Q}$ to $\hat{H}(S)$ is called a <u>Herbrand interpretation of S</u>. Any Herbrand interpretation $\mathcal{Q}$ of S determines an interpretation of S in the usual sense as follows:

(1) H(S) is the universe (domain) of the interpretation.

(2) The denotation f^* of f, a function letter occurring in S is given by: $f^*(t_1,\ldots,t_n) = f(t_1,\ldots t_n), t_i \in \hat{H}(S)$.

(3) The denotation P^* of P, a predicate letter occurring in S is given by: $P^*(t_1,\ldots,t_n)$ if and only if $P(t_1,\ldots,t_n) \in \mathcal{Q}$.

Notice that $P(t_1,\ldots,t_n)$ in (3) above, need not belong to $\hat{H}(S)$. As a result if S is a finite set of ground clauses and H(S) is infinite then the interpretation corresponding to $\mathcal{Q}$ is infinite. It is the interpretation given by (1) - (3) above which we have in mind when we refer to a clause or set of clauses as being satisfied by a Herbrand interpretation.

Given any interpretation M of a set of clauses S we denote by $M \models S$ the relation of M satisfying S. If $S = \{C\}$ then we also write $M \models C$. We let the symbol $\neg$ denote logical negation.

<u>Proposition 2.1.1.</u>/

Proposition 2.1.1. Given a set of clauses S and a Herbrand interpretation M of S

(1) $M \models S$ if and only if $M \cap C\sigma \neq \emptyset$ for all ground instances $C\sigma$ of a clause $C \in S$.

(2) $M \models \neg S$ if and only if $\overline{C\sigma} \subseteq M$ for some ground instance $C\sigma$ of a clause $C \in S$.

Proof. It suffices to prove (1) since (2) is just the contrapositive of (1). Suppose $M \models S$ then $M \models C$ for all $C \in S$. But then $M \models C\sigma$ for all ground instances $C\sigma$ of C (since C is interpreted as universally quantified and the domain of M is H(S)). $M \models C\sigma$ implies that $M \models \{L\}$ for some $L \in C\sigma$ and therefore implies that $M \cap C\sigma = \{L\} \neq \emptyset$.

Conversely if $M \cap C\sigma \neq \emptyset$ for all $C \in S$ and for all ground instances $C\sigma$ of C then $M \models C$ and therefore $M \models S$.

Proposition 2.1.2. Given a set of clauses S and a model M of S (i.e. $M \models S$) there exists a Herbrand model M' of S (i.e. $M' \models S$).

Proof. Note first that if S contains an individual constant then every $t \in H(S)$ denotes some element t^* in the domain of M. If S contains no such constant and b is the constant symbol introduced into H(S) then let b^* be some arbitrary element of the non-empty domain of M. Then in this case as well every $t \in H(S)$ denotes some element t^* in the domain of M.

If $L \in \hat{H}(S)$ then $L = P(t_1, \ldots, t_n)$ for some P occurring in S and $t_1, \ldots, t_n \in H(S)$. But then $L^* = P^*(t_1^*, \ldots, t_n^*)$ is either true or false in M where P* is the predicate in M denoted by P. Let M' be the complete assignment to $\hat{H}(S)$ where for all L $\in$ $\hat{H}(S)$,

L $\in$ M' if and only if L* is true in M,

$\overline{L}$ $\in$ M' if and only if L* is false in M.

Suppose M $\models$ S and M' $\models$ $\neg$ S. Then $\overline{C\sigma} \subseteq$ M' for some C $\in$ S and some ground instance C σ of C. But then $\overline{L}$ $\in$ M' for each L $\in$ Cσ and therefore each such L* is false in M. If C = C($x_1,\ldots,x_n$) and C σ = C($t_1,\ldots,t_n$) then, since C*(t_1*,...,t_n*) is false in M, C is also false in M and M $\models$ $\neg$ S.

<u>Corollary 2.1.3.</u> A set of clauses S is unsatisfiable if and only if S has no Herbrand models.

<u>Proposition 2.1.4.</u> Let S be a set of clauses and S' an unsatisfiable set of instances of clauses in S. Then S is unsatisfiable.

<u>Proof.</u> If S is satisfiable then M $\models$ S for some Herbrand model M of S. But then M $\cap$ C$\sigma \neq \emptyset$ for all C $\in$ S and all ground instances C σ. But each ground instance C'σ' of a clause C' $\in$ S' is a ground instance Cσ of a clause C $\in$ S (where C' = Cθ and $\sigma = \theta\sigma'$). Therefore M $\cap$ C'σ' $\neq \emptyset$ for each ground instance C' σ' of each C' $\in$ S' and therefore M' is a Herbrand model of S' where M' $\subsetneq$ M is the subset of M which is a complete assignment to $\hat{H}(S')$.

2.2 Semantic Trees.

The notion of a semantic tree was introduced by Robinson in [43] to obtain extensions of resolution for first-order logic with equality. The semantic trees studied below are, however, limited to first-order logic without equality. We extend Robinson's original definition of failure and concern ourselves more with establishing specific applications than with extending the general theory. Further research on semantic trees is reported on in Robinson's recent over-view of theorem-proving [46].

Let K be a set of ground atoms, T a tree and $\mathcal{Q}$ a function defined on nodes of T having assignments to K as values. If X is a subset of T let $\mathcal{Q}(X) = \{\mathcal{Q}(N) : N \in X\}$. Then $\mathcal{S} = (T, \mathcal{Q})$ is a <u>semantic tree for K</u> if

(1) $\mathcal{Q}(N_0) = \emptyset$ for $N_0 = r(T)$,

(2) $\mathcal{Q}(s(N)) \subseteq \mathcal{Q}(N)$ for $N \neq r(T)$,

(3) $\mathcal{Q}(\mathcal{B})$ is a complete assignment to K for $\mathcal{B}$ a complete branch of T and

(4) for $N \in T$ such that $s^{-1}(N) = \{N_1, \ldots, N_n\}$, $B_1 \vee \ldots \vee B_n$ is a tautology where B_i is the conjunction of the literals in $\mathcal{Q}(N_i) - \mathcal{Q}(N)$.

Note that because our convention of considering trees as growing upward, the orientation of semantic trees in this paper is opposite to their orientation in [17] . If $K = \hat{H}(S)$ for some S then $\mathcal{Q}(\mathcal{B})$ is a Herbrand interpretation of S if $\mathcal{B}$ is a complete branch of a semantic tree for K.

That conversely for every Herbrand interpretation M of S there exists a complete branch $\mathcal{B}$ of $\mathcal{S}$ such that $\mathcal{Q}(\mathcal{B}) = M$ is a consequence of the following

Proposition 2.2.1. If $\mathcal{S} = (T, \mathcal{Q})$ is a semantic tree for K and M is a complete assignment to K then $M = \mathcal{Q}(\mathcal{B})$ for some complete branch $\mathcal{B}$ of $\mathcal{S}$.

Proof. Given M construct $\mathcal{B}$ as follows: $r(T) \in \mathcal{B}$. If $N \in \mathcal{B}$ and $s^{-1}(N) = \{ N_1, \ldots, N_n \}$ then since M is an interpretation of $B_1 \vee \ldots \vee B_n$, where B_i is the conjunction of the literals in $\mathcal{Q}(N_i) - \mathcal{Q}(N)$, and since $B_1 \vee \ldots \vee B_n$ is true in M, some B_i moreover is true in M and therefore each literal in $\mathcal{Q}(N_i) - \mathcal{Q}(N)$ is true in M. So $\mathcal{Q}(N_i) - \mathcal{Q}(N) \subseteq M$. Let $N_i \in \mathcal{B}$. If $\mathcal{B}$ is the complete branch of T defined in this way then $\mathcal{Q}(\mathcal{B}) \subseteq M$. But $M \subseteq \mathcal{Q}(\mathcal{B})$ and $M = \mathcal{Q}(\mathcal{B})$ since $\mathcal{Q}(\mathcal{B})$ is a complete assignment to K.

Clash Trees. A semantic tree $\mathcal{S} = (T, \mathcal{Q})$ for a set of ground atoms K is a clash tree when for any $N \in T$, $s^{-1}(N) = \{N_1, \ldots, N_{m+1}\}$ implies that

$$\mathcal{Q}(N_i) = \mathcal{Q}(N) \mathbin{\dot{\cup}} \{L_i\}, \quad 1 \leq i \leq m \qquad \text{and}$$

$$\mathcal{Q}(N_{m+1}) = \mathcal{Q}(N) \mathbin{\dot{\cup}} \{\bar{L}_1, \ldots, \bar{L}_m\} ,$$

for some $L_1, \ldots, L_m$ such that $|L_1|, \ldots, |L_m| \in K$.

The nodes $N_1, \ldots, N_m$ are satellite nodes and N_{m+1} a nucleus node of $\mathcal{S}$. All of the clash trees investigated in this paper will be one of the two following kinds.

Binary Semantic Tree for Ordered K. Let K be a totally ordered (finite or infinite) non-empty set of ground atoms, $K = \{L_1, \ldots, L_n, \ldots\}$, where $i < j$ implies that A_i precedes A_j in the given ordering of K. The binary semantic tree $\mathcal{S} = (T, \mathcal{A})$ for K ordered in this way is given by:

(1) $\mathcal{A}(r(T)) = \emptyset$.

(2) If $N \in T$ and $\mathcal{A}(N)$ is a complete assignment to some $K' \subseteq K$ then

(a) If $K' = K$ then $s^{-1}(N) = \emptyset$, otherwise

(b) If $K' = \{L_i, L_{i+1}, \ldots L_n, \ldots\}$ then $s^{-1}(N) = \{N_1, N_2\}$ for some $N_1, N_2 \in T$ and $\mathcal{A}(N_1) = \mathcal{A}(N) \overset{\circ}{\cup} \{L_i\}$, $\mathcal{A}(N_2) = \mathcal{A}(N) \overset{\circ}{\cup} \{\overline{L}_i\}$.

Note that if K' is an initial segment of K and if M' is a complete assignment to K' then $M' = \mathcal{A}(N)$ for some $N \in T$.

M -Clash Tree for K. Let K be a finite set of ground atoms and M a complete assignment to K, then the M-clash tree $\mathcal{S} = (T, \mathcal{A})$ for K is defined by:

(1) $\mathcal{A}(r(T)) = \emptyset$.

(2) If $N \in T$ and $\mathcal{A}(N)$ is a complete assignment to some $K' \subseteq K$ then

(a) If $K' = K$ then $s^{-1}(N) = \emptyset$ otherwise

(b) $\mathcal{A}(N) \subset M$. Let $M - \mathcal{A}(N) = \{L_1, \ldots, L_m\}$. Then $s^{-1}(N) = \{N_1, \ldots N_{m+1}\}$ for some $N_1, \ldots, N_{m+1} \in T$ and $\mathcal{A}(N_i) = \mathcal{A}(N) \overset{\circ}{\cup} \{L_i\}$ for $1 \leq i \leq m$, $\mathcal{A}(N_{m+1}) = \mathcal{A}(N) \overset{\circ}{\cup} \{\overline{L}_1, \ldots \overline{L}_m\}$.

We need to verify that given K and M the M-clash tree for K actually exists. For this purpose it suffices to verify that $\mathcal{A}(N) \subset M$ whenever $\mathcal{A}(N)$ is a complete assignment to $K' \subset K$.

Suppose this is not the case, then because T is well-founded there exists a lowest interior node N_0 such that $\mathcal{Q}(N_0) \not\subset M$ (i.e. $\mathcal{Q}(N_0) \not\subset M$ and $\mathcal{Q}(s\ (N_0)) \subset M$). $N_0 \neq r\ (T)$ since $\mathcal{Q}(\ r\ (T)) = \emptyset \subset M$. But $\mathcal{Q}(s(N_0)) \subset M$ implies that either

$\mathcal{Q}(N_0) = \mathcal{Q}(s(N_0)) \quad \dot{\cup} \quad \{L_i\}$, for some $L_i \in M - \mathcal{Q}(N)$, or

$\mathcal{Q}(N_0) = \mathcal{Q}(s(N_0)) \quad \dot{\cup} \quad \{\overline{L}_1, \dots, \overline{L}_m\}$ for $M - \mathcal{Q}(N) =$

$\{L_1, \dots, L_m\}$.

In the first case $\mathcal{Q}(N_0) \subset M$, in the second case $K' = K$. It follows that the M-clash tree for K does in fact always exist.

Note that if $K' \subseteq K$ and M' is a complete assignment to K' then $M' = \mathcal{Q}(N)$ for some $N \in T$.

Failure. Let $\mathcal{S} = (T, \mathcal{Q})$ be a semantic tree for some set of ground atoms K and let S be a set of clauses. A clause $C \in S$ fails at $N \in T$, if $\overline{C}\sigma \subseteq \mathcal{Q}(N)$. Note that

(1) C fails at r(T) if and only if $C = \Box$.

(2) If $K = \hat{H}(S)$, $\mathcal{B}$ is a complete branch of $\mathcal{S}$ and C fails at $N \in \mathcal{B}$, then $\mathcal{Q}(\mathcal{B}) \models \neg C$.

(3) If C fails at N then C is not a tautology. (If C were a tautology and $\overline{C}\sigma \subseteq \mathcal{Q}(N)$ then $C\sigma$ would be a tautology and $\mathcal{Q}(N)$ would contain complementary literals.)

(4) If C fails at N then C subsumes the clause $\overline{\mathcal{Q}(N)}$.

$C \in S$ fails properly at $N \in T$ if C fails at N and either $N = r(T)$ or C does not fail at s (N). A node $N \in T$ is free for S if no $C \in S$ fails at N. A node $N \in T$ is a failure point for S if some $C \in S$ fails at N and either

$N = r(T)$ or $s(N)$ is free for S. If $N \in T$ and T_N is the subtree of T rooted in $N \in T$ then a cut X through T_N is a <u>frontier of T_N for S</u> if every node in N is a failure point for S. T_N is <u>closed for S</u> if some cut X through T_N is a frontier of T_N for S. If $N = r(T)$ and T_N is closed for S then we also say that <u>$\mathcal{S}$ is closed for S.</u>

<u>Proposition 2.2.2.</u> If T_N is closed for S then T_N is closed for some finite set S' of ground instances of clauses in S.

<u>Proof.</u> Let X be a frontier for S. X is finite by 1.7.4. For each $N \in X$ let C'_N be some ground instance of a clause $C \in S$ which fails at N (i.e. $C'_N = C\sigma$ where $\overline{C}\sigma \subseteq \mathcal{A}(N)$). Then $S' = \{ C'_N : N \in X\}$ is finite and X is a frontier for S.

<u>Proposition 2.2.3.</u> If some semantic tree $\mathcal{S} = (T, \mathcal{A})$ for some K is closed for S then S is unsatisfiable.

<u>Proof.</u> Let $K' = K \cap \hat{H}(S)$ and let M be a Herbrand interpretation of S. Let $M' \subseteq M$ be the complete assignment to K' contained in M and let M" be any extension of M' to a complete assignment to K. Then $M'' = \mathcal{A}(\mathcal{B})$ for some complete branch of T. Since $\mathcal{S}$ is closed for S, $\overline{C}\sigma \subseteq \mathcal{A}(\mathcal{B})$ for some ground instance of a clause $C \in S$. But then $\overline{C}\sigma \subseteq M' \subseteq M$, so C and therefore S is false in M. Because M was an arbitrary Herbrand interpretation of S, S is unsatisfiable since it has no Herbrand models.

Proposition 2.2.4. Let S be unsatisfiable and let $\mathcal{S} = (T, \mathcal{A})$ be a semantic tree for $\hat{H}(S)$. Then $\mathcal{S}$ is closed for S.

Proof. We need to show that some cut through T is a frontier for S or equivalently that every complete branch $\mathcal{B}$ of T contains a failure point for S. Let $\mathcal{B}$ be such a branch then the unsatisfiability of S implies that $\mathcal{A}(\mathcal{B}) \models \neg S$, i.e. $\bar{C}\sigma \subseteq \mathcal{A}(B)$ for some ground instance of some $C \in S$. Since $\mathcal{B}$ is well-founded either r(T) is a failure point for S or there exists a node $N \in \mathcal{B}$ such that some $C \in S$ fails at N but no $D \in S$ fails at s(N). In either case $\mathcal{B}$ contains a failure point for S.

Corollary 2.2.5 (Herbrand's Theorem). If S is unsatisfiable then some finite set S' of instances of clauses in S is unsatisfiable.

Proof. Let $\mathcal{S}$ be the binary semantic tree for $\hat{H}(S)$ ordered in some way. Then $\mathcal{S}$ is closed for S and is therefore closed for some finite set S' of instances of clauses in S. It follows that S' is unsatisfiable.

2.3 Semantic Trees and Derivations.

Let $\mathcal{S} = (T, \mathcal{A})$ be a semantic tree and S a set of clauses, $N \in T$ is an inference node for S if $s^{-1}(N)$ is a set of failure points for S.

Proposition 2.3.1. If $\mathcal{S} = (T, \mathcal{A})$ is a semantic tree and T_N is closed for S where $N \in T$, then either T_N contains an inference node for S or some C $\in$ S fails at N.

Proof. If no C $\in$ S fails at N and X is some frontier of T_N for S then by 1.7.5, since $X \neq \{N\}$, $s^{-1}(N') \subsetneq X$ for some $N' \in T_N$. But then N' is an inference node for S.

The following theorem and its first corollary provide the basis for two methods of applying semantic trees to establish the completeness of resolution inference systems.

Theorem 2.3.2. Let $\mathcal{S} = (T, \mathcal{A})$ be a clash tree and let T_{N_0}, $N_0 \in T$, be closed for a set of clauses S. Then there exists a derivation $\mathcal{D} = (T', c)$ from S of a clause C which fails at N_0. There is a 1-1 mapping $\psi : T' \to T_{N_0}$ such that

(1) If $N \in T'$ is a tip then $c(N) \in S$ fails properly at $\psi(N)$ and $\psi(N)$ is a failure point for S in T_{N_0}.

(2) If $N \in T'$ is an interior node then $\psi(N)$ is interior to T_{N_0}. If $\mathcal{C}$ is the clash $c(s^{-1}(N))$ at N with resolvent c (N) then

(a) $\mathcal{C}$ is restricted.

(b) the satellites of $\mathcal{C}$ fail properly at satellite nodes of $s^{-1}(\psi(N))$,

(c) The nucleus of $\mathcal{C}$ fails properly at the nucleus node of $s^{-1}(\psi(N))$,

(d) c (N) fails at $\psi(N)$ and

(e) If A $\in \mathcal{C}$ fails properly at $N' \in s^{-1}(\psi(N))$ and $\bar{A}\sigma \subseteq \mathcal{A}(N')$ then

$L \in A$ is resolved upon in $\mathcal{C}$ if and only if

$$\overline{L}\sigma \subseteq \mathcal{Q}(N') - \mathcal{Q}(N).$$

(3) No $c(N)$, for $N \in T'$, is a tautology.

(4) $\mathcal{D}$ is minimal.

Proof. Let X be a frontier of T_{N_0} for S. Let $\mathcal{D}' = (T_{N_0}/X, c')$ be the ground derivation defined by $c'(N) = \overline{\mathcal{Q}(N)}$ for all $N \in T_{N_0}/X$. The definition of clash tree guarantees that if N is interior to T_{N_0}/X then $c'(s^{-1}(N))$ is a restricted clash with resolvent $c'(N)$. Thus $\mathcal{D}$ is a derivation. The conditions on assignments that they contain no complementary literals implies that $\mathcal{D}'$ contains no tautologies. The condition that $\mathcal{Q}(s(N)) \subset \mathcal{Q}(N)$, for N interior to T, implies that $\mathcal{D}'$ is minimal.

For every tip $N \in T_{N_0}/X$ (i.e. for $N \in X$) let $A_N \in S$ be a clause which fails at N (i.e. $\overline{A_N\sigma} \subseteq \mathcal{Q}(N)$ for some σ). Then A_N subsumes $c'(N)$. Let $S' = \{A_N : N \in X\}$ be standardised. By the contraction theorem there exists a derivation $\mathcal{D} = (T', c)$ from S' and therefore from S of a clause which subsumes $c'(N_0)$, i.e. of a clause which fails at N_0. $\mathcal{D}$ is a contraction of $\mathcal{D}'$ and therefore $\mathcal{D}$ contains no tautologies and is minimal. If ψ is the mapping associated with the contraction then $\mathcal{D}$ satisfies properties (1) and (2) of the theorem.

Corollary 2.3.3. Let $\mathcal{S} = (T, \mathcal{Q})$ be a clash tree and let $N_0 \in T$ be an inference node for S. Then there exists a clash $\mathcal{C}$ such that each clause in $\mathcal{C}$ is a variant of a clause in S and

(a) $\mathcal{C}$ is restricted,

(b) the satellites of $\mathcal{C}$ fail properly at satellite nodes of $s^{-1}(N)$,

(c) the nucleus of $\mathcal{C}$ fails properly at the nucleus node of $s^{-1}(N)$,

(d) the resolvent C of $\mathcal{C}$ fails at N,

(e) if $A \in \mathcal{C}$ fails properly at $N' \in s^{-1}(N)$ and $\bar{A}\sigma \subseteq \mathcal{A}(N')$ then $L \in A$ is resolved upon in $\mathcal{C}$ if and only if $\bar{L}\sigma \subseteq \mathcal{A}(N') - \mathcal{A}(N)$ and

(d) neither C nor any of the clauses in $\mathcal{C}$ are tautologies.

<u>Proof.</u> T_N is closed for S. The corresponding derivation $\mathcal{D}'$ of a clause which fails at N consists of just the single clash $\mathcal{C}$.

<u>Corollary 2.3.4.</u> If S is unsatisfiable then there exists a minimal binary refutation of S containing no tautologies.

<u>Proof.</u> Let $\mathcal{S} = (T, \mathcal{A})$ be the binary semantic tree for $\hat{H}(S)$ ordered in some way. Theorem 2.3.2 guarantees the existence of a minimal refutation $\mathcal{D}$ of S containing no tautologies. Since $s^{-1}(N)$ contains exactly two elements for N interior to T, $\mathcal{D}$ is binary.

A theorem similar to corollary 2.3.4 was proved by Loveland in [23] for the case of ground sets of clauses S. In section 2.6 we shall see that corollary 2.3.4 can be strengthened by introducing the notion of α-ordering in order to make use of the ordering of $\hat{H}(S)$ in the proof of 2.3.4.

2.4 M-Clash Derivations.

Let S be a set of clauses, M a Herbrand interpretation of S and $\mathcal{C}$ a clash with satellites $A_1, \ldots, A_n$, nucleus B and m.g.s.u. θ. Then $\mathcal{C}$ is an M-clash if

(1) $A_1\theta, \ldots, A_n\theta$ and C are false in M,

(2) $B \in S$ and $B\theta$ has an instance $B\theta\lambda$ true in M, $\{ L : L\theta\lambda \in B\theta\lambda \cap M\}$ is the subset of literals in B resolved upon in $\mathcal{C}$ and

(3) $\mathcal{C}$ is restricted.

A clash derivation $\mathcal{D}$ is an M-clash derivation if each clash in $\mathcal{D}$ is an M-clash. The definition of M-clash introduced by Slagle in [51] is less restrictive and is generally easier to apply. Conditions (1) and (2) above are replaced by

(1') $A_1, \ldots, A_n$ and C are false in M and

(2') B has an instance true in M.

The following theorem is a third corollary of Theorem 2.3.2. Because 2.3.2 was proved by applying the contraction theorem, the proof of Theorem 2.4.1 is equivalent to a proof that M-clash derivations are preserved under contractions.

Theorem 2.4.1. Let $\mathcal{J} = (T, \mathcal{Q})$ be an M-clash tree and let T_{N_0}, $N_0 \in T$, be closed for S. Then there exists a minimal M-clash derivation $\mathcal{D}' = (T', c)$ from S of a clause which fails at N_0. $\mathcal{D}'$ contains no tautologies.

Proof. Let $\mathcal{D}' = (T', c)$ be the minimal derivation containing no tautologies of Theorem 2.3.2 corresponding to T_{N_0}. Let $\psi : T' \to T_{N_0}$ be the associated mapping. It suffices to show that if $N \in T'$ is interior to T' and if $\mathcal{C} = \{A_1, \ldots, A_n, B\}$ is the clash at N then $\mathcal{C}$ is an M-clash.

Let $N' = \psi(N)$. The satellites $A_1, \ldots, A_n$ of $\mathcal{C}$ fail properly at satellite nodes $N_1', \ldots, N_n'$ of $s^{-1}(N')$ and the nucleus B of $\mathcal{C}$ fails properly at the nucleus node N'_{n+1} of $s^{-1}(N')$. Since $\mathcal{C}$ is standardised there is a single substitution σ such that

$$\bar{A}_i \sigma \subseteq \mathcal{Q}(N_i') \text{ and } \bar{B}\sigma \subseteq \mathcal{Q}(N_{n+1}')$$

But then σ unifies $\mathcal{C}$ and therefore $\sigma = \theta\lambda$ for some λ where θ is an m.g.s.u. of $\mathcal{C}$. The resolvent C of $\mathcal{C}$ fails at N' which is a satellite node of $\mathcal{S}$. Thus $A_1\theta, \ldots, A_n\theta$ and C fail at satellite nodes of $\mathcal{S}$. But if a clause D fails at a satellite node N" of $\mathcal{S}$ then for some substitution λ, $D\lambda \subseteq \mathcal{Q}(N'') \subseteq M$, i.e. D is false in M.

$B\theta\lambda \subseteq \mathcal{Q}(N'_{n+1})$ and $L \in B$ is resolved upon in $\mathcal{C}$ if and only if

$\bar{L}\theta\lambda \subseteq \mathcal{Q}(N_{n+1}') - \mathcal{Q}(N') \subseteq \bar{M}$, i.e. if and only if

$L\theta\lambda \in B\theta\lambda \cap M$.

The instance $B\theta\lambda$ of $B\theta$ is therefore true in M. Since N_{n+1}' is a tip of T, $B \in S$. $\mathcal{C}$ is restricted and therefore $\mathcal{C}$ is an M-clash and $\mathcal{D}$ is an M-clash derivation.

<u>Corollary 2.4.2.</u> If S is unsatisfiable and M is a Herbrand interpretation of S then there exists a minimal M-clash refutation of S containing no tautologies.

<u>Proof.</u> Let S' be a finite unsatisfiable set of ground instances of clauses in S and let K be the finite set of ground atoms occurring in clauses in S'. Then $K = \hat{H}(S') \subseteq \hat{H}(S)$ and some subset M' of M is a complete assignment to K. Since S' is unsatisfiable, the M'-clash tree $\mathcal{S} = (T, \mathcal{A})$ for K is closed for S' and is therefore closed for S. By 2.4.1 (letting $N_0 = r(T)$ and M' be the M of 2.4.1) there exists a minimal M'-clash refutation $\mathcal{D}$ of S containing no tautologies. But since $M' \subseteq M$, $\mathcal{D}$ is also an M-clash refutation of S.

<u>Remarks.</u>

It is the existence of M-clash derivations satisfying conditions (1) and (2) rather than (1') and (2') which is necessary to justify the completeness of extending M-clash resolution to systems which employ factoring. If $\mathcal{C} = \{A_1, \ldots, A_n, B\}$ is an M-clash with resolvent C and m.g.s.u. θ then there is a set of factors $\mathcal{C}' = \{A_1', \ldots A_n', B'\}$ with resolvent C and m.g.s.u. θ' where

$$A_i' = A_i\theta_i \quad , \quad B' = B\theta_{n+1} \text{ and}$$
$$\theta = \theta_1, \ldots, \theta_{n+1}\theta'$$

The clash $\mathcal{C}'$ is restricted, each $A_i'\theta' = A_i\theta$ is false in M and $B'\theta' = B\theta$ has an instance $B'\theta'\lambda$ true in M. The literals $L' \in B'$ resolved upon in $\mathcal{C}'$ are precisely those literals for which $L'\theta'\lambda \in M$. Thus $\mathcal{C}'$ is an M-clash with resolvent C.

(2) M-clash resolution is a theoretically interesting resolution method. Its potentiality for efficient theorem-proving however seems quite limited. To implement M-clash resolution for a given Herbrand interpretation M it is necessary to find efficient procedures for determining both when clauses are false in M and when clauses have instances true in M. Such procedures exist forvery few Herbrand interpretations.

For example, suppose that S_0 is a set of clauses representing the axioms for group theory and the negation of some proposed theorem. Suppose that M_0 is some finite group of small cardinality. First it is necessary to extend M_0 to a Herbrand interpretation M by introducing denotations for the Skolem function symbols of S_0. It is then necessary to provide an algorithm for deciding when instances of clauses C over $H(S_0)$ are true or false in M. In most cases this will have to be done by enumerating all ground instances of C and by individually deciding the validity in M of each such instance. This process will in general be a very lengthy one even for models M_0 of small cardinality.

(3) Perhaps the most interesting use of M-clashes is for establishing connections among hyper-resolution [40] renaming [25] and set of support. As noted by Slagle [51] all of these resolution methods are examples of M-clash resolution.

Hyper-resolution is obtained by choosing as the Herbrand interpretation M, for a given set of clauses S_0, the set $M = \overline{H}$ where $H = \hat{H}(S_0)$. Although M is usually infinite, this case of M-clash resolution is especially easy to apply since a clause A is false in M if and only if it is positive. A clause B has an instance true in M if and only if it is non-positive; precisely the negative literals in B are resolved upon in any M-clash (hyper-resolution clash) $\mathcal{C}$ containing B as nucleus.

Let $\Lambda = \{P_1, \ldots, P_n\}$ be the set of all predicate symbols occurring in a given set of clauses S_0. Let $\Lambda' = \{P_1, \ldots P_m\}$, $0 \leq m \leq n$, . be a subset of Λ and let

$$M = \{L : L \in \hat{H}(S_0) \text{ and } L = P_i(t_1, \ldots t_{n_i}), i \leq m\}$$
$$\dot{\cup} \{L : L \in \hat{H}(S_0) \text{ and } L = P_j(t_1, \ldots t_{n_j}), m \leq j \leq n\}.$$

Then M is a Herbrand interpretation of S. In this case M-clash resolution is equivalent to hyper-resolution after <u>renaming</u>, i.e. after replacing in S_0 each literal $L \in C \in S_0$ by $\overline{L}$ when $L = P_i(s_1, \ldots, s_{n_i})$ and $P_i \in \Lambda'$.

Given a set of clauses S_0 and a satisfiable subset $S' \subseteq S_0$, let M be a Herbrand model of S'. Then the satellites and resolvent of every M-clash $\mathcal{C} = \{A_1, \ldots, A_n, B\}$ are false in M and therefore do not belong to S'. Since $\mathcal{C}$ is restricted the resolvent C of $\mathcal{C}$ can be obtained by resolving a sequence of binary clashes $\mathcal{C}_1, \ldots, \mathcal{C}_n$ where $\mathcal{C}_1 = \{A_1, B\}$ and for $2 \leq i \leq n$, $\mathcal{C}_i = \{A_i, C_{i-1}\}$ where C_{i-1} is the resolvent of $\mathcal{C}_{i-1}$ (see section 2.8 below). The resolvent of $\mathcal{C}$ is C and no two clauses from S' are resolved together to obtain C. This last condition that no two clauses from S' are resolved in a binary clash can be interpreted as the definition of the <u>set of support</u> resolution method.

<u>2.5 Deduction Completeness.</u>

Much of the efficiency of resolution derives from the fact that it is not a complete rule for deriving logical consequences. More precisely, given a set of clauses S_0, the process of searching for a refutation of S_0 is accelerated by not generating certain of the logical consequences of S_0 along the way.

Theorem 2.5.1, which generalises the subsumption theorem of [20] and the deduction completeness theorems of [52] provides information about the extent of deduction completeness for resolution. Theorem 2.5.1 is used to establish the permutation theorem of Chapter 3.

<u>Theorem 2.5.1.</u> Let S be a set of clauses, $S \neq \emptyset$, and C a clause which is not a tautology, logically implied by S.

(1) There exists a minimal binary derivation $\mathcal{D}_1$ from S of a clause D which subsumes C.

(2) If M is a Herbrand interpretation of S and if C is false in M then there exists a minimal M-clash derivation $\mathcal{D}_2$ from S of a clause D which subsumes C.

(3) Neither $\mathcal{D}_1$ nor $\mathcal{D}_2$ contain tautologies.

<u>Proof.</u> If S logically implies C, then $S \cup \neg C$ is

unsatisfiable. Let $C = \{L_1(x_1,\ldots,x_n),\ldots,L_m(x_1,\ldots,x_n)\}$ where $x_1,\ldots,x_n$ are all the variables occurring in C and $L_i(x_1,\ldots,x_n)$ indicates all occurrences of these variables in $L_i \in C$. $\neg C$ is logically equivalent to $\exists x_1,\ldots,x_n$ $(\overline{L}_1(x_1,\ldots,x_n)$ & ... & $\overline{L}_m(x_1,\ldots,x_n))$. Let $a_1,\ldots a_n$ be constant symbols not occurring in $S \cup \neg C$ and let $(\neg C)^*$ $= \{\{\overline{L}_1(a_1,\ldots,a_n)\},\ldots,\{\overline{L}_m(a_1,\ldots,a_n)\}\}$. then $S_0 = S \cup (\neg C)^*$ is unsatisfiable.

(1) Let $\mathcal{S} = (T, \mathcal{Q})$ be the binary semantic tree for $\hat{H}(S_0)$ ordered in such a way that the atoms $|L_1(a_1,\ldots a_n)|$, ,..., $|L_m(a_1,\ldots,a_n)|$ precede all others in the ordering of $\hat{H}(S_0)$. Then, because C is not a tautology there exists a node $N \in T$ such that $\mathcal{Q}(N) = \cup(\neg C)^*$. T_N is closed for S_0 unless some clause $D \in S_0$ and therefore $D \in S$ fails (improperly) at N. In this case let $\mathcal{D} = (T', c)$, where $T' = \{N_0\}$ and $c(N_0) = D$. If T_N is closed for S_0 then it is closed for S since no clause in $(\neg C)^*$ fails in T_N). In this case also, by Theorem 2.3.2, there exists a minimal binary derivation $\mathcal{D}$ of a clause D which fails at N. We shall show that any such clause D subsumes C. But first:

(2) Let C be false in M. Then $\overline{C}\lambda \subseteq M$ for some ground substitution $\lambda = \{t_1/x_1,\ldots,t_n/x_n\}$ where $t_i \in H(S)$. Extend M to a Herbrand interpretation M^* of S_0 by defining

$L(a_1,\ldots,a_n) \in M^*$ if and only if $L(t_1,\ldots,t_n) \in M$ and
$\overline{L}(a_1,\ldots,a_n) \in M^*$ if and only if $\overline{L}(t_1,\ldots,t_n) \in M$.

M^* contains no complementary literals and contains either L or $\bar{L}$ for each $L \in \hat{H}(S_0)$. Therefore M^* is a Herbrand interpretation of S_0 and $M \subseteq M^*$. Note moreover that $\bar{C}\lambda \subseteq M$ implies that $\cup(\neg C)^* \subseteq M^*$.

Let S_0' be a finite unsatisfiable set of ground instances of clauses in S_0 and let $M' \subseteq M^*$ be the subset of M^* which is a complete assignment to the atoms occurring in clauses of S_0'. Let $\mathcal{S} = (T, \mathcal{C})$ be the M'-clash tree for S_0'. $\mathcal{S}$ is closed for S_0' and therefore for S_0. Since C is not a tautology, $\mathcal{C}(N) = \cup(\neg C)^*$ for some $N \in T$. Either some $D \in S$ fails at N or T_N is closed for S. In either case there exists a minimal M'-clash derivation $\mathcal{D} = (T', c)$ from S of a clause D which fails at N. $\mathcal{D}$ is also an M^*-clash derivation of D since $M' \subseteq M^*$.

Thus each satellite and resolvent of a clash in $\mathcal{D}$ is false in M^* and each nucleus has an instance true in M^*. But no clause $A = c(N')$, $N' \in T'$, contains any of the constants $a_1, \ldots, a_n$. If σ^* is a ground substitution all of whose terms belong to $H(S_0)$ let σ be the ground substitution which differs from σ^* by having the term t_i whenever σ^* has a_i. Then

$$\bar{A}\sigma^* \subseteq M^* \quad \text{if and only if } \bar{A}\sigma \subseteq M \quad \text{and}$$

$$A\sigma^* \cap M^* \neq \emptyset \quad \text{if and only if } A\sigma \cap M \neq \emptyset .$$

Thus each satellite and resolvent in $\mathcal{D}$ is false in M and each nucleus in $\mathcal{D}$ has an instance true in M and therefore $\mathcal{D}$ is an M-clash derivation of D. (For M-clashes as defined by (1) and (2) instead of (1') and (2') in section 2.4 a slightly more detailed argument along the same lines as above is needed.)

(1) and (2) concluded: It remains to show that if a clause D fails at a node N, where $\mathcal{Q}(N) = \cup (\neg C)^*$, then D subsumes C. But

$$\bar{D} \sigma^* \subseteq \mathcal{Q}(N) = \cup (\neg C)^* \text{ for some } \sigma^*.$$

Let σ differ from σ^* by having x_i whenever σ^* has a_i in any of its terms, for each i, $1 \leq i \leq n$. Then $D \sigma \subseteq \{L_1(x_1, \ldots, x_n), \ldots L_m(x_1, \ldots, x_n)\}$, i.e. D subsumes C.

It should be noted that Theorem 2.5.1 does not settle the problem of generating consequences from assumptions by resolution. That this is so is due to the fact that if A and B are sentences of first-order logic, if A implies B and if A^* and B^* are the sets of clauses corresponding to A and B, then it is not generally true that A^* implies B^*. $A = \exists y \forall x\, P(x,y)$ and $B = \forall x\, \exists y\, P(x,y)$ provide a simple counterexample.

2.6 α-Ordering and Binary Resolution.

Let S be a set of clauses and $\leq_A$ a total ordering of $\hat{H}(S)$. (Write $L_1 <_A L_2$ for $L_1 \leq_A L_2$ and not $L_2 \leq_A L_1$.) The notion of A-restriction, which extends Slagle's definition [51], provides the basis for studying the completeness of the more effective α-restriction (called A-restriction in [17] and α-restriction in [20]) .

Let $\mathcal{D} = (T,c)$ be a derivation and let $N \in T$, $N \neq r(T)$. Then $c(N)$ satisfies the <u>A-restriction</u> if

$$|L\theta\lambda| >_A |L'\theta\lambda| \qquad \text{for some } \lambda, \text{ for } L \in c(N)$$

resolved upon at N, for $L' \in c(N)$ not resolved upon at N and for θ m.g.s.u. of the clash at s(N).

The weaker restriction that

$$|L\sigma| >_A |L'\sigma| \qquad \text{for some } \sigma, \text{ for } L \in c(N)$$

resolved upon at N and for $L' \in c(N)$ not resolved upon at N

(as in the case of the corresponding weakening of the M-clash rule) is not sufficiently restrictive to justify extending A-restrictions to clashes of factors (compare remark (1) section 2.4).

The following theorem translates the ordering for binary semantic trees into A-restrictions on the corresponding binary derivation. The second half of the proof of 2.6.1 is equivalent to a demonstration that α-restrictions are preserved under contractions.

<u>Theorem 2.6.1.</u> Given S unsatisfiable and A a total ordering of $\hat{H}(S)$ there exists a minimal binary refutation $\mathcal{D} = (T,c)$ of S such that $\mathcal{D}$ contains no tautologies and, for all $N \in T$, $N \neq r(T)$, $c(N)$ satisfies the A-restriction.

<u>Proof.</u> Let S' be a finite unsatisfiable set of instances of clauses in S. Then $\hat{H}(S') \subseteq \hat{H}(S)$ and A totally orders S'. Let $\mathcal{S} = (T',Q)$ be the binary semantic tree for $\hat{H}(S')$

ordered by A. Then $\mathcal{S}$ is closed for S' and therefore for S.

Let $\mathcal{D} = (T, c)$ and $\Psi : T \rightarrow T'$ be as in Theorem 2.3.2 where $N_0 = r(T')$. Then $\mathcal{D}$ is a minimal binary refutation of S containing no tautologies. Let $N \in T$, $N \neq r(T)$. Then $C = c(N)$ fails properly at some node $N' \in s^{-1}(\Psi(s(N)))$, $N' \in T'$. Therefore $\overline{C}\sigma \subseteq \mathcal{Q}(N')$ for some ground substitution σ. If $L \in C$ is resolved upon at N and $L' \in C$ is not resolved upon at N then

$\{\overline{L}\sigma\} = \mathcal{Q}(N') - \mathcal{Q}(s(N'))$ and $L' \in \mathcal{Q}(s(N'))$.

But by the construction of $\mathcal{S}$, $|\overline{L}\sigma| >_A |\overline{L'}\sigma|$, i.e. $|L\sigma| \leq_A |L'\sigma|$. But $\sigma = \theta\lambda$ for some λ where θ is an m.g.s.u. of the clash at s(N). Therefore c(N) satisfies the A-restriction.

In general A-restrictions may be very difficult to verify, What is wanted is a notion of ordering and corresponding restriction which applies directly to literals occurring in clauses of derivations instead of to literals occurring in ground instances of such clauses. The P-orderings of Slagle [51] meet this requirement and are particularly easy to apply.

Given a set of clauses S and $P_1, \ldots, P_n$ and ordering of the predicate symbols occurring in S, let the partial ordering $\leq_P$ of $\overset{o}{H}(S)$ be defined by *

$L \leq_P L'$ if and only if $L = P_i(t_1, \ldots, t_{n_i})$ and
$L' = P_j(t_1, \ldots, t_{n_j})$ implies $1 \leq i \leq j \leq k$.

* $\overset{o}{H}(S)$ is the set of all atoms obtained by instantiating, by means of any substitution, the atomic formulae ocurring in S.

The partial ordering $\leq_P$ is called a P-ordering for S.
The P-restriction corresponding to a P-ordering $\leq_P$ is defined as follows: Let $\mathfrak{D} = (T,c)$ be a derivation and let $N \in T$, $N \neq r(T)$. Then $c(N)$ violates the P-restriction if

$|L| <_P |L'|$ for $L \in c(N)$ resolved upon at N and

$L' \in c(N)$ not resolved upon at N.

Otherwise $c(N)$ satisfies the P-restriction. Given a P-ordering and a clause C there may be several literals L in C which contain the same predicate letter and such that $L \geq_P L'$ for all $L' \in C$. In this case the P-restriction imposes no restriction on which one of these literals L are to be resolved upon when C occurs in a clash.

The notion of α-restriction includes the case of P-restriction and allows a stricter limitation of the literals which can be resolved upon. Let $\leq_\alpha$ be a partial ordering of $\overset{\circ}{H}(S)$ then $\leq_\alpha$ is an α-ordering for S, if for any $L_1, L_2 \in \overset{\circ}{H}(S)$ and for any substitution σ

$L_1 \leq_\alpha L_2$ implies $L_1\sigma \leq_\alpha L_2\sigma$.

Let $\mathfrak{D} = (T,c)$ be a derivation and let $N \in T$, $N \neq r(T)$.
Then $c(N)$ violates the α-restriction if

$|L\theta| <_\alpha |L'\theta|$ for $L \in c(N)$ resolved upon at N

for $L' \in c(N)$ not resolved upon at N and for θ

m.g.s.u. of the clash at s (N).

Otherwise $c(N)$ satisfies the α-restriction. α-orderings can often be conveniently represented by finite sets of inequality schemes.

Examples.

(1) If the atoms $P(a)$, $P(f(x))$, $P(g(y))$ and $Q(y)$ or their complements occur in a set of clauses S then the inequalities

$P(a) <_{\alpha} P(f(t_1)) <_{\alpha} P(g(t_2))$ and

$P(t_1) <_{\alpha} Q(t_2)$, for all terms t_1 and t_2,

determines an α-ordering for S. If $C = \{P(x), \overline{P}(a), Q(f(x))\}$ then the α-restriction for C implies that $Q(f(x))$ may not be resolved upon in C. If $C = \{P(f\ (x), P(a), Q(f(x))\}$ then only $P(a)$ may be resolved upon in C.

(2) The condition,

$P(t) <_{\alpha} P(f(t))$, for all terms t, imposes an α-ordering for any set of clauses containing P and f. However the condition

$P(t_1) <_{\alpha} P(f(t_2))$, for all terms t_1 and t_2, does not. (Because $P(t_1) \leq_{\alpha} P(f(t_2))$ implies that $P(f(x)) \leq_{\alpha} P(f(x))$, which violates the reflexivity of partial orderings).

(3) In systems which incorporate the use of marked factors α-restrictions can serve to restrict the generation of factors of clauses. Let $\leq_{\alpha}$ be the α-ordering of example (1) and let $C = \{P(x), P(f(y)), P(g(z))\}$. Then C has a total of 5 marked factors (3 of them i-factors). Only 3 marked factors of C are compatible with the α-restriction.

Lemma 2.6.2. Given a set of clauses S and $\leq_{\alpha}$ an α-ordering for S, there exists a total ordering $\leq_{A}$ of $\hat{H}(S)$ such that for any derivation $\mathcal{D} = (T,c)$ from S, for any $N \in T$, $N \neq r(T)$, c(N) satisfies the α-restriction if and only if c(N) satisfies the A-restriction.

Proof. Given S and $\leq_\alpha$ there is at least one total ordering $\leq_A$ of $\hat{H}(S)$ which is compatible with $\leq_\alpha$, i.e. such that

$$L \leq_A L' \text{ whenever } L \leq_\alpha L' \text{ and } L, L' \in \hat{H}(S).$$

(Just extend the restriction of $\leq_\alpha$ to $\hat{H}(S)$ to a total ordering of $\hat{H}(S)$). Let $\mathcal{D} = (T, c)$ be any derivation from S and let c(N), N $\in$ T, satisfy the A-restriction. If c(N) violates the α-restriction then

$|L\theta| \leq_\alpha |L'\theta|$ for some L $\in$ c(N) resolved upon at N, for L' $\in$ c(N) not resolved upon at N and for θ m.g.s.u. of the clash at s(N).

But then $|L\theta\lambda| \leq_\alpha |L'\theta\lambda|$ for all λ and therefore $|L\theta\lambda| \leq_A |L'\theta\lambda|$ for all ground $|L\theta\lambda|, |L'\theta\lambda| \in \hat{H}(S)$.

It follows that c(N) violates the A-restriction contrary to assumption.

Corollary 2.6.3. Given S unsatisfiable and $\leq_\alpha$ an α-ordering for S there exists a minimal binary refutation $\mathcal{D} = (T, c)$ of S such that $\mathcal{D}$ contains no tautologies and, for all N $\in$ T, N $\neq$ r(T), c(N) satisfies the α-restriction.

Proof. Let $\leq_A$ be the total ordering of $\hat{H}(S)$ corresponding to $\leq_\alpha$ by 2.6.2. Let $\mathcal{D}$ be the refutation of S for $\leq_A$ of 2.6.1. They by 2.6.2 each c(N), N $\neq$ r(T), satisfies the α-restriction.

2.7 α-Ordering and M-clashes

Corollary 2.7.2 was proved by Slagle [51] for the case of PM-clash resolution. Theorem 2.7.1 is proved by modifying Slagle's argument.

Theorem 2.7.1. Let S be unsatisfiable, M a Herbrand interpretation of S and $\leq_A$ a total ordering of $\hat{H}(S)$. There exists an M-clash refutation $\mathcal{D}$ of S such that $\mathcal{D}$ contains no tautologies and each satellite clause in $\mathcal{D}$ satisfies the A-restriction.

Proof. Let $\mathcal{S} = (T, \mathcal{Q})$ be an M'-clash tree closed for S where $M' \subseteq M$. ($\mathcal{S}$ exists by the construction in the proof of 2.4.2.) The proof is by induction on the number n of nodes in T free for S. If n=0 then $\square \in S$ and $\mathcal{D} = (T', c)$, where $T' = \{ N_0 \}$ and $c(N_0) = \square$, is the desired refutation of S. Suppose that $n > 0$ and that the theorem holds for any S' such that $\mathcal{S}$ is closed for S' and such that the number of nodes in T free for S' is less than n.

Let $M' = \{ L_1, \ldots, L_m \}$ where $| L_i | \leq_A | L_j |$ for $i \leq j$. Construct $M'' \subseteq M'$ as follows:

$M_0'' = \emptyset$

$M''_{i+1} = M''_i$ if $M''_i \dot{\cup} \{ L_{i+1} \}$ falsifies some $A \in S$,

i.e. if $\bar{A}\sigma \subseteq M_i'' \dot{\cup} \{ L_{i+1} \}$ for some σ ,

otherwise

$M''_{i+1} = M''_i \dot{\cup} \{ L_{i+1} \}$

$M'' = M''_m$.

M'' falsifies no clause in S (since no M''_i does). $M'' \neq M'$ since M' falsifies some clause in S. Let $N \in T$ be such that

$\mathcal{Q}(N) = M''$. Let $s^{-1}(N) = \{N_1, \ldots, N_{k+1}\}$ where N_{k+1} is nucleus.

N_{k+1} is a tip of T and therefore $\mathcal{Q}(N_{k+1})$ falsifies some $B \in S$. Moreover B fails properly at N_{k+1} since B does not fail at N. Each satellite N_i, $1 \leq i \leq k$, is a failure point for S since

$$\mathcal{Q}(N_i) = \{L\} \dot{\cup} \mathcal{Q}(N) = \{L\} \dot{\cup} M'' \text{ for}$$

some $L \in M'$.

Thus N is an inference node for S and some set $\mathcal{C}$ of variants of clauses in S is a clash satisfying conditions (a) - (f) of 2.3.3. Let C be the resolvent of $\mathcal{C}$ and θ an m.g.s.u. of $\mathcal{C}$. Each satellite $A \in \mathcal{C}$ satisfies the A-restriction. For if $E \subseteq A$ is the set of literals in A resolved upon in $\mathcal{C}$ then for some λ

$$\bar{A}\theta\lambda \subseteq \{L\} \cup M'' \text{ for some } L{=}L_i \in M',$$

$$\bar{E}\theta\lambda \quad \{L\} = \{L_i\} \quad \text{and}$$

$$\bar{L'}\theta\lambda \in M'' \text{ for all } L' \in A\text{-}E.$$

But, by the construction of M'', λ may be chosen such that $M''_{i-1} \dot{\cup} \{L_i\}$ falsifies A. So if $L' \in A\text{-}E$ then $\bar{L'}\theta\lambda \in M''_{i+1}$ and therefore $\bar{L'}\theta\lambda = \{L_j\}$ for some $j < i$ and $|E\theta\lambda| >_A |L'\theta\lambda|$. So the clause A in $\mathcal{C}$ satisfies the A-restriction.

Let $\mathcal{D}_0 = (T_0, c_0)$ be the derivation of C from $\mathcal{C}$, i.e.

$$T_0 = \{r(T_0)\} \dot{\cup} s^{-1}(r(T_0)), \; c_0(r(T_0)) = C.$$

$c_0(s^{-1}(r(T_0))) = C$. $\mathcal{D}_0$ is an M-clash derivation containing no tautologies and every satellite in $\mathcal{D}_0$ satisfies the A-restriction.

Let $S' = S \dot{\cup} \{C\}$. Then $\mathcal{S}$ is closed for S' and has fewer than n nodes free for S' (since C fails at N). By induction hypothesis there exists an M-clash refutation $\mathcal{D}_1 = (T_1, c_1)$ of S' such that $\mathcal{D}_1$ contains no tautologies and each satellite in $\mathcal{D}_1$ satisfies the A-restriction. Let $\mathcal{D} = (T', c)$ be obtained from $\mathcal{D}_0$ and $\mathcal{D}_1$ by identifying any tip N of T, such that $c_1(N) = C$ with the root of a copy of $\mathcal{D}_0$. Then $\mathcal{D}$ is the desired refutation of S.

<u>Corollary 2.7.2.</u> Let S be unsatisfiable. M a Herbrand interpretation of S and $\leq_\alpha$ an α-ordering for S. There exists an M-clash refutation $\mathcal{D}$ of S such that $\mathcal{D}$ contains no tautologies and each satellite in $\mathcal{D}$ satisfies the A-restriction.

<u>Proof.</u> By Theorem 2.7.1 and Lemma 2.6.2.

2.7.2 cannot be improved either by insisting that nuclei in also satisfy the α-restriction or by requiring that $\mathcal{D}$ be minimal. Let $S = \{ \{L_1, L_2\}, \{L_1, \bar{L}_2\}, \{\bar{L}_2, L_1\}, \{\bar{L}_1, \bar{L}_2\} \}$, $M = \{L_1, L_2\}$ and $L_1 <_A L_2$. Then S is unsatisfiable but no refutation of S exists having either of the two properties mentioned above.

<u>2.8 Pseudo-clashes.</u>

For a variety of reasons it is usually desirable to reduce the problem of searching for clash refutations to the problem of searching for binary clash refutations (see e.g. section 2.9). This reduction, which can be obtained for restricted clashes, is investigated by introducing the notion of pseudo-clash. In section 2.10 we prove the completeness of resolving maximal pseudo-clashes (the corresponding completeness theorem fails to hold for maximal clashes.)

Let $\hat{\mathcal{C}}$ be a standardised sequence of clauses $(A_1,\dots,A_n, B)$. For $1 \leq i \leq n$, let

$$A_i = E_i \overset{\circ}{\cup} A_{0i} , \qquad E_i \neq \emptyset ,$$

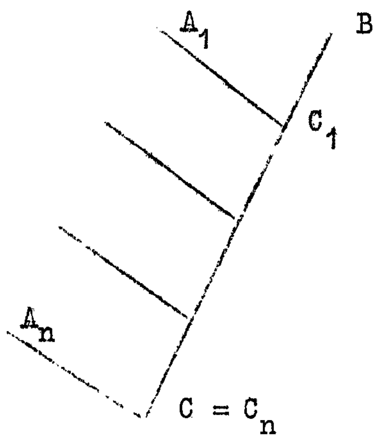
$$B = F_1 \overset{\circ}{\cup} \dots \overset{\circ}{\cup} F_n \overset{\circ}{\cup} B_0 , \qquad F_i \neq \emptyset .$$

Let $C_0 = B$ and suppose $\{A_{i+1}, C_i\}$ is a clash with resolvent C_{i+1} $(0 \leq i \leq n-1)$ where the literals resolved upon in A_{i+1} are E_{i+1} and in C_i the descendants of F_{i+1}. Then $\hat{\mathcal{C}}$ is a *preudo-clash on $(F_1,\dots,F_n)$* with *resolvent* $C = C_n$. The clauses $A_1,\dots,A_n$ are the *satellites* and B the *nucleus* of $\hat{\mathcal{C}}$. $\hat{\mathcal{C}}$ is *restricted* if none of $C_1,\dots,C_{n-1}$ are tautologies when none of $A_1,\dots,A_n$, B and C are tautologies. A derivation $\mathcal{D} = (T, c)$ is a *pseudo-clash derivation* if $s^{-1}(N) \neq \emptyset$ implies that some sequence $\hat{\mathcal{C}}$ consisting of the clauses $c(s^{-1}(N))$ is a pseudo-clash with resolvent $c(N)$.

Remarks.

(1) To every pseudo-clash $\hat{\mathcal{C}}$ there corresponds a binary derivation of the resolvent C of $\hat{\mathcal{C}}$ from the clauses in $\hat{\mathcal{C}}$ (see the figure on next page). The literals resolved upon at interior nodes of the derivation all descend from

literals in the nucleus of $\hat{\mathcal{C}}$. Therefore to any pseudo-clash derivation $\mathcal{D}$ there corresponds the binary derivation $\mathcal{D}'$ obtained by "decomposing" each pseudo-clash in $\mathcal{D}$.



(2) If $\hat{\mathcal{C}}$ is a ground pseudo-clash then $A_i = \{L_i\} \dot{\cup} A_{0i}$, $B = \{\bar{L}_1,\dots,\bar{L}_n\} \dot{\cup} B_0$, $1 \leq i \leq n$, and

$$C = (A_{0i} - \{\bar{L}_2,\dots,\bar{L}_n\}) \cup \dots$$
$$\cup (A_{0i} - \{\bar{L}_{i+1},\dots,\bar{L}_n\}) \cup \dots$$
$$\cup A_{0n} \cup B_0 .$$

If none of $A_1,\dots,A_n$, B and C are tautologies then, for $1 \leq i \leq n$, none of $L_{i+1},\dots,L_n$ belong to A_i (since $\{\bar{L}_{i+1},\dots,\bar{L}_n\} \subseteq C_i$ and $A_{0i} \subseteq C_i$). Conversely if none of $L_{i+1},\dots,L_n$ belong to A_i for all i, $1 \leq i \leq n$, then $\hat{\mathcal{C}}$ is restricted.

(3) If $\hat{\mathcal{C}}' = (A_1',\dots,A_n', B')$ is a ground pseudo-clash with resolvent C' and if $A_1',\dots,A_n'$ and B' are all instances of $A_1,\dots,A_n$ and B respectively then (by the contraction theorem applied to the binary derivation corresponding to $\hat{\mathcal{C}}$) $\hat{\mathcal{C}} = (A_1,\dots,A_n,B)$ is a pseudo-clash and C' is an instance of C, the resolvent of $\hat{\mathcal{C}}$. Moreover $\hat{\mathcal{C}}$ is restricted if $\hat{\mathcal{C}}'$ is.

(4) $\hat{\mathcal{C}} = (A_1,\dots,A_n,B)$ is a pseudo-clash if and only if $\mathcal{C} = \{A_1,\dots,A_n,B\}$ is a clash. If $\mathcal{C}$ is a clash with

m.g.s.u. θ then the resolvent of $\mathcal{C}$ is

$$(A_{01} \cup \ldots \cup A_{0n} \cup B_0)\,\theta\ .$$

However the resolvent of $\hat{\mathcal{C}}$ is

$$(A_{01}\theta - \{F_2,\ldots,F_n\}\,\theta) \cup \ldots$$
$$\cup\,(A_{0i}\theta - \{F_{i+1},\ldots,F_n\}\,\theta) \cup \ldots$$
$$\cup\,A_{0n}\theta \cup B_0\theta\ .$$

Theorem 2.8.1 below implies that if $\mathcal{C} = \{A_1,\ldots,A_n,B\}$ is any restricted clash with resolvent C and if $(A_{\pi(1)},\ldots,A_{\pi(n)})$ is any permutation of the sequence $(A_1,\ldots,A_n)$ then $\hat{\mathcal{C}}_{\pi} = (A_{\pi(1)},\ldots,A_{\pi(n)},B)$ is a restricted pseudo-clash with resolvent C. If $\hat{\mathcal{C}} = (A_1,\ldots,A_n,B)$ is a pseudo-clash then $\mathcal{C} = \{A_1,\ldots,A_n,B\}$ is a clash; however even if $\hat{\mathcal{C}}$ is restricted the resolvent of $\hat{\mathcal{C}}$ may differ from the resolvent of $\mathcal{C}$.

<u>Theorem 2.8.1.</u> Let $\hat{\mathcal{C}} = (A_1,\ldots,A_n,B)$ then $\hat{\mathcal{C}}$ is a restricted pseudo-clash if $\mathcal{C} = \{A_1,\ldots,A_n,B\}$ is a restricted clash and if B θ is not a tautology where θ is an m.g.s.u. of $\mathcal{C}$. The resolvent of $\hat{\mathcal{C}}$ is the resolvent of $\mathcal{C}$.

<u>Proof.</u> We use the notation in the definition of pseudo-clash above. Let $\mathcal{E} = \{E_1 \cup \bar{F}_1,\ldots,E_n \cup \bar{F}_n\}$ and $\mathcal{E}_i = \{E_i \cup \bar{F}_i\}$. If $\mathcal{C}$ is a clash with resolvent C then $\mathcal{E}$ and $\mathcal{E}_i$ are unifiable for $1 \leq i \leq m$. The substitution $\theta_1,\ldots,\theta_m$ is an m.g.s.u. of $\mathcal{E}$ where

θ_i is an m.g.s.u. of $\mathcal{E}_i\,\theta_0\ldots\,\theta_{i-1}$ $(\theta_0 = \varepsilon)$ and

$C = (A_{01} \cup \ldots \cup A_{0n} \cup B_0)\,\theta_1\ldots\,\theta_n$.

Suppose that $\mathcal{C}$ is restricted and that none of $A_1,\dots,A_n,B,C$ are tautologies. It suffices to show that, for $0 \le i \le n$, no C_i is a tautology and

$$C_i = (A_{01} \cup \dots \cup A_{0i} \cup F_{i+1} \cup \dots \cup F_n \cup B_0)\,\theta_0 \dots \theta_i .$$

The proof is by induction on i. If $i = 0$ then $C_0 = B = B\theta_0$ and C_0 is not a tautology. Suppose that the equation for C_i above holds for a given i $(0 \le i \le n)$ and that C_i is not a tautology. We need to show that C_{i+1} is not a tautology and that

$$C_{i+1} = (A_{01} \cup \dots \cup A_{0i+1} \cup F_{i+2} \cup \dots \cup F_n \cup B_0)\,\theta_0 \dots \theta_{i+1} .$$

But C_{i+1} is the resolvent of A_{i+1} and C_i where the literals E_{i+1} are resolved upon in A_{i+1} and the literals $F_{i+1}\,\theta_0 \dots \theta_i$ are resolved upon in C_i. But because $\mathcal{C}$ is standardised $E_{i+1} = E_{i+1}\,\theta_0 \dots \theta_i$. So θ_{i+1} is an m.g.s.u. of $\{E_{i+1} \cup \bar{F}_{i+1}\,\theta_0 \dots \theta_i\}$ and

$$C_{i+1} = A_{0i+1}\,\theta_{i+1} \cup (C_i\,\theta_{i+1} - F_{i+1}\,\theta_0 \dots \theta_{i+1}).$$

Since $\mathcal{C}$ is restricted

$$C_i\,\theta_{i+1} - F_{i+1}\,\theta_0 \dots \theta_{i+1} =$$
$$(A_{01} \cup \dots \cup A_{0i} \cup F_{i+2} \cup \dots \cup F_n \cup B_0)\,\theta_0 \dots \theta_{i+1}.$$

Since $A_{0\,i+1}\theta_{i+1} = A_{0\,i+1}\,\theta_0 \dots \theta_{i+1}$,

$$C_{i+1} = (A_{01} \cup \dots \cup A_{0i+1} \cup F_{i+2} \cup \dots \cup F_n \cup B_0)\,\theta_0 \dots \theta_{i+1}.$$

If C_{i+1} is a tautology then so is $C_{i+1}\,\theta_{i+2} \dots \theta_n = C' \cup C''$ where $C' = (A_{01} \cup \dots \cup A_{0i+1} \cup B_0)\,\theta_0 \dots \theta_n$ and $C'' = (F_{i+2} \cup \dots \cup F_n)\,\theta_0 \dots \theta_n$. But $C' \subseteq C$, so C' is not a tautology. $C'' \subseteq B\theta_0 \dots \theta_n$, so C'' is not a tautology. Therefore $C' \cup C''$ is a tautology only if

$\bar{F}_j\, \theta_0 \ldots \theta_n \cap X \neq \emptyset$ for $X = A_{0k}\, \theta_0 \ldots \theta_n$ or for $X = B_0\, \theta_0 \ldots \theta_n$ for some j and k such that $i + 2 \leq j \leq n$ and $1 \leq k \leq i + 1$. But then $\mathcal{C}$ is not restricted contrary to assumption.

Theorem 2.8.1 fails to hold without the assumption that $B\theta$ is not a tautology. Therefore 2.8.1 does not fully justify the second half of remark (5) above. On the other hand it is not difficult to verify that all completeness theorems which assert the existence of derivations $\mathcal{D} = (T, c)$ containing no tautologies continue to hold under the stronger restriction that no $c(N)\theta$ is a tautology for $N \in T$, $N \neq r(T)$, and θ m.g.s.u. of the clash at $s(N)$. (In fact these theorems hold under the still stronger condition that no $c(N)\, \theta_1 \ldots \theta_m$ is a tautology where $N \neq r(T)$, θ_i is the m.g.s.u. of the clash at $s^i(N)$ and $s^m(N) = r(T)$, $s^{m+1}(N) \neq r(T)$). For this reason the completeness of searching for restricted clash refutations containing no tautologies is not lost by a corresponding search for binary refutations containing no tautologies. We shall ignore these complications in the following section.

<u>2.9 Hyper-resolution and P_1-resolution.</u>

A clash $\mathcal{C}$ is a <u>P_1-clash</u> and its resolvent C is a <u>P_1-resolvent</u> if $\mathcal{C}$ is binary and one parent of C is positive. The completeness of P_1-resolution follows from the completeness of hyper-resolution, for given any hyper-resolution derivation

$\mathcal{D}$ each hyper-resolution clash $\mathcal{C}$ in $\mathcal{D}$ can be replaced by a pseudo-clash $\hat{\mathcal{C}}$ and $\hat{\mathcal{C}}$ can be replaced by a P_1-derivation of the resolvent of $\mathcal{C}$.

Search strategies for hyper-resolution have the advantage over those for P_1-resolution that they avoid the redundancy involved in calculating the n! hyper-resolvent pseudo-clashes $\hat{\mathcal{C}}$ associated with a hyper-resolution clash $\mathcal{C}$ having n satellites. On the other hand search strategies for P_1-resolution have the advantage of computing hyper-resolvents incrementally and of saving the intermediate resolvents for resolution with possibly other positive satellites. More precisely if $\hat{\mathcal{C}} = (A_1,\ldots,A_n,B)$ is a hyper-resolution pseudo-clash with resolvent C and associated sequence of P_1-resolvents $C_1,\ldots,C_n = C$ then each C_i can be used as an intermediate resolvent for some hyper-resolution pseudo-clash $\hat{\mathcal{C}}' = (A_1,\ldots,A_i, A'_{i+1}, \ldots,A'_n,B)$ without recalculating for $\hat{\mathcal{C}}'$ the intermediate resolvents $C_1,\ldots,C_i$ already calculated for $\hat{\mathcal{C}}$. A second point in favour of P_1-resolution is that the problem of developing search strategies for binary resolution systems is much better understood than the corresponding problem for clashes of larger cardinality.

The following version of P_1-resolution incorporating the use of marked factors offers the advantage of both P_1-resolution and hyper-resolution without being subject to any of the disadvantages of either.

(1) An input positive clause is factored as a satellite clause.

(2) An input non-positive clause B is factored as a nucleus clause for hyper-resolution (i.e. generate all factors $B\theta$ of B where θ is an m.g.s.u. of a unifiable complete partition of all the negative literals in B). For each such factor B' of B choose a total ordering of the negative (i.e. distinguished) literals in B'.

(3) Resolve a positive factor on its single distinguished literal with a non-positive factor on its first remaining negative literal.

(4) Factor a positive resolvent as a satellite clause. Let a non-positive resolvent be its own trivial i-factor and let its first distinguished literal be the next remaining distinguished literal descending from its non-positive parent.

Clearly the device of choosing a unique total ordering of the negative literals in non-positive factors amounts to associating a unique pseudo-clash with every hyper-resolution clash. The generation of factors of positive clauses can be further restricted by choosing an α-ordering and then only generating those satellite factors compatible with the α-restriction. The method of decomposing hyper-resolution clashes outlined above can be extended to hyper-resolution after renaming without any complications. A system employing renaming, α-ordering and (1) - (4) has been suggested by Darlington [6] for application to information retrieval.

Darlington's system also incorporates Meltzer's device of using renaming to simulate set of support. [26]

2.10 Maximal Pseudo-clash Refutations.

Part (a) of Lemma 2.10.1 is used in the proof of Theorem 2.10.2 which asserts the existence of maximal pseudo-clash refutations. Part (b) is used in Chapter 3 to prove a permutation theorem for paramodulation refutations.

Lemma 2.10.1. Let $S \overset{\bullet}{\cup} \{B\}$ be an unsatisfiable set of ground clauses where $B = \{L_1,\ldots,L_n\}$ is not a tautology.

(ai) For $1 \leq i \leq n$ let $\hat{R}_i$ be the set of resolvents of restricted pseudo-clashes $\hat{\mathcal{C}} = (A_1,\ldots,A_i,B)$ on $\hat{B}_i = (L_1,\ldots,L_i)$ where $A_1,\ldots,A_i \in S$ are not tautologies.

(bi) For $1 \leq i \leq n$ let R_i be the set of resolvents of clashes $\mathcal{C} = \{A_1,\ldots,A_i,B\}$ where $A_1,\ldots,A_i \in S$ are not tautologies and the set of literals resolved upon in B is $\{L_1,\ldots,L_i\}$.

then

(aii) $\hat{S}_i = S \cup \hat{R}_i$ is unsatisfiable for all i, $1 \leq i \leq n$,

and

(bii) $S_i = S \cup R_i$ is unsatisfiable for all i, $1 \leq i \leq n$.

Proof. (a) By induction on i. Extend the definition of $\hat{R}_i$ and $\hat{S}_i$ to the case $i = 0$ by letting $\hat{R}_0 = \{B\}$. Then, since $\hat{S}_0 = S \cup \{B\}$, (aii) holds trivially for $i = 0$. Suppose that (aii) holds for a given i. Then we want to show that $\hat{S}_{i+1}$ is unsatisfiable assuming that $\hat{S}_i$ is unsatisfiable.

But $\hat{R}_{i+1}$ is the set of non-tautologous binary resolvents of clashes $\{A_{i+1}, C_i\}$ where $A_{i+1} \in S$ and $C_i \in \hat{R}_i$ are not tautologies and where L_{i+1} is the literal resolved upon in C_i. $\hat{S}_{i+1} = (\hat{S}_i - \hat{R}_i) \cup \hat{R}_{i+1}$. It suffices to construct a semantic tree which is closed for $\hat{S}_{i+1}$. Let $\mathcal{S} = (T, \mathcal{Q})$ be a binary semantic tree for the set of atoms in $\hat{S}_i$ ordered with L_{i+1} last. Then $\mathcal{S}$ is closed for S_i and any clause $C_i \in \hat{R}_i$ which fails in $\mathcal{S}$ fails properly only at a tip of T (since C_i contains L_{i+1} and $\overline{L}_{i+1} \notin \mathcal{Q}(N)$ if N is not a tip of T).

Suppose $\mathcal{S}$ is not closed for $\hat{S}_{i+1}$. Then some complete branch $\mathcal{B}_1$ of $\mathcal{S}$ does not contain a failure point for $\hat{S}_{i+1}$. But then some $C_i \in \hat{R}_i$ fails properly at the tip N_1 of $\mathcal{B}_1$, Let $N = s(N_1)$ and $s^{-1}(N) = \{N_1, N_2\}$. Then some $A_{i+1} \in S$ fails properly at N_2(since the complete branch $\mathcal{B}_2$ differing from $\mathcal{B}_1$ only in the tip N_1 contains a failure point for S). By 2.3.3. the resolvent $C \in \hat{R}_{i+1}$ of A_{i+1} and C_i fails at N on $\mathcal{B}_1$ contrary to assumption. So $\mathcal{S}$ is closed for $\hat{S}_{i+1}$ and $\hat{S}_{i+1}$ is unsatisfiable.

(b) $$R_i = \{C : C = (A_1 - \{\overline{L}_1\} \cup \dots \cup (A_i - \{\overline{L}_i\}) \cup (B - \{L_1, \dots, L_i\}), \text{ for } A_1, \dots, A_i \in S \text{ where } \overline{L}_1 \in A_1, \dots, \overline{L}_i \in A_i\}.$$

Let $\mathcal{S} = (T, \mathcal{Q})$ be a binary semantic tree for the set of atoms in $S_0 = S \dot{\cup} R$. Then $\mathcal{S}$ is closed for S_0. Suppose $\mathcal{S}$ is not closed for some S_i, $1 \leq i \leq n$. Then some complete branch $\mathcal{B}$ of $\mathcal{S}$ contains a failure point for S_0 but not for S_i.

Therefore B fails on $\mathcal{B}$ and $\{\overline{L}_1,\ldots,\overline{L}_i\} \subseteq \mathcal{A}(\mathcal{B})$. Let $\mathcal{B}_j$ be the complete branch of $\mathcal{S}$ which differs from $\mathcal{B}$ only in L_j, i.e. $\mathcal{A}(\mathcal{B}_j) = (\mathcal{A}(\mathcal{B}) - \{\overline{L}_j\}) \dot{\cup} \{L_j\}$, for $1 \leq j \leq i$. Then each $\mathcal{B}_j$ contains a failure point for S. Let $A_j \in S$ fail on $\mathcal{B}_j$. If $\overline{L}_j \notin A_j$ for some j then A_j fails on $\mathcal{B}$ already and so contrary to assumption $\mathcal{B}$ contains a failure point for S and therefore for S_i. Therefore $\overline{L}_j \in A_j$. Let

$$C = (A_1 - \{\overline{L}_1\}) \cup \ldots \cup (A_i - \{\overline{L}_i\}) \cup (B - \{L_1,\ldots,L_i\}) .$$

then $C \in R_i$ and C fails on $\mathcal{B}$ i.e. $\overline{C} \subseteq \mathcal{A}(\mathcal{B})$ since

$$\overline{A}_j \subseteq \mathcal{A}(\mathcal{B}_j) = (\mathcal{A}(\mathcal{B}) - \{\overline{L}_j\}) \dot{\cup} \{L_j\} \quad \text{and}$$
$$\overline{A_j - \{\overline{L}_j\}} \subseteq \mathcal{A}(\mathcal{B}) \quad \text{and}$$
$$\overline{B - \{L_1,\ldots,L_i\}} \subseteq \mathcal{A}(\mathcal{B}) .$$

Therefore $\mathcal{B}$ contains a failure point for S_i. It follows that $\mathcal{S}$ is closed for S_i and that S_i is unsatisfiable.

<u>Theorem 2.10.2.</u> If S is unsatisfiable then there exists a refutation $\mathcal{D} = (T,c)$ of S such that $\mathcal{D}$ is a pseudo-clash derivation and the sequence $\hat{B}$ of literals resolved upon in the nucleus B of any pseudo-clash $\hat{\mathcal{C}}$ in $\mathcal{D}$ contains all the literals in B. $\mathcal{D}$ contains no tautologies and every pseudo-clash in $\mathcal{D}$ is restricted.

<u>Proof.</u> By the contraction theorem it suffices to consider the case where S is a set of ground clauses. The proof is by induction on the total number n of distinct atoms contained in S. If n = 1 then $S \subseteq \{\{L\}, \{\overline{L}\}, \{L,\overline{L}\}\}$ for

some atom L. The derivation $\mathcal{D}$ consisting of the pseudo-clash $\hat{\mathcal{C}} = (\{L\}, \{\overline{L}\})$ with resolvent $\square$ satisfies the theorem. Suppose that $n>1$ and that the theorem holds for any unsatisfiable set containing fewer than a total of n distinct atoms.

Suppose that S contains only the distinct atoms $L_1,\ldots L_n$ Let $C_1,\ldots,C_m \in S$ be all the clauses in S containing the atom L_1 positively. Thus L_1 has exactly m distinct positive occurrences in S. Let $\hat{C}_i$ be a sequence of literals containing all and only the literals in C_i where L_1 occurs last in $\hat{C}_i$. Let $\hat{S}_0 = S$ and for $1 \leq i \leq m$ let $\hat{S}_i = (\hat{S}_{i-1} - \{C_i\}) \cup \hat{R}_i$ where $\hat{R}_i$ is the set of all non-tautologous resolvents of restricted pseudo-clashes with satellites in $\hat{S}_{i-1}$ and nucleus C_i where $\hat{C}_i$ is the sequence of literals resolved upon in C_i. Each $\hat{S}_i$ is unsatisfiable since $\hat{S}_0 = S$ is unsatisfiable and if $\hat{S}_{i-1}$ is unsatisfiable then so is $\hat{S}_i$, by 2.10.1, for $1 \leq i \leq m$. Notice that $\hat{S}_i$ contains m-i positive occurrences of L_1 because if $\hat{S}_{i-1}$ contains $m - i+1$ such occurrences, then $\hat{S}_{i-1} - \{C_i\}$ contains $m - i$ such occurrences, but $\hat{R}_i$ contains no positive occurrences of L_1 by virtue of the fact that L_1 is last in the sequence $\hat{C}_i$. Thus $\hat{S}_m$ contains no positive occurrences of L_1. Therefore, by the purity principle [39] some $S' \subseteq \hat{S}_m$ is unsatisfiable where the atom L_1 does not occur in S'.

By induction hypothesis there exists a refutation $\mathcal{D} = (T',c')$ of S' satisfying the theorem. By appending to each tip $N' \in T'$ where $c'(N') \notin S$ the derivation of $c'(N')$ from S associated with the construction above we obtain the desired refutation $\mathcal{D}$ of S.

The proposition analogous to Theorem 2.10.2 for clashes does not hold. If $S = \{ \{L_1,L_2\}, \{L_1,L_2\}, \{L_1, L_2\}, \{L_1,L_2\} \}$ then S is unsatisfiable but there exists no clash refutation $\mathcal{D}$ of S such that the set of literals resolved upon in the nucleus B of any clash $\mathcal{C}$ in $\mathcal{D}$ coincides with the set of all literals in B. No such refutation $\mathcal{D}$ of S exists even if $\mathcal{D}$ is allowed to contain unrestricted clashes and tautologies.

CHAPTER 3.

The use of equality axioms in resolution systems has seemed to be especially inefficient. In order to remedy this problem several modifications of resolution have been proposed (e.g. [3] , [6] , [28] ,[38] , [43], [48] , and [55]). Of these the paramodulation method of [38] seems to be particularly simple and efficient. In this chapter we compare paramodulation with hyper-resolution using axioms for equality. These two methods are first described in sections 3.1 and 3.2 and then compared in 3.3. A simple interpretation of hyper-resolution with equality axioms is found in the subsystem of paramodulation providing a straightforward proof for the completeness of this subsystem. In sections 3.4 and 3.5 modifications of the hyper-resolution method are proposed and these modifications are seen to induce corresponding modifications of the paramodulation method. A principal conclusion of this chapter is that systems designed especially to deal with equality need not be more efficient than existing resolution systems.

Chapter 3 is essentially [20] with only minor modifications.

3.1 Hyper-resolution with Equality Axioms.

Let S_0 be a set of clauses and let $E = E_1 \cup E_2 \cup E_3$ where

$$E_1 = \{\{ x = x \}\},$$

$$E_2 = \{\{ x_i \neq y_i,\ f(x_1,\ldots,x_i,\ldots,x_n) = f(x_1,\ldots,y_i,\ldots,x_n) \} : f \text{ in the vocabulary of } S_0 \text{ and } 1 \leq i \leq n\},$$

$$E_3 = \{\{ x_i \neq y_i,\ \overline{P}(x_1,\ldots,x_i,\ldots,x_n),\ P(x_1,\ldots,y_i,\ldots,x_n)\} : P \text{ the equality symbol or } P \text{ in the vocabulary of } S_0 \text{ and } 1 \leq i \leq n \}.$$

We write $s = t$ instead of $=(s,t)$ and $s \neq t$ instead of $\overline{=(s,t)}$. We adopt the convention that "s=t" is syntactically indistinguishable from "t=s". This convention allows us to simplify notation and in particular allows us to consider as tautologies clauses of the form $\{s \neq t\ ,\ t = s\}$.

If S_0 has no normal model (i.e. no model in which the equality symbol is interpreted as a substitutive identity relation) then $S = S_0 \cup E$ is unsatisfiable. Therefore there exists a hyper-resolution refutation $\mathcal{D}$ of S. This needs to be verified by appropriately modifying earlier definitions and proofs to accomodate the indistinguishability of "t=s" and "s=t". In this connection we note only that two equations $t_1=s_1$ and $t_2=s_2$ may have two non-equivalent m.g.s.us, i.e. one for $\mathcal{E}_1 = \{\{ t_1,t_2 \},\ \{ s_1,s_2\}\}$ and another for $\mathcal{E}_2 = \{\{t_1,s_2\}\ ,\ \{t_2,s_1\}\}$. In section 3.3 we shall compare hyper-resolution refutations $\mathcal{D}$ of S with paramodulation refutations of S_0.

The efficiency of obtaining the refutation $\mathcal{D}$ can probably be improved by imposing restrictions which have not been investigated for paramodulation (e.g. deletion of tautologies and subsumed clauses, α-ordering restrictions, various factoring methods, unique decomposition of restricted clashes, diagonal search and preprocessing procedures). In addition we note the following improvement for obtaining the hyper-resolution refutation $\mathcal{D}$ of $S = S_0 \cup E$. If S_0 results by eliminating quantifiers from a set of sentences S_0^* then it is unnecessary to include in $E_2 \subset E$ axioms for the Skolem function symbols introduced in obtaining S_0 from S_0^*. This follows directly from the fact that $S_0^* \cup E^*$ is unsatisfiable where $E^* = E_1 \cup E_2^* \cup E_3$ and E_2^* contains axioms only for the function symbols occurring in S_0^*.

3.2 Paramodulation.

Given a clause B and a single occurrence of a term t in B we write $B[t]$ to indicate the given occurrence of t in B. For ground clauses $A = \{t = s\} \dot{\cup} A_0$ and $B = B[t]$ paramodulant of A and B is the clause $C = B[s/t] \cup A_0$ where $B[s/t]$ indicates the result of replacing the single distinguished occurrence of t in $B[t]$ by s. Note that t may occur as a subterm of another term in B, both in its distinguished occurrence in $B[t]$ as well as in otner occurrences in B.

For the general case paramodulation is most conveniently defined in the context of refutation systems which include a separate rule for factoring clauses. For standardised factors

$A = \{t_1 = s\} \overset{\circ}{\cup} A_0$ and $B = B[t_2]$ where $\{t_1, t_2\}$ is unifiable with m.g.u. θ, a paramodulant of A and B is the clause

$$\begin{aligned} C &= A_0 \cup B\theta[s\theta/t_2\theta] \\ &= (A_0 \cup B[s/t_2])\theta . \end{aligned}$$

The factors A and B are respectively the first and second parents of C. We call the distinguished occurrence of t_2 in $B[t_2]$ the term paramodulated upon in B. (In a more precise terminology we would refer instead to "the distinguished occurrence of the term paramodulated upon.") The literal $t_1 = s_1$ in A and the literal in $B[t]$ containing the distinguished occurrence of t are both called the literals paramodulated upon.

An important case of paramodulation occurs when the occurrence of the term paramodulated upon in the second parent $B[t]$ does not occur as a proper subterm of another term in B. In this case the distinguished occurrence of t in $B[t]$ is said to be primary in $B[t]$ and the application of paramodulation is also primary. (The terminology here is borrowed from Sibert [48]). For example if $B = \{ f(c) \neq c\}$ then both the single occurrence of f(c) and the second occurrence of c in B are primary in B but the first occurrence of c is not.

A derivation $\mathfrak{D} = (T, c)$ is a p-derivation if $N \in T$ and $s^{-1}(N) = \{N_1, \ldots, N_n\} \neq \emptyset$ implies that either

(1) $c(s^{-1}(N))$ is a clash with resolvent $c(N)$ or

(2) $n=2$ and $c(N)$ is a paramodulant of factors of $c(N_1)$ and $c(N_2)$.

Notice that we do not explicitly exhibit factoring in either clash-derivations or p-derivations. A p-derivation $\mathfrak{D}$ is <u>binary</u> if all clashes in $\mathfrak{D}$ are binary.

Given a set of clauses S_0 let

$$E_4 = \{\{f(x_1,\ldots,x_n) = f(x_1,\ldots,x_n)\} : f \text{ in the vocabulary of } S_0,\ n \geq 0\} .$$

The following completeness theorem for paramodulation was reported by G. Robinson and Wos in [38] .

<u>Theorem 3.2.1.</u> If S_0 has no normal model then there exists a binary p-refutation of $S_0 \cup E_4$.

Robinson and Wos have also proposed the following unsolved

<u>Conjecture [38].</u> If S_0 has no normal model then there exists a binary p-refutation of $S_0 \cup E_1$.

<u>3.3 Comparison of the Paramodulation and Hyper-resolution Methods.</u>

The basis for our comparison rests upon the observation that most hyper-resolvents with nucleus parents in $E_2 \cup E_3$ can be interpreted as paramodulants. This same observation was noted independently by Chang in [4] . For later applications in section 3.5 it is useful to formulate this observation more generally for n-resolvents with nucleus parents in $E_2 \cup E_3$.

A clash $\mathfrak{C}$ is an <u>n-clash</u> and its resolvent is an <u>n-resolvent</u> if the set of literals resolved upon in the nucleus $B \in \mathfrak{C}$ coincides with the set of negative literals contained in B.

Thus any hyper-resolvent is an n-resolvent and any n-resolvent of a restricted n-clash all of whose satellite parents are positive, is a hyper-resolvent. Note that we do not require that n-clashes be restricted.

In the sequel in order to simplify terminology we shall often treat variants of the same clause as though they were identical. This convention allows us in the statement of 3.3.1 below to refer to "$B \in E_3$" instead of "B a variant of a clause in E_3".

Lemma 3.3.1. Let $\mathcal{C} = \{A_1,\ldots,A_n,B\}$ be an n-clash with nucleus B and n-resolvent C. Then

(1) if $B \in E_3$ and $n = 2$ then C is a primary paramodulant of factors of A_1 and A_2,

(2) if $B \in E_2$ then C is a paramodulant whose first parent is a factor of A_1 and second parent is an appropriate clause $B^* \in E_4$.
(If $B = \{x_i \neq y_i,\ f(x_1,\ldots,x_i,\ldots,x_n) = f(x_1,\ldots,y_i,\ldots,x_n)\}$
then $B^* = \{f(x_1,\ldots,x_n) = f(x_1,\ldots,x_n)\}$.)

Proof. (1) Let $\mathcal{F} = \{A_1', A_2', B'\}$ be the set of factors of clauses in $\mathcal{C}$ having C as resolvent. $B' = B$ since otherwise the two negative literals of B would be unified in B' implying that $\mathcal{C}$ has only one satellite in which case n would be 1. Let

$$A_1' = \{t = s\} \dot{\cup} A_{01},$$
$$A_2' = \{P(t_1,\dots,t_i,\dots,t_n)\} \dot{\cup} A_{02} \text{ and}$$
$$B' = \{x_i \neq y_i, \bar{P}(x_1,\dots,x_i,\dots,x_n), P(x_1,\dots,y_i,\dots,x_n)\}.$$

Then $C = (\{P(t_1,\dots,s,\dots,t_n)\} \cup A_{01} \cup A_{02})\theta$ where θ is an m.g.s.u. of $\{t, t_i\}$. C is therefore a primary paramodulant of A_1' and A_2'.

(2) Let $\mathcal{C}' = \{A', B\}$ be the appropriate set of factors of clauses in $\mathcal{C}$ having C as resolvent. $B' = B$.

Let

$$A' = \{t = s\} \dot{\cup} A_0 \quad \text{and}$$
$$B' = \{x_i \neq y_i, f(x_1,\dots,x_i,\dots,x_n) = f(x_1,\dots,y_i,\dots,x_n)\}.$$

Then $C = \{f(x_1,\dots,t,\dots,x_n = f(x_1,\dots,s,\dots,x_n)\} \cup A_0$ and C is a paramodulant of A' and of

$$B^* = \{f(x_1,\dots,x_n) = f(x_1,\dots,x_n)\} \in E_4.$$

Lemma 3.3.1 implies that any hyper-resolution derviation $\mathcal{D}$ of a clause C from clauses $S_0 \cup E \cup E_4$ can be transformed into a p-derivation of C from $S_0 \cup E_1 \cup E_4$, provided that every clash in $\mathcal{D}$ with nucleus $B \in E_3$ has exactly two satellites. Furthermore, in order to obtain the application of section 3.4 to trivilization of inequalities, it is desirable that the clause $\{x = x\}$ does not occur as parent of a paramodulant in the resulting p-derivation. These two desiderata motivate the following definition and lemma.

A clash derivation $\mathcal{D}$ is _normal_ if whenever a clause

$B \in E_2 \cup E_3$ occurs in $\mathcal{D}$

(1) B occurs as nucleus of an n-clash $\mathcal{C}$,

(2) $\mathcal{C} \cap (E_1 \cup E_4) = \emptyset$ and

(3) If $B \in E_3$ then $\mathcal{C}$ contains exactly two satellites.

Theorem 3.3.3 below states that any normal derivation from clauses $S_0 \cup E \cup E_4$ can be transformed into a p-derivation from $S_0 \cup E_1 \cup E_4$. Lemma 3.3.2 both guarantees the existence of normal derivations and also serves as a lemma in the proof of 3.4.2.

Lemma 3.3.2. Let $S = S_0 \cup E'$, where $E' = E$ or $E' = E_2 \cup E_3 \cup E_4$, be unsatisfiable. Then there exists a finite unsatisfiable set S' of ground instances of clauses in S such that if a clash derivation $\mathcal{D} = (T, c)$ from S results from lifting a clash derivation $\mathcal{D}' = (T, c')$ from S' (i.e. $c'(N)$ is an instance of $c(N)$ for all $N \in T$) then for every clash $\mathcal{C}$ in $\mathcal{D}$

(1) if the nucleus of $\mathcal{C}$ is in $E_2 \cup E_3$ then $C \cap (E_1 \cup E_4) = \emptyset$ and

(2) if $\mathcal{C}$ is an n-clash with nucleus $B \in E_3$ then $\mathcal{C}$ contains exactly two satellites.

Proof. Choose S' containing no tautologies. In addition let S' contain no instances of clauses in E_2 of the form

(*) $\{t_i \neq t_i, \; f(t_1, \ldots, t_i, \ldots, t_n) = f(t_1, \ldots, t_i, \ldots, t_n)\}$.

S' may so be chosen because any instance of a clause in E_2 of form (*) is subsumed by the corresponding instance $\{f(t_1, \ldots, t_n) = f(t_1, \ldots, t_n)\}$ of a clause in E_1 or E_4.

Let $\mathcal{D}'$ and $\mathcal{D}$ be derivations from S' and S respectively where $\mathcal{D}$ lifts $\mathcal{D}'$. Suppose that (1) is violated. Then there are clauses A, B $\in \mathcal{C}$ where $A \in E_1 \cup E_4$ and $B \in E_2 \cup E_3$ is nucleus of $\mathcal{C}$ which is some clash in $\mathcal{D}$. If $B \in E_2$ then, since A is positive, the negative literal in B is resolved upon in $\mathcal{C}$ and the corresponding instance B' of B in $\mathcal{D}'$ is of the form (*). If $B \in E_3$ then the first or second negative literal in B is resolved upon in $\mathcal{C}$ and the corresponding instance B' of B in $\mathcal{D}'$ is a tautology.

If (2) is violated and $\mathcal{C}$ is an n-clash in $\mathcal{D}$ with nucleus $B \in E_3$ and only one satellite then the corresponding instance B' of B in $\mathcal{D}'$ is a tautology of the form $\{s \neq t, s=t\}$.

Lemma 3.3.2 guarantees the existence of normal refutations $\mathcal{D}$ of unsatisfiable $S = S_0 \cup E'$. For if $\mathcal{D}'$ is a hyper-resolution refutation of S' then $\mathcal{D}$ is normal if $\mathcal{D}$ lifts $\mathcal{D}'$ and S' is the finite set of instances of clauses in S, asserted to exist by 3.3.2.

Given a normal derivation $\mathcal{D}$ we denote the corresponding p-derivation by $\phi(\mathcal{D})$. Let $\mathcal{D} = (T,c)$ be a normal derivation from $S_0 \cup E'$ where $E' = E$ or $E' = E_2 \cup E_3 \cup E_4$. Then $\phi(\mathcal{D}) = (T',c')$ is defined as follows:

(1) $T' \subseteq T$,

(2) $N \in T-T'$ if and only if $c(N) \in E_3$ and

(3) for $N \in T'$

(a) if $c(N) \in E_2$ then $c'(N) \in E_4$

(i.e. if $c(N) = \{ x_i \neq y_i, f(x_1,\ldots,x_i,\ldots,x_n) = f(x_1,\ldots,y_i,\ldots,x_n) \}$
then $c'(N) = \{ f(x_1,\ldots,x_n) = f(x_1,\ldots,x_n) \}$)
and

(b) $c'(N) = c(N)$ otherwise.

Theorem 3.3.3. Let $\mathfrak{D}$ be a normal derivation of a clause C from $S_0 \cup E'$ where $E' = E$ or $E' = E_2 \cup E_3 \cup E_4$. Then $\phi(\mathfrak{D})$ is a p-derivation from $S_0 \cup E''$ where $E'' = E_1 \cup E_4$ if $E' = E$ and $E'' = E_4$ if $E' = E_2 \cup E_3 \cup E_4$.

Proof. 3.3.3 is a direct consequence of Lemma 3.3.1.

Theorem 3.3.4. If S_0 has no normal model then there exists a p-refutation $\mathfrak{D}$ of $S_0 \cup E_4$ such that:

(r1) Both parents of every paramodulant in $\mathfrak{D}$ are positive.

(r2) Every resolvent in $\mathfrak{D}$ is a hyper-resolvent.

(r3) All applications of paramodulation in $\mathfrak{D}$ are primary except when the second parent is in E_4. Moreover no clause in E_4 is first parent of a paramodulant in $\mathfrak{D}$.

(r4) Given any α-ordering for $S_0 \cup E_4$ all parents of paramodulants and satellite parents of hyper-resolvents in $\mathfrak{D}$ satisfy the α-restriction.

Proof. Since S_0 has no normal model, $S_0 \cup E_2 \cup E_3 \cup E_4$ is unsatisfiable. Let $\leq_\alpha$ be an α-ordering for $S_0 \cup E_4$ and therefore for S as well. Let $\mathfrak{D}$ be a hyper-resolution refutation of S satisfying the α-restriction for satellites in $\mathfrak{D}_0$. Moreover let $\mathfrak{D}_0$ be obtained by lifting a ground refutation of a set S' of ground instances of clauses in S, where S' satisfies the conditions of Lemma 3.3.2. Then $\mathcal{D}_0$ is normal and $\phi(\mathcal{D}_0) = \mathcal{D}$ is a p-refutation of $S_0 \cup E_4$. That $\mathcal{D}$ satisfies (r1), (r2) and (r4) follows directly from the definition of ϕ. The first part of (r3) follows from Lemma 3.3.1 and the second part of (r3) follows from Lemma 3.3.2.

Note that Theorem 3.3.4 implies Theorem 3.2.1 because all applications of hyper-resolution can be replaced by sequences of P_1-resolutions.

Theorem 3.3.5. If S_0 has no normal model then there exists a p-refutation $\mathcal{D}$ of $S_0 \cup E_1 \cup E_4$ such that

(r1) - (r4) and

(r5) The clause $\{x = x\}$ does not occur in $\mathcal{D}$ as parent of a paramodulant and clauses in E_4 occur in $\mathcal{D}$ only as second parents of paramodulants.

Proof. The proof of 3.3.5 is identical to that of 3.3.4 except that we let $S = S_0 \cup E$. Restriction (r5) then follows from Lemma 3.3.2 and the definition of ϕ.

Theorems 3.3.4 and 3.3.5 establish the most direct of our comparisons between paramodulation and hyper-resolution with equality axioms. These comparisons are refined further in the next two sections. We note first that 3.3.4 and 3.3.5 already establish the compatibility with the paramodulation method of the deletion of tautologies and subsumed clauses. (Since these deletion rules are compatible with hyper-resolution refutations $\mathcal{D}$ they are also compatible with p-refutations $\phi(\mathcal{D})$.)

It is also possible to impose minimality restrictions on p-refutations $\phi(\mathfrak{D})$ corresponding to the minimality restrictions imposable on hyper-resolution derivations. Finally the compatability of renaming for hyper-resolution implies the same for paramodulation provided that the equality symbol is not enamed.

3.4 Trivialization of Inequalities.

Resolving a factor $C = \{s \neq t\} \dot{\cup} C_0$ with $\{x = x\}$ produces the clause $C_0\theta$ where θ is an m.g.u. of $\{s,t\}$. We call such an application of resolution the operation of trivializing an inequality (compare [3] and [48]).

Corollary 3.7.2 of Theorem 3.4.1 implies that if S_0 has no normal model then S_0 may be effectively preprocessed by trivializing inequalities in clauses of S_0 obtaining a set of clauses S_0^* such that $S^* = S_0^* \cup E_2 \cup E_3$ is unsatisifiable. S_0^* consists of S_0 together with all resolvents of clashes $\mathfrak{C} = \{A_1,\ldots,A_n,B\}$ where $B \in S_0$ and $A_i = \{x=x\}$ for all i, $1 \leq i \leq n$ (simultaneous trivialization of inequalities). Clearly the number of applications of resolution involved in obtaining S_0^* from S_0 is finite and therefore S_0^* can effectively be obtained from S_0.

If $\mathfrak{D}$ is a normal hyper-resolution refutation of S_0^* then $\phi(\mathfrak{D})$ is a p-refutation of $S_0^* \cup E_4$ such that (r1) - (r5) of Theorems 3.3.4 and 3.3.5 hold for $\phi(\mathfrak{D})$, with the obvious strengthening of (r5) that $\{x=x\}$ does not occur in $\phi(\mathfrak{D})$. If we enlarge $\phi(\mathfrak{D})$ so as to exhibit trivializations of inequalities then we obtain a p-refutation $\mathfrak{D}'$ of $S_0^* \cup E_1 \cup E_4$ which may differ from the p-refutation of 3.3.5 in that (r2) may fail to hold for these applications of resolution which trivialize inequalities. On the other hand all trivializations of inequalities in $\mathfrak{D}'$ occur before all other applications of resolution and before all applications of paramodulation in $\mathfrak{D}'$, whereas no such ordering of these operations need occur in the p-refutation asserted to exist by 3.3.5.

Theorem 3.4.1 implies more generally that satisfiable sets S_1 of unit clauses may be effectively preprocessed out of a set of clauses $S = S_0 \cup S_1$ before attempting to find a refutation of the resulting set S_0^*. Our intuition is that such preprocessing is likely to increase the efficiency of obtaining proofs of more difficult theorems. The figure below gives a simple example of two derivations of the same clause. Only the first derivation will be generated if the original set S_0 is preprocessed. If the entire set S_0^* must be generated before attempting to find a refutation then this method of preprocessing may be inefficient for proving theorems which have a simple proof which can be detected for instance with less effort than that involved in generating all of S_0^* itself. On the other hand since resolving a clause $A \in S_0$ with a unit clause in S_1 produces a clause containing fewer literals than are contained in A we may expect that this preprocessing procedure will tend to retain the simplest of th se derivations which differ by permuting occurrences of clauses from S_1 along their branches. Finally even for the case of simpler theorems preprocessing can be made more efficient by simultaneously generating S_0^* and generating resolvents from S_0^* before completing the generation of S_0^*. (Such a procedure would be similar to the diagonal search strategy of Chapter 4.)

Example.

$$\begin{tikzcd}
\{L_1,L_2\} \in S_0 \arrow[dr, no head] & \{\overline{L}_1\} \in S_1 \arrow[d, no head] \\
\{\overline{L}_2,L_3\} \in S_0 \arrow[dr, no head] & \{L_2\} \in S_0^* \arrow[d, no head] \\
\{\overline{L}_3,L_4\} \in S_0 \arrow[dr, no head] & \{L_3\} \arrow[d, no head] \\
 & \{L_4\}
\end{tikzcd}
\qquad
\begin{tikzcd}
\{L_1,L_2\} \in S_0 \arrow[dr, no head] & \{\overline{L}_2,L_3\} \in S_0 \arrow[d, no head] \\
\{\overline{L}_3,L_4\} \in S_0 \arrow[dr, no head] & \{L_2,L_3\} \arrow[d, no head] \\
\{\overline{L}_1\} \in S_1 \arrow[dr, no head] & \{L_1, L_4\} \arrow[d, no head] \\
 & \{L_4\}
\end{tikzcd}$$

Notice that the redundancy exemplified here can not be removed by implementing singly connectedness [44] and is not removed by hyper-resolution. Notice also that in this case the eliminated proof involved resolvents of length 2 and is therefore more complicated than the first proof in the sense that it is generated after the first proof by diagonal search (Chapter 4.)

Theorem 3.41. Let $S = S_0 \cup S_1$ be unsatisfiable where S_1 is a satisfiable set of unit clauses. Then the set

$S_0^* = S_0 \cup R$ is unsatisfiable where R is the set of resolvents of clashes with nucleus in S_0 and satellites in S_1.

<u>Proof</u>. Assume first that S is a set of ground clauses. Let $S_1 = \{\{L_1\},\dots,\{L_n\}\}$. We prove the theorem for this case showing by induction that, for all $k \leq n$,

$$U_k = S_0 \cup R_k \cup (S_1 - \{\{L_1\},\dots,\{L_k\}\})$$

is unsatisfiable where R_k is the set of resolvents of clashes with nucleus in S_0 and satellites in $\{\{L_1\},\dots,\{L_k\}\}$. U_0 is unsatisfiable since $U_0 = S$.

Suppose that U_k is unsatisfiable for some given k. By Lemma 2.10.1 (a) or (b), $U = (U_k - \{L_{k+1}\}) \cup R'$ is unsatisfiable where R' is the set of binary resolvents C of clashes $\mathcal{C}$ with nucleus $\{L_{k+1}\}$ and satellites A in $U_k - \{L_{k+1}\}$. But then $A \notin S_1$ (since S_1 is satisfiable). So $A \in S_0 \cup R_k$ and therefore $C \in R_{k+1}$. But then $R' \subseteq R_{k+1}$ and since $R_k \subseteq R_{k+1}$, $U_{k+1} = U$ is unsatisfiable. It follows therefore that $U_n = S_0 \cup R = S_0^*$ is unsatisfiable.

If S is not a set of ground clauses let $S' = S_0' \cup S_1'$ be a finite set of ground instances of clauses in S where S_0' and S_1' are instances of clauses in S_0 and S_1 respectively. Then $S_0' \cup R''$ is unsatisfiable where R'' is the set of resolvents of clashes with nucleus in S_0' and satellites in S_1'. But by the contraction theorem $S_0' \cup R'$ is a set of ground instances of clauses in $S_0 \cup R$ which is therefore unsatisfiable.

<u>Corollary 3.4.2</u>. Let $S = S_0 \cup E$ be unsatisfiable. Then $S^* = S_0 \cup R \cup E_2 \cup E_3$ is unsatisfiable where R is the set of resolvents of clashes with nucleus in S_0 and satellites in E_1.

Proof. Let S' be a finite unsatisfiable set of ground instances of clauses in S satisfying Lemma 3.3.2. Take $S_0 \cup E_2 \cup E_3$ to be the S_0 of Theorem 3.41. Then $S_0 \cup E_2 \cup E_3 \cup R_0$ is unsatisfiable where each C in R_0 is the resolvent of a clash $\mathcal{C}$ having nucleus in $S_0 \cup E_2 \cup E_3$ and satellites $\{x = x\}$. But each C $\in R_0$ is obtained in 3.4.1 by lifting a ground clash $\mathcal{C}' \subseteq S'$. If $\mathcal{D}$ is the derivation of C from $\mathcal{C}$ and $\mathcal{D}'$ the derivation of an instance of C from $\mathcal{C}'$ then it follows by 3.3.2 that $\mathcal{D}$ is normal and therefore that the nucleus parent of C is not in $E_2 \cup E_3$, i.e. $R_0 = R$.

3.5 Permutation of Inferences.

Theorem 3.41 is a permutation theorem in the sense that it states that certain clashes may be permuted toward the tips of a derivation. Theorem 3.51 and its corollary are permutation theorems in the same sense. Corollary 3.5.2 implies that applications of paramodulation may be made to occur before applications of resolution in a p-refutation. Together 3.4.2 and 3.5.2 imply that a p-refutation may be obtained either in the form where trivializations of inequalities precede paramodulations which in turn precede other resolutions or in the form where paramodulations precede trivializations which precede other resolutions. For a p-refutation in either of these forms we may insist that (r3) and (r5) still hold ((r5) suitably modified as for the p-refutation corresponding to Corollary 3.4.2). Restriction (r1) must be weakened to assert only that literals paramodulated upon are positive. Restrictions (r2) and (r4) need to be modified to assert that resolutions which do not trivialize inequalities can be applications of any fixed complete resolution rule (e.g. set of support, M-clash resolution, binary α-restricted clash resolution etc.). The ordering restriction of Corollary 3.5.2 can be effectively implemented by insisting that no resolvent be the parent of a paramodulant. On the other hand 3.5.2 unlike 3.4.2 does not imply that the initial set S_0 can be effectively preprocessedd by applying paramodulation to obtain an unsatisfiable set S_0^*.

A theorem similar to 3.5.2 can be obtained by analyzing the abstract of the Robinson - Wos completeness proof for paramodulation, [37] and [56] . It was in fact this observation which motivated the discovery of Theorem 3.5.1 and its corollary. Unlike the case of obtaining the p-refutations corresponding to 3.4.2 we do not have any intuition on the efficiency of finding the p-refutations corresponding to 3.5.2.

Given $S = S_0 \dot{\cup} S_1$ where S_1 is a set of non-positive clauses, let $\mathcal{D}$ be a clash derivation of a clause C from S such that every clash in $\mathcal{D}$ is an n-clash with nucleus in S_1 and satellites not in S_1. Then C is said to be an <u>S_1-resolvent from S_0</u> and the derivation $\mathcal{D}$ is said to be <u>associated</u> with C. If $S_1 = E_2 \cup E_3$ and if $\mathcal{D}$ is a normal derivation associated with an S_1-resolvent C from S_0, then

$\phi(\mathfrak{D})$ is a p-derivation of C from S_0 containing no application of clash resolution.

<u>Theorem 3.5.1.</u> Let $S = S_0 \dot{\cup} S_1$ be unsatisfiable where S_1 is a set of non-positive clauses. Then some finite set S* of S_1-resolvents from S_0 is unsatisfiable.

<u>Corollary 3.5.2.</u> Let $S = S_0 \dot{\cup} E_2 \dot{\cup} E_3$ be unsatisfiable. Then some finite set S* of clauses derivable by paramodulation with resolution from $S_0 \cup E_4$ is unsatisfiable.

<u>Proof of 3.5.2.</u> Let $E_2 \cup E_3$ be the S_1 of 3.5.1 and let S* be the resulting finite unsatisfiable set of S_1-resolvents from S_0. As in the proof of 3.4.2 we may choose a set S' of ground instances of clauses in S such that each derivation $\mathfrak{D}_C$ associated with $C \in S^*$ is normal. By 3.3.3 each $\phi(\mathfrak{D}_C)$ is a p-derivation and since $\mathfrak{D}_C$ contains no clash with nucleus not in $E_2 \cup E_3$, $\phi(\mathfrak{D}_C)$ contains no application of clash resolution.

The proof of Theorem 3.5.1 requires the following lemma.

<u>Lemma 3.5.3.</u> Let $\mathfrak{D}$ be a hyper-resolution derivation of a positive ground clause C from ground clauses $S \dot{\cup} \{D\}$ where D is non-positive and occurs in $\mathfrak{D}$ only at the nucleus node immediately above the root. Suppose that $\mathfrak{D}$ contains no tautologies. Then there is a hyper-resolution derivation $\mathfrak{D}'$ of a clause $C' \subseteq C$ from clauses $S \cup R$ where R is the set of n-resolvents of n-clashes with satellites in S and nucleus D. $\mathfrak{D}'$ contains no tautologies.

<u>Proof of 3.5.3.</u> Let $D = \{\overline{L}_1,\ldots,\overline{L}_n\} \dot{\cup} D_0$ where D_0 is positive and each L_i is negative. Let $C = \{K_1,\ldots,K_m\}$ and $\neg C = \{\{\overline{K}_1\},\ldots,\{\overline{K}_m\}\}$. Notice that $|L_i| \neq |K_j|$ for all i, j where $1 \leq i \leq n$, $1 \leq j \leq m$, since the clash at the root of $\mathfrak{D}$ is restricted. $S \cup \{D\} \cup \neg C$ is unsatisfiable because $S \dot{\cup} \{D\}$ implies C. By Lemma 2.10.1 (b), $S \cup R' \cup \neg C$ is unsatisfiable where R' is the set of n-clash resolvents with nucleus D and satellites in $S \cup \neg C$. But R' = R since no n-clash with nucleus D has a clause $\{\overline{K}_j\}$ as satellite. Therefore $S \cup R \cup \neg C$ is unsatisfiable and $S \cup R$ implies C. By Theorem 2.5.1, because C is positive, there exists a hyper-resolution derivation $\mathfrak{D}'$ from $S \cup R$ of a clause C' which subsumes C, i.e. $C' \subseteq C$. $\mathfrak{D}'$ contains no tautologies.

<u>Proof of 3.5.1.</u> As in the proof of 3.4.1 it suffices to consider the case where S is a set of ground clauses. Let $\mathfrak{D} = (T,c)$ be a hyper-resolution refutation of S containing no tautologies. The proof is by induction on the number n of clashes in $\mathfrak{D}$ having a clause in S_1 as nucleus. Recall that each nucleus node of $\mathfrak{D}$ is a tip of T. If n = 0 then $\mathfrak{D}$ is a refutation of S_0 and S_0 is unsatisfiable. But S_0 is trivially a set of S_1-resolvents from S_0.

Suppose that n > 0 and that the theorem holds for any hyper-resolution refutation $\mathfrak{D}'$ of a set $S_0' \dot{\cup} S_1$ whenever $\mathfrak{D}'$ contains no tautologies and the number of occurrences of clauses in S_1 at nucleus nodes is less than n. Let $N \in T$

be such that for all $N' \in T_N$, $c(N') \in S_1$ if and only if N' is the nucleus node of $s^{-1}(N)$. In other words choose N such that if $\mathcal{D}_N = (T_N, c)$ then a clause $D \in S_1$ occurs in $\mathcal{D}_N$ only at the nucleus node lying immediately above the root of $\mathcal{D}_N$. Then $\mathcal{D}_N$ is a hyper-resolution derivation from $S_0 \dot{\cup} \{D\}$ of a positive clause $C = c(N)$ and $\mathcal{D}_N$ contains no tautologies. It follows by Lemma 3.5.3, that there exists a hyper-resolution derivation $\mathcal{D}_0$ of a clause $C' \subseteq C$ from $S_0 \dot{\cup} R_0$ where R_0 is a set of $\{D\}$-resolvents from S_0, i.e. R_0 is a set of S_1-resolvents from S_0. $\mathcal{D}_0$ contains no tautologies.

Let $\mathcal{D}_1$ be the subderivation of $\mathcal{D}$ obtained by ignoring all of $\mathcal{D}_N$ except for the root N of T_N. Then $\mathcal{D}_1$ is a hyper-resolution refutation containing no tautologies of the set $(S_0 \dot{\cup} S_1) \cup C$ and $\mathcal{D}_1$ contains fewer than n occurrences of clauses from S_1 at its tips. Let $\mathcal{D}_2$ be obtained from $\mathcal{D}_1$ by applying the contraction theorem to $\mathcal{D}_1$, associating with the node N in $\mathcal{D}_1$ the clause $A_N = C' \subseteq C$ and otherwise associating to every other tip N' in $\mathcal{D}_1$ the clause $A_{N'} = c(N')$. Then $\mathcal{D}_2$ is a contraction of $\mathcal{D}_1$ and is a hyper-resolution refutation of $(S_0 \dot{\cup} S_1) \cup C'$. $\mathcal{D}_2 = (T_2, c_2)$ contains no tautologies, fewer than n occurrences of clauses from S_1 at its tips and one node N_0, corresponding to N, such that $c_2(N_0) = C'$.

Let $\mathcal{D}'$ be obtained by identifying the root of $\mathcal{D}_0$ with the tip N_0 of $\mathcal{D}_2$. Then $\mathcal{D}'$ is a hyper-resolution refutation of $(S_0 \dot{\cup} S_1) \cup R_0$ containing no tautologies and fewer than n occurrences of clauses from S_1 at its tips. By induction hypothesis there exists an unsatisfiable set R of S_1-resolvents from $S_0 \cup R_0$. But since R_0 is already a set of S_1-resolvents from S_0, R is as well.

CHAPTER 4

The first half of this chapter (sections 4.1 - 4.5) extends the Hart-Nilsson-Raphael [16] and Pohl [32] theories of heuristic search to the case of theorem-proving graphs and theorem-proving problems. In particular the admissibility and optimality theorems of [16] are generalized for the classes $\mathfrak{D}$ and $\mathfrak{D}^u$ of diagonal search strategies for abstract theorem-proving problems. Both ordinary tree (or graph) searching problems [8] , [32] , and resolution problems are shown to be special cases of the more general notion of abstract theorem-proving problems with non-negative costs. In section 4.4 concrete algorithms are discussed for applying diagonal search strategies to theorem-proving by resolution.

The last two sections of this chapter contain an investigation of two complete factoring methods. The first method, when applied to hyper-resolution, amounts to never unifying negative literals. The second method, m-factoring, is shown to be always more efficient than the Wos-Robinson method.

The material in sections 4.1 - 4.5 is nearly identical to that reported in [21] . The completeness result of section 4.6 was announced without proof in [17] for the special case of hyper-resolution systems.

4.1 Theorem-Proving Graphs.

In the theorem-proving problem we begin with an initial non-empty set of sentences S_0 and with a set of inference rules Γ.

If $\varphi \in \Gamma$ and S is a set of sentences then $\varphi(S)$ is another set of sentences. $\varphi(S) = \emptyset$ if φ is not applicable to S. In particular $\varphi(S) = \emptyset$ if S is not finite. In applications to resolution systems, S_0 is a set of clauses and Γ consists of a single resolution rule or of a factoring rule and a separate rule for resolving factors. If φ is binary resolution of factors then $\varphi(S) = S' \neq \emptyset$ if S contains two factors which resolve or one factor which resolves with itself and each $C' \in S'$ is a resolvent of the clauses in S. If φ is the operation of unifying literals in a single clause (the Wos-Robinson method of factoring [53]) then $\varphi(S) = S' \neq \emptyset$ if S is a singleton $S = \{C\}$ and each $C' \in S'$ is a factor of C.

Given an initial set of sentences S_0 and a set of inference rules Γ let S^* be the set of all sentences which can be derived from S_0 by iterated application of the rules in Γ. Each sentence $C \in S^*$ can be assigned a level: if $C \in S_0$ then the level of C is zero, otherwise $C \in \varphi(S)$ for some $\varphi \in \Gamma$ and for some $S \subseteq S^*$ and the level of C is one greater than the maximum of the levels of the sentences $D \in S$. If S_i is the set of all sentences of level i then $S^* = \bigcup_{0 \leq i} S_i$. Since a sentence $C \in S^*$ may have several distinct derivations, the level of C need not be unique. Since $\varphi(S) \neq \emptyset$ only if S is finite, the set of sentences which occur in a given derivation of a sentence $C \in S^*$ is always finite. The theorem-proving problem for a triple (S_0, Γ, F), $F \subseteq S^*$, is that of generating by means of a search strategy Σ some $C^* \in F$ by iterated application of the rules in Γ beginning with the sentences in S_0. For certain applications it may be required to derive a sentence $C^* \in F$ having minimum level in F or, more generally, having minimum cost in F, where cost is determined by some "costing function" defined on the sentences in S*. The tree (or graph) searching problem [8] , [32] can be interpreted as a theorem-proving problem (S_0, Γ, F) where each operator $\varphi \in \Gamma$ has the property that $\varphi(S) = \emptyset$ whenever S is not a singleton.

A triple (S_0, Γ, F) determines a directed graph whose nodes are single sentences $C \in S^*$. C' is an immediate successor of C (i.e. C' is connected to C by an arc directed from C to C') if for some $S \subseteq S^*$ and $\varphi \in \Gamma$, $C \in S$ and $C' \in \varphi(S)$. The situation is similar to that which exists for ordinary graph searching problems as distinguished from tree searching problems. Searching in a directed graph for a path from a node a to a node b can be interpreted as searching in a directed labelled tree for a path from a node N_1, with label $c(N_1) = a$, to a node N_2, with label $c(N_2) = b$. The tree search interpretation of graph searching has the property of representing a single node d in a graph as distinct nodes $N_1, \ldots, N_k$ in a tree when the node d can be generated in k different ways as the end node of k different paths from the initial node a. This property of the tree search representation is one which we find useful when extended to deal with the more general theorem-proving problem. In particular the extended tree search representation associates distinct nodes with distinct derivations. This 1 - 1 correspondence between nodes and derivations allows the number of nodes generated by a search strategy Σ in the course of obtaining a terminal node to be treated as a measure of the efficiency of Σ for the given problem.

We define the notion of an <u>abstract theorem-proving graph</u> ("abstract graph" or simply "graph") (G,s). The extended tree representation of an <u>interpreted theorem-proving graph</u> (S_0, Γ) can be obtained from (G,s) by labelling the nodes $N \in G$ by use of a labelling function $c:G \to S^*$, and by interpreting each application of the function s to a subset $G' \subseteq G$ as an application of a function $\varphi \in \Gamma$ to the subset $\{c(N): N \in G'\}$. An abstract theorem-proving graph is a pair (G,s) where G is a set of nodes, $s:2^G \to 2^G$ is a successor function defined on subsets of G taking subsets of G as values. G and s satisfy the following conditions:

(1) $s(\emptyset) = \emptyset$.

(2) $s(G') \neq \emptyset$ implies that G' is finite.

(3) $G' \neq G''$ implies that $s(G') \cap s(G'') = \emptyset$.

(4) Let $\mathcal{S}_0 = \{N \in G : N \notin s(G') \text{ for any } G' \subseteq G\}$,

let $\mathcal{S}_{k+1} = \{N \in G : N \in s(G') \text{ for some}$

$$G' \subseteq \bigcup_{i \leq k} \mathcal{S}_i,\ G' \cap \mathcal{S}_k \neq \emptyset\}.$$

Then (a) $\mathcal{S}_0 \neq \emptyset$,

(b) $G = \bigcup_{0 \leq i} \mathcal{S}_i$,

(c) $\mathcal{S}_i \cap \mathcal{S}_j = \emptyset$ for $i \neq j$.

The graph (G,s) reduces to an ordinary tree if $s(G') \neq \emptyset$ implies that G' is a singleton. For this case condition (3) states that distinct nodes have distinct sets of successors. More generally, (3) states that distinct sets of nodes have distinct sets of successors. It is precisely this condition which ensures that the graphs (G,s) extend the ordinary tree representation of search spaces. Condition (5) states that (G,s) is a levelled acyclic directed graph. In other words each $N \in G$ can be assigned a unique <u>level</u> i where $N \in \mathcal{S}_i$ and $N \notin \mathcal{S}_j$ for all $j \neq i$. If (S_0, Γ) is an interpretation of (G,s) with labelling function $c : G \to S^*$ then $S_i = \{c(N) : N \in \mathcal{S}_i\}$ is just the set of labelled nodes of level i. Condition (3) guarantees that for each $C \in S^*$ and for each distinct derivation of C from S_0 there is a distinct node $N \in G$ such that $C = c(N)$. There is no restriction that $\mathcal{S}_0$ or S_0 be finite. The case where $\mathcal{S}_0$ is infinite allows us to deal with axiom schemes in theorem-proving and more generally with potentially infinite sets of initial nodes $\mathcal{S}_0$.

The successor function s of (G,s) determines a partial ordering of the nodes in G: N' is an <u>immediate successor</u> of N (and N an <u>immediate ancestor</u> of N') if

$$N' \in s(G') \text{ and } N \in G' \text{ for some } G' \subseteq G.$$

A node N' is a <u>successor</u> of N (and N an <u>ancestor</u> of N'), written $N' > N$,

if N' is an immediate successor of N or

if N' is a successor of an immediate successor of N.

we write $N \leq N'$ if $N < N'$ or $N = N'$. The definition of (G,s) guarantees that for all $N \in G$ the set $\{N' : N' \geq N\}$ is finite, although the set $\{N' : N' \geq N\}$ may be infinite. Notice that in the theorem-proving interpretation of graphs (G,s), a derivation of a sentence $c(N)$ consists of all the sentences $c(N')$ where $N' \leq N$. Each such derivation contains only finitely many sentences $c(N')$.

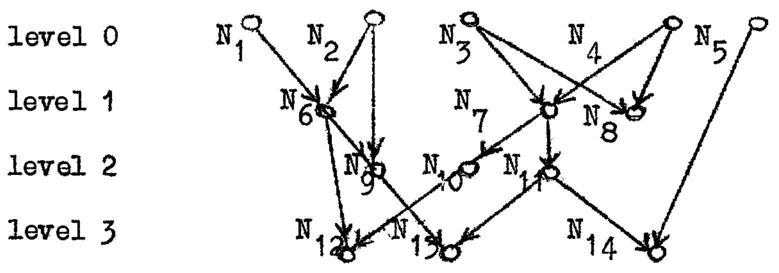


Figure 1.

Figure 1 illustrates a graph (G,s) where nodes are represented as points and where points N and N' are connected by a directed line from N to N' if N' is an immediate successor of N. In general it is convenient to picture graphs as directed downward, so that N lies above N' if N' is a successor of N. To determine in Figure 1 if $N \in s(G')$ it suffices to verify that G' is the set of all nodes connected to N by an arc directed to N. Thus, for example,

$$s(N_1,N_2) = \{N_6\},$$
$$s(N_2,N_6) = \{N_9\},$$
$$s(N_3,N_4) = \{N_7,N_8\},$$
$$s(N_7) = \{N_{10},N_{11}\},$$
$$s(N_2) = s(N_5) = s(N_8) = s(N_1,N_2,N_6) = \emptyset .$$

If the graph of Figure 1 is interpreted as a resolution graph by a labelling function $c : G \rightarrow S^*$ then the two clauses

$c(N_7)$ and $c(N_8)$ must be all the resolvents of the pair $c(N_3)$, $c(N_4)$. The clause $c(N_8)$ resolves with none of the clauses $c(N_i)$, $1 \leq i \leq 14$. The clauses $c(N_{10})$ and $c(N_{11})$ are either factors of $c(N_7)$ or are obtained from $c(N_7)$ by resolving $c(N_7)$ with itself. If $C = c(N_6) = c(N_7) = c(N_{14})$ then C has three derivations, two of level one and one of level three. Derivations are not necessarily represented by derivation trees. For instance the derivation of $c(N_{12})$ consists of the caluses $c(N_1)$, $c(N_2)$, $c(N_3)$, $c(N_4)$, $c(N_6)$, $c(N_7)$, $c(N_9)$, $c(N_{11})$, $c(N_{13})$. The clause $c(N_2)$ is used twice in the derivation of $c(N_{13})$ but is represented by only one node in G.

An abstract theorem-proving problem with non-negative costs ("abstract problem with costs" or simply "problem") is a tuple $\mathcal{P} = (G, s, F, g)$ where $F \subseteq G$, the set of terminal nodes for $\mathcal{P}$ (or solution nodes), and $g : G \to \mathbb{R}$, the costing function of $\mathcal{P}$, ($\mathbb{R}$, the set of real numbers) are such that

(1) $N \in F$ implies that $s(G') = \emptyset$ whenever $N \in G' \subseteq G$,

(2) (a) $g(N) \geq 0$ for all $N \in G$,

(b) if $N \in s(N_1, \ldots, N_k)$ (we write $s(N_1, \ldots, N_k)$ instead of $s(\{N_1, \ldots N_k\} \quad) \quad)$ then

$$g(N) \geq \max_{1 \leq i \leq k} g(N_i).$$

A solution to the problem $\mathcal{P}$ is obtained by constructing an algorithm Σ which generates from S_0 a node $N \in F$. Each node $N \in F$ is assigned a cost and it is often required to solve $\mathcal{P}$ by generating a node $N \in F$ having minimal cost in F. If $g(N) = 0$ for all $N \in G$ then in effect we have a problem without costs.

Alternatively $g(N)$ may be taken to be the level of N, the number of nodes $N' \leq N$ or any other value which satisfies (3) above. In applications to resolution theory $g(N)$ is usually taken to be the level of the clause $c(N)$. For $N \in \mathfrak{S}_0$ we do not require that $g(N) = 0$. This freedom allows us to assign different costs to distinct nodes in $\mathfrak{S}_0$ and is especially useful when $\mathfrak{S}_0$ is infinite. The set $\mathfrak{F}$ may be empty in which case the problem has no solution. In resolution applications when $\mathfrak{F} = \{N \in G : c(N) = \Box\}$ then $\mathfrak{F}$ is empty if the set $S_0 = \{c(N) : N \in \mathfrak{S}_0\}$ is satisfiable. The general problem $\mathfrak{P} = (G, s, \mathfrak{F}, g)$ reduces to the ordinary tree searching problem when (G,s) is a tree.

4.2 Search Strategies for Abstract Theorem-Proving Problems.

A <u>search strategy</u> Σ for $\mathfrak{P}$ is a function $\Sigma: 2^G \rightarrow 2^G$ which generates subsets of G from other subsets of G. Given such a function Σ for $\mathfrak{P}$ we define the sets Σ_i of nodes already <u>generated by Σ before the i+1-st stage</u> and $\tilde{\Sigma}_i$ of nodes which are <u>candidates for generation by Σ at the i+1-st stage</u>:

(1) $\Sigma_0 = \emptyset, \; \tilde{\Sigma}_0 = \mathfrak{S}_0,$

(2) $\Sigma_{i+1} = \Sigma_i \cup \Sigma(\Sigma_i),$

$\tilde{\Sigma}_{i+1} = (\{ N : N \in s(G'), G' \subseteq \Sigma_{i+1} \} \cup \tilde{\Sigma}_i) - \Sigma_{i+1}.$

We require that Σ satisfy

(3) $\Sigma(\Sigma_i) \subseteq \tilde{\Sigma}_i$.

The set of nodes $\Sigma(\Sigma_i)$, chosen from the set of candidates $\tilde{\Sigma}_i$, is the set of nodes <u>newly generated by Σ at the i+1-st stage</u> (it is easy to verify that $\Sigma_i \cap \tilde{\Sigma}_i = \emptyset$ for all $i \geq 0$).

The function Σ should be interpreted as selecting subsets G' of Σ_i and generating nodes $N \in s(G')$ which have not been previously generated. The definitions above only partially formalize the intuitive notion of search strategy for $\mathcal{P}$. In particular the search strategies Σ are never allowed to display any redundancy, i.e. generate the same node twice. This restriction is not essential because given any concrete, possibly redundant, algorithm for generating nodes in G there corresponds a unique search strategy Σ which, except for redundancies, generates the same nodes in the same order.

Notice that $\Sigma(\Sigma_i)$ may contain more than one node - as is common with resolution strategies which simultaneously generate several resolvents of a single clash or several factors of a single clause. Notice too that nodes in S_0 can be generated at any stage. We do not require that $\Sigma(\Sigma_i)$ contain a node $N \in F$ when $\tilde{\Sigma}_i \cap F \neq \emptyset$. If $\mathcal{P}$ is an ordinary tree search problem then the definition of search strategy for $\mathcal{P}$ provides a formal notion which applies to the usual strategies employed in searching for nodes in trees.

A search strategy Σ for $\mathcal{P} = (G,s,F,g)$ is complete for $\mathcal{P}$ if for all $N \in G$ there exists an $i > 0$ such that $N \in \Sigma_i$. It is possible to define completeness in this way since we do not insist that Σ generates no additional nodes after generating a first node $N^* \in F$. We say that Σ <u>terminates</u> at stage i if $F \cap \Sigma_{i-1} = \emptyset$ and either

(1) $F \cap \Sigma_i \neq \emptyset$ or

(2) $\Sigma_i = \Sigma_{i-1}$.

In the first case Σ terminates with a solution and without a solution in the second case.

In the terminology of [16] a search strategy Σ is said to be admissible for $\mathcal{P}$ if Σ is complete for $\mathcal{P}$ and terminates with a solution having least cost in $\mathcal{F}$ if $\mathcal{F} \neq \emptyset$, i.e. $N^* \in \mathcal{F} \cap \Sigma_i$, $\mathcal{F} \cap \Sigma_{i-1} = \emptyset$ implies that $g(N^*) \leq g(N)$ for all $N \in \mathcal{F}$. In resolution applications admissible search strategies are of special interest for robot control and automatic program writing [13], where minimal cost solutions are related to simplest strategies and most efficient programs. More generally intuition suggests that, in the absence of special information about the location of non-minimal solutions, admissible search strategies will tend to be more efficient than non-admissible strategies for finding arbitrary solutions. An important step towards formalizing this intuition has already been made in the optimality theorems of Hart, Nilsson and Raphael [16]

We define the notion of a search strategy Σ for a problem $\mathcal{P} = (G,s,\mathcal{F},g)$ being compatible with a merit ordering $\preccurlyeq$ defined on the nodes of G. For the moment we require only that $\preccurlyeq$ be reflexive and defined for all pairs of nodes in G. We write $N_1 \prec N_2$ (N_1 has better merit than N_2) when $N_1 \preccurlyeq N_2$ and not $N_2 \preccurlyeq N_1$. We write $N_1 \approx N_2$ (N_1 and N_2 have equal merit) if $N_1 \preccurlyeq N_2$ and $N_2 \preccurlyeq N_1$. A search strategy Σ for $\mathcal{P}$ is compatible with a merit ordering $\preccurlyeq$ if for all $i \geq 0$,

(1) $\tilde{\Sigma}_i \neq \emptyset$ implies that $\Sigma(\Sigma_i) \neq \emptyset$,

(2) $N \in \Sigma(\Sigma_i)$ implies that $N \preccurlyeq N'$ for all $N' \in \tilde{\Sigma}_i$.

In other words, Σ always generates, from a non-empty set $\tilde{\Sigma}_i$, at least one node $N \in \tilde{\Sigma}_i$ and no node $N' \in \tilde{\Sigma}_i$ which is not generated by Σ has better merit than any node $N \in \tilde{\Sigma}_i$ which is generated by Σ. Since a node N may have better merit than an ancestor $N' < N$, Σ need not generate nodes in order of merit. Distinct strategies Σ and Σ' for the same $\mathcal{P}$ compatible with the same merit ordering $\preccurlyeq$ differ only with regard to tie breaking rules for choosing which nodes to generate from a set of candidates having equal merit.* If $\preccurlyeq$ is the trivial ordering, $N \preccurlyeq N'$ for all $N, N' \in G$, then $\preccurlyeq$ is a merit ordering for G and any search strategy Σ for $\mathcal{P}$ is compatible with $\preccurlyeq$. If $\preccurlyeq$ is the ordering by levels, $N \preccurlyeq N'$ if and only if $N \in S_i$, $N' \in S_{i'}$ and $i \leq i'$, then any search strategy for $\mathcal{P}$ compatible with $\preccurlyeq$ is a level saturation (or breadth first) strategy for $\mathcal{P}$. If $\preccurlyeq$ is the ordering by costs, $N \preccurlyeq N'$ if and only if $g(N) \leq g(N')$, then Σ compatible with $\preccurlyeq$ is a cost saturation strategy for $\mathcal{P}$. If $\preccurlyeq$ is the inverse ordering by levels, $N \preccurlyeq N'$ if and only if $N \in S_i$, $N' \in S_{i'}$ and $i \geq i'$, then Σ compatible with $\preccurlyeq$ is a depth first strategy for $\mathcal{P}$.

Lemma 4.21 states the fundamental properties of search strategies Σ compatible with merit orderings: (a) any node $N_2 \in G$ is generated by Σ before any node N_1 which has worse merit than N_2 and than all the ancestors of N_2, (b) if N_1 is generated before N_2 then N_2 or some ancestor of N_2 has worse or equal merit to N_1.

* This remark is not strictly correct, since the consequences of breaking a tie differently may be the generation of different untied successors of the originally tied nodes.

<u>Lemma 4.2.1</u>. Let $\mathcal{P} = (G, s, \mathcal{F}, g)$ be a problem, $\preccurlyeq$ a merit ordering for G and Σ a search strategy for $\mathcal{P}$ compatible with $\preccurlyeq$.

(a) If $N_1 \in \Sigma_i$ and $N_2 \in G$ are such that $N \prec N_1$ for all $N \leq N_2$ then $N_2 \in \Sigma_{i-1}$.

(b) If $N_1 \in \Sigma_i$ and $N_2 \in \Sigma(\Sigma_i)$ then $N_1 \preccurlyeq N$ for some $N \leq N_2$.

<u>Proof</u>. (a) Let N_1 be generated at the j + 1 - st stage, i.e. $N_1 \in \Sigma(\Sigma_j)$, $N_1 \notin \Sigma_j$ and $j < i$. If $N_2 \notin \Sigma_j$ then for some $N \leq N_2$, $N \notin \Sigma_j$ and $N \in \tilde{\Sigma}_j$. But $N \prec N_1$ and therefore Σ is not compatible with $\preccurlyeq$ since it generates N_1 instead of N at the j + 1 - st stage. Therefore $N_2 \in \Sigma_j$ and $N_2 \in \Sigma_{i-1}$ since $j < i$.
(b) Suppose $N \prec N_1$ for all $N \leq N_2$. Then by (a), $N_2 \in \Sigma_{i-1}$ and therefore $N_2 \notin \Sigma(\Sigma_i)$.

A merit ordering $\preccurlyeq$ for G is <u>δ-finite</u> if for all $N \in G$ the set $\{N' \in G : N' \preccurlyeq N\}$ is finite (compare [16]). The importance of δ-finite merit orderings is a consequence of Theorem 4.2.2.: any search strategy compatible with a δ-finite merit ordering is complete. Any merit ordering for a finite set G is δ-finite. Ordering by levels is δ-finite if $\mathcal{S}_0$ is finite and s(G') is finite for all $G' \subseteq G$, under the same conditions inverse ordering by levels is not δ-finite if G is infinite (by König's Lemma).

<u>Theorem 4.2.2</u>. If $\mathcal{P} = (G, s, \mathcal{F}, g)$ is a problem, $\preccurlyeq$ a δ-finite merit ordering for G and Σ a search strategy for $\mathcal{P}$ compatible with $\preccurlyeq$, then Σ is complete for $\mathcal{P}$.

<u>Proof.</u> Let $N^* \in G$ be given. We need to show that $N^* \in \Sigma_j$ for some $j > 0$. If G is finite then $G = \bigcup_{i>0} \Sigma_i$ since $\tilde{\leqq}_i \neq \emptyset$ implies that $\leqq(\Sigma_i) \neq \emptyset$ and since $\leqq(\Sigma_i) \cap \Sigma_i \neq \emptyset$. Otherwise if G is infinite let $N' \leq N^*$ be a node such that $N \preccurlyeq N'$ for all $N \leq N^*$. Since $\preccurlyeq$ is δ-finite, since $\tilde{\leqq}_i \neq \emptyset$ implies that $\leqq(\leqq_i) \neq \emptyset$ and since $\leqq(\leqq_i) \subseteq \tilde{\leqq}_i$ it follows for some $j > 0$ and for some $N_1 \in \Sigma_j$, $N' \prec N_1$, and therefore $N \prec N_1$ for all $N \leq N^*$ and by Lemma 4.2.1 (a), $N^* \in \Sigma_j$.

<u>4.3 Heuristic Functions and Diagonal Search</u>.

There is special interest in merit orderings which can be expressed in terms of the cost function g of $\mathcal{P} = (G, s, \mathcal{F}, g)$ and of an additional heuristic function h [8], [30], [33]. A <u>heuristic function</u> h for $\mathcal{P}$ is a function $h: G \to \mathbb{R}$ such that $h(N) \geq 0$, for all $N \in G$. Let $f(N) = g(N) + h(N)$ for all $N \in G$. The intended interpretation of the heuristic function h is that $f(N) = g(N) + h(N)$ is an estimate of the cost $g(N^*)$ of a terminal node $N^* \in \mathcal{F}$, such that $N \leq N^*$, i.e. $h(N)$ is an estimate of $g(N^*) - g(N)$. If it is desired that Σ be admissible then $h(N)$ is intended to estimate the minimum value of $g(N^*) - g(N)$ for $N^* \in \mathcal{F}$ such that $N \leq N^*$.

Suppose, for example, that we know of a given problem

$\mathcal{P}_0 = (G_0, s_0, \mathcal{F}_0, g_0)$ that if it has a solution then its minimum cost is k. Suppose for simplicity that no $N \in G$ has cost $g_0(N)$ greater than k. Given only this information then an appropriate definition of a heuristic function h_0 for $\mathcal{P}$ is $h_0(N) = k - g_0(N)$ for all $N \in G$.

Suppose that a given problem $\mathcal{P}_1 = (G_1, s_1, \mathcal{F}_1, g_1)$ is interpreted as a resolution problem by a labelling function $c : G_1 \rightarrow S^*$. Suppose that the inference rules Γ consist of a factoring operation for unifying two literals in a clause and of a separate resolution rule for resolving at most two factors. Let $g_1(N)$ be the level of N and $\mathcal{F}_1 = \{ N : c(N) = \square \}$ For $N \in G_1$ let $l(c(N))$ be the length of c(N) (number of literals in c(N)). The heuristic function h_1 for $\mathcal{P}_1$ is defined by the letting $h_1(N)$ be the expected length of c(N):

(1) for $N \in \mathcal{S}$, $h_1(N) = l(c(N))$,

(2) for c(N) a resolvent of $c(N_1)$ and $c(N_2)$
$h_1(N) = l(c(N_1)) + l(c(N_2)) - 2$,

(3) for c(N) a factor of c(N') (the result of unifying two literals in c(N'))
$h(N) = l(c(N')) - 1$.

To the extent that merging does not occur (i.e. so long as $h_1(N) = l(c(N))$), $h_1(N)$ is a lower bound on the value of $g_1(N^*) - g_1(N)$ for $c(N^*) = \square$ when c(N) occurs in a derivation of $\square$. Notice that since Γ contains a factoring operation, this operation is explicitly exhibited in derivations, contrary to the conventions employed in chapters 1 - 3.

The costing function g and heuristic function h allow us to define two important classes of search strategies for $\mathcal{P}$. Given $\mathcal{P} = (G,s,\Gamma,g)$ and h a heuristic function for $\mathcal{P}$. Let the merit orderings $\preccurlyeq_d$ and $\preccurlyeq_d^u$ for G be defined for all $N_1, N_2 \in G$, by

(1) $N_1 \preccurlyeq_d N_2$ if and only if $f(N_1) \leq f(N_2)$,

(2) $N_1 \preccurlyeq_d^u N_2$ if and only if $f(N_1) \leq f(N_2)$ and

$h(N_1) \leq h(N_2)$ when $f(N_1) = f(N_2)$.

A search strategy Σ for $\mathcal{P}$ is a <u>diagonal search strategy</u> for $\mathcal{P}$ and h (written $\Sigma \in \mathcal{D}(\mathcal{P},h)$) if and only if Σ is compatible with the merit ordering $\preccurlyeq_d$. Σ is an <u>upwards diagonal search strategy</u> for $\mathcal{P}$ and h ($\Sigma \in \mathcal{D}^u(\mathcal{P},h)$) if and only if Σ is compatible with the merit ordering $\preccurlyeq_d^u$. Notice that $\mathcal{D}^u(\mathcal{P},h) \subseteq \mathcal{D}(\mathcal{P},h)$ and that $\mathcal{D}^u(\mathcal{P},h) = \mathcal{D}(\mathcal{P},h)$ if $h(N) = 0$ for all $N \in G$.

Except for minor differences, the search strategies $\Sigma \in \mathcal{D}(\mathcal{P},h)$ coincide with those investigated in [16] for the case of ordinary tree search. The search strategies $\Sigma \in \mathcal{D}^u(\mathcal{P},h)$ differ from those in $\mathcal{D}(\mathcal{P},h)$ by generating, from among candidate nodes which have equal merit for $\preccurlyeq_d$, those nodes whose cost is estimated to be closest to the cost of a solution node. In the case of the problem $\mathcal{P}_0$ and heuristic function h_0, defined above, $f_0(N) = g_0(N) + h_0(N) = k$ for all $N \in G$. All nodes in G have equal merit for search strategies $\Sigma \in \mathcal{D}(\mathcal{P}_0,h_0)$. For $\Sigma \in \mathcal{D}^u(\mathcal{P}_0,h_0)$ nodes which have cost closer to k have better merit than nodes which have smaller cost. In case g_0 (N) is the level of N for all $N \in G$ then $\Sigma \in \mathcal{D}^u(\mathcal{P}_0, h_0)$ is a depth-first search strategy, which intuitively seems the most efficient search strategy for $\mathcal{P}$, given only the information that a minimal solution of $\mathcal{P}$ must have level k. Concrete search algorithms for $\Sigma \in \mathcal{D}^u(\mathcal{P}_1, h_1)$ are discussed in the next section.

The terminology, diagonal and upwards diagonal search, is suggested by representing nodes $N \in G$ as occupying positions in the plane with co-ordinates (h(N),g(N)), where h increases rightwards away from the origin and g increases downwards away from the origin (see Figure 2). $\Sigma \in \mathcal{D}(\mathcal{P}, h)$ attempts to generate nodes on consecutive diagonals in order of increasing distance from the origin (0,0). In addition if $\Sigma \in \mathcal{D}^u(\mathcal{P}, h)$ then Σ attempts to generate nodes, lying on a given diagonal d, upwards in order of increasing h. If $\precsim_d$ or $\precsim_d^u$ are δ-finite then each diagonal contains only finitely many nodes $N \in G$ and for every diagonal d there are only finitely many diagonals which contain nodes $N \in G$ and which are closer than d to (0,0).

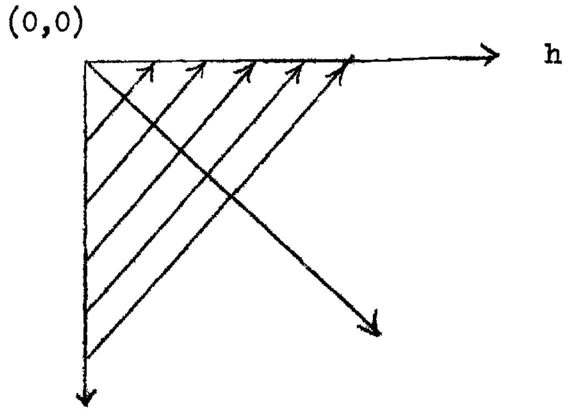


Figure 2.

Figure 3 illustrates Lemma 4.2.1 and Theorem 4.2.2 for a problem $\mathcal{P}$ and for a search strategy $\Sigma \in \mathcal{D}^u(\mathcal{P},h)$ where $\preccurlyeq_{d^u}$ is assumed to be δ-finite. The node $N^* \in \hat{F}$ has minimum cost in $\hat{F}$ and $N' \leq N^*$ is a node having worst merit in the set consisting of N^* and all ancestors of N^*. The node $N \in G$ has better merit than N^* and $N'' \leq N$ has worst merit in the set consisting of N and all ancestors of N. Dots represent nodes, lying on diagonals, generated by Σ before the generation of N^*. The small circles represent nodes generated by Σ after the generation of N^*. The diagonal d contains the node N'. By Lemma 4.2.1 Σ generates N^* before generating nodes having worse merit than N', i.e. before generating nodes lying above N' on d and before generating nodes lying on diagonals to the right of d.

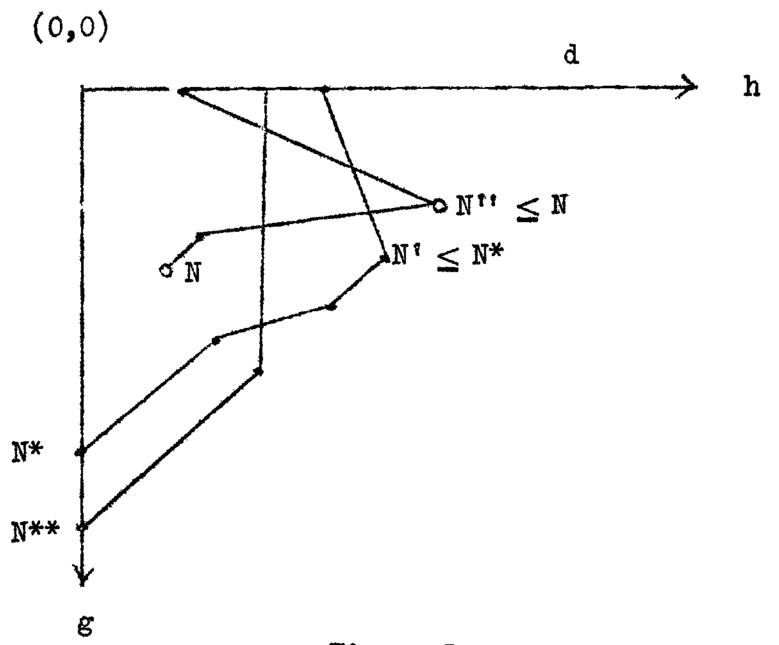


Figure 3.

The heuristic function h satisfies no conditions other than $h(N^*) = h(N^{**}) = 0$ and those imposed by the δ-finiteness of $\preccurlyeq_d^u$. Σ may fail to be admissible because some $N^{**} \in \mathcal{F}$ having worse merit than N^* will be generated before N^* if N^{**} and all ancestors of N^{**} have better merit than N'. The node $N \in G$ will not be generated before N^* if N'' lies to the right of d or above N' on d.

4.4 Upwards Diagonal Search Strategies for Resolution.

The algorithm Σ^* defined below approximates an upwards diagonal search strategy for the resolution problem $\mathcal{P}_1$ and heuristic function h_1. The same algorithm Σ^* when applied to the resolution problem $\mathcal{P}_2$ and heuristic function h_2 defined below is a pure upwards diagonal search strategy for $\mathcal{P}_2$ and h_2. The admissibility and optimality theorems of the next section apply to Σ^* for $\mathcal{P}_2$ and h_2 and to Σ^* for $\mathcal{P}_1$ and h_1, except when merging occurs in $\mathcal{P}_1$. A search strategy which differs inessentially from Σ^* has been implemented in POP2 by Miss Isobel Smith for a problem and heuristic function similar to $\mathcal{P}_1$ and h_1.

The definition and identification of the problem $\mathcal{P}_2$ was motivated by a suggestion of Mr. Donald Kuehner.

$\mathcal{P}_2 = (G_2, s_2, \mathcal{F}_2, g_2)$ differs from $\mathcal{P}_1$ by interpreting clauses $c(N)$ as lists of literals and by explicitly exhibiting and assigning cost to the operation (treated as a special case of factoring) of identifying two copies of the same literal within a clause. The length $l(c(N))$ of $c(N)$ is defined as the number of literals in the clause $c(N)$, counting duplications. $g_2(N)$ and $h_2(N)$ are still defined respectively

as the level of N and expected length of $c(N)$.
$h_2(N) = l(c(N))$ for all $N \in G_2$ and $h_2(N)$ is always a lower bound on the value of $g_2(N^*) - g_2(N)$ when $N \leq N^*$ and $N^* \in F_2 = \{N \in G_2 : c(N_2) = \square\}$.

Throughout the remainder of this section, $\mathcal{P} = (G,s,F,g)$ and h are either $\mathcal{P}_1$ and h_1 or $\mathcal{P}_2$ and h_2. The definition of Σ^* for $\mathcal{P}$ and h is the same for both of these cases except for the details remarked upon at the end of this section.

Clauses $c(N)$ are stored upon the generation of N in cells $A(i,j)$ of a two-dimensional array A. $c(N)$ is stored in $A(i,j)$ if $l(c(N)) = i$ and $g(N) = j$. Although it is natural to represent cells $A(i,j)$ as lists of clauses, we write $c(N) \in A(i,j)$ if $c(N)$ is stored in $A(i,j)$ when N is generated. The _merit_ of a node $N \in G$ is defined to be the cell $A(h(N),g(N))$. The cell $A(i,j)$ is said to be _better_ than $A(i',j')$ (written $A(i,j) \prec A(i',j')$) if

(1) $i + j < i' + j'$ or

(2) $i + j = i' + j'$ and $i < i'$.

Thus _a node $N \in G$ has better upwards diagonal merit than a node $N' \in G$ if and only if the merit of N is better than the merit of N'_, equivalently $N \prec_{\overset{u}{d}} N'$, if and only if $A(h(N),g(N))) \prec A(h(N'),g(N'))$. Notice that for $\mathcal{P}_2$ and h_2, $N \in G_2$ has merit $A(i,j)$ if and only if $c(N) \in A(i,j)$. For $\mathcal{P}_1$ and h_1, if $N \in G_1$ has merit $A(i,j)$ then $c(N) \in A(i',j)$ where $i' = l(c(N)) \leq h(N) = i$. Σ^*, on the whole, attempts to generate nodes of merit $A(i,j)$ before attempting to generate nodes of worse merit $A(i',j') \succ A(i,j)$.

A node of merit $A(i,j)$ is generated either

(0) by inserting into $A(i,0)$, when $j=0$, a clause $c(N)$ where $l(c(N)) = i$ and $g(N) = 0$,

(1) by unifying two literals within a clause $c(N) \in A(i+1,j-1)$ or

(2) by resolving a factor $c(N_1) \in A(i_1,j_1)$ with a factor $c(N_2) \in A(i_2,j_2)$ where N_1 may be identical to N_2 and where

$$i = i_1 + i_2 - 2 \quad \text{and}$$
$$j = \max(j_1,j_2) + 1.$$

Σ^* employs two subalgorithms for generating nodes $N \in G$. The principal subalgorithm Fill(i,j), generates in all possible ways, from nodes already generated, nodes N of merit $A(i,j)$ which have worse merit than all their ancestors. Fill (i,j) terminates when all such nodes have been generated. Fill (i',j'), where $A(i',j')$ is the next cell after $A(i,j)$, begins when Fill (i,j) terminates. Σ begins by invoking Fill $(0,0)$.

Whenever a node N_0 is generated by Fill (i,j) the second subalgorithm Recurse$(c(N_0))$ interrupts Fill (i,j) and generates in all possible ways, from nodes already generated, nodes N_1 which are immediate successors of N_0 and which are of merit $A(i_1,j_1)$ better than $A(i,j)$. In general whenever a node N is generated, either directly by Fill (i,j) or by some call of Recurse $(c(N'))$ which is local to Fill (i,j), Recurse $(c(N))$ generates, from nodes already generated, immediate successors of N which are of better merit than $A(i,j)$. Notice that if N is generated by Recurse $(c(N'))$ during Fill (i,j) then N has better merit than some ancestor of merit $A(i,j)$. Notice too that the depth of recursion involved in Recurse $(c(N'))$ is bounded by the sum $i+j$.

Remarks.

(1) If $\mathcal{P}$ and h are $\mathcal{P}_2$ and h_2 and if $c(N_0)$ is generated directly by Fill (i,j) then $c(N_0) \in A(i,j)$ and the only immediate successors of N_0 which are of better merit than $A(i,j)$ are nodes $N_1 \in A(i-1,j+1)$. Any such N_1 generated by Recurse $(c(N_0))$ is obtained either by factoring $c(N_0)$ or by resolving $c(N_0)$ with a unit clause $c(N)$ of level $g(N) \leq j$. More generally if N_0 is generated by Recurse during Fill (i,j) and if $c(N_0) \in A(i_0,j_0)$ then it is easy to verify that $i_0 + j_0 = i + j$ and therefore any immediate successor N_1 of better merit than $A(i,j)$ is of merit $A(i_0 - 1, j_0 + 1)$.

(2) If $\mathcal{P}$ and h are $\mathcal{P}_1$ and h_1 then Σ^* may fail to do upwards diagonal search because of merging, i.e. nodes may be generated by Recurse which have worse merit than other candidates for generation. Suppose that N_0 is generated by Fill (i,j) and that $c(N_0) \in A(i',j)$ where $i' < i$. Suppose that N_1 and N_2 of merit $A(i'-1,j+1)$ are generated by Recurse $(c(N_0))$, N_1 before N_2. Suppose that N_3 of merit $A(i'-1, j+2) \prec A(i,j)$ is generated by Recurse $(c(N_1))$. Then N_2 has better merit than N_3 but N_3 is generated before N_2 since Recurse $(c(N_1))$ must terminate before Recurse $(c(N_0))$ generates N_2.

(3) For both $\wp_1$, h_1 and $\wp_2$, h_2, Σ^* has the desirable property of attempting to resolve every unit clause $c(N_0)$ with all previously generated units $c(N_1)$ as soon as $c(N_0)$ is generated. If N_0 is generated furing Fill (i,j) and if $c(N_0) \in A(1,j_0)$ and $c(N_1) \in A(1,j_1)$ then $A(0, \max(j_0,j_1)+1) \prec A(i,j)$ and an attempt will be made to resolve $c(N_0)$ with $c(N_1)$ during Recurse $(c(N_0))$.

(4) Suppose that Fill (i,j) has just begun, then Σ^* has not yet generated any nodes of merit worse than $A(i,j)$. Thus if N has merit $A(i,j)$ then either $j = 0$ and $g(N) = 0$ or $c(N)$ is a resolvent of factors $c(N_1)$ and $c(N_2)$ and both N_1 and N_2 are of merit better than $A(i,j)$. In order to generate all such nodes N it suffices to attempt to resolve all clauses $c(N_1)$ with clauses $c(N_2)$ where

$$C_1 \in A(l,k),\ C_2 \in A(i-l+2, j-1)$$

for $0 \leq k \leq j-1$ and

$1 \leq l \leq \frac{i}{2}$ if i is even or

$1 \leq l \leq \frac{i+1}{2}$ if i is odd.

(5) The details for generating nodes during Recurse $(c(N))$ have already been discussed for $\wp_2$ and h_2 in remark (1). For $\wp_1$ and h_1 these details are more complicated. Suppose that N has been generated during Fill (i^*,j^*) and that $c(N) \in A(i,j)$. The following procedure will generate without redundancy, from nodes generated before N, immediate successors of N which are of better merit than $A(i^*,j^*)$:

(a) First resolve $c(N)$ with clauses in $A(i',j')$ where $j-1 \leq j' \leq i^*+j^*-i+2$, in order of decreasing j', and for given j', where $1 \leq i' \leq i^*+j^*-j'-i+1$ in arbitrary order but preferably in order of increasing i'.

(b) Next generate factors of $c(N)$ by attempting to unify, in all possible ways, two literals in $c(N)$.

(c) Finally resolve $c(N)$ with clauses in $A(i',j')$ where

$1 \leq i' \leq i^* + j^* - i - j + 1$

$0 \leq j' \leq j$ in arbitrary order but preferably in order of increasing i'.

(6) Let $\mathcal{P}_3 = (G_1, s_1, \mathcal{F}_1, g_3)$ where g_3 is defined as g_1 except for nodes N such that $c(N)$ is an immediate factor of a clause $c(N')$ in which case $g_3(N) = g_3(N')$. In other words $\mathcal{P}_3$ is identical to $\mathcal{P}_1$ except that cost is not assigned to the factoring operation. $h_3(N)$ is still defined as the expected length of $c(N)$.

With only minor modifications Σ^* can be applied to $\mathcal{P}_3$. The details differ little from those already discussed for applying Σ^* to $\mathcal{P}_1$.

4.5 Admissibility and Optimality of $\mathcal{O}$ and $\mathcal{O}^u$.

Let $\mathcal{P} = (G, s, \mathcal{F}, g)$ be an abstract theorem-proving problem. For $N \in G$ let

$$H(N) := \{g(N^*) - g(N) : N^* \in \mathcal{F}, N \leq N^*\},$$

$$h^*(N) = \inf H(N) \text{ when } H(N) \neq \emptyset,$$

$$h^*(N) = \infty \text{ when } H(N) = \emptyset.$$

Then when $N \leq N^*$, for some $N^* \in \mathcal{F}$, $h^*(N)$ is the greatest lower bound on the additional cost over $g(N)$ of $g(N^*)$. The heuristic function h is intended to be an estimate of h^*. The only property of ∞ needed below is that $k < \infty$ for all real numbers k. Since we do not allow $h(N) = \infty$, it is often impossible to construct a heuristic function h which gives a perfect estimate of h^*. In particular it is impossible to incorporate into a definition of h any information that a node N is not an ancestor a node $N^* \in \mathcal{F}$. However such heuristic information can be applied to a problem $\wp$ by defining a new problem $\wp'$ which differs from $\wp$ by containing no such nodes N. Alternatively it is possible to allow $h(N) = \infty$ in which case several complexities need to be introduced in preceding definitions (e.g. in the definition of δ-finiteness).

A heuristic function h for $\wp$ satisfies the <u>lower bound condition</u> for $\wp$ if

$$h(N) \leq h^*(N) \text{ for all } N \in G$$

i.e. if $h(N) \leq g(N^*) - g(N)$ whenever $N^* \in \mathcal{F}$ and $N \leq N^*$. Thus the lower bound condition constrains in effect only the value of $h(N)$ when N is an ancestor of some solution node. Recall that h_2 satisfies the lower bound condition for $\wp_2$ while h_1 does the same for $\wp_1$ except for merging.

Lemma 4.5.1 states certain fundamental properties of heuristic functions h satisfying the lower bound condition: (a) $h(N^*) = 0$ for $N^* \in \mathcal{F}$, (b) no ancestor of a solution node $N^* \in \mathcal{F}$ has worse diagonal merit than N^*, (c) there exists a solution node $N^* \in \mathcal{F}$ having minimum cost in $\mathcal{F}$ if diagonal merit is δ-finite.

Lemma 4.5.1 Let $\mathcal{P} = (G,s,\mathcal{F},g)$ be an abstract theorem-proving problem and let the heuristic function h for $\mathcal{P}$ satisfy the lower bound condition.

(a) If $N^* \in \mathcal{F}$ then $h(N^*) = h^*(N^*) = 0$ and therefore $f(N^*) = g(N^*)$.

(b) If $N^* \in \mathcal{F}$ and $N \leq N^*$ then $f(N) \leq f(N^*)$.

(c) If $\precsim_d$ is δ-finite then for some $N^* \in \mathcal{F}$ $g(N^*) \leq g(N)$ for all $N \in \mathcal{F}$, provided $\mathcal{F} \neq \emptyset$.

Proof. (a) is obvious, since $H(N^*) = \{0\}$ and $h^*(N^*) = 0$.
(b) If $N^* \in \mathcal{F}$ and $N \leq N^*$ then $h(N) \leq g(N^*) - g(N)$. But then $f(N) = g(N) + h(N) \leq g(N^*) = f(N^*)$.
(c) If $\precsim_d$ is δ-finite then for all $N \in G$, the set $\{N' \mid f(N') < f(N),\ N' \in G\}$ is finite. In particular for $N \in \mathcal{F}$ the set $\{N' \mid g(N') \leq g(N),\ N' \in \mathcal{F}\}$ is finite and therefore contains an element N^* such that $g(N^*)$ is minimal. But then $g(N^*) \leq g(N')$ for all $N' \in \mathcal{F}$.

Theorem 4.5.2. If $\precsim_d$ is δ-finite for $\mathcal{P} = (G,s,\mathcal{F},g)$ and if h satisfies the lower bound condition for $\mathcal{P}$ then $\Sigma \in \mathcal{D}(\mathcal{P},h)$ is admissible for $\mathcal{P}$.

Proof. Assume that $\mathcal{F} \neq \emptyset$. Let $N^* \in \mathcal{F}$ be such that $g(N^*) \leq g(N)$ for all $N \in \mathcal{F}$ (such an $N^* \in \mathcal{F}$ exists by Lemma 4.5.1 (c)). By Theorem 4.2.2. Σ is complete and therefore there is a stage i such that for some N,

$$N \in \mathcal{F} \cap \Sigma_i \text{ and } \mathcal{F} \cap \Sigma_{i-1} = \emptyset.$$

Suppose that Σ is not admissible for $\wp$. Then $g(N^*) < g(N)$. But, by Lemma 4.5.1 for all $N' \leq N^*$, $f(N') \leq f(N^*) = g(N^*) < f(N)$. So $f(N') < f(N)$ for all $N' \leq N^*$. But then $N' \prec N$ for all $N' \leq N^*$. By Lemma 4.5.1 (a), $N^* \in \Sigma_{i-1}$ and therefore $\wp \cap \Sigma_{i-1} \neq \emptyset$, contrary to assumption.

Theorem 4.5.2 specializes to a generalisation of Theorem 1 in [16] when $s(G') = \emptyset$ for all $G' \subseteq G$ which are not singletons. In particular it is not necessary to require that $\mathfrak{S}_0$ be finite or that $g(N)$ be strictly greater than $g(N')$ whenever $N' \leq N$. Since the specialization yields a tree representation of graph search, it is unnecessary to distinguish between the cost $g(N)$ and the total cost along some minimal path to N.

Figure 4 illustrates Lemma 4.5.1 and Theorem 4.5.2. $\wp$, Σ, N^*, N', N and N'' are as in Figure 3, but h satisfies the lower bound condition. By Lemma 4.5.1, N' lies on the same diagonal d as does N^*. Σ is admissible since any $N^{**} \in \wp$ having worse merit than N^* lies on a diagonal to the right of d and is not generated before N^*. It is still possible for a node $N \in G$ to have better merit than N^* and not be generated before N^* because N'' has worse merit than N'.

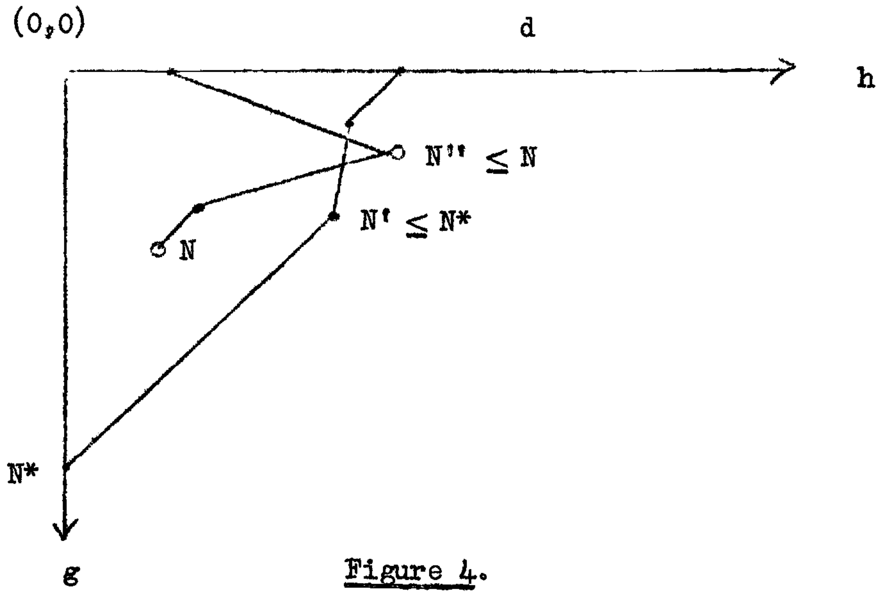


Figure 4.

To prove the appropriate extension of the Hart-Nilsson-Raphael Theorem on the optamality of $\Sigma \in \mathcal{O}^u$, we need to formulate an assumption equivalent to their "consistency assumption". The reader familiar with [16] will easily convince himself that the following condition is equivalent to the consistency assumption. We say that the evaluation function f satisfies the montonicity condition if

$$f(N') \leq f(N) \text{ for } N' \leq N \text{ and}$$

$$f(N^*) = g(N^*) \text{ for } N^* \in F.$$

(The first condition is equivalent to

$$h(N) \geq h(N') + (g(N') - g(N)) \text{ for } N' \leq N).$$

Notice that for $\mathcal{P}_2$ the evaluation function $f_2 = g_2 + h_2$ satisfies the monotonicity condition whereas for $\mathcal{P}_1$ the function $f_1 = g_1 + h_1$ is monotonic except for merging.

Figure 5 illustrates upwards diagonal search when the function f satisfies the monotonicity condition. $\mathcal{P}$, Σ, N^*, N', N and N'' are as in Figures 3 and 4. By Lemma 4.5.3, h satisfies the lower bound condition and therefore Σ is admissible and N' lies on the same diagonal as N^*. The monotonicity condition implies that if N has better diagonal merit than N^* then all ancestors of N have better merit than N^* and therefore, by Lemma 4.2.1, N is generated before N^*.

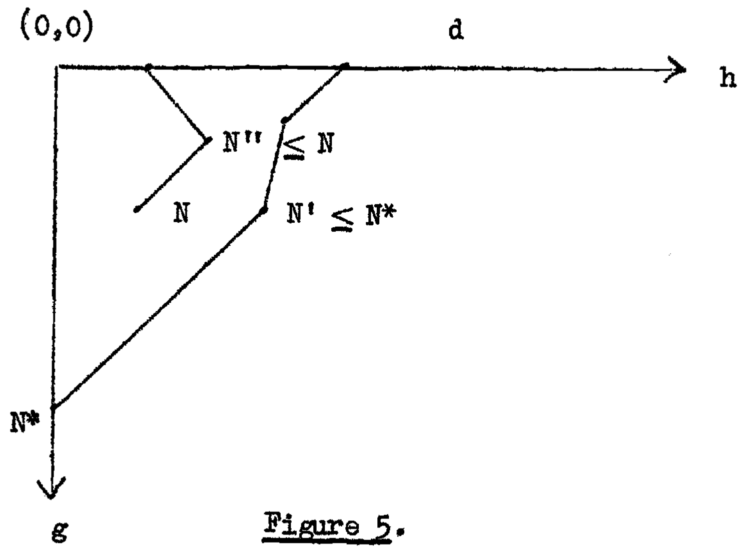


Figure 5.

Lemma 4.5.3. Let $\wp = (G, s, \digamma, g)$ be an abstract theorem-proving problem, let h be a heuristic function for $\wp$ and let f satisfy the monotonicity condition, where $f(N) = g(N) + h(N)$, $N \in G$. Then

(a) h satisfies the lower bound condition,

(b) If $\Sigma \in \mathfrak{D}(\wp, h)$, $N_1 \in \Sigma_i$ and $N_2 \in \Sigma(\Sigma_i)$ then $f(N_1) \leq f(N_2)$.

Proof. (a) h satisfies the lower bound condition if $h(N) \leq g(N^*) - g(N)$ whenever $N^* \in \digamma$ and $N \leq N^*$.

But the monotonicity of f implies that

$$f(N) = g(N) + h(N) \leq f(N^*) = g(N^*).$$

So $h(N) \leq g(N^*) - g(N)$.

(b) Suppose the contrary, namely that

$$N_1 \in \Sigma_i,\ N_2 \in \Sigma(\Sigma_i) \text{ and } f(N_1) > f(N_2).$$

But then, since $f(N') \leq f(N_2) < f(N_1)$ for all $N' \leq N_2$, it follows that $N' \prec N_1$ for all $N' \leq N_2$. By Lemma 4.2.1 (a), $N_2 \in \Sigma_{i-1}$, contradicting the assumption that $N_2 \in \Sigma(\Sigma_i)$.

For the case of ordinary graphs, the optimality theorem (Theorem 2) of [16] compares, in effect, search strategies $\Sigma \in \mathfrak{D}(\wp, h)$ with strategies

$\Sigma' \in \mathcal{D}(\mathcal{P},h')$ where $h'(N) \leq h(N)$ for all $N \in G$ and where $f = g + h$ is monotonic. (In [16] the search strategy Σ' is assumed only to be "no better informed" than Σ - we interpret this to mean that $h'(N) \leq h(N)$ and $\Sigma' \in \mathcal{D}(\mathcal{P},h')$.) If Σ_i and $\Sigma'_{i'}$ are the first sets which contain nodes $N^* \in \mathcal{F}$ then $\Sigma_i \subseteq \Sigma'_{i'} \cup G'$ where G' is the set of nodes $N \in \Sigma_i$ which have diagonal merit equal to $N^* \in \Sigma_i \cap \mathcal{F}$, i.e. before termination Σ' generates all the nodes generated by Σ except possibly for unlucky choices by Σ of nodes tied for merit with the solution node $N^* \in \Sigma_i$. Theorem 4.5.4 below generalizes Theorem 2 of [16] and implies in addition that $\mathcal{D}^u$ is an optimal subclass of $\mathcal{D}$.

It should be noted that the monotonicity condition on f in Theorem 4.5.4 can be replaced by the lower bound condition on h with the result that Σ' may now fail to generate nodes in the larger set G' of nodes $N \in \Sigma_i$ where some $N'' \leq N$ has diagonal merit tied with the solution node $N^* \in \Sigma_i$. A special case of this modification of Theorem 4.5.4 is illustrated by the example of Figure 6, following the proof of Theorem 4.5.4

Theorem 4.5.4.

Let $\mathcal{P} = (G,s,\mathcal{F},g)$ and let h and h' be heuristic functions for $\mathcal{P}$ such that

$$h'(N) \leq h(N) \text{ for all } N \in G.$$

Let $f(N) = g(N) + h(N)$ and $f'(N) = g(N) + h'(N)$. Suppose that f is monotonic. Given $\Sigma \in \mathcal{D}^u(\mathcal{P},h)$ and $\Sigma' \in \mathcal{D}(\mathcal{P},h')$, suppose that

$N_1 \in F \cap \Sigma_i$, $F \cap \Sigma_{i-1} = \emptyset$,

$N_2 \in F \cap \Sigma'_{i'}$ and $F \cap \Sigma'_{i'-1} = \emptyset$.

Then $\Sigma_i \subseteq \Sigma'_{i'} \cup G^*$ where

$G^* = \{ N : N \in \Sigma_i$ and for some $N' \leq N_1$,

$f(N) = f(N') = f(N_1)$ and $h(N) \leq h(N') \}$.

<u>Proof.</u> Σ' satisfies the lower bound condition since $h'(N) \leq h(N)$ for all $N \in G$ and since Σ satisfies the lower bound condition. Therefore both Σ and Σ' are admissible and $g(N_1) = g(N_2)$, $f(N_1) = f(N_2)$. Suppose that $N \in \Sigma_i$ and that $N \notin \Sigma'_{i'}$. It suffices to show that $N \in G^*$.

By Lemma 4.2.1 (b), $N \in \Sigma_i$ implies that $N \underset{d}{\overset{u}{\lessdot}} N'$ for some $N' \leq N_1$. But by Lemma 4.5.3 (b), since f is monotonic

$f(N) \leq f(N_1)$,

$f(N') \leq f(N_1)$,

$f(N'') \leq f(N)$ for all $N'' \leq N$.

But $h'(N'') \leq h(N'')$ implies

$f'(N'') \leq f(N'')$. So

$f'(N'') \leq f(N)$ for all $N'' \leq N$.

Also $N \notin \Sigma'_{i'}$ and $N_2 \in \Sigma'_{i'}$ imply by Lemma 4.2.1 (a) that for some $N'' \leq N$, $N'' \gtrdot_d N_2$, i.e.

$f'(N'') \geq f'(N_2) = f(N_1)$. So

$f(N) \geq f(N_1)$ and

$f(N) = f(N_1)$.

$N \underset{d}{\overset{u}{\lessdot}} N'$ implies

$f(N) \leq f(N') \leq f(N_1)$. So

$f(N) = f(N') = f(N_1)$ and

$h(N) \leq h(N')$, i.e.

$N \in G^*$.

Figure 6 compares nodes generated, before the generation of a given $N^* \in \mathcal{F}$, by different search strategies $\Sigma_i \in \mathcal{D}(\mathcal{P}, h_i)$ for a fixed problem $\mathcal{P} = (G, s, \mathcal{F}, g)$ and for different heuristic functions h_i. $h_1(N)$ is assumed to be a greatest lower bound on the value of $h^*(N)$ when $N \leq N^*$, where N^* has least cost in $\mathcal{F}$. Nodes $N \in G$ are represented as points with co-ordinates $(h_1(N), g(N))$. The node N' has worst upwards diagonal merit in the set consisting of N^* and the ancestors of N^*. The functions h_i are defined by $h_i(N) = i.h_1(N)$ for all $N \in G$, $0 \leq i \in \mathbb{R}$.

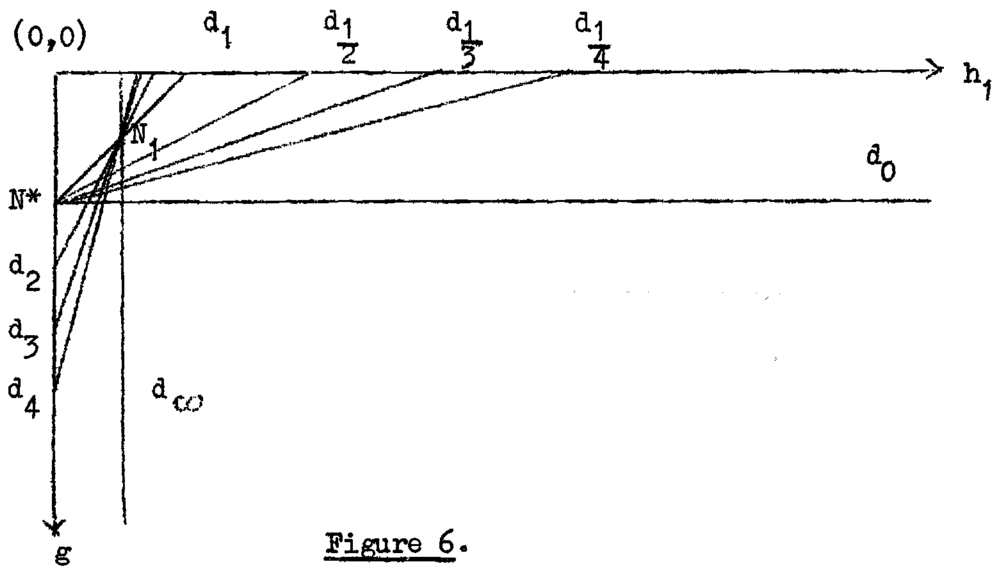


Figure 6.

For $0 \leq i \leq 1$, h_i satisfies the lower bound condition for $\mathcal{P}$ and Σ_i is admissible for $\mathcal{P}$. Σ_i need not be admissible for $\mathcal{P}$ when $i > 1$. The area to the left of the line d_i contains nodes generated by Σ_i before the generation of N^*. For $0 \leq i \leq 1$, Σ_i generates all the nodes generated by Σ_1. For $i > 1$, Σ_i generates all the nodes left of d_i which have been generated by Σ_1. No Σ_i is more efficient than Σ_1 if $i < 1$. Some Σ_i may generate fewer nodes than Σ_1 if $i > 1$, but this possibility becomes more remote as i increases. However even for large i, Σ_i may be more efficient than Σ_1 for generating solution nodes of arbitrary cost. A more thorough analysis of relationships similar to those discussed here has been made by Ira Pohl in [32] and [33].

4.6 Resolution of Marked Factors with i-Factor as Nucleus.

For resolution systems which employ separate rules for factoring and for resolving clashes of factors, Theorem 4.6.2 implies the refutation completeness of generating only i-factors of nucleus clauses. This completeness, which is subject to certain restrictions on the given resolution rule, applies to AM-resolution and to hyper-resolution in particular. For the case of systems which employ marked factoring and P_1-resolution of marked factors, 4.6.2 implies that non-positive clauses need never be factored. As reported in [17], this restriction can be combined with the method of section 2.9 for obtaining unique decompositions of hyper-resolution clashes. A theorem related to 4.6.2 was reported by Raphael in [36].

Lemma 4.6.1. Let $\mathcal{C} = \{A_1, \ldots, A_n, B\}$ be a clash with resolvent C. Then there exists a clash of marked factors $\mathcal{C}' = \{A'_{11}, \ldots, A'_{1m_1}, \ldots A'_{n1}, \ldots, A'_{nm_n}, B'\}$ with resolvent C' such that

(a) B' is an i-factor of B,

(b) A'_{ij} is a variant of a satellite factor of A_i,

$1 \leq j \leq m_i$,

(c) C is an instance of C'.

(d) $\mathcal{C}$ is restricted if $\mathcal{C}$ is.

Proof. By 1.6.1 there exist marked factors

$B'' = \{K_1,\ldots,K_n\} \overset{\bullet}{\cup} B_0''$ of B and

$A'_i = \{L_i\} \overset{\bullet}{\cup} A'_{0i}$ of A_i such that

$C = (A_{01}' \cup \ldots \cup A_{0n}' \cup B_0'')\theta_1$ where

θ_1 is an m.g.s.u. of

$$\mathcal{E}_1 = \{\{L_1,\bar{K}_1\},\ldots,\{L_n,\bar{K}_n\}\},$$

$B'' = B\theta_2$ where θ_2 is an m.g.s. u . of

$\mathcal{E}_2 = \{F_1,\ldots,F_n\}$, $B = F_1 \overset{\bullet}{\cup} \ldots \overset{\bullet}{\cup} F_n \cup B_0$

and $\cup(F_1 \overset{\bullet}{\cup} \ldots \overset{\bullet}{\cup} F_n)$ is the set of distinguished literals resolved upon in B..

For $1 \leq i \leq n$ let

$F_i = \{K_{i1},\ldots,K_{im_i}\}$ and let B' be the i-factor of B with distinguished literals $\cup(F_1 \overset{\bullet}{\cup} \ldots \overset{\bullet}{\cup} F_n)$.

Let $A^!_{ij} = \{L_{ij}\} \overset{\bullet}{\cup} A'_{0ij}$ be a variant of A_i' such that $A^!_{i1} = A^!_i$ and such that $\mathcal{C}'$ above is standardized.

Let η be a substitution such that

$$A^!_{ij}\eta = A^!_{i1} = A^!_i \text{ for all } i,j,\ 1 \leq i \leq n,\ 1 \leq j \leq m_i.$$

Let $\mathcal{E}_3 = \{\{L_{11},\bar{K}_{11}\},\ldots,\{L_{ij},\bar{K}_{ij}\},\ldots,\{L_{nm_n},\bar{K}_{nm_n}\}\}$.

Then $\mathcal{E}_3\eta\theta_2 = \mathcal{E}_1$.

Therefore $\eta\theta_2\theta_1$ unifies $\mathcal{E}_3$. Let θ_3 be an m.g.s. u of $\mathcal{E}_3$. Then for some λ, $\eta\theta_2\theta_1 = \theta_3\lambda$.

The resolvent of $\mathcal{C}'$ is

$$C' = (A'_{011} \cup \dots \cup A'_{0ij} \cup \dots \cup A'_{0nm_n} \cup B_0)\,\theta_3.$$

$$\begin{aligned} C'\lambda &= A'_{011}\,\theta_2\theta_1 \cup \dots \cup A'_{0ij}\,\theta_2\theta_1 \cup \dots \cup A'_{0nm_n}\,\theta_2\theta_1 \cup B_0\,\theta_2\theta_1 \\ &= A'_{01}\,\theta_1 \cup \dots \cup A'_{0i}\,\theta_1 \cup \dots \cup A'_{0n}\,\theta_1 \cup B_0''\,\theta_1 \\ &= C. \end{aligned}$$

We have shown that (a) - (c) hold. Suppose that $\mathcal{C}$ is restricted and that $\mathcal{C}'$ is not. Then

$$L_{ij}\theta_3 \in C' \text{ or } K_{ij}\theta_3 \in C'$$

for some i and j. But then

$$L_{ij}\,\theta_3\lambda = L_{ij}\,\eta\,\theta_2\,\theta_1 = L_i\,\theta_1 \in C'\lambda = C \quad \text{or}$$

$$K_{ij}\,\theta_3\lambda = K_{ij}\,\eta\,\theta_2\,\theta_1 = K_i\,\theta_1 \in C'\lambda = C.$$

So $\mathcal{C}'' = \{A_1', \dots, A_n', B''\}$ and $\mathcal{C}$ are not restricted.

For the statement of Theorem 4.6.2 and for the remainder of this chapter it is convenient to exhibit clashes of factors explicitly as clashes in derivations. A <u>factor-derivation</u> $\mathcal{D} = (T,c)$ of a clause C from a set of clauses S is a derivation such that

(1) for each tip $N \in T$, $c(N)$ is a factor of a clause in S,

(2) for each interior node $N \in T$, $c(N)$ is a factor of the resolvent of the clash of factors $c(s^{-1}(N))$,

Notice that the factoring operation is not exhibited in factor-derivations. If every clash in a factor-derivation $\mathcal{D}$ is a clash of marked factors then $\mathcal{D}$ is a <u>marked factor-derivation</u>. To simplify the statement and proof of Theorem 4.6.2 we allow the clause $c(r(T))$ in a marked factor derivation $\mathcal{D} = (T,c)$ to be unfactored.

<u>Theorem 4.6.2.</u> Let $\mathcal{D} = (T,c)$ be a clash derivation of a clause C from a set of clauses S. There exists a marked factor derivation $\mathcal{D}' = (T',c')$ from S of a clause C' which has C as instance. Every nucleus factor in $\mathcal{D}'$ is an i-factor.

To every node $N' \in T'$ there corresponds a node $N \in T$ satisfying the following conditions:

(a) If N' is a tip of T' then N is a tip of T and $c'(N')$ is a marked factor of $c(N)$.

(b) If N' is interior to T' then N is interior to T and $c(N)$ is an instance of $c'(N')$. Let $\mathcal{C} = c(s^{-1}(N))$ and $\mathcal{C}' = c'(s^{-1}(N'))$.

(i) Satellites of $\mathcal{C}$ correspond to satellites of $\mathcal{C}'$ and the nucleus of $\mathcal{C}$ corresponds to the nucleus of $\mathcal{C}'$.

(ii) $\mathcal{C}'$ is restricted if $\mathcal{C}$ is.

<u>Proof</u> (by induction on the number n of nodes in T). If n = 1 then $\mathcal{D}' = \mathcal{D}$ satisfies the requirements of the theorem. If $n > 1$ let $N_0 = r(T)$ and $s^{-1}(N_0) = \{N_1,\ldots,N_m\}$. We may assume by way of induction hypothesis that to each derivation $\mathcal{D}_i = (T_{N_i}, c)$, $1 \leq i \leq m$, there corresponds a marked factor derivation $\mathcal{D}_i' = (T_i', c'_i)$ which satisfies the theorem relative to $\mathcal{D}_i$.

Let $N_i' = r(T_i')$, $\mathcal{C}^* = \{c_1'(N_1'),\ldots,c'_m(N'_m)\}$ and $\mathcal{C} = c(s^{-1}(N_0))$. $\mathcal{C}^*$ subsumes $\mathcal{C}$. By Lemma 1.10.1, $\mathcal{C}^*$ is a clash which covers $\mathcal{C}$, $c(N_0)$ is an instance of the resolvent C^* of $\mathcal{C}^*$. Let $\mathcal{C}^*$ be the $\mathcal{C}$ of Lemma 4.6.1 and let $\mathcal{C}$ be the corresponding set of marked factors with resolvent C' which has C* and therefore $c(N_0)$ as an instance.

The desired marked factor derivation $\mathcal{D}' = (T', c')$ is determined by the following conditions.

(1) $r(T') = N_0'$ and $c'(N_0') = C'$

(2) $c'(s^{-1}(N_0')) = \mathcal{C}'$. For $N' \in s^{-1}(N_0')$, $T'_{N'}$ is an isomorphic copy of T_i' where $c'(N')$ is a marked factor of $c_i'(N_i')$, $(T'_{N'}, c')$ is a copy of $\mathcal{D}'_i$, except that $c'(N')$ is a marked factor of $c'_i(N_i')$.

$\mathcal{D}'$ satisfies (a) and (b) of the theorem. N_0 corresponds to N_0' and if N corresponds to N' in $\mathcal{D}'_i$ then N corresponds to the appropriate copy of N' in $\mathcal{D}'$.

Notice that the level of C' in $\mathcal{D}'$ is the same as the level of C in $\mathcal{D}$, however the number of factors in a clash $\mathcal{C}'$ of $\mathcal{D}'$ is often greater than the number of clauses in the corresponding clash $\mathcal{C}$ of $\mathcal{D}$.

4.7 m-Factor Derivations.

m-Factor derivations are of interest for at least two reasons: First (Theorem 4.7.1), m-factoring provides an effective method for implementing merging restrictions. In particular the restrictions investigated by Andrews for ground derivations [2] can be lifted to general derivations by imposing m-factor restrictions. Second (Theorem 4.7.3), m-factoring is always more efficient than the Wos-Robinson factoring method (for search strategies Σ which give preference, among clauses of equal level, to clauses of shorter length).

Recall that a factor of a clause C is a clause $C\theta$ where θ is an m.g.s.u. of some partition of C (equivalently, some complete partition of C). m-factors are defined only for resolvents of clashes : if $C\theta$ is a factor of a resolvent C of a clash $\mathcal{C}$ then $C\theta$ is an m-factor (merging-factor) of C if θ does not unify literals in C which descend from the same parent in $\mathcal{C}$.

Let $\mathcal{C} = \{D_1 = E_1 \dot{\cup} D_{01}, \ldots, D_n = E_n \dot{\cup} D_{0n}\}$ be a clash where E_i is the set of literals in D_i resolved upon in $\mathcal{C}$. Then $C = (D_{01} \cup \ldots \cup D_{0n})\theta'$ is the resolvent of $\mathcal{C}$, where θ' is an m.g.s.u. of $\mathcal{C}$. A factor $C\theta$ of C is an m-factor if

$L_1, L_2 \in D_{0i}$ and $L_1\theta' \neq L_2\theta'$ imply
$L_1\theta'\theta \neq L_2\theta'\theta$.

m-factoring restrictions can be strengthened by limiting attention to m-factors of m-resolvents. C is an m-resolvent of $\mathcal{C}$ if

$L_1, L_2 \in D_{0i}$ and $L_1 \neq L_2$ imply
$L_1\theta' \neq L_2\theta'$.

Thus $C\theta$ is an m-factor of an m-resolvent C of a clash $\mathcal{C}$ if and only if

$L_1, L_2 \in D_{0i}$ and $L_1 \neq L_2$ imply
$L_1\theta'\theta \neq L_2\theta'\theta$.

A factor derivation $\mathcal{D} = (T, c)$ is an m-factor derivation if for every interior node $N \in T$, c(N) is an m-factor of an m-resolvent of the clash of factors $c(s^{-1}(N))$.

An m-factor $C\theta$ is a <u>merge</u> if at least two literals in distinct parents of C are unified by $\theta'\theta$, i.e. if for some

$$L_1 \in D_{0i}, \; L_2 \in D_{0j}, \text{ where } i \neq j,$$
$$L_1\theta'\theta = L_2\theta'\theta \; .$$

This definition coincides with Andrews' in the case of ground clashes and ground resolvents. Notice that if $\mathcal{C} = \{D_1,\ldots,D_n\}$ is a clash of n factors then $C\theta$ is a merge if and only if

$$l(C\theta) < l(D_1)+\ldots+ l(D_n)-2(n-1).$$

<u>Theorem 4.7.1.</u> Let $\mathcal{D} = (T,c)$ be a ground derivation from a set of clauses S and let S be a set of instances of clauses in S'. Then there exists an isomorphic m-factor derivation $\mathcal{D}' = (T,c')$ from S'.

For all $N \in T$

(a) $c(N)$ is an instance of $c'(N)$,

(b) $l(c(N)) = l(c'(N))$,

(c) if N is interior to T then

(i) $c'(N)$ is a merge if and only if $c(N)$ is and

(ii) $c'(s^{-1}(N))$ is restricted if $c(s^{-1}(N))$ is.

<u>Proof</u> (by induction on the number of nodes n in T). If $n = 1$ then $T = \{N_0\}$. Let $C = c(N_0)$. Then C is an instance $C'\sigma$ of some clause $C' \in S'$.

Let $C = \{L_1,\ldots,L_m\}$ and $\mathcal{E} = \{E_1,\ldots,E_m\}$ where $E_i = \{L' \in C' : L'\sigma = L_i\}$. $\mathcal{E}$ is a complete partition of C' unified by σ. Let θ be an m.g.s.u. of $\mathcal{E}$ then

$\sigma = \theta\lambda$, for some λ. Let $c'(N_0) = C'\theta$ then $c(N_0)$ is an instance of $c'(N_0)$ and contains the same number of literals as $c'(N_0)$.

Suppose that $n > 1$ and that the theorem holds for any ground derivation containing fewer than n nodes. Let $N_0 = r(T)$, $s^{-1}(N_0) = \{N_1,\ldots,N_m\}$ and $\mathcal{D}_i = (T_{N_i}, c)$ for $1 \leq i \leq m$. By the induction hypothesis there exist m-factor derivations $\mathcal{D}'_i = (T_{N_i}, c'_i)$ from S' satisfying (a)-(c) for $N \in T_{N_i}$, $1 \leq i \leq m$.

Let $\mathcal{C} = c(s^{-1}(N_0))$, $C = c(N_0)$ and $\mathcal{C}' = \{c_1'(N_1),\ldots,c'_m(N_m)\}$. Then $\mathcal{C}'$ subsumes $\mathcal{C}$ and therefore covers $\mathcal{C}$. $\mathcal{C}'$ is restricted if $\mathcal{C}$ is and the resolvent C' of $\mathcal{C}'$ has C as an instance. Let $c'(N_0) = C'\theta$ be defined as in the case $n = 1$. Then $c(N_0)$ is an instance of $c'(N_0)$ and contains the same number of literals as $c'(N_0)$. Let $\mathcal{D}' = (T, c')$ be defined by $c'(N_0) = C'\theta$ and $c'(N) = c'_i(N)$ for $N \in T_{N_i}$.

It suffices now to show that $C'\theta$ is an m-factor of an m-resolvent of $\mathcal{C}'$. Suppose that, on the contrary, there are distinct literals L_1 and L_2 in some $c'(N_i)$ such that $L_1\theta'\theta = L_2\theta'\theta$ where θ' is the m.g.s.u. of $\mathcal{C}'$ at N_0. But then, since $\mathcal{C}'$ covers $\mathcal{C}$ and since C is a ground resolvent,

$c'(N_i)\sigma = c(N_i)$ for some σ and

$c'(N_0)\lambda = c(N_0)$ for some λ such that $\sigma = \theta'\lambda$.

$L_1\sigma$ and $L_2\sigma$ are distinct in $c(N_i)$ (since $c'(N_i)$ and $c(N_i)$ contain the same number of literals). Therefore $L_1\sigma$ and $L_2\sigma$ are distinct in $c(N_0)$. But then $L_1\theta'\theta$ and $L_2\theta'\theta$

are distinct, contrary to assumption.

<u>Lemma 4.7.2.</u> Let $\mathcal{C} = \{D_1 = E_1 \dot{\cup} D_{01}, \ldots, D_n = E_n \dot{\cup} D_{0n}\}$ be a clash of factors with resolvent $C = (D_{01} \cup \ldots \cup D_{0n})\,\theta$. Let $D = C\,\theta'$ be a factor of C. Then there exists a clash of factors $\mathcal{C}' = \{D'_1 = E'_1 \dot{\cup} D'_{01}, \ldots, D'_n = E'_n \dot{\cup} D'_{0n}\}$ where for each i, $1 \leq i \leq n$, D'_i is a factor $D_i\,\theta_i$ of D_i, and D is an m-factor of the m-resolvent C' of $\mathcal{C}'$.

<u>Proof.</u> Let θ' be an m.g.s.u. of the complete partition $\mathcal{E}'$ of C. Then we can represent $\mathcal{E}'$ as

$$\mathcal{E}' = \{G_1\,\theta, \ldots, G_k\,\theta\} \text{ where}$$
$$G_j = G_{j1} \dot{\cup} \ldots \dot{\cup} G_{jn},\ G_{ji} \subseteq D_{0i} \text{ and}$$
$$L \in D_{0i},\ L\,\theta \in G_{ji}\theta \text{ imply } L \in G_{ji}.$$

Then $\mathcal{E}_i = \{G_{1i}, \ldots, G_{ki}\}$ is a complete partition of D_{0i}.

Let

$$\mathcal{E}'' = \mathcal{E} \cup \{G_1, \ldots, G_k\}$$

where $\mathcal{E}$ is the family of literals resolved upon in $\mathcal{C}$. Then $\theta\,\theta'$ is an m.g.s.u. of $\mathcal{E}''$ since $\mathcal{E}$ is a refinement of $\mathcal{E}''$, θ is an m.g.s.u. of $\mathcal{E}$ and θ' is an m.g.s.u. of $\mathcal{E}''\,\theta$. On the other hand, each $\mathcal{E}_i$ is a refinement of $\mathcal{E}''$ and none of the refinements $\mathcal{E}_1, \ldots, \mathcal{E}_n$ share variables. Let θ_i be an m.g.s.u. of $\mathcal{E}_i$. By 1.3.5, $\theta_1 \ldots \theta_n \theta''$ is an m.g.s.u. of $\mathcal{E}''$ where θ'' is an m.g.s.u. of

$$\mathcal{E}''\,\theta_1 \ldots \theta_n = \mathcal{E}\,\theta_1 \ldots \theta_n \cup \{G_1, \ldots, G_k\}\,\theta_1 \ldots \theta_n.$$

Let $D'_i = D_i\theta_i$. Then $\mathcal{C}' = \{D'_1, \ldots, D'_n\}$ is a clash and the family of literals resolved upon in $\mathcal{C}'$ is $\mathcal{E}\theta_1\ldots\theta_n$. Let θ^* be an m.g.s.u. of $\mathcal{E}\theta_1\ldots\theta_n$. The resolvent of $\mathcal{C}'$ is

$$C' = D_{01}\theta_1\ \theta^* \cup \ldots \cup D_{0n}\ \theta_n\ \theta^*.$$

Since $\mathcal{E}\,\theta_1\ldots\theta_n$ is a refinement of $\mathcal{E}''\ \theta_1\ldots\theta_n$, $\theta^*\ \theta^{**}$ is an m.g.s.u. of $\mathcal{E}''\theta_1\ldots\theta_n$ where θ^{**} is an m.g.s.u. of

$$\mathcal{E}''\,\theta_1\ldots\theta_n\theta^* = \mathcal{E}\ \theta_1\ldots\theta_n\theta^* \cup \{G_1,\ldots,G_k\}\theta_1\ldots\theta_n\theta^*.$$

But then $\theta_1\ldots\theta_n\theta^*\theta^{**}$ is an m.g.s.u. of $\mathcal{E}''$ and because $\mathcal{E}\ \theta_1\ldots\theta_n\ \theta^*$ is already unified we may take θ^{**} to be an m.g.s.u. of

$$\{G_1,\ldots,G_k\}\ \theta_1\ldots\theta_n\theta^*,$$

which is a partition of C'. Let $D' = C'\ \theta^{**}$

Then

$$\begin{aligned} D' &= (D_{01}\theta_1\theta^* \cup\ldots\cup D_{0n}\theta_n\theta^*)\ \theta^{**} \\ &= (D_{01} \cup\ldots\cup D_{0n})\ \theta_1\ldots\theta_n\theta^*\ \theta^{**} \\ &= (D_{01} \cup\ldots\cup D_{0n})\ \theta\ \theta' \\ &= D. \end{aligned}$$

It suffices to show that D' is an m-factor of an m-resolvent of $\mathcal{C}'$. Suppose not. Then for distinct literals L'_1 and L'_2 in some $D'_{0i} = D_{0i}\ \theta_i$,

$$L'_1\ \theta^*\ \theta^{**} = L'_2\ \theta^*\ \theta^{**}.$$

But then there are distinct literals L_1 and L_2 in D_{0i} such that $L'_1 = L_1\ \theta_i$, $L'_2 = L_2\ \theta_i$ and $L_1\ \theta_i\ \theta^*\ \theta^{**} = L_2\ \theta_i\ \theta^*\ \theta^{**}$.

Therefore

$$L_1\ \theta_1\ldots\theta_n\ \theta^*\ \theta^{**} = L_2\ \theta_1\ldots\ \theta_n\ \theta^*\ \theta^{**} \text{ and}$$

$$L_1\ \theta\ \theta' = L_2\ \theta\ \theta'.$$

So $L_1, L_2 \subseteq G_{ji}$ for some j. But $G_{ji}\ \theta_i$ is a singleton and therefore $L_1\ \theta_i = L_2\ \theta_i$ and $L'_1 = L'_2$ contrary to assumption.

Theorem 4.7.3. Let $\mathfrak{D} = (T,c)$ be a factor derivation of a factor C from a set of clauses S. Then there exists an isomorphic m-factor derivation $\mathfrak{D}' = (T,c')$ of C from S such that

$c'(N)$ is a factor of $c(N)$ for all $N \in T$.

Proof. (by induction on the number n of nodes in T). If $n = 1$ then $\mathfrak{D}' = \mathfrak{D}$ suffices. Suppose that $n > 1$ and that the theorem holds for any factor derivation containing fewer than n nodes.

Let $N_0 = r(T)$, $s^{-1}(N_0) = \{N_1,\ldots,N_m\}$, $C = c(N_0)$ and $\mathcal{C} = \{c(N_1),\ldots,c(N_m)\}$. By Lemma 4.7.2. there exists a clash $\mathcal{C}' = \{D_1,\ldots,D_m\}$, where D_i is a factor of $c(N_i)$, $1 \leq i \leq m$, and C is an m-factor of the m-resolvent of $\mathcal{C}'$. Let $\mathfrak{D}_i$ be the factor derivation (T_{N_i},c_i) which is identical to (T_{N_i},c) except that $c_i(N_i) = D_i$ instead of $c(N_i)$. By the induction hypothesis for each i, $1 \leq i \leq m$, there exists an m-factor derivation $\mathfrak{D}_i' = (T_{N_i},c'_i)$ of D_i such that $c'_i(N)$ is a factor of $c_i(N)$ for every $N \in T_{N_i}$.

Let $\mathfrak{D}' = (T,c')$ where $c'(N_0) = C$ and $c'(N) = c'_i(N)$ for $N \in T_{N_i}$. $\mathfrak{D}'$ is the required m-factor derivation of C.

Theorem 4.7.3 states that any clause (or factor) C, derivable by a factor derivation $\mathfrak{D}$, can be derived by an isomorphic m-factor derivation $\mathfrak{D}'$. Moreover $\mathfrak{D}'$ is no more complicated than $\mathfrak{D}$ in the sense that no factor in $\mathfrak{D}'$ contains more literals than the corresponding factor in $\mathfrak{D}$. Search strategies Σ (such as level saturation or upwards diagonal search) which generate simpler before more complex derivations will generate m-factor derivations before isomorphic factor derivations which are not m-factor derivations. Let Σ be such a strategy and let $(\mathfrak{I}, \Sigma)$ be a proof procedure employing Σ and an inference system $\mathfrak{I}$ which incorporates the Wos-Robinson factoring method (W-R method). Let $(\mathfrak{I}^1, \Sigma)$ differ from $(\mathfrak{I}, \Sigma)$ only by using the m-factoring method instead of the W-R method. If $(\mathfrak{I}, \Sigma)$ generates n derivations and $(\mathfrak{I}^1, \Sigma)$ generates n' derivations before obtaining a first refutation then $n = n' + k$ where k is the number of non m-factor derivations generated by $(\mathfrak{I}, \Sigma)$ before the generation of a first refutation.

REFERENCES

[1) Allen, J., and Luckham, D., An interactive theorem-proving program. Machine Intelligence 5, Edinburgh University Press (1970) pp.321-336.

[2] Andrews, P.B., Resolution with merging. Journal of the Association for Computing Machinery, volume 15 (1968) pp.367-381.

[3] Brown, T.C., Resolution with covering strategies and equality theory. California Institute of Technology, California (1968).

[4] Chang, C.L., Renamable paramodulation for automatic theorem-proving with equality. National Institute of Health, Bethseda, Maryland (1969).

[5] Darlington, J.L., Automatic theorem-proving with equality substitutions and mathematical induction. Machine Intelligence 3, Edinburgh University Press (1968) pp. 113-127.

[6] Darlington, J.L., Theorem-proving and information retrieval. Machine Intelligence 4, Edinburgh University Press (1969) pp.173-181.

[7] Davis, M., Eliminating the irrelevant from mechanical proofs. Proceedings of Symposia in Applied Mathematics, volume 15, American Mathematical Society (1963) pp.15-30.

[8] Doran, J. and Michie, D., Experiments with the graph traverser program. Proceedings of the Royal Society (A), volume 294 (1966) pp.235-259.

[9] Gelernter, H., Realization of a geometry theorem-proving machine. Proceedings of the IFIP Congress 1959, pp.273-282.

[10] Gilmore, P.C., A proof method for quantification theory. IBM Journal of Research and Development, volume 4 (1960) pp.28-35.

[11] Gödel, K., Über die Länge von Beweisen. Ergebnisse eines math. Koll., volume 7 (1936) pp.23-24.

[12] Gould, W.E., A matching procedure for w-order logic. Science Report No.4, AFCRL 66-781 (1966).

[13] Green, C.C., The application of theorem-proving to question-answering systems. Ph.D. thesis, Stanford University, Stanford, California. Also Stanford Artificial Intelligence Project Memo AI - 76 (1969).

[14] Guard, J.R., Oglesby, F.C., Bennet, J.H. and Settle, L.G., Semi-automated mathematics. Journal of the Association for Computing Machinery, volume 16 (1969) pp. 49-62.

[15] Hart, T.P., A useful algebraic property of Robinson's unification algorithm. Artificial Intelligence Project Memo 91, Project MAC, M.I.T., Cambridge, Massachusetts (1965).

[16] Hart, P.E., Nilsson, N.J. and Raphael, B., A formal basis for the heuristic determination of minimum cost paths. I.E.E.E. Transactions on System Sciences and Cybernetics (July 1968).

[17] Hayes, P.J. and Kowalski, R.A., Semantic trees in in automatic theorem-proving, Machine Intelligence 4. Edinburgh University Press (1969) pp. 87-101.

[18] Kleene, S.C., Mathematical Logic. John Wiley and Sons, Inc., New York (1967).

[19] Kowalski, R.A., Panel discussion, Formal systems and non-numerical problem solving by computers. Fourth Systems Symposium, Case Western Reserve University, Cleveland, Ohio (November 1968).

[20] Kowalski, R.A., The case for using equality axioms in automatic demonstration, Proceedings IRIA Symposium on Automatic Demonstration, Versailles, France, December 1968. Springer-Verlag (in press).

[21] Kowalski, R.A., Search strategies for theorem-proving. Machine Intelligence 5, Edinburgh University Press (1970) pp. 181-201.

[22] Loveland, D.W., Mechanical theorem-proving by model elimination. Journal of the Association for Computing Machinery, volume 15 (1968).pp.236-251.

[23] Loveland, D.W., A linear format for resolution. Proceedings IRIA Symposium on Automatic Demonatration, Versailles, France, December 1968. Springer-Verlag (in press).

[24] Luckham, D. Refinement theorems in resolution theory. Proceedings IRIA Symposium on Automatic Demonstration, Versailles, France, December 1968. Springer-Verlag (in press).

[25] Meltzer, B., Theorem-proving for computers: some results on resolution and renaming. The Computer Journal, volume 8 (1966) pp. 341-343.

[26] Meltzer, B., Some notes on resolution strategies. Machine Intelligence 3, Edinburgh University Press (1968) pp. 71-75.

[27] Meltzer, B., Power amplification for theorem-provers. Machine Intelligence 5, Edinburgh University Press (1970) pp. 165-179.

[28] Morris, J.B., E-resolution: extension of resolution to include the equality relation. Proceedings of the International Joint Conference on Artificial Intelligence, Washington, D.C. (7-9 May 1969) pp. 287-294.

[29] Nerode, A. and Smullyan, R.M., Review of E.W. Beth, The foundations of mathematics, a study in the philosophy of science. The Journal of Symbolic Logic, volume 27 (1962) pp. 73-75.

[30] Nilsson, N.J., Searching problem-solving and game-playing trees for mininal cost solutions. IFIP Congress Reprints (1968) pp. H 125-130.

[31] Norton, M.N., ADEPT - a heuristic program for proving theorems of group theory. Ph.D. thesis, M.I.T., Cambridge, Massachusetts (1966).

[32] Pohl, I., Bi-directional and heuristic search in path problems. Ph.D. thesis. Stanford University, Stanford, California. Also SLAC Report No.104 (1969).

[33] Pohl, I., First results on the effect of error in heuristic search. Machine Intelligence 5, Edinburgh University Press (1970).

[34] Prawitz, D., An improved proof procedure. Theoria, volume 26 (1960) pp. 102-139.

[35] Prawitz, D., Advances and problems in mechanical proof procedures. Machine Intelligence 4, Edinburgh University Press (1969) pp. 59-71.

[36] Raphael, B., Some results about proof by resolution. SIGART Newsletter No.14 (February 1969) pp. 22-25.

[37] Robinson, G.A. and Wos,L., Completeness of para-modulation. Abstract, The Journal of Symbolic Logic, volume 34 (1969) p.160.

[38] Robinson, G.A. and Wos, L., Paramodulation and theorem-proving in first-order theories with equality. Machine Intelligence 4, Edinburgh University Press (1969) pp. 135-150.

[39] Robinson, J.A., A machine-oriented logic based on the resolution principle. Journal of the Association for Computing Machinery, volume 12 (1965) pp.23-41.

[40] Robinson, J.A., Automatic deduction with hyper-resolution. International Journal of Computer Mathematics, volume 1 (1965) pp. 227-234.

[41] Robinson, J.A., Heuristic and complete processes in the mechanization of theorem-proving. Systems and Computer Science, University of Toronto Press (1967) pp. 116-124

[42] Robinson, J.A., A review of automatic theorem-proving. Proceedings of Symposia in Applied Mathematics, volume 19 (1967) pp.1-18.

[43] Robinson, J.A., The generalised resolution principle. Machine Intelligence 3, Edinburgh University Press (1968) pp. 77-94.

[44] Robinson, J.A., New directions in mechanical theorem-proving. Proceedings of the IFIP Congress 1968. pp.206-210 (Invited papers).

[45] Robinson, J.A., Mechanizing higher-order logic. Machine Intelligence 4, Edinburgh University Press (1969) pp. 151-120.

[46] Robinson, J.A., The present state of mechanical theorem-proving. Proceedings of the Fourth Systems Symposium 1968 (in press).

[47] Sandewall, E., Concepts and methods for heuristic search. Proceedings of the International Joont Conference on Artificial Intelligence, Washington, D.C. (7-9 May 1969) pp. 199-218.

[48] Sibert, E.E., A machine-oriented logic incorporating the equality relation. Machine Intelligence 4, Edinburgh University Press (1969) pp.103-133.

[49] Slagle, J.R., A heuristic program that solves symbolic integration problems in freshman calculus. Computers and Thought. McGraw-Hill, New York (1963).

[50] Slagle, J.R., A proposed preference strategy using sufficiency-resolution for answering questions. Lawrence Radiation Laboratories Memo. UCRL - 14361 (1965)

[51] Slagle, J.R., Automatic Theorem-proving with renamable and semantic resolution. Journal of the Association for Computing Machinery, volume 14 (1967) pp.687-697.

[52] Slagle, J.R., Chang, C.L. and Lee, R.C.T., Completeness theorems for semantic resolution in consequence finding. Proceedings of the International Joint Conference on Artificial Intelligence, Washington, D.C. (7-9 May 1969) pp.281-285.

[53] Wos, L., Carson, D.F. and Robinson, G.A., The unit preference strategy in theorem-proving. Proceedings of the AFIPS 1964 Fall Joint Computer Conference, volume 26, pp. 616-621.

[54] Wos, L., Robinson, G.A. and Carson, D.F., Efficiency and completeness of the set of support strategy in theorem-proving. _Journal of the Association for Computing Machinery_, volume 12 (1965) pp.536-541.

[55] Wos, L., Robinson, G.A., Carson D.F. and Shalla, L., The concept of demodulation in theorem-proving. _Journal of the Association for Computing Machinery_, volume 14 (1967) pp. 698-709.

[56] Wos, L. and Robinson, G.A., Maximal model theorem. Abstract, _The Journal of Symbolic Logic_, volume 34 (1969) pp. 159-160.