Linear Resolution with Selection Function

Robert Kowalski and Donald Kuehner
Metamathematics Unit, University of Edinburgh

Recommended by B. Meltzer

ABSTRACT

Linear resolution with selection function (SL-resolution) is a restricted form of linear resolution. The main restriction is effected by a selection function which chooses from each clause a single literal to be resolved upon in that clause. This and other restrictions are adapted to linear resolution from Loveland's model elimination.

We show that SL-resolution achieves a substantial reduction in the generation of redundant and irrelevant derivations and does so without significantly increasing the complexity of simplest proofs. We base our argument for the increased efficiency of SL-resolution upon precise calculation of these quantities.

A more far reaching advantage of SL-resolution is its suitability for heuristic search. In particular, classification trees, subgoals, lemmas, and and/or search trees can all be used to increase the efficiency of finding refutations. These considerations alone suggest the superiority of SL-resolution to theorem-proving procedures constructed solely for their heuristic attraction.

From comparison with other theorem-proving methods, we conjecture that best proof procedures for first order logic will be obtained by further elaboration of SL-resolution.

1. Introduction

Inference Systems and Efficiency

A good inference system for mechanising deduction should satisfy the following criteria:

(1) It should admit few redundant derivations and limit those which are irrelevant to a proof.
(2) It should admit simple proofs.
(3) It should determine a search space which is amenable to a variety of methods for heuristic search.

We shall argue that, on the basis of these three criteria, linear resolution with selection function (SL-resolution) is a good inference system and is better than other systems which have been investigated for first order logic. We shall calculate precise bounds on the number of redundant derivations eliminated and on the complexity of simplest proofs admitted by SL-resolution. Finally, we describe some of the useful heuristics that can be applied for searching the SL-resolution derivation space.

Artificial Intelligence 2 (1971), 227–260

We shall assume that the reader is familiar with the basic terminology of resolution theory. For this purpose, the review paper [20] by Robinson is a more than adequate preliminary. We regard a *proof procedure* as consisting of an inference system supplemented by a search strategy. The *inference system* of a proof procedure consists of axioms and rules of inference which determine for a given theorem the *search space* of all derivations obtainable from the axioms by means of the inference rules. One inference system is a *refinement* of another, if the search space of the first is contained in the search space of the second. A *search strategy* for a given inference system is an algorithm for consecutively generating derivations in the search space until a first proof is found. In general, every proof procedure generates a number of unnecessary derivations in addition to the first proof of the theorem it tries to prove.

We propose that the number of derivations generated by a search strategy before the generation of a first proof be treated as a measure of *the difficulty* of proving the theorem by means of the given proof procedure. We shall use this measure of difficulty as a means of comparing the efficiency of different proof procedures.

It is important to distinguish the difficulty of proving a theorem from the complexity of a proof of that theorem. Let the *size* of a resolution derivation be the number of distinct occurrences of resolvents in the derivation. We shall treat size as a first approximation to the *complexity* of a derivation. (More sophisticated measures will be considered in a later section.) The difficulty of proving a theorem depends upon the proof procedure and therefore upon both the inference system and the search strategy employed. The complexity of a derivation depends only on the inference system which admits the derivation, but not upon any search strategy.

The efficiency of a proof procedure is related to the complexity of the simplest proofs its inference system allows. In general, the simpler the proofs admitted by an inference system, the easier it is for a search strategy to generate a first proof. The fewer unnecessary derivations admitted by the inference system, the fewer are generated by the search strategy. The more amenable the search space to the application of intelligent heuristic methods, the more intelligent is the resulting proof procedure.

Outline of the Paper

The paper is divided into three main parts. The first defines linear, t-linear and SL-derivations and search spaces. The discussion of t-linear resolution is included primarily to clarify the definition of SL-resolution and to simplify the comparison with other linear resolution systems. The formal definition of SL-resolution uses the chain format employed by Loveland for model elimination.

Where the first part of the paper compares the successive reduction of branching rates at nodes of the search trees for refinements of linear resolution, the second part uses minimal derivations to compare the complexities of the simplest refutations obtainable by these refinements. Proofs that no system admits simpler refutations than minimal and s-linear resolution and that there exist SL-refutations as simple as minimal refutations are included in the appendix.

The last main part of the paper deals with the application of heuristics to the determination of search spaces and search strategies for SL-resolution. The search spaces can take the form either of search trees labelled by derivations or of and/or trees labelled by goals and subgoals. Efficient search strategies which search for simplest proofs can be constructed, employing heuristic functions and diagonal search. Further improvements can be obtained by using lemmas and by employing deletion rules for subsumed chains.

More detailed investigation of search strategies for SL-resolution is contained in Kuehner's thesis [10]. The theory of efficiency which underlies our arguments for SL-resolution is investigated in Kowalski's thesis [8] and outlined further in Meltzer's lectures [17].

This paper is a revised and shortened version of an earlier one [11]. It differs from the original primarily by the incorporation of more examples, by the substitution of proof outlines in the appendix for detailed proofs in the text, and by an altered emphasis which reflects the results of further investigations. Certain shortcomings in the earlier treatment of t-linear resolution and lemma generation were called to our attention by Michael Gordon and Donald Loveland respectively. These faults have now been corrected.

Since the completion of the original paper, we have learned of the related investigations of Donald Loveland [14] and Raymond Reiter [19]. Loveland investigates in detail the relationship between model elimination and linear resolution, and includes an interesting comparison of these systems with the Prawitz matrix reduction method [18]. Reiter investigates two ordering restrictions and establishes their compatibility with linear resolution and the merging restriction [2]. Reiter's second ordering restriction coincides with the selection function restriction for ground derivations.

In SL-resolution, we have attempted to construct the best inference system possible and have borrowed freely from what seems, to us, the best in other systems. The resulting system can be regarded as a form of either model elimination or linear resolution. When compared with the systems investigated by Loveland and Reiter, it bears the greatest resemblance to model elimination.

2. Linear Resolution

Linear resolution was independently discovered by Loveland [13], Luckham [15] and Zamov and Sharonov [23]. It is a refinement of unrestricted

resolution which significantly reduces the number of redundancies derivable. For certain measures of complexity (*rm*-size, Section 5), linear resolution is as *powerful* as unrestricted resolution, in the sense that no resolution system admits refutations which are simpler than the simplest derivable by linear resolution. This latter property is shared by other refinements (notably minimal resolution, Section 5). Among the most powerful refinements of unrestricted resolution, linear resolution offers exceptional opportunities for the application of heuristic search by virtue of the relative uncomplicated structure of its search spaces.

Linear Derivations

A *linear derivation* **D**, from a set of clauses S, is a sequence of clauses $(C_1, \ldots, C_n)$ such that $C_1 \in S$ and each C_{i+1} is a resolvent of C_i (the *near parent* of C_{i+1}) and B, where either

(1) B is in S (the *input parent* of C_{i+1}), or

(2) B is some *ancestor* C_j of C_i, $j < i$, (the *far parent* of C_{i+1}).

C_1 is the *top clause* of **D** and C_n is the clause *derived* by **D**. In case (1), C_{i+1} is obtained by *input resolution* and, in case (2), by *ancestor resolution.* If **D** derives the null clause from S, then **D** is a *linear refutation* of S.

The sequence $\mathbf{D} = (PQ, Q, R, S, \bar{R}T, T, P, \square)$ is a linear refutation of $S = \{PQ, \bar{P}, \bar{Q}R, \bar{R}S, \bar{R}\bar{S}T, P\bar{T}\}$. (For examples, we omit the set theoretical brackets and commas employed in the representation of clauses as sets of literals.) Notice that only near parents of resolvents are displayed in linear derivations. The other parent of a resolvent can be regarded as an operator. Notice, too, that, in the preceding example, C_6 is obtained by resolving the near parent C_5 with its ancestor C_3. All other resolvents are obtained by input resolution. An example of a general-level linear derivation is the refutation

$$(\bar{P}(x)\bar{P}(a), R(a), Q(y), \bar{R}(y), \square) \text{ of}$$

$$\{\bar{P}(x)\bar{P}(a), P(x)R(a), \bar{R}(x)Q(y), \bar{Q}(y)\bar{R}(y)\}.$$

Search Trees

For linear derivations from S with a common top clause C_1, it is useful to organise the search space as a *search tree* $T = T(C_1)$:

(1) The linear derivation (C_1) is the *root* of T.

(2) If $\mathbf{D} = (C_1, \ldots, C_n)$ belongs to T then all linear derivations $(C_1, \ldots, C_n, C_{n+1})$ from S belong to T and are *immediate descendants* of **D**.

The complete linear resolution search space for S consists of all the search

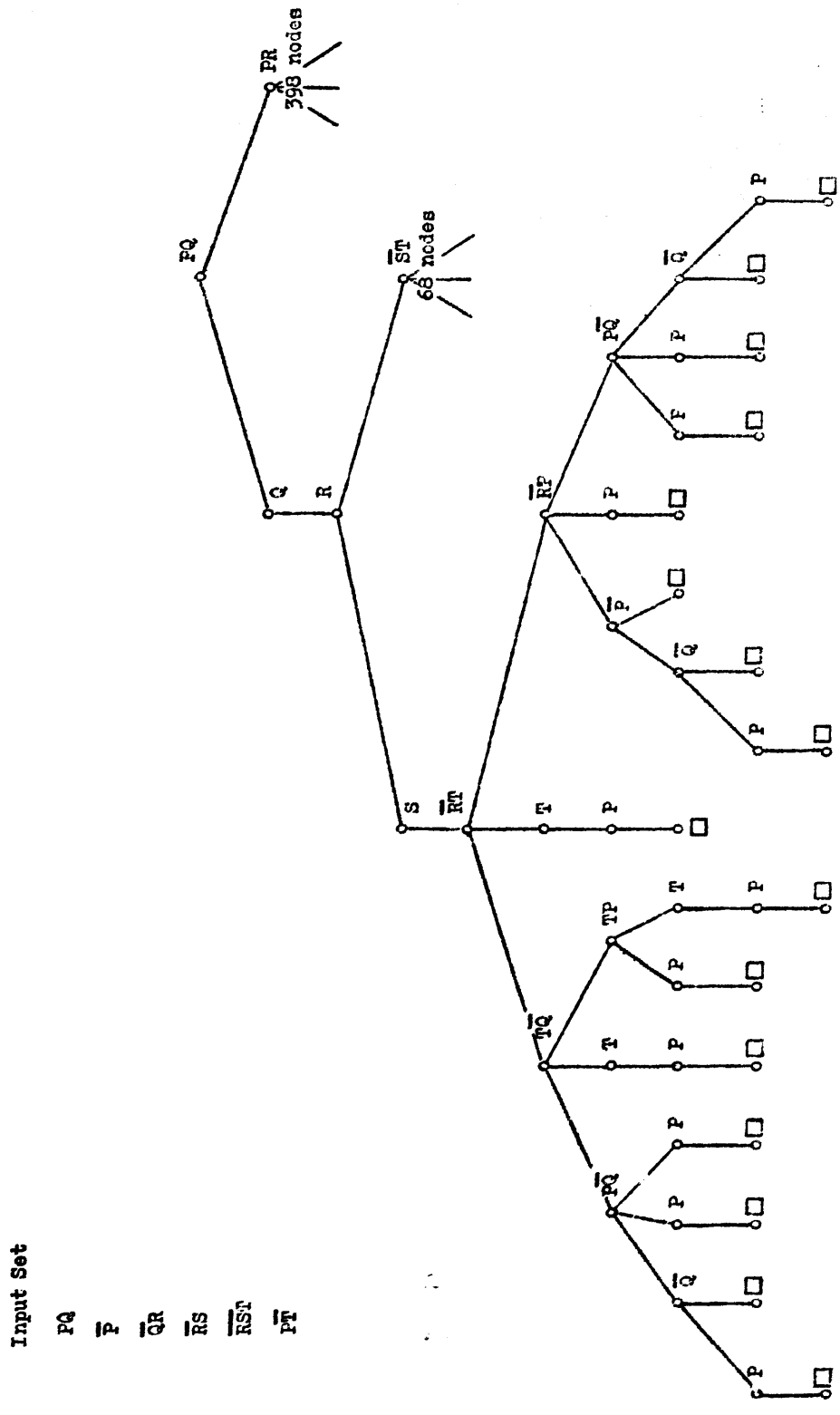

FIG. 1. Search tree for linear resolution (513 nodes).

trees $T(C_1)$ for each clause C_1 in the input set S. For the purpose of displaying a search tree T, each node is labelled by the clause derived by that node.

The search tree shown in Fig. 1 illustrates the great number of redundant derivations admitted by linear resolution. In general, the efficiency of proof procedures is improved by eliminating redundancies from the search space and by not significantly increasing the complexity of simplest proofs. It is to this end that we investigate refinements of linear resolution.

Refinements of Linear Resolution

It is possible to impose on linear resolution the restrictions that no resolvent is a tautology and that the top clause belongs to a given support set of the input set S. (A subset T of S is a *support set* of S if $S - T$ is satisfiable, Wos et al. [21].) Both restrictions eliminate unnecessary derivations without decreasing the power of linear resolution.

The support set restriction is especially useful because it limits the number of search trees which need to be investigated in the course of searching for a refutation. The easily recognisable support subsets of S include the set of all positive clauses, the set of all negative clauses, and the set of all clauses which come from the negation of the conclusion of the theorem (when the axioms and special hypotheses in S are satisfiable). For the example of Fig. 1, the top clause is the only clause in the support set of positive clauses. All of the refinements of linear resolution discussed in this paper are compatible with both the support set and no-tautologies restrictions.

Other restrictions which have been investigated for linear resolution include the s-linear restriction (Loveland [13] and Zamov and Sharanov [23]) and merging restrictions (Anderson and Bledsoe [1], Yates et. al. [22], and Kieburtz and Luckham [6]). The t-linear and SL-resolution systems investigated in this paper are both refinements of s-linear resolution with the support set and no-tautologies restrictions. The merging restriction does not seem to be a useful one and we have not investigated it in connection with SL-resolution. The following table compares, for various refinements, the size of a simplest proof and the number of derivations of the same or smaller size for the input set and top clause of the example of Fig. 1. The combination of linear resolution and the merging restriction defined in [1] is denoted by "m-linear";

	linear	s-linear	m-linear	ms-linear	t-linear	SL(1)	SL(2)
Size n of simplest refutation	6	6	7	7	6	7	6
Number of derivations of size $\leqslant n$	193	171	224	224	74	13	12

and the combination of *m*-linear resolution and the s-linear restriction, by "*ms*-linear." "SL(1)" and "SL(2)" denote SL-resolution with different selection functions. (The selection function chooses and resolves upon the alphabetically least atom for SL(1) and the alphabetically greatest atom for SL(2).) The selection function for SL-resolution acts in much the same way as the support set for set-of-support resolution: each specification of a selection function determines a complete SL-search space.

3. *t*-Linear Derivations

Definition

The three new restrictions incorporated in *t*-linear resolution are defined only for ground derivations from input sets of ground clauses.

Let $\mathbf{D} = (C_1, \ldots, C_n)$ be a ground linear derivation from S. A literal L in C_i *descends* from L in an ancestor C_j iff L occurs in every intermediate clause $C_k, j \leqslant k \leqslant i$. An ancestor C_j of C_i is an *A-ancestor* of C_i iff C_{j+1} has an input parent and all literals in C_j, except for the literal K resolved upon in obtaining C_{j+1}, have descendants in C_i. The literal K is called the *A-literal* of C_i from the A-ancestor C_j.

In the derivation $(PQ, Q, R, S, \bar{R}T, T)$ from the input set $\{PQ, \bar{P}, \bar{Q}R, \bar{R}S, \bar{R}\bar{S}T\}$, the derived clause, C_5, has A-ancestors C_2, C_3 and C_4 and A-literals Q from C_2, R from C_3 and S from C_4. C_5 is not an A-ancestor of C_6 because C_6 is not obtained by input resolution.

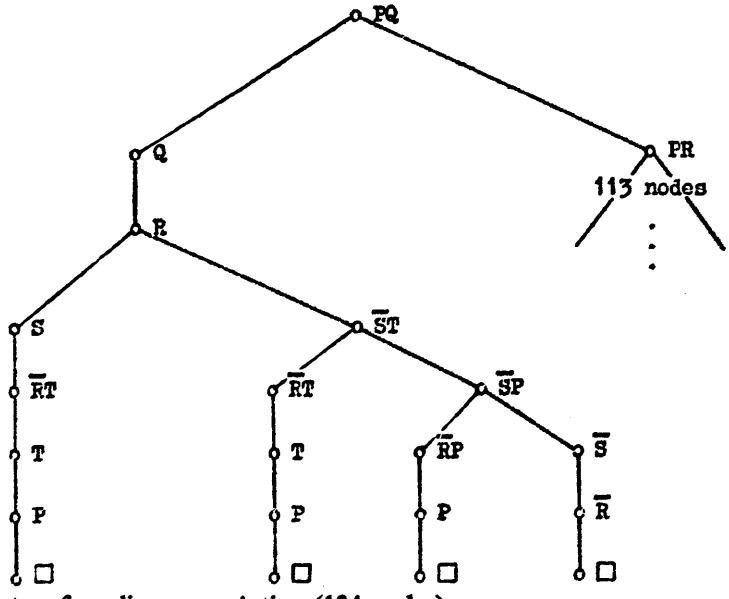

FIG. 2. Search tree for *t*-linear resolution (134 nodes).

A linear derivation **D** is *t-linear* if it satisfies the following three restrictions:

(1) If C_{i+1} is obtained by ancestor resolution, then it is obtained by resolution with an A-ancestor of C_i.

(2) If C_i contains a literal complementary to one of its A-literals, then C_{i+1} is obtained by ancestor resolution.

(3) A-literals of C_i from distinct A-ancestors have distinct atoms.

We have already remarked that the no-tautologies and support set restrictions are compatible with t-linear resolution. Fig. 2 illustrates part of the t-linear search space for the example of Fig. 1.

Remarks

(1) Notice that the first condition implies that if C_i resolves with an A-ancestor C_j then the literal resolved upon in C_j is the A-literal of C_i from C_j (for otherwise C_i would be a tautology). Thus the resolvent C_{i+1} is contained in its near parent. (This last property is Loveland's *s-linear restriction* [13] and Zamov and Sharonov's *absorption restriction* [23].) The second condition states that ancestor resolution is compulsory in the sense that it must be performed as soon as it can be performed.

(2) Clearly, for an efficient implementation of the t-linear restrictions, it would seem desirable to find an efficient way of associating with each clause C_i a list of its A-ancestors and A-literals. In fact, it is only necessary to associate A-literals, since all the other literals in A-ancestors are already contained in C_i. Restrictions (1) and (2) can then be implemented by simply deleting any literal in C_i which is complementary to an associated A-literal. The implementation of (3) is equally simplified. In the next section, we shall define a chain format for SL-derivations which provides just such a way of associating A-literals with clauses.

(3) It is instructive to compare ancestor resolution in t-linear derivations with the implicit merging operation. A single *merging operation* occurs when a literal in a resolvent occurs in both its parents. Thus the resolvent QR of PQR and $\bar{P}QR$ is obtained from its parents by resolution and two merging operations. The merging operation is implicit in the representation of clauses as sets of literals. If clauses were replaced by sequences of literals, the merging operation would need to be performed explicitly. So far, for t-linear resolution, ancestor resolution resembles the merging operation in that both remove a single literal from a clause and both are compulsory. For SL-resolution, the resemblance is more marked and both operations are treated as special cases of a single rule. For SL-derivations from sets of general clauses, ancestor resolution resembles factoring.

4. SL-Derivations

Informal Definition

SL-resolution is t-linear resolution with an additional restriction which calls for a single literal to be selected from each clause C_i in an SL-derivation. The

selected literal is the only literal in C_i which is ever resolved upon when C_i is used as near parent for input resolution. The choice of selected literal is constrained by the condition that it be a literal most recently introduced into the derivation of C_i. Thus, in the derivation (PQ, PR) only R may be selected in C_2, and therefore (PQ, PR, R) corresponds to no SL-derivation for any legitimate way of selecting literals.

For each derivation **D** in an SL-search tree, there is only one literal in the derived clause C, which is resolved upon in obtaining all immediate descendants by input resolution. If the same derivation occurs in a t-linear search tree then there are additional immediate descendants obtained by resolving on all other literals in C. Thus, if C contains m-literals, then there are, on the average, m times as many immediate descendants of **D** in the t-linear search tree as there are in the SL-search tree. If m is the average number of literals in clauses derived by t-linear derivations of size $\leqslant n$, then there are, on the average, m^n more t-linear derivations of size n than there are SL-derivations of the same size.

Fig. 3 illustrates, for the example of Fig. 1 and 2, the entire search tree for SL-resolution with the selection function which chooses the literal having alphabetically greatest atom.

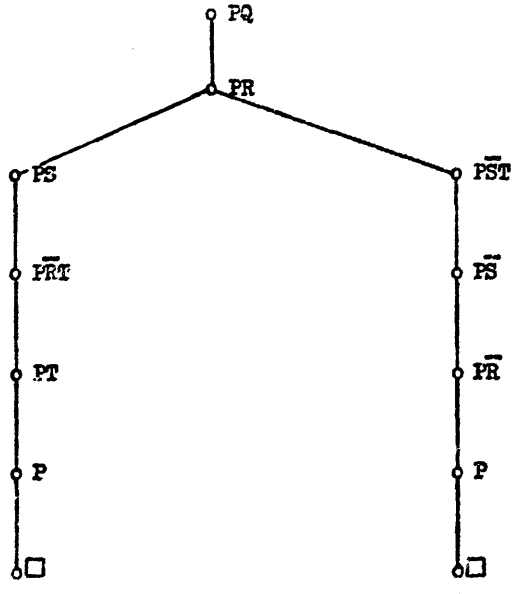

FIG. 3. Search tree for SL-resolution (12 nodes).

Notice that when a clause is used as near parent for ancestor resolution, the literal resolved upon is already constrained by the compulsory ancestor resolution restriction on t-linear derivations. Thus, in the clause $P\bar{R}T$ in Fig. 3, only the literal $\bar{R}$ may be resolved upon, even though both $\bar{R}$ and T are most recently introduced and T is alphabetically greater than R.

In the formal definition of SL-derivations, clauses are replaced by sequences of literals, called chains. When a near parent resolves with an input parent, the resolvent is obtained by concatenating literals from the near parent to the left of literals from the input parent. Between these two subsequences of literals we insert the selected literal resolved upon in the near parent. This literal is the *A*-literal of the resolvent from its near parent. More generally, each resolvent chain contains all of its *A*-literals. *A*-literals are deleted when they no longer belong to *A*-ancestors. Those literals in a chain which are not *A*-literals are called *B*-literals.

Fig. 4 illustrates in chain format the SL-search tree of Fig. 3. *A*-literals are enclosed in boxes. Merging operations are displayed explicitly. Of two identical literals in a chain, the rightmost is deleted. We underline literals resolved upon and also literals removed by the merging operation. The operation of deleting *A*-literals is not displayed, although defined explicitly in the formal definition.

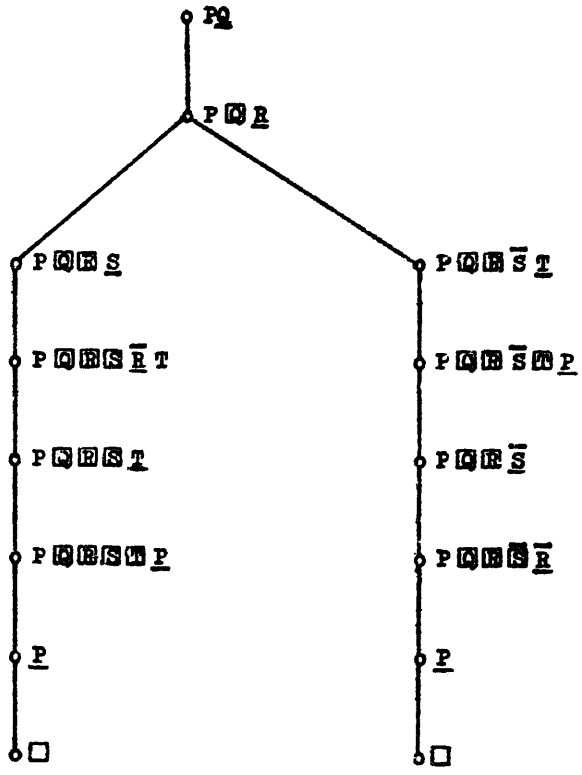

FIG. 4. Search tree for SL-resolution in chain format.

For the efficient implementation of the general resolution rule, it is useful to treat this single operation as a sequence of two suboperations: factoring of clauses and resolution of factors. If C is a clause and $\mathbf{E}$ a unifiable partition of the literals in C, having most general unifier (mgu) θ, then $C\theta$ is a *factor*

of C. If exactly one component of E contains two literals and every other component exactly one, then C is a *basic factor*. If $\{L\} \cup A$ and $\{\bar{K}\} \cup B$ are factors and the literals L and K are unifiable with mgu θ, then $(A \cup B)\theta$ is a resolvent. ($\cup$ denotes the union of disjoint sets.)

The definition of SL-resolution treats chains in the same way that separate and explicit rules for factoring and resolution of factors treat clauses. Altogether there are three operations which can be applied in order to obtain chains in SL-derivations. The extension operation is input resolution of factored chains. The reduction operation incorporates, as special cases, both basic factoring and ancestor resolution of factored chains. The truncation operation is a bookkeeping device for eliminating A-literals.

Formal Definition

Let S be a given input set of clauses. For each factor C of a clause in S and for each literal L in C, choose exactly one sequence C^* consisting of all literals in C, with L leftmost in C^*. C^* is an *input chain*. (Only the leftmost literal in C^* is resolved upon when C^* is input parent for an extension operation.) For the input set of clauses

$$\{\bar{P}(x)\bar{P}(a), P(x)R(a), \bar{R}(x)Q(y), \bar{Q}(y)\bar{R}(y)\},$$

there is only one corresponding set of 9 input chains. For

$$S = \{PQ, \bar{P}, \bar{Q}R, \bar{R}S, \bar{R}\bar{S}T, P\bar{T}\},$$

each corresponding set of input chains contains exactly 12 members. Each such set contains exactly one of $\bar{R}\bar{S}T$ and $\bar{R}T\bar{S}$, one of $\bar{S}\bar{R}T$ and $\bar{S}T\bar{R}$, and one of $T\bar{R}\bar{S}$ and $T\bar{S}\bar{R}$. For the purposes of SL-resolution, it is of no importance which one of these sets is chosen to specify the set of input chains.

In general, a *chain* is any sequence of literals, each of which is assigned the *status* of either A- or B-literal. All literals in input chains are *B-literals*. Two B-literals in a chain belong to the same *cell* if they are not separated by an A-literal. Thus the chain $P\boxed{Q}\boxed{\bar{R}}\bar{S}T$ has two cells: one containing only the B-literal P and the other, the rightmost cell, containing the B-literals $\bar{S}$ and T. The literal T is the rightmost literal in the chain.

Let ϕ be a function defined on non-empty chains, having chains as values. ϕ is a *selection function* iff $\phi(C^*)$ is C^* or can be obtained from C^* by interchanging the rightmost B-literal in C^* with another B-literal in the rightmost cell. Thus if C^* is $P\boxed{Q}\boxed{\bar{R}}\bar{S}T$ then $\phi(C^*)$ is $P\boxed{Q}\boxed{\bar{R}}T\bar{S}$ or C^* itself. The rightmost literal in $\phi(C^*)$ is the *selected literal* in C^*. (The extension operation applied to C^* resolves $\phi(C^*)$ on its rightmost B-literal with an input chain on its leftmost literal.) We require further that equivalent chains have the same selected literals.

For a given set of clauses S, support set S_0 and selection function ϕ, an *SL-derivation* from S is a sequence of chains $\mathbf{D}^* = (C_1^*, \ldots, C_n^*)$ satisfying (1)–(3).

(1) C_1^* is an input chain from S_0.

(2) Each C_{i+1}^* is obtained from C_i^* by one of extension, reduction or truncation.

(3) Unless C_{i+1}^* is obtained from C_i^* by reduction, then no two literals occurring at distinct positions in C_i^* have the same atom (*admissibility* restriction).

C_{i+1}^* is obtained from C_i^* by *truncation* iff (*a*) and (b):

(a) The rightmost literal in C_i^* is an A-literal.

(b) C_{i+1}^* is the longest initial subsequence of C_i^* whose rightmost literal is a B-literal. The status of a literal in C_{i+1}^* is the same as its status in C_i^*.

C_{i+1}^* is obtained from C_i^* by *reduction* iff (a)–(e);

(a) The rightmost literal in C_i^* is a B-literal.

(b) C_i^* is not obtained from C_{i-1}^* by truncation.

(c) The rightmost cell of C_i^* contains a B-literal L and either

(i) C_i^* contains a B-literal K, which is not in the rightmost cell of C_i^*, (basic factoring) or

(ii) C_i^* contains an A-literal $\bar{K}$, which is not the rightmost A-literal of C_i^*, (ancestor resolution).

(d) L and K are unifiable with mgu θ.

(e) Let C_i^{**} be obtained by deleting the given occurrence of L in C_i^*. Then $C_{i+1}^* = C_i^{**}\theta$. The status of a literal in C_{i+1}^* is the same as the status of the literal from which it descends in C_i^*.

C_{i+1}^* is obtained from C_i^* by *extension* with an input chain B^* iff (a)–(d):

(a) The rightmost literal in C_i^* is a B-literal.

(b) C_i^* and B^* share no variables.

(c) The selected literal L in C_i^* and the complement K of the leftmost literal $\bar{K}$ in B^* are unifiable with mgu θ.

(d) Let B^{**} be obtained by deleting the leftmost literal $\bar{K}$ from B^*. Then C_{i+1}^* is the chain $(\phi(C_i^*)B^{**})\theta$ obtained by applying θ to the result of concatenating $\phi(C_i^*)$ and B^{**} in that order. The literal $L\,\theta$ in C_{i+1}^*, descending from the rightmost literal in $\phi(C_i^*)$ is an A-literal in C_{i+1}^*. Every other literal in C_{i+1}^* has the same status as the literal from which it descends in C_i^* or B^{**}.

Remarks

(1) It is not difficult to verify that the admissibility restriction, together with (b) in the definition of reduction, incorporates the three restrictions on t-linear derivations as well as the compulsory merging and no-tautologies restriction. The effect of (b) is to guarantee that if a literal can be removed by reduction, then this is done before any extension operations are performed.

(2) The restrictions (c) (i) and (c) (ii) on reduction are both concerned with restrictions on the factoring operation. If reduction is performed with a *B*-literal K in the rightmost cell, then the effect of this factoring operation is to generate a chain already derivable by choosing a different factor for the input chain of the last extension operation. Similarly, if reduction is performed with the rightmost *A*-literal $\bar{K}$, then a variant chain can be derived without this reduction operation by using a different factor for the most recent input chain.

The factoring restrictions incorporated in the reduction operation correspond to restrictions which can be imposed on arbitrary resolution systems. The factoring method involved (called *m*-factoring [8]) imposes no constraints on the generation of factors of input clauses but allows only those factors of resolvents which do not involve the merging of literals which descend from the same parent. It is easy to show that *m*-factoring is the least redundant factoring method which generates short clauses as soon as possible and does not increase the complexity of derivations.

(3) The truncation operation can be eliminated and incorporated into more complicated definitions of extension and reduction. Nevertheless there is a good reason for treating it as a separate operation: The admissibility restriction applies to the parents of chains obtained by truncation.

(4) Case (ii) of the reduction operation does not, in fact, completely correspond to ancestor resolution in linear resolution systems. It corresponds, rather, to resolution with an instance of an ancestor. In linear resolution a clause C_i resolves with an ancestor C_j which is standardised apart to share no variables. The corresponding case of reduction in SL-resolution can be interpreted as resolving C_i^* with $C_j^*\,\theta$ where θ is the result of composing all mgu's generated in obtaining the sequence of chains C_{j+1}^* to C_i^*. Moreover, the resolvent C_{i+1}^* is obtained without renaming the variables which occur in its parents. This way of defining ancestor resolution can be applied to linear resolution systems in general and can be justified by resolution theoretic arguments. In the context of SL-resolution, it has several noteworthy advantages: it provides the most efficient and restrictive way of implementing ancestor resolution in SL-derivations, without in any way complicating simplest refutations. Moreover, it reflects on the general level the relationship between ancestor resolution and factoring which is the analogue of the relationship between ancestor resolution and merging for SL-derivation from sets of ground clauses.

Model Elimination

SL-resolution is more closely related to Loveland's model elimination system [12] than it is to other resolution systems. In particular, chain format,

A- and *B*-literals, extension, ancestor resolution reduction, and truncation all derive from model elimination. (We have used Loveland's terminology, except for " contraction" which we have renamed "truncation" in order to distinguish it more easily from "reduction".)

SL-resolution differs from model elimination primarily in that, for ground derivations, model elimination has no merging operation. At the general level, a limited amount of factoring is obtained in model elimination by allowing ancestor resolution with rightmost *A*-literals. For these reasons, only a weakened version of the admissibility restriction holds for model elimination.

Although not explicitly incorporated in Loveland's original definition, it is easy to verify that compulsory ancestor resolution is compatible with model elimination. For certain restricted selection functions, resolution with selected literals is already incorporated in model elimination. (The selected literal is the rightmost literal in a chain and is determined, therefore, by the initial choice of input chains.) The compatibility of the more liberal employment of selection functions can be established for model elimination by the same method used for SL-resolution.

It is not difficult to show that, in most cases, SL-resolution yields simpler refutations and fewer unnecessary derivations than model elimination. (The anomalous case arises when a simplest SL-refutation involves no basic factoring reduction operations and these operations are performed in unnecessarily generated SL-derivations.)

In the next section, we compare the power of SL-resolution with that of other resolution systems. Comparison of these systems with model elimination will not be investigated beyond that which is implied by the preceding comparison of SL-resolution with model elimination. The preliminary investigations reported in this paper suggest that the study and implementation of model elimination procedures have been unprofitably neglected in favour of less efficient resolution procedures.

5. Complexity

In order to investigate the complexity of linear and SL-refutations, we shall compare them with minimal refutations. Minimal refutations include the simplest obtainable by any resolution system. Moreover, every minimal refutation (whether simplest or not) can be regarded as reasonably simple for the theorem it proves. We show that for every minimal refutation there exists an *s*-linear refutation of the same complexity for the same set of clauses; and for every unsatisfiable set of clauses there exists an SL-refutation as simple as some minimal refutation. Proof outlines for the three theorems, which establish these results, are included in the appendix.

Minimal Derivations

A *non-linear derivation* is a tree of nodes labelled by clauses, which are said to be *at* the nodes. Clauses from the input set are at the *tips* and the derived clause is at the *root*. Every node which is not a tip is labelled by a resolvent of the clauses at the immediate predecessor nodes. A literal is *resolved upon* at a node if it occurs in the clause at that node and is removed when obtaining the resolvent at the immediate descendant node.

A *branch* of a non-linear derivation is a set of nodes consisting of a single tip and the immediate descendant of every node, except the root, which is contained in the set. The number of nodes in a derivation, which are not tips, is the *size* of the derivation. Its *level* is the number of non-tip nodes contained in a largest branch.

A ground non-linear refutation is *minimal* if, for every branch, the literals resolved upon at distinct nodes have distinct atoms. A ground non-refutation is *minimal* if it can occur as a subderivation of a minimal ground refutation; i.e., if it derives a non-tautology and, for every literal resolved upon at a node, its atom does not occur in any clauses at a descendant node. A general derivation is *minimal* if it lifts a minimal ground derivation; i.e. is tree-isomorphic, the clause at any node has as an instance the clause at the corresponding node, etc.

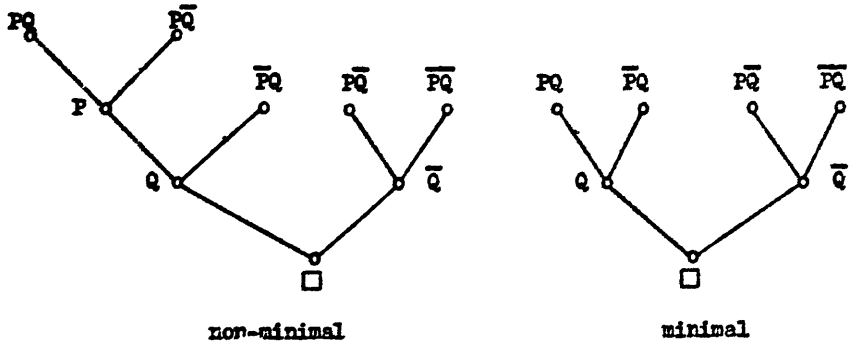

FIG. 5. Non-linear refutations of $\{PQ, \bar{P}Q, P\bar{Q}, \bar{P}\bar{Q}\}$.

Fig. 5 illustrates minimal and non-minimal refutations of the same input set. The minimal refutation has 4 branches, size 3 and level 2; the non-minimal refutations, 5 branches, size 4 and level 3. The literal Q is resolved upon twice in the leftmost branch of the non-minimal derivation.

If a ground set S of clauses contains exactly n distinct atoms, then there are only finitely many minimal derivations from S, none of which has level greater than n or size greater than 2^n-1. Under quite general conditions on S (which apply, in particular to the example of Fig. 5) there are infinitely many non-minimal derivations and refutations of unbounded level and size.

(The conditions are that some minimally unsatisfiable subset of S contains at least two clauses containing a literal L and two other clauses containing $\bar{L}$.)

The notion of minimal derivations was introduced by Loveland [13] and investigated independently by Kowalski [8] in conjunction with Pat Hayes. Minimal derivations are just those derivations which can be obtained by the construction of semantic trees (Hayes and Kowalski [4]). Loveland defines a ground derivation to be minimal if it cannot be "pruned". The two definitions are not equivalent. Every unprunable derivation is minimal in our sense, but not conversely. It follows from Loveland's Corollary 2 that there exist minimal refutations as simple as the simplest obtainable by any resolution system (Theorem 1 below).

rm-size

Ancestor resolution in linear derivations resembles the factoring (and merging) operation more closely than it does the resolution operation. For this reason, the size of derivations is not entirely appropriate for comparing the complexities of linear with non-linear derivations.

We define the *rm*-size of a non-linear derivation to be the pair (r,m) where m is the number of basic factoring (and merging) operations performed in the derivation and r the number of resolution (of factors) operations.[1] For a linear derivation, the *rm*-size is (r,m) where r is the number of input resolution operations and m the number of both ancestor resolution and basic factoring (and merging) operations. (For an SL-derivation, r is just the number of extension operations and m the number of reduction operations.) In Fig. 5, the minimal derivation has *rm*-size (3,2) and the non-minimal derivation *rm*-size (4,2). In Fig. 4, both SL-refutations have *rm*-size (5,2)

For both linear and non-linear derivations, we do not include in m the number of initial factoring operations applied to input clauses. For linear (but not SL-) derivations, the definition of *rm*-size is deliberately ambiguous when a near parent resolves with a top clause, which can be treated as either an input or far parent.

Simplest Refutations

If complexity is defined as any function of r and m then two derivations (linear and (or) non-linear) have the same complexity if they have the same *rm*-size. In order to compare the complexities of derivations having different *rm*-sizes, we shall assume only that complexity is non-decreasing with increasing r and m and that an increase in m does not increase complexity more than the same increase in r. More precisely, if $(r_1,m_1) \leqslant (r_2,m_2)$ means that no derivation of *rm*-size (r_1,m_1) is more complex than one of *rm*-size (r_2,m_2) then the assumptions are that

[1] i.e., the number of non-tip nodes in the derivation.

$$r_1 \leqslant r_2 \text{ and } m_1 \leqslant m_2 \text{ imply } (r_1,m_1) \leqslant (r_2,m_2), \text{ and}$$
$$(r,m) \leqslant (r+n, m-n).$$

If we were to assume that $(r,m) < (r+n, m-n)$ then Theorem 1 could be strengthened to assert that every simplest non-linear refutation is minimal.

THEOREM 1. *For every unsatisfiable set of clauses there exists a simplest refutation which is also minimal.*

THEOREM 2. *For any unsatisfiable set S and support subset S_0 there exists an s-linear refutation of S with top clause in S_0 such that no non-linear refutation of S is simpler.*

(To prove Theorem 2, it is necessary to verify that any ground *s*-linear refutation can be lifted to a general *s*-linear refutation of the same *rm*-size. This verification fails unless ancestor resolution is defined for linear derivations in a manner similar to that for SL-derivations and mentioned in Remark (4) of the preceding section. The effect of such a definition is to yield a lower value for *m* in the calculation of *rm*-size.)

The Complexity of SL-Refutations

The simplest SL-refutation of a set of clauses may be more complex than a refutation obtainable in some other resolution system. Theorem 3 establishes that the complexity of a most complex minimal refutation is a bound on the complexity of a simplest SL-refutation.

THEOREM 3. *For every unsatisfiable set S, support subset S_0 and selection function ϕ, there exists an SL-refutation of S which has the same rm-size as some minimal refutation of S.*

Better bounds can be obtained for special cases. We conjecture that an improved bound can also be established for the general case. It is easy to verify that, for every unsatisfiable set of two-literal ground clauses *S*, no SL-refutation has *rm*-size worse than $(2n-1,2)$ where *n* is the number of distinct atoms occurring in *S*. On the other hand, for each *n* there exists an unsatisfiable set of two literal clauses *S* and a minimal refutation of *S* with *rm*-size $(2^n-1,2)$.

We have only found one example of a set *S* such that no selection function or support set yields an SL-refutation as simple as can be obtained by unrestricted, minimal or *s*-linear resolution: For

$$S = \{L\bar{M}, \bar{L}\bar{P}, L\bar{Q}, \bar{L}\bar{R}, NMQ, NPR, \bar{N}\bar{T}, T\}$$

a simplest refutation has *rm*-size (7,3). The simplest SL-refutation obtainable has *rm*-size (9,2), (10,4), (11,3), (12,3), (14,2) or (15,1) depending on the specification of selection function and support set.

We have not found any examples where SL-resolution significantly increases the complexity of a simplest proof. For a number of other systems it is easy to construct refutations which are the simplest obtainable by those systems and which exceed in complexity the bound established for SL-refutations. In particular, for $S = \{PQ, P\bar{Q}, \bar{P}Q, \bar{P}\bar{Q}\}$, P_1 – deduction yields as simplest proof no refutation of *rm*-size better than (4,2). All minimal and SL-refutations of S have *rm*-size (3,2). For the same set of clauses, resolution with any singleton set of support also yields simplest proofs more complex than minimal refutations. It is an open question whether the complexity of simplest proofs obtainable by *m*-linear resolution exceed the bound of the complexity of minimal refutations. Our analysis of the completeness proofs for *m*-linear resolution yields bounds on complexity which are worse than have been established for SL-resolution.

Proof Procedures for SL-Resolution

Heuristic considerations can be used for the construction of the search spaces and search strategies involved in SL-resolution proof procedures. Heuristics for choosing selection functions and support sets apply both to the determination of SL-search trees and to the and/or tree search spaces obtained by the generation of goals and subgoals. For both representations of search spaces, efficient search strategies can be constructed by employing length-of-chain as a heuristic, by employing strategies for the deletion of subsumed chains and by generating lemmas.

The Specification of Support Sets and Selection Functions

For a given input set, the search space for SL-resolution is determined by the specification of a support subset and a selection function. Heuristic criteria can be applied in both these cases with the goal of reducing the branching rate at nodes of the SL-search trees. Since the choice of support set and selection function does not affect the bound on the complexity of simplest SL-refutations, consistent reduction of branching rates results in an overall reduction of the size of the subspace which needs to be generated before finding a first refutation. (This assumes that the search strategy itself favours the generation of simplest refutations in preference to more complex ones. The same assumption is necessary for proofs of increased efficiency in other cases which have been investigated ([8]).)

In general, support sets containing a small number of clauses are preferable to those containing more. The choice of small support sets improves efficiency by reducing the number of search trees which have to be examined by the search strategy.

The choice of selected literals (and therefore of selection function) need

not be fixed in advance of the generation of chains by the search strategy. This choice can be made dynamically and be deferred until the search strategy first considers using a chain as near parent for application of the extension operation. At that time, the heuristic selects a literal in the chain which can be resolved upon with the least number of input chains. Good estimates of this number can be calculated quickly for each B-literal in the rightmost cell of the chain by employing a classification of input chains, arranged in the form of a classification tree.

Classification Trees

For each literal which can be encountered in a chain, there corresponds exactly one branch of a classification tree. With the tip of this branch is associated all those input chains which might resolve with a literal corresponding to the branch.

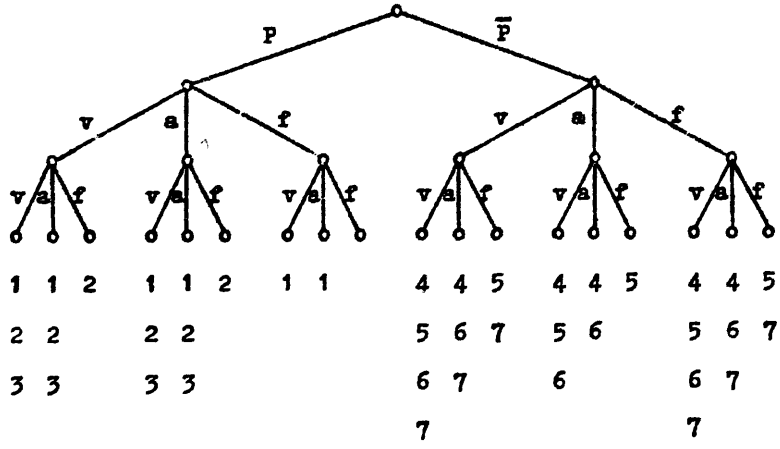

FIG. 6. An operator classification tree.

The classification tree of Fig. 6 classifies input chains for the unsatisfiable set of clauses

$$S = \{\bar{P}(x,a)\bar{P}(a,x),\ P(x,a)P(x,f(x)),\ P(x,a)P(f(x),x)\}.$$

The corresponding set of input chains has 7 members

$$B_1^* = \bar{P}(x,a)\bar{P}(a,x), \qquad B_2^* = \bar{P}(a,x)\bar{P}(x,a), \qquad B_3^* = \bar{P}(a,a),$$
$$B_4^* = P(x,a)P(x,f(x)), \qquad B_5^* = P(x,f(x))P(x,a),$$
$$B_6^* = P(x,a)P(f(x),x), \qquad B_7^* = P(f(x),x)P(x,a).$$

The two arcs branching from the root test, from left to right, whether a literal is positive or negative. The three arcs branching from the nodes immediately below the root test whether the first argument place contains a variable, the constant a, or a term beginning with the function symbol f. The three arcs branching from the nodes just above the tips test whether the

second argument place contains a variable, the constant a, or a term beginning with f. The column of numbers at the tip of a branch contains the subscripts of just those input chains which can resolve (on their leftmost literal) with some literal which passes the tests for all arcs along the branch. Thus, for instance, only the input chains B_5^* and B_7^* could possibly be used for extension with a selected literal of the form $\bar{P}(f(s),f(t))$. No input chain resolves with a literal of the form $P(f(s),f(t))$.

The complete SL-search tree, with the top chain B_4^* and for the selection function determined by the classification tree of Fig. 6, is shown in Fig. 7. Above each B-literal in a rightmost cell is written the number of input chains associated in the classification tree with the branch corresponding to the literal. The selected literal is the one having the smallest number written above it, and is underlined. If the other literal were selected in the top clause then the corresponding complete SL-search tree would contain 17 nodes instead of 9.

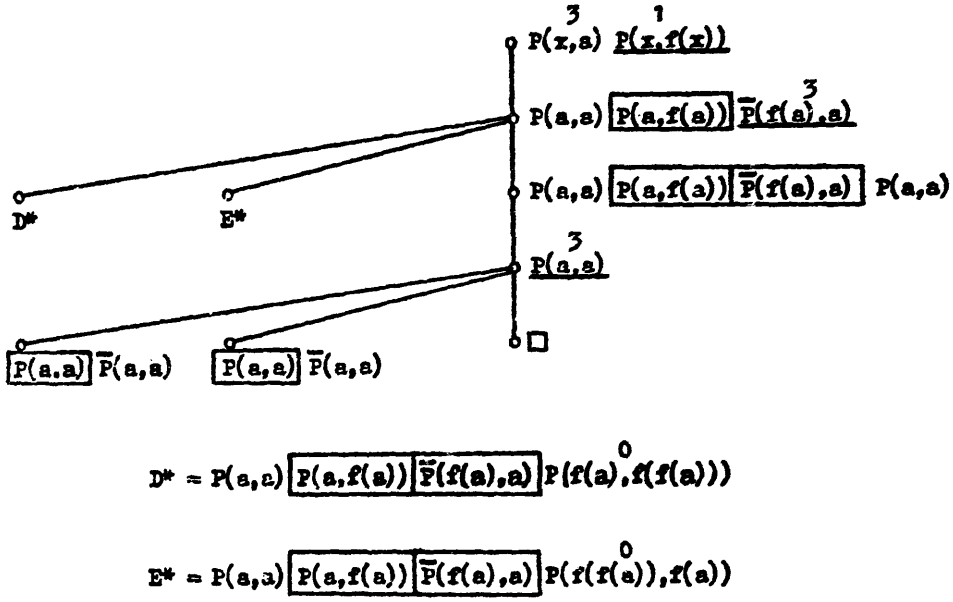

FIG. 7. Selecting literals, using Fig. 6.

The notion of classification tree can be usefully extended in a number of directions. Our experience with these trees encourages us to believe that they will become an essential and increasingly important part of SL-resolution proof procedures.

Search Strategies

All of the search strategies which have been investigated for resolution systems can be regarded as based on a merit ordering of derivations in the

search space. At any given step, the search strategy generates from among all derivations, which are candidates for generation, a derivation of best merit. A derivation is a *candidate* for generation if it has not been generated but its immediate subderivations have. Two search strategies based on the same merit ordering differ only with respect to tie-breaking rules, which choose a single derivation for generation out of a number of equally meritorious candidates. For a given ordering to be a *merit ordering*, we insist only that, for any two derivations, it can be decided whether they have equal merit or whether one has better merit than the other.

A search strategy is *exhaustive* if it potentially (if left to continue without termination) generates all derivations in the search space. A merit ordering is *δ-finite* if, for every derivation in the search space, only finitely many derivations have better or equal merit. Any search strategy based on a δ-finite merit ordering is exhaustive [7].

If, for a given measure of complexity, merit is defined so that one derivation has better merit than another iff it is simpler, than a search strategy based on such a merit ordering is called a *complexity saturation* strategy. Most measures of complexity determine δ-finite merit orderings and therefore determine complexity saturation strategies which are exhaustive. Provided only that no derivation is simpler than any of its subderivations, then every such complexity saturation strategy always generates a first refutation which is simplest in the search space.

It is interesting to note that the three basic proof procedures, outlined by Loveland [12], for model elimination, all employ some form of δ-finite complexity saturation search. The first procedure is saturation for the complexity of a derivation measured by the number of extension operations performed in it; the second, for complexity measured by the largest number of A-literals occurring in any chain in the derivation; the third is identical to the second, except that lemmas can be constructed and used as input chains for extension. The use of lemmas with SL-resolution will be discussed briefly at the end of this section.

The efficiency of complexity saturation can be significantly improved by diagonal search, which is an extension of the Hart-Nilsson-Raphael algorithm for path-finding in graphs ([3] and [7]). For a given complexity measure g and heuristic function h (both non-negative real-valued functions defined on derivations), a *diagonal search strategy* (called upwards diagonal in [7]) is any search strategy based on the merit ordering:

D has *better merit* than **D′** iff

$$g(\mathbf{D})+h(\mathbf{D})<g(\mathbf{D}')+h(\mathbf{D}'), \text{ or}$$
$$g(\mathbf{D})+h(\mathbf{D}) = g(\mathbf{D}')+h(\mathbf{D}') \text{ and } h(\mathbf{D})<h(\mathbf{D}').$$

The value $h(\mathbf{D})$ of the *heuristic function* is intended to be an estimate of the

additional complexity (in addition to $g(\mathbf{D})$) of a simplest refutation containing D as a sub-derivation.

Suppose that the complexity of an SL-derivation $\mathbf{D}^*$ of *rm*-size (r,m) is defined as a weighted sum $ar+bm$, where $a \geqslant b$. Suppose that the derived chain has l_1+l_2 *B*-literals, l_2 of which belong to the rightmost cell. Let $h(\mathbf{D}^*) = al_1+bl_2$. Then $h(\mathbf{D}^*)$ is a lower bound on the additional complexity (in addition to $g(\mathbf{D}^*)$) of a simplest SL-refutation containing $\mathbf{D}^*$. (Thus, for example, if $\mathbf{D}^*$ has *rm*-size (r,m) and derives $PQ\boxed{R}S\boxed{T}UV$, then no SL-refutation which extends $\mathbf{D}^*$ can have *rm*-size less than $(r+3, m+2)$.) It can easily be shown (as in [7]) that diagonal search is easily implemented, that it always generates a first refutation which is simplest in the search space, and that it does so by generating significantly fewer unnecessary derivations than are generated by complexity saturation. The following table compares the numbers of derivations generated by complexity saturation and diagonal search, for the example and refinements in the table of section 2. For both strategies, complexity is defined by $r+m$.

	linear	*s*-linear	*m*-linear	*ms*-linear	*t*-linear	SL(1)	SL(2)
Complexity Saturation	282	224	357	357	95	13	14
Diagonal Search	42	42	171	171	40	11	12

A useful rule, for helping to decide between the generation of candidate derivations of equal best merit, can be applied in diagonal search following a suggestion of Loveland for model elimination: Generate a derivation of best merit whose derived chain contains the greatest number of A-literals. Such a derivation offers the greatest possibilities for eliminating B-literals by reduction and therefore for eliminating B-literals in the course of generating an empty chain.

Deletion of Subsumed Chains

Among the methods most often used in resolution proof procedures are strategies for the deletion of subsumed clauses. Corresponding methods can be applied in SL-resolution for the deletion of subsumed chains. Deletion strategies need to be defined carefully in order to preserve completeness and even then cannot always be guaranteed to increase efficiency.

Two chains are said to be *equivalent* if either can be obtained from the other by permuting the order of *B*-literals in cells. (Thus $PQ\boxed{R}S\boxed{T}UV$ is equivalent to a total of four distinct chains which include, for instance, both itself and $QP\boxed{R}S\boxed{T}VU$. It is not equivalent to $PS\boxed{R}Q\boxed{T}UV$.) A chain C^*

subsumes another C'^* if some instance $C^*\sigma$ is an initial subchain of a chain equivalent to C'^*. (Thus $\boxed{P(x)}Q(x)$ subsumes both $\boxed{P(a)}Q(a)R(a)$ and $\boxed{P(a)}R(a)Q(a)$ but not $Q(a)\boxed{P(a)}R(a)$.)

Let Σ by any search strategy for SL-resolution, Σ can be modified to obtain a new strategy Σ' which step by step generates the same derivations as Σ, in the same order, but deletes derivations of subsumed chains and does not generate derivations which extend previously deleted derivations:

(1) Both search strategies generate the same first SL-derivation.
(2) If Σ generates a derivation, then Σ' generates the same derivation provided that its immediate subderivation has been generated by Σ' and has not been previously deleted.
(3) If Σ' generates a derivation $\mathbf{D}^*$ of a chain C^* then
   (a) $\mathbf{D}^*$ is deleted if C^* is subsumed by the chain derived by some previously generated and undeleted derivation,
   (b) otherwise every previously generated and undeleted derivation of a chain, subsumed by C^*, is deleted.

The search strategy Σ' is complete, relative to Σ, i.e., Σ' eventually generates a refutation if Σ does.

Deletion of subsumed clauses can be defined for other resolution systems in a manner analogous to the preceding definition for SL-resolution. In the case of P_1-deduction, for instance, an incomplete deletion strategy is obtained by interchanging the analogues of steps (3a) and (3b). In general, if step (3b), or its analogue, is omitted, then increased efficiency can be guaranteed for any search strategy Σ which generates a first refutation which is simplest for its search space. The inclusion of (3b) is a possible source of decreased efficiency. Although deletion of subsumed chains and clauses seems to be a desirable addition to proof procedures, we have not found good modifications of (3b) or restrictions on Σ which always guarantee the increased efficiency of incorporating such deletion rules. A more thorough investigation of these problems for non-linear resolution systems is contained in [8].

Generation of Subgoals and Lemmas

Possibilities for the generation of subgoals and for the processing of their solutions in the form of lemmas are unique to SL-resolution and model elimination. To avoid various complications, we shall discuss in detail only the case of ground SL-resolution.

Suppose that a derivation of a chain C^* has been generated and that no truncation or reduction operation can be applied to C^*. It is easy to verify that if $\phi(C^*) = C_0^*L$ then C_0^* must occur as a descendant of C^* in any SL-refutation containing C^*. Thus the *goal* $(C^* \rightarrow \square)$ of deriving the null chain from C^* can be decomposed into the *immediate subgoal* $(C_0^*L \rightarrow C_0^*)$ of

deriving C_0^* from C^* and the further goal $(C_0^* \rightarrow \square)$ of deriving the null chain from C_0^*. The solution of the immediate subgoal determines a lemma which can be reused to solve *analogous immediate subgoals* of the form $(C_0'^* L \rightarrow C_0'^*)$.

For example, the goal $(N\boxed{P}QR \rightarrow \square)$ can be *completely decomposed* to obtain the immediate subgoals $(N\boxed{P}QR \rightarrow N\boxed{P}Q)$, $(N\boxed{P}Q \rightarrow N\boxed{P})$ and $(N \rightarrow \square)$. The derivation

$$(N\boxed{P}QR,\ N\boxed{P}Q\boxed{R}S,\ N\boxed{P}Q\boxed{R}\|\boxed{S}\bar{R},\ N\boxed{P}Q)$$

is a solution to the immediate subgoal $(N\boxed{P}QR \rightarrow N\boxed{P}Q)$. Having solved such a subgoal, the fact can be recorded and applied later for solving analogous immediate subgoals such as $S\boxed{T}R \rightarrow S\boxed{T})$. In particular, we may generate the *lemma* $\bar{R}$ which can be used as input chain for extension. If the solution to $(N\boxed{P}QR \rightarrow N\boxed{P}Q)$ were

$$(N\boxed{P}QR,\ N\boxed{P}Q\boxed{R}S,\ N\boxed{P}Q\boxed{R}\|\boxed{S}\bar{P},\ N\boxed{P}Q),$$

then the corresponding lemma would be $\bar{R}\bar{P}$ and could be restricted in application to those analogous subgoals $(C_0^* R \rightarrow C_0^*)$ where C_0^* contains P as A-literal or $\bar{P}$ as B-literal.

The preceding examples of lemma construction are easy to generalize (see, for instance, Loveland [12]). The restricted use of such lemmas can be shown to increase efficiency by always leading to the generation of fewer unnecessary derivations before the generation of a first refutation.

And/or Tree Search Space

The consistent application of subgoal generation leads to an *and/or tree representation* of the search space for SL-resolution.

(1) The top chain C_1^* at the root of an SL-search tree is replaced by the goal $(C_1^* \rightarrow \square)$ at the *and-node* which is the *root* of the corresponding and/or tree.

(2) Each *and-node*, labelled by a goal of the form $(C^* \rightarrow C'^*)$ has as many immediate descendant *or-nodes* as there are B-literals in $C^* - C'^*$. These or-nodes are labelled by the immediate subgoals obtained by complete decomposition of $(C^* \rightarrow C'^*)$.

(3) Each *or-node*, labelled by an immediate subgoal $(C_0^* L \rightarrow C_0^*)$, has as many immediate descendant *and-nodes* as there are ways of applying extension to $C_0^* L$ with input chains of the form $\bar{L}B_0^*$. Each such and-node is labelled by the corresponding goal $(C_0^*\boxed{L}B_0^* \rightarrow C_0^*)$.

The precise details for dealing with truncation and reduction can be formulated without great difficulty.

Problems arise with the implementation of and/or trees for general sets of input clauses, because the solution of subgoals cannot be accomplished independently. This complication can be dealt with and search strategies can be designed for searching and/or trees. Such strategies can synthesise the use of heuristics for estimating branching rates at *and-nodes* and *or-nodes*. In addition, the use of lemmas and classification trees can be incorporated with other methods in order to obtain search strategies which employ look-ahead and learning to estimate the difficulty of solving goals and subgoals. Such estimates can be improved by straightforward methods for comparing estimated with actual difficulty.

Search strategies suggested by the and/or tree representation can be translated to SL-search trees. The general topic of synthesising the advantages of the several methods remarked upon in this section is a promising area for further improving the efficiency of SL-resolution.

Conclusions

In this paper, we have attempted to support our belief that SL-resolution offers a substantial contribution to the more efficient mechanisation of first order logic. We have argued that SL-resolution achieves a significant reduction in the generation of unnecessary derivations without intolerably complicating simplest proofs. Moreover, the amenability of SL-resolution to the application of heuristic methods suggests that, on these grounds alone, it is at least competitive with theorem-proving procedures designed solely from heuristic considerations.

Arguments for SL-resolution can be extracted from a broader basis. Regarded as either a model elimination or linear resolution system, it seeks to incorporate the best features of both systems in a way which improves upon the original. Still other proof procedures, such as the inverse method (Maslov [16]) can be compared with resolution (Kuehner [9]) to obtain a further comparison in the favour of SL-resolution. Other arguments for resolution systems in general (Kowalski [8]) apply to SL-resolution in particular.

It is interesting to note that none of the preceding arguments for SL-resolution appeal directly to its completeness and that some of the more convincing ones rely upon its heuristic attraction.

We do not pretend that SL-resolution solves the problems of automatic theorem-proving. The intelligent performance of deductive activity involves numerous sub-activities. These sub-activities include learning theorems and proofs, formulating worthwhile conjectures, searching for proofs and counter-examples, correcting faulty theorems, proofs and counter-examples, and improving successful ones. These sub-activities so depend upon one another

that we do not expect efficient theorem-proving to be realised in isolation from the remainder of intelligent activity.

Despite its present shortcomings, we remain optimistic for the continuing progress of theorem-proving based upon SL-resolution. The applicability of classification trees and of subgoal and lemma generation were largely unanticipated in our early investigations and have not yet been fully exploited. Our continuing optimism seems to be justified by past experience and by more recent developments which have not yet been formulated in enough detail to include in this paper.

ACKNOWLEDGMENTS

We are grateful for the advice and encouragement of our colleagues, Pat Hayes and Bernard Meltzer, of the Metamathematics Unit. We are indebted also to the postgraduate students, in the Department of Machine Intelligence and Perception, who directly or indirectly suggested improvements or discovered errors in our earlier presentation. In these respects, we owe special thanks to Michael Gordon, Gordon Plotkin and Ed Wilson. We thank Donald Loveland, Nils Nilsson and Ed Wilson for reading the earlier draft of this paper and for making observations which eventually led to this revision.

During the course of this research, Robert Kowalski was supported by a Science Research Council grant to the Metamathematics Unit. During 1970/71, Donald Kuehner was supported by an IBM fellowship awarded by Imperial College.

REFERENCES

1. Anderson, R., and Bledsoe, W. W. A linear format for resolution with merging and a new technique for establishing completeness. *J. ACM* **17** (July 1970), 525–534.
2. Andrews, P. B. Resolution with merging, *J. ACM* **15** (1968) 367–381.
3. Hart, P. E., Nilsson, N. J. and Raphael, R. A formal basis for the heuristic determination of minimum cost paths. *I.E.E.E. Transactions on System Sciences and Cybernetics* (July 1968).
4. Hayes, P. J. and Kowalski, R. A. Semantic trees in automatic theorem proving. *Machine Intelligence* **4**, Edinburgh University Press, 1969, pp. 87–101.
5. Hayes, P. J. and Kowalski, R. A. Lecture notes on automatic theorem-proving, Metamathematics Unit Memo. 40, University of Edinburgh (March 1971).
6. Kieburtz, R. and Luckham, D. Compatibility of Refinements of the Resolution Principle (1969).
7. Kowalski, R. A. Search strategies for theorem-proving. *Machine Intelligence* **5**, Edinburgh University Press, 1970, pp. 181–201.
8. Kowalski, R. A. Studies in the completeness and efficiency of theorem-proving by resolution. Ph.D. Thesis, University of Edinburgh, 1970.
9. Kuehner, D. G. A note on the relation between resolution and Maslov's inverse method. *Machine Intelligence* **6**, Edinburgh University Press, 1971, pp. 73–76.
10. Kuehner, D. G. Strategies for improving the efficiency of automatic theorem-proving. Ph.D. Thesis, University of Edinburgh, 1971.
11. Kowalski, R. A. and Kuehner, D. G. Linear resolution with selection function. Metamathematics Unit Memo 34, University of Edinburgh, (October 1970).
12. Loveland, D. W. A simplified format for the model-elimination theorem-proving procedure. *J. ACM* **16**, (July 1969), 349–363.

13. Loveland, D. W A linear format for resolution. *Symposium on Automatic Demonstration, Lecture Notes in Mathematics*, **125**. Springer-Verlag, Berlin and New York, 1970, pp. 147–163.
14. Loveland, D. W. Some linear Herbrand proof procedures: an analysis. Department of Computer Science, Carnegie-Mellon University (December 1970).
15. Luckham, D., Refinement theorems in resolution theory. *Symposium on Automatic Demonstration, Lecture Notes in Mathematics* **125**. Springer-Verlag, Berlin and New York, 1970, pp. 163–191.
16. Maslov, S. J. Proof-search strategies for methods of the resolution type. *Machine Intelligence* **4**. Edinburgh University Press, 1971, pp. 77–90.
17. Meltzer, B. Prolegomena to a theory of efficiency of proof procedures. *Proceedings of the NATO Advanced Study Institute on Artificial Intelligence and Heuristic Programming*, Edinburgh University Press, 1971, pp. 15–33.
18. Prawitz, D. Advances and problems in mechanical proof procedures. *Machine Intelligence* **4**. Edinburgh University Press, 1969, pp. 59–71.
19. Reiter, R. Two results on ordering for resolution with merging and linear format. Department of Computer Science, University of British Columbia (July 1970).
20. Robinson, J. A. A review of automatic theorem-proving. *Proceedings of Symposia in Applied Mathematics* **19**, (1967), 1–18.
21. Wos, L. T., Carson, D. F. and Robinson, G. A. Efficiency and completeness of the set of support strategy in theorem-proving, *J. ACM* **12**, (1965), 687–697.
22. Yates, R. A., Raphael, B. and Hart, J. P. Resolution Graphs. *Artificial Intelligence* **1** (1970), 257–289.
23. Zamov, N. K. and Sharanov, V. I. On a class of strategies which can be used to establish decidability by the resolution principle. (In Russian.) *Issled, po konstruktivnoye matematikye i matematicheskoie logikye* III, 16 (1969), 54–64. (National Lending Library, Russian Translating Program 5857, Boston Spa, Yorkshire.)

Appendix

All of the Lemmas and Theorems proved in this appendix are stated for refutations of unsatisfiable sets of clauses. Most of these propositions can be stated in a more general form which applies to derivations from arbitrary sets of clauses. In each case, we have chosen the simplest formulation which is adequate for establishing the main theorem stated in section 5.

LEMMA 1. *Let* **D′** *be a non-linear ground refutation of a set of ground instances of clauses in S. Then there exists a refutation* **D** *of S which lifts* **D′** *and has the same rm-size.*

The proof is not difficult and is similar to that of Theorem 4.7.1 in [8].

LEMMA 2. *For any unsatisfiable set of clauses, there exists a simplest non-linear refutation which lifts, and has the same rm-size as, a simplest ground refutation of a set of ground instances of clauses in S.*

Proof Outline. Let **D** be a simplest non-linear refutation of *S* and assume it lifts a ground refutation **D′**. Note that **D** cannot be simpler than **D′**. By using Lemma 1 and the fact that **D** is simplest and lifts **D′**, it is easy to verify that **D** and **D′** have the same *rm*-size. It follows from a second application

of Lemma 1 that **D**′ is a simplest ground refutation of a set of instances of clauses in *S*.

If **D** is a simplest non-linear refutation which lifts no ground refutation, then it is necessary to show that there exists another simplest refutation which does. This can be done by first constructing a ground "pseudo-derivation" isomorphic to and having same *rm*-size as **D**. (The pseudo-derivation fails to be a derivation because certain compulsory merging operations are not performed.) The pseudo-derivation, in turn, can be " contracted " to obtain a ground derivation from instances of clauses in *S*. The contracted derivation has fewer resolution operations and, at worst, has no more merging operations than it has fewer resolution operations. Therefore it is at least as simple as the pseudo-derivation. By Lemma 1, the contracted derivation can be lifted to a refutation of *S* which has the same *rm*-size This derivation is obviously at least as simple as **D** and is therefore a simplest refutation of *S*.

(The definition of pseudo-derivation is given in [5] and the contraction operation for pseudo-derivations is the analogue of the contraction operation for derivations studied in [5] and [8].)

LEMMA 3. *For every unsatisfiable set of ground clauses, there exists a simplest ground refutation which is also minimal.*

Proof Outline. Let **D** be a simplest ground refutation of the set, *S*. By Loveland's Corollary 2 [13], if **D** is not minimal then it can be "pruned" to obtain a minimal refutation **D**′ of *S*. The pruning operation removes resolution operations and introduces no more merging operations than the resolution operations it removes. Therefore **D**′ is a simplest refutation of *S*.

THEOREM 1. *For every unsatisfiable set of clauses, there exists a simplest refutation which is also minimal.*

Proof. By Lemma 2, there is a simplest refutation of the set *S* which lifts and has the same *rm*-size as a simplest refutation **D**′ of a set *S*′ of instances of clauses in *S*. By Lemma 3, there is a minimal refutation **D**″ of *S*′ which is as simple as **D**′. By Lemma 1, there is a refutation **D** of *S*, which lifts **D**″ and has the same *rm*-size as **D**″. Therefore **D** is a minimal and simplest refutation of *S*.

LEMMA 4. Let **D** be a minimal ground refutation of a set *S* of ground clauses. For any clause C_1 at a tip of **D**, there is an *s*-linear refutation of *S* with top clause C_1 and having the same *rm*-size as **D**.

Proof Outline (illustrated in Fig. 8). The proof is by induction on the size n of **D**. If $n = 0$, then the desired refutation is just the one clause s-linear derivation of $\square$. Suppose $n > 0$.

Let the two immediate subderivations of **D** derive the unit clauses $\{L\}$ and $\{\bar{L}\}$. Because **D** is minimal, if we delete from all clauses at nodes of **D**

the literals L and $\bar{L}$, we obtain minimal refutations $\mathbf{D}_1$ of S_1, and $\mathbf{D}_2$ of S_2, tree-isomorphic respectively to the subderivations of $\{L\}$ and $\{\bar{L}\}$. Suppose that $C_1 - \{L\}$ occurs at a tip of $\mathbf{D}_1$ and $B - \{\bar{L}\}$ (where $\bar{L} \in B$ and $B \in S$) at a tip of $\mathbf{D}_2$.

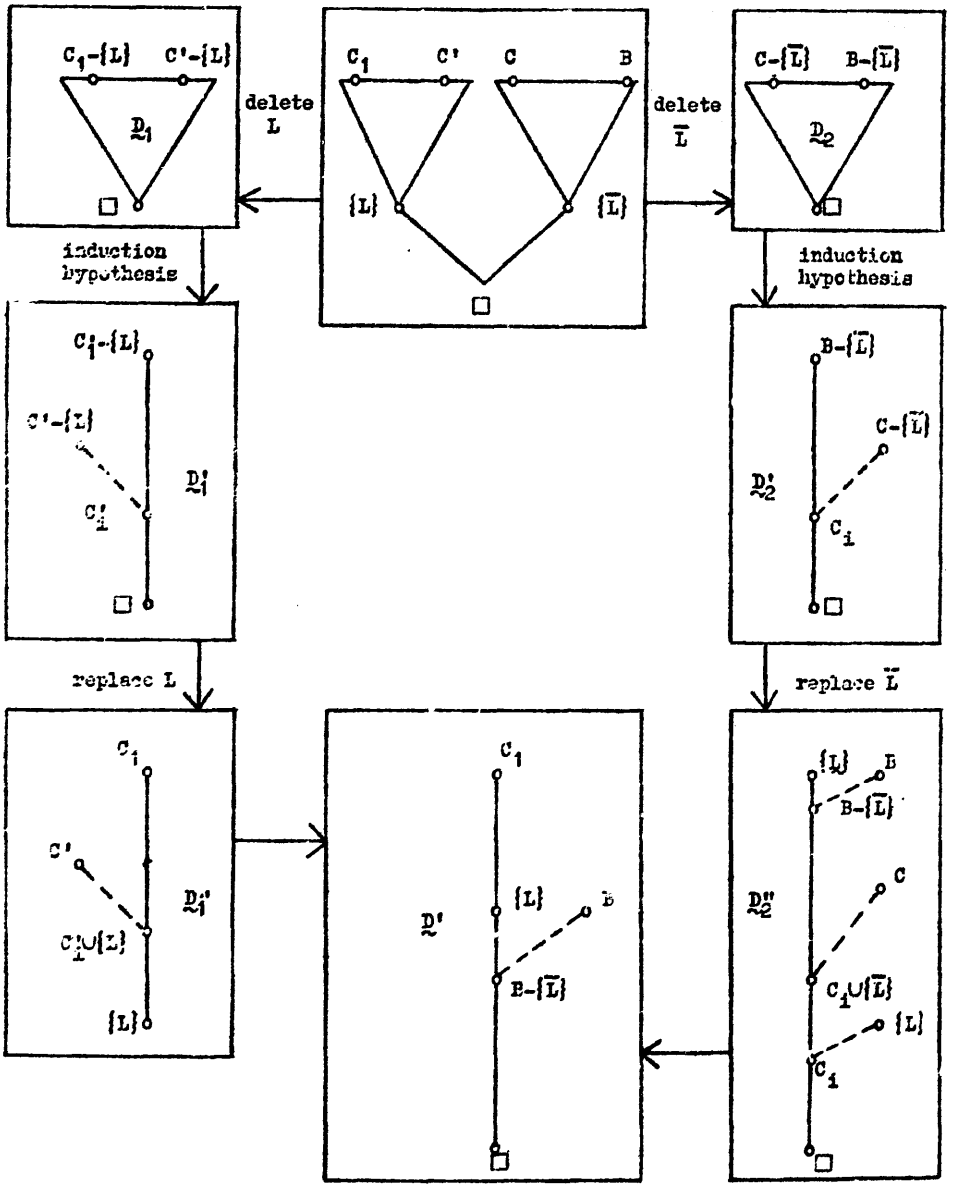

FIG. 8. Outline of the proof of Lemma 4. (A broken line, here and in Fig. 9, connects a resolvent with its input or far parent.)

By the induction hypothesis, there exists s-linear refutations $\mathbf{D}_1'$ of S_1 with top clause $C_1 - \{L\}$, and $\mathbf{D}_2'$ of S_2 with top clause $B - \{\bar{L}\}$. $\mathbf{D}_i$ and $\mathbf{D}_i$ have the same *rm*-size.

Let $\mathbf{D}_1''$ be the *s*-linear derivation of $\{L\}$ from S, isomorphic to $\mathbf{D}_1'$, with top clause C_1, obtained by replacing L in all input parents from which L was deleted in obtaining S_1. (L is inserted into all resolvents of such parents and into all descendants of such resolvents.)

Let $\mathbf{D}_2''$ be obtained from $\mathbf{D}_2'$ by first inserting $\{L\}$ as new top clause before $B-\{\bar{L}\}$ and by next inserting immediately before any resolvent C_i with input parent of the form $C-\{\bar{L}\}$, where $C \in S$ and $\bar{L} \in C$, the clause $C_i \cup \{\bar{L}\}$. It is easy to check that $\mathbf{D}_2''$ is an s-linear refutation of $S \cup \{\{L\}\}$ where $\{L\}$ occurs only as top clause. ($\{L\}$ is treated as far parent for resolvents C_i in $\mathbf{D}_2''$ with near parents $C_i \cup \{\bar{L}\}$.)

The desired s-linear refutation $\mathbf{D}'$ of S is obtained by appending $\mathbf{D}_2''$ to $\mathbf{D}_1''$ and deleting the duplicated occurrence of $\{L\}$. It is straightforward to verify that $\mathbf{D}$ and $\mathbf{D}'$ can be constructed so that they have the same *rm*-size.

LEMMA 5. *Let* $\mathbf{D}$ *be an s-linear ground refutation of a set of ground instances of clauses in* S. *Then there exists an s-linear refutation of* S *which lifts* $\mathbf{D}$ *and has the same rm-size.*

The proof of Lemma 5 is similar to, but much simpler than, the proof of Lemma 7.

THEOREM 2. *For any unsatisfiable set* S *and support subset* S_0, *there exists an s-linear refutation of* S *with top clause in* S_0 *such that no non-linear refutation of* S *is simpler.*

Proof. As in the proof of Theorem 1, there is a simplest non-linear refutation $\mathbf{D}$ of S which lifts and has the same *rm*-size, as a simplest minimal refutation $\mathbf{D}'$ of a set of ground instances S' of clauses in S. Some tip of $\mathbf{D}'$ is labelled by a clause C_1' which is an instance of some clause in S_0. By Lemma 4, there is an *s*-linear refutation $\mathbf{D}''$ of S' with top clause C_1' and having the same *rm*-size as $\mathbf{D}'$. By Lemma 5, there exists an *s*-linear refutation of S, with top clause C_1 in S_0, which has the same *rm*-size as $\mathbf{D}''$ and therefore is as simple as a simplest non-linear refutation of S.

LEMMA 6. For every unsatisfiable set S of ground clauses, support set S_0 and selection function ϕ, there exists an SL-refutation of S which has the same *rm*-size as some minimal ground refutation of S.

Proof Outline (illustrated in figure 9). The proof is by induction on the number n of distinct atoms in S. If $n = 0$ then the desired *SL*-refutation contains just the null chain and has *rm*-size (0,0). Suppose $n > 0$.

It suffices to consider the case where S is minimally unsatisfiable and S_0 contains just one clause C_1. Choose as top chain any input chain C_1^* formed from this clause. The selection function ϕ determines a unique order in which literals descending from those in C_1^* are resolved upon in any SL-derivation with top chain C_1^*. In particular, ϕ determines a literal L in C_1^* whose

descendants are the last to be resolved upon, among all descendants of literals in C_1^*.

It is easy to verify that the set of clauses obtained from S by deleting all occurrences of L and ignoring clauses containing $\bar{L}$ is unsatisfiable and contains therefore a minimally unsatisfiable subset S_1. Obtain the corresponding

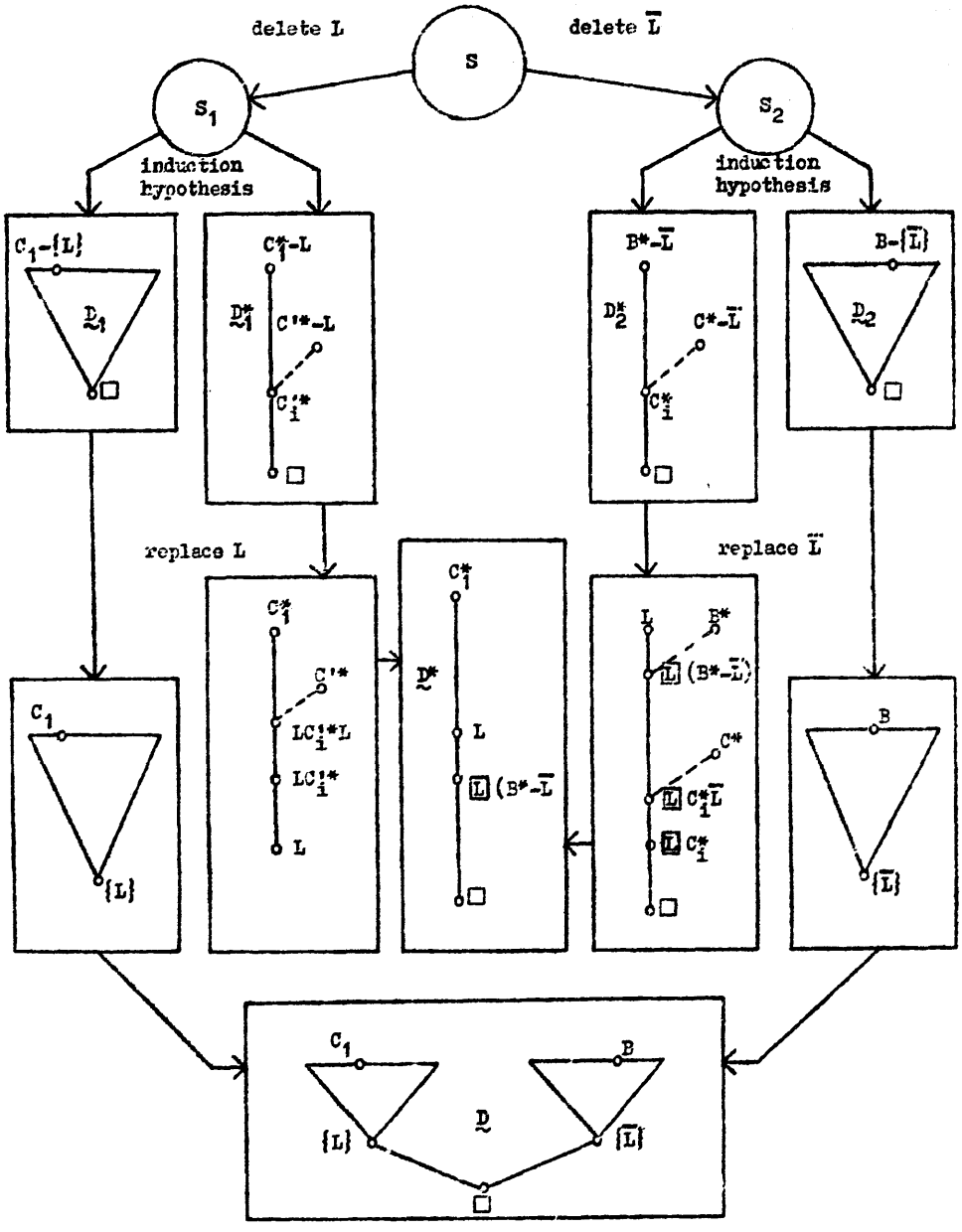

FIG. 9. Outline of the proof of Lemma 6.

set of chains S_1^* from the input chains S^* by deleting L, ignoring chains containing $\bar{L}$ and, of the remaining chains, choosing those which correspond

to clauses in S_1. It is easy to check that the chain $C_1^* - L$, obtained by deleting L from C_1^*, belongs to S_1^*.

Similarly, there exists a minimally unsatisfiable set of clauses S_2 and a corresponding set of chains S_2^*, obtained by deleting $\bar{L}$ from clauses in S and chains in S^* ignoring clauses and chains containing L. S_2^* contains a chain $B^* - \bar{L}$, obtained by deleting $\bar{L}$ from some chain $B^* \in S^*$ which contains $\bar{L}$.

We shall apply the induction hypothesis to the sets of clauses S_1 and S_2 with respective support sets $\{C_1 - \{L\}\}$ and $\{B - \{\bar{L}\}\}$. For this purpose, we define selection functions ϕ_1 for S_1 and ϕ_2 for S_2. Suppose that C^* is any chain obtainable by an SL-derivation from S with top chain C_1^* for the selection function ϕ. Let C^{**} be C^* with all occurrences of L and $\bar{L}$ deleted. If L occurs in C^* only as a B-literal in the leftmost cell then $\phi_1(C^{**}) = \phi(C^*)$. If L occurs in C^* only as the leftmost A-literal, then $\phi_2(C^{**}) = \phi(C^*)$. The values of ϕ_1 and ϕ_2 for other chains may be defined arbitrarily.

By the induction hypothesis, there exist minimal refutations $\mathbf{D}_1$ of S_1 and $\mathbf{D}_2$ of S_2 and SL-refutations $\mathbf{D}_1^*$ of S_1 for ϕ_1 with top chain $C_1^* - L$ and $\mathbf{D}_2^*$ of S_2 for ϕ_2 with top chain $B^* - \bar{L}$. $\mathbf{D}_i$ and $\mathbf{D}_i^*$ have the same *rm*-size.

The desired SL-refutation $\mathbf{D}^*$ of S can now be obtained from $\mathbf{D}_1^*$ and $\mathbf{D}_2^*$ as in the similar construction of the *s*-linear refutation of Lemma 4: Introduce L as new B-literal in the leftmost cell of all chains in $\mathbf{D}_1^*$. Introduce L as new top chain and as a new A-literal to the left of all literals in chains of $\mathbf{D}_2^*$ and insert $C_i^*\bar{L}$ immediately before any chain C_i^* obtained by extension in $\mathbf{D}_2^*$ with a chain $C^* - \bar{L} \in S_2^*$ where $\bar{L} \in C^*$ and $C^* \in S^*$. $\mathbf{D}^*$ is then obtained by appending the second derivation to the first, deleting the duplicated occurrence of the chain L. It is not difficult to verify that D^* is an SL-refutation of S^* for the selection function ϕ with top chain C_1^*.

The minimal refutation $\mathbf{D}$ of S, with same *rm*-size as $\mathbf{D}^*$, is obtained from $\mathbf{D}_1$ and $\mathbf{D}_2$: To each clause $C - \{L\}$ at a tip of $\mathbf{D}_1$, where $L \in C$ and $C \in S$, add the literal L. Also add L to the clauses at all nodes in $\mathbf{D}_1$ which descend from such tips. The resulting derivation is a minimal derivation of $\{L\}$ from S. In a similar manner obtain from $\mathbf{D}_2$ a minimal derivation of $\{\bar{L}\}$ from S. $\mathbf{D}$ is then the minimal refutation of S, having these two minimal derivations as immediate subderivations. It is quite straightforward to check that $\mathbf{D}^*$ and $\mathbf{D}$ can be constructed so that they have the same *rm*-size.

LEMMA 7. *For every unsatisfiable set S, support set S_0 and selection function ϕ, there exists a set of ground instances S' of clauses in S, support subset S_0' of S' and selection function ϕ'; such that, for every ground SL-refutation of S', for S_0' and ϕ', there exists an SL-refutation of S, for S_0 and ϕ, which has the same rm-size.*

Proof Outline. For simplicity, we may assume that S is minimally unsatisfiable and that S_0 consists of a single clause C_1. Let S' be any minimally

unsatisfiable set of ground instances of clauses in S. S' contains some instance C_1' of C_1. Let S^* be a set of input chains corresponding to S, let S'^* be the corresponding set of input chains for S' and let C_1^* by any chain (in S^*) corresponding to C_1 and $C_1'^*$ be any chain (in S'^*) corresponding to C_1'.

We construct a tree T, each node of which is labelled both by an SL-derivation $\mathbf{D}^*$ from S^* for ϕ with top chain C_1^* and by a ground SL-derivation $\mathbf{D}'^*$ from S'^* with top chain $C_1'^*$. Both derivations have the same *rm*-size and $\mathbf{D}'^*$ derives an instance of the chain derived by $\mathbf{D}^*$. The root of T is labelled by the SL-derivations (C_1^*) and $(C_1'^*)$. Suppose that a node N and the SL-derivations $\mathbf{D}^* = (C_1^*, \ldots, C_n^*)$ and $\mathbf{D}'^* = (C_1'^*, \ldots, C_n'^*)$ at N have been constructed and verified to have the desired properties. We need to specify the immediate descendant nodes and the SL-derivations labelling them.

If $\mathbf{D}'^*$ violates the admissibility restriction then N has no immediate descendants. If truncation can be performed on $C_n'^*$ then it can be performed on C_n^* and N has one immediate descendant obtained by adding to $\mathbf{D}^*$ and $\mathbf{D}'^*$ the chains which result from truncation.

If reduction needs to be performed on $C_n'^*$ then one way of doing reduction is chosen and performed in order to obtain the single node which is the immediate descendant of N. The new node is labelled by adding to $\mathbf{D}'^*$ the chain which results from reduction. A similar reduction operation can be performed on C_n^* and the result is added to $\mathbf{D}^*$ and also labels the new node.

Let L be the selected literal in C_n^* and let L' in $C_n'^*$ be the corresponding instance of L. Treat L' as the selected literal in $C_n'^*$. If the preceding cases do not apply and no extension operation with a chain B'^* from S'^* can be performed on $C_n'^*$ then N has no immediate descendants. Otherwise, N has immediate descendants for each such B'^*. Each new node is labelled by adding to $\mathbf{D}'^*$ the chain which results from extension. A similar extension operation can be performed on C_n^* with a chain B^* from S^*. The SL-derivation which results from the performance of this extension operation also labels the new node.

In each of the preceding cases, it is straightforward to verify that all new nodes have the desired properties.

T labelled by its ground derivations may fail to be an SL-search tree for some selection function ϕ'. There may be distinct nodes N and N' labelled by ground derivations of the same ground chain C'^*, but by general derivations of distinct general chains. The selected literals in the general chains correspond to different literals in C'^*. In such a case, a single such node N can be selected and all subtrees of T rooted at nodes N' can be replaced by the subtree rooted at N. It can now be verified that the modified tree, together with the ground SL-derivations labelling its nodes, constitutes an SL-search tree T for some selection function ϕ', for the top chain C'^* and for the

input set S'. It follows that, for every ground SL-refutation $\mathbf{D'^*}$ of S' for ϕ', there is an SL-refutation of S for ϕ with top chain C_1^*, having same *rm*-size as D'^*.

THEOREM 3. *For every unsatisfiable set S, support set S_0 and selection function ϕ, there exists an SL-refutation of S which has the same rm-size as some minimal refutation of S.*

Proof. Let S', S_0' and ϕ' be as stated in Lemma 7. By Lemma 6, there exists an *SL*-refutation $\mathbf{D'^*}$ of S' for ϕ' with top chain in $S_0'^*$, and $\mathbf{D'^*}$ has the same *rm*-size as some minimal refutation $\mathbf{D'}$ of S'. But, by Lemma 2, there is a minimal refutation $\mathbf{D}$ of S which has the same *rm*-size as $\mathbf{D'}$ and, by Lemma 7, there is an *SL*-refutation $\mathbf{D^*}$ of S for ϕ with top chain in S_0^* which has the same *rm*-size as $\mathbf{D'^*}$. Therefore, the *SL*-refutation $\mathbf{D^*}$ has the same *rm*-size as the minimal refutation $\mathbf{D}$.