THE UNIVERSITY *of* EDINBURGH

STRATEGIES FOR IMPROVING THE EFFICIENCY

OF AUTOMATIC THEOREM-PROVING

by

Donald Kuehner

Ph.D. Thesis

University of Edinburgh

July, 1971.

ABSTRACT

In an attempt to overcome the great inefficiency of theorem-proving methods, several existing methods are studied, and several new ones are proposed. A concentrated attempt is made to devise a unified proof procedure whose inference rules are designed for the efficient use by a search strategy.

For unsatisfiable sets of Horn clauses, it is shown that P_1-resolution and selective linear negative (SLN) resolution can be alternated heuristically to conduct a bi-directional search. This bi-directional search is shown to be more efficient than either of P_1-resolution and SLN-resolution.

The extreme sparseness of the SLN-search spaces lead to the extension of SLN-resolution to a more general and more powerful resolution rule, selective linear (SL) resolution, which resembles Loveland's model elimination strategy. With SL-resolution, all immediate descendants of a clause are obtained by resolving upon a single selected literal of that clause.

Linear resolution, s-linear resolution and t-linear resolution are shown to be as powerful as the most powerful resolution systems. By slightly decreasing the power, considerable increase in the sparseness of search spaces is obtained by using SL-resolution. The amenability of SL-resolution to applications of heuristic methods suggest that, on these grounds alone, it is at least competitive with theorem-proving procedures designed solely from heuristic considerations.

Considerable attention is devoted to various anticipation

procedures which allow an estimate of the sparseness of search trees before their generation. Unlimited anticipation takes the form of pseudo-search trees which construct outlines of possible proofs. Anticipation procedures together with a number of heuristic measures are suggested for the implementation of an exhaustive search strategy for SL-resolution.

ACKNOWLEDGMENTS

This research has been supported during 1970-71 by an IBM fellowship awarded by Imperial College.

I am indebted to my supervisor, Dr. Bernard Meltzer, for his encouragement and suggestions, and for the free and stimulating atmosphere of his Metamathomatics Unit. I am thankful for the illuminating and constructive discussions with my colleagues of the Unit, Bob Kowalski, Pat Hayes and J Moore. I am particularly indebted to Bob Kowalski, who co-authored Chapters 3 and 4 of this thesis, as well as offering suggestions for the other chapters.

TABLE OF CONTENTS

Chapter 1. Basic Concepts

1.1 Introduction

Since the publication of Herbrand's theorem in 1930 [8], it has been theoretically possible to prove theorems by using an algorithm. However, there has been considerable difficulty in finding an automatic procedure which is efficient. The work of Prawitz [24], Davis [5] and Robinson [28] first suggested that efficiency is possible.

This thesis develops several related strategies in an attempt to form a unified proof procedure for efficient theorem proving. The desire for efficiency has motivated every step, so that an efficient and detailed search strategy was conceived of from the beginning. Thus the form of the inference rules was dictated in part by an awareness of a complementary search strategy.

In all of the following, attention is restricted to derivations obtainable by the resolution rule [28]. Several other inference rules, including Loveland's model elimination [16] and the Maslov inverse method [20] have been shown to be equivalent to resolution by Loveland [18] and Kuehner [13]. Other arguments in favour of resolution are supplied by Kowalski [12].

The unrestricted use of the resolution rule generates far more derivations than are needed for obtaining a proof. An increase in the efficiency of obtaining a proof can only be achieved by a selective search strategy together with a restricted use of the resolution rule. Thus the achievement of an efficient proof

procedure involves the devising of suitable restrictions for the resolution rule.

1.2 Derivations and Search Spaces

A familiarity with the basic terminology of resolution theory is assumed. Robinson [30] provides an excellent background. The resolution rule will be treated as having only two clauses as input, the parents of the resolvent. For any restriction R of the resolution rule and any finite set S of input clauses, an R-derivation $\underset{\sim}{D}$ of the clause C from S is a tree of nodes, each node being labelled by a clause which is at the node. $\underset{\sim}{D}$ has the following properties. If $C \in S$, then $\underset{\sim}{D}$ is a single node labelled by C. If $C \notin S$, then there exist R-derivations $\underset{\sim}{D}_1$ and $\underset{\sim}{D}_2$ of C_1 and C_2 from S, such that C is an R-resolvent of C_1 and C_2. Then $\underset{\sim}{D}$ is composed of the labelled nodes of $\underset{\sim}{D}_1$ and D_2 together with the node labelled by C which is the immediate descendant of the nodes labelled by C_1 and C_2. The immediate descendant relationship between nodes of $\underset{\sim}{D}_1$ and of $\underset{\sim}{D}_2$ is inherited by the nodes of $\underset{\sim}{D}$. The derivations $\underset{\sim}{D}_1$ and $\underset{\sim}{D}_2$ are the immediate subderivations of $\underset{\sim}{D}$. Any clause $C' \in S$ which labels a node of $\underset{\sim}{D}$ is an input clause of $\underset{\sim}{D}$. If $\underset{\sim}{D}$ is an R-derivation of the null clause, $\square$, from S, then $\underset{\sim}{D}$ is an R-refutation of S. The clauses, other than C, at the nodes of $\underset{\sim}{D}$ are the ancestors of C, and C is the descendant of each of its ancestors. C is the immediate descendant of its parents. Note that the same clause may occur at different nodes of a derivation.

For any finite set S of input clauses, and any restriction R of the resolution rule, a graph $G_R(S)$ of labelled nodes is the R-search space for S, iff for each R-derivation of a clause C from S there is a node of $G_R(S)$ which is labelled by C. The node N of

$G_R(S)$ is an <u>immediate descendant</u> of the nodes N_1 and N_2 of $G_R(S)$ iff the R-derivations of the clauses at N_1 and N_2 are the immediate subderivations of the R-derivation of the clause at N. The clauses labelling nodes of $G_R(S)$ are referred to as the clauses <u>of</u> $G_R(S)$. For any clause C of $G_R(S)$, the portion of $G_R(S)$ whose nodes are labelled by the ancestors of C is the derivation of C. It follows that for any R-derivation $\underset{\sim}{D}$ of C from S, $\underset{\sim}{D}$ is isomorphic to a descendancy related subset of the labelled nodes of $G_R(S)$.

Kowalski's thesis [12] contains an extended examination of resolution graphs.

1.3 Search Strategies and Layered Search Spaces

A resolution proof procedure is a restriction R of the resolution rule together with a search strategy. A search strategy is an algorithm which determines, for any finite set S of input clauses, the order of generation of the clauses of $G_R(S)$. This distinction between the inference rule and the search strategy of a proof procedure has been investigated in detail by Kowalski [12] and has been elucidated by Meltzer [22].

A search strategy is exhaustive iff for any clause C of $G_R(S)$, it generates only a finite number of clauses before generating C. If S is an unsatisfiable set of input clauses, and R is a complete inference rule, then the null clause is a clause of $G_R(S)$. If Σ is an exhaustive search strategy, then Σ generates only a finite number of clauses of $G_R(S)$ before generating a null clause. Let $G_R^*(S)$ be the descendancy related subset of the labelled nodes of $G_R(S)$ corresponding to the clauses of $G_R(S)$ generated by Σ when Σ first generates a null clause. Thus $G_R^*(S)$ is finite if S is unsatisfiable, R is complete and Σ is exhaustive. The difficulty of finding a refutation of S may be measured by the number of nodes of $G_R^*(S)$ which are not labelled by input clauses. The efficiency of a proof procedure is improved by choosing R and Σ in such a way as to decrease the difficulty of finding a refutation of any unsatisfiable set S.

It seems impossible to ensure that $G_R^*(S)$ is finite for all unsatisfiable S and all complete R unless Σ is exhaustive. This thesis will consider only exhaustive search strategies. In order

to ensure that Σ is exhaustive, it is convenient to subdivide $G_R(\mathcal{S})$ into a denumerable collection of layers, each layer having only a finite number of nodes. The search space is thought of as being layered from the top down, the first layer being at the top of the search tree, the second layer just below it, and so on. In order that Σ be exhaustive, it is then necessary only that Σ generate all clauses in one layer before generating any clauses in the next lower layer.

Clearly, any algorithm for layering search spaces partially specifies the search strategy. Certain methods of layering are so easy and so natural that they should be considered when designing an inference rule. In this way, it is possible to design a unified proof procedure for theorem-proving.

In Chapter 6, several methods for layering search spaces are considered. For each of these, a detailed search strategy uses heuristic criteria for determining the order of generating clauses within layers of the search space.

The requirement that a search strategy generate all clauses of one layer before generating any of the next layer implies that the ancestors of a clause C must be in the same layer as C or in higher layers. Let the merit of a clause be the number of the layer in which it occurs. Thus each layer of a search space is a set of clauses of equal merit. The layers are the merit levels of the search space. Thus the merit of each ancestor of a clause is less than or equal to the merit of the clause itself.

1.4 Efficient Search Spaces

A search space should be layered so that null clauses with simple derivations occur at higher levels than null clauses with more complex derivations. The measure of complexity which is easiest to work with is the size of the derivation, the number of resolvents in the derivation. A simplest R-refutation of S is then one such that no other R-refutation of S has fewer resolvents. In Chapter 6, there is a thorough discussion of alternate measures of the complexity of a derivation, and their relation to the difficulty of finding a refutation. It is argued that size is the best of the easy measures of complexity.

The most straightforward method of ensuring that simple refutations are on higher layers than more complex refutations is to assign all clauses with simple derivations to layers which are higher than clauses with more complex derivations. The merit of a clause is then the complexity of its derivation. A more efficient layering defines the merit of a clause C to be a lower bound on the complexity of the simplest refutation obtainable from C. If the cost of a clause C is the complexity of its derivation, then the merit of C is an upper bound on the cost of the least costly null clause which could be a descendant of C. This is the merit used by Kowalski's diagonal search [11].

Let R be the unrestricted resolution rule, and let R' be any restriction of R. Let $G_R(S)$ and $G_{R'}(S)$ be layered in some way compatible with the preceding discussion. There are two factors to consider when designing the resolution rule R'. One is that

$G_{R'}(S)$ should have fewer clauses on each layer than does $G_R(S)$. The other is that the highest null clause of $G_{R'}(S)$ should not be much lower than the highest null clause of $G_R(S)$. The rule R' is more efficient than R if $G_{R'}(S)$ has fewer clauses on each level, and its highest null clause is as high as the highest null clause of $G_R(S)$. One may consider that $G_{R'}(S)$ is obtained by pruning the search space $G_R(S)$ without pruning out all highest null clauses.

For any resolution rules R' and R, $G_{R'}(S)$ is more sparse than $G_R(S)$ if each layer of $G_{R'}(S)$ has fewer clauses than the corresponding layer of $G_R(S)$. R' is a refinement of R if the clauses on each layer of $G_{R'}(S)$ is a subset of the clauses on the corresponding layer of $G_R(S)$. R' is as powerful as R if the highest null clause of $G_{R'}(S)$ is as high as the highest null clause of $G_R(S)$. Thus, in designing a proof procedure, one tries to construct a resolution rule whose search spaces are as sparse as possible with very little if any loss in power.

The construction of a sparse search space without loss of power may be considered to be the process of pruning away clauses which are irrelevant to the derivation of the highest null clauses. Furthermore, all redundant rederivations of the same clause should be pruned out. This pruning of redundant derivations should extend to all but one of the highest null clauses. If all redundant and irrelevant clauses were pruned out, then there would remain exactly one derivation of the null clause.

The SLN-resolution rule of Chapter 2 produces extremely sparse search spaces, but only for certain types of input sets. It

is proved to be at least as powerful as P_1-resolution. By combining SLN- with P_1-resolution in a bi-directional search, the resulting search space is more sparse than the search space for either rule alone, and the corresponding rule is at least as powerful as P_1-resolution.

The s-linear and t-linear rules of Chapter 3 produce very sparse search spaces, and are proved to be as powerful as unrestricted resolution. The t-linear rule is a refinement of the s-linear rule so the t-linear search spaces are always at least as sparse as the s-linear search spaces.

The branching rate of a clause C in $G_R(S)$ is the number of clauses of $G_R(S)$ which are immediate descendants of C. If the branching rate of a clause C is reduced, then there are fewer descendants of C. Thus there are fewer clauses either in the layer that C is in or in some layers below that. It follows that a resolution rule which uniformly decreases the branching rate of clauses produces sparser search spaces. That is, if R' differs from R only in that any clause C has a lower R'-branching rate than an R-branching rate, then $G_{R'}(S)$ is more sparse than $G_R(S)$.

SL-resolution of Chapter 4 has a markedly lower branching rate for all non-unit clauses than does t-linear resolution, but SL-resolution is not quite as powerful as t-linear resolution, and thus is not as powerful as unrestricted resolution. However, it is proved that SL-resolution is as powerful as the rule for obtaining minimal-derivations. With this bound on the power of SL-resolution, it is felt that the advantage of its increased sparseness

outweighs the loss of power. This feeling is supported by experimental evidence.

As well as being relatively powerful and having notably sparse search spaces, s-linear, t-linear and SL-resolution have the added advantage of determining search spaces which are exceptionally amenable to a variety of methods for heuristic search.

Chapters 4 and 5 which investigate t-linear and SL-resolution are extracted from papers [14 and 15] written in conjunction with Robert Kowalski. Since the completion of the original paper, we have learned of the related investigations of Donald Loveland [18] and Raymond Reiter [26]. 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 [25]. 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.

Chapter 2. SLN-Resolution

2.1 Introduction

Although bi-directional search has been investigated as a general problem-solving technique by Pohl [23] and others, there are difficulties in applying it to theorem-proving. However, bi-directional search is feasible for certain kinds of sets of input clauses, the simplest of which are Reynolds' transformational systems [29].

A transformational system is any finite set of clauses such that each clause is either a one literal clause (a unit), or it contains exactly one negative and one positive literal (a transformation). Any unsatisfiable transformational system has a P_1-refutation which is purely linear. (A P_1-resolution [29] has one parent all of whose literals are positive. A purely linear resolution has one parent which is an input clause. Chang [3] calls purely linear resolution, input resolution.)

Let $S = \{\{L\}, \{\overline{L},M\}, \{\overline{M},N\}, \{\overline{N},P\}, \{\overline{P}\}\}$

represent a ground-level (no variables) transformational system. In all examples of clauses, set theoretical brackets will be omitted together with the commas which separate elements of a set. Using this notation,

$$S = \{L, \overline{L}M, \overline{M}N, \overline{N}P, \overline{P}\}.$$

Then figure 1 illustrates a purely linear P_1-refutation of S.

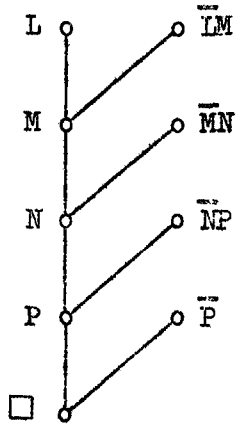


Figure 1.

One remarkable property of purely linear P_1-refutations is that they are reversible. Figure 2 illustrates the reversal of the refutation of figure 1. Here again the refutation is linear, but uses N_1-resolution, in which all of the literals of one parent are negative.

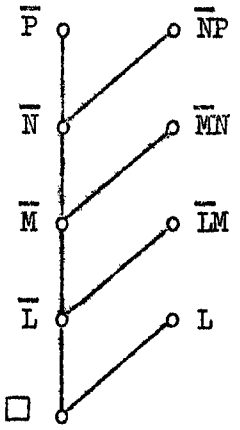


Figure 2.

Figure 3 illustrates a bi-directional refutation combining both P_1- and N_1-resolution.

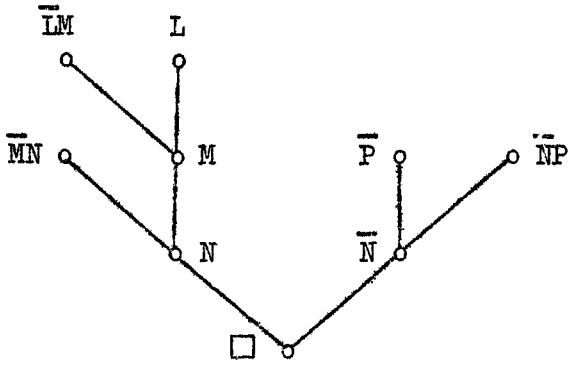


Figure 3.

The fact that the refutations of figures 1 and 2 can be combined to form a bi-directional refutation suggests that two searches may be conducted simultaneously, and their results combined to form a refutation. The advantage of conducting a search in this way is that a search to level n is usually less than half as difficult as a search to level 2n. It follows that the sum of the difficulties of the simultaneous searches should be less than the difficulty of either search alone.

Bob Kowalski and Pat Hayes have suggested that some properties of transformational systems might apply to Horn clauses [9]. A clause is a Horn clause iff it has at most one positive literal. Thus, a positive Horn clause is a unit clause, and a mixed Horn clause has any number of negative literals and one positive literal. A negative Horn clause has any number of negative literals and no positive literals. There are many problems which can be expressed using only Horn clauses. Such problems are characterised by Cohn [4] and include many theorems of group therapy.

As with transformational systems, a P_1-refutation of a set

of Horn clauses is reversible as an N_1-refutation. However, in this case only the N_1-refutation is purely linear. For the unsatisfiable set

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

of ground-level Horn clauses, figure 4 illustrates a P_1-refutation $\underset{\sim}{D}$, and figure 5 illustrates its linear N_1-reversal $\underset{\sim}{D}^*$.

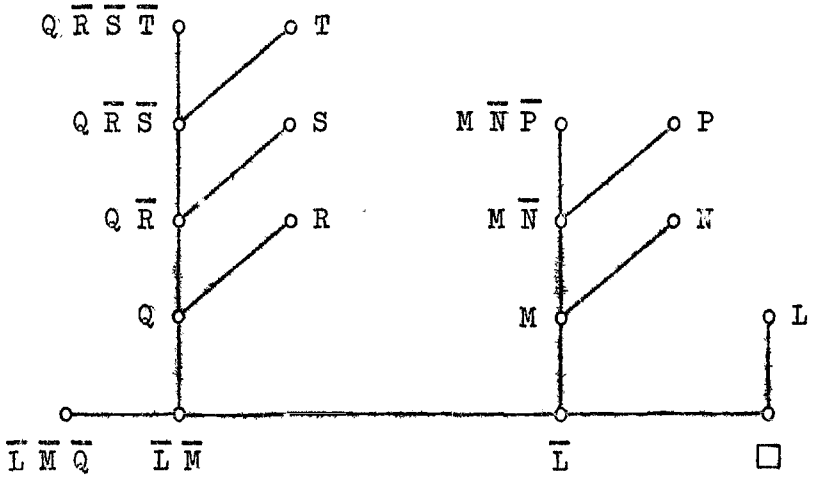


Figure 4.

Note that for every resolvent C* of $\underset{\sim}{D}^*$, there are subderivations of $\underset{\sim}{D}$ which derive units complementary to each literal in C*. For $\bar{L}\,\bar{M}\,\bar{R}$ of $\underset{\sim}{D}^*$, there are subderivations of L, M and R of $\underset{\sim}{D}$. The derivation of $\bar{L}\,\bar{M}\,\bar{R}$ has size 3, and the derivations of L, M and N have sizes 0, 2 and 0. The number k of resolutions needed to obtain the null clause from these clauses is 3. The total number of resolutions involved is 3+0+2+0+3 = 8. The same total is obtained for every resolvent of $\underset{\sim}{D}^*$ as is shown by the table following figure 5.

In section 2.4, it will be shown that analogous results hold for any unsatisfiable set of Horn clauses.

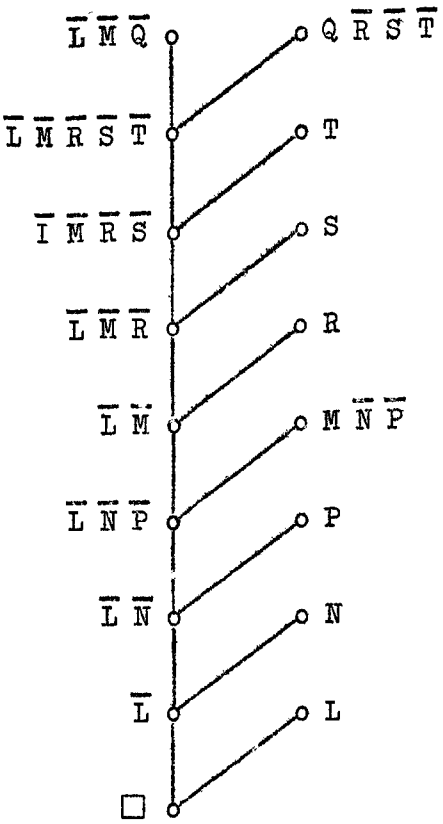


Figure 5.

derived by $\underset{\sim}{D}^*$	size	derived by $\underset{\sim}{D}$	sizes	k	total
$\bar{L}\,\bar{M}\,\bar{Q}$	0	L,M,Q	0,2,3	3	8
$\bar{L}\,\bar{M}\,\bar{R}\,\bar{S}\,\bar{T}$	1	L,M,R,S,T	0,2,0,0,0	5	8
$\bar{L}\,\bar{M}\,\bar{R}\,\bar{S}$	2	L,M,R,S	0,2,0,0	4	8
$\bar{L}\,\bar{M}\,\bar{R}$	3	L,M,R	0,2,0	3	8
$\bar{L}\,\bar{M}$	4	L,M	0,2	2	8
$\bar{L}\,\bar{N}\,\bar{P}$	5	L,N,P	0,0,0	3	8
$\bar{L}\,\bar{N}$	6	L,N	0,0	2	8
$\bar{L}$	7	L	0	1	8

2.2 The Definition of SLN-resolution

The example of the preceding section suggests that there is a linear N_1-refutation of any unsatisfiable set of Horn clauses. In figure 5 only the rightmost literal of each negative clause is resolved upon. That is, the clauses may be considered to be ordered. An extension of the concept of ordering allows a selection function to choose which literal is to be resolved upon. SLN-resolution is selective linear negative resolution.

For notational convenience and for efficient computer implementation, it is useful to treat resolution as a sequence of two operations, factoring followed by resolution of factored clauses. If C is a clause and $\underset{\sim}{E}$ a unifiable partition of the literals of C, having most general unifier (m.g.u.) θ, then $C\theta$ is a factor of C. If exactly one component of $\underset{\sim}{E}$ contains two literals and every other component exactly one, then $C\theta$ is a basic factor of C, and $C\theta$ is obtained from C by one factoring operation. The resolution of factored clauses unifies one literal from one parent with the complement of one literal from the other parent. Although other factoring methods are compatible with SLN-resolution, only Kowalski's m-factoring [12] is considered in the following discussion. The method of implementing m-factoring is to factor input clauses in all possible ways, and to factor resolvents in all possible ways provided that the literals which are unified in the factoring descend from different input parents. This last restriction ensures against redundant factoring. This form of factoring is built into the definition of SLN-resolution.

For any Horn clause C, an ordered Horn clause C* is a sequence consisting of the literals of C written in some fixed order. If C contains a positive literal, then that positive literal is the leftmost literal of C*. For any set S of Horn clauses, the set S* is the set of all ordered Horn clauses obtainable from factors of clauses of S. That is, if $S = \{P(a,x)\ P(x,y)\}$ then $S^* = \{P(a,x)\ P(x,y),\ P(x,y)\ P(a,x),\ P(a,x)\}$. For any ordered Horn clause C* and C'*, let C*C'* be the ordered clause beginning with the literals in C* in the order they appear in C*, followed on the right by the literals of C'* in the order that they appear in C'*. Thus $C^*\bar{L}$ is the ordered Horn clause whose rightmost literal is $\bar{L}$.

An ordered N_1-resolution has as one parent an ordered negative Horn clause of the form $C^*\bar{L}$ and as the other parent a mixed or unit Horn clause of the form KC'* where C'* may be empty. Neither parent contains two literals which have the same atom. If L and K are unifiable with m.g.u. θ, then the ordered N_1-resolvent is $C^*C'^*\theta$. The literal resolved upon in $C^*\bar{L}$ is $\bar{L}$, and in KC'* it is K. If C'* is not empty, then the literals of C'* which are to the right of the literals of $C^*\theta$ in $C^*C'^*\theta$ are the new literals of $C^*C'^*\theta$.

Let C* be an ordered N_1-resolvent, and let $\bar{K}$ occur as a new literal of C*. If there is a non-new literal $\bar{L}$ occurring in C* such that L and K are unifiable with m.g.u. θ, then an ordered factor of C* is $C_0^*\theta$, where C_0^* is obtained by deleting the given occurrence of $\bar{L}$ from C*. The literal $\bar{L}$ is factored out of C*. The factoring operation is said to have been applied to C*. If

the n rightmost literals of C^* are new in C^*, then define the n rightmost literals of $C_0^*\theta$ to be new in $C_0^*\theta$. With this extension to the definition of a new literal, the preceding definition of an ordered factor also applies when C^* itself is an ordered factor. The definition of an ordered factor in terms of new literals avoids redundant factoring. If the mgu θ is the null substitution, then $\overline{L} = \overline{K}$. In this case $\overline{L}$ and $\overline{K}$ are said to merge, and $\overline{L}$ is merged out of C^*. The merging operation is an instance of the factoring operation.

An SLN-derivation from a set S of Horn clauses is a sequence $(C_1^*, \ldots, C_n^*)$ of ordered clauses satisfying the following conditions.

(1) The initial clause $C_1^* \in S^*$ and is negative.

(2) C_{i+1}^* is an ordered N_1-resolvent of C_i^* (the near parent) and an ordered clause (the input parent) from S^*, or C_{i+1}^* is an ordered factor of C_i^*.

It should be noted that factoring must precede resolution if it is to be done at all. That is, if a new literal can be factored out of a clause, this factoring must be done before any other new literal is resolved upon. This is because only new literals can be factored out, while all of the new literals of a resolvent descend from the input parent.

SLN-derivations are not represented in the standard derivation format. Together with other linear derivations, SLN-derivations are conceived of as being vine-like trees consisting of an initial node together with a sequence of its descendants. The input parent of an ordered clause may be attached to the arc joining the

clause to its near parent. This device is particularly useful in simplifying the appearance of SLN-search spaces. Using this representation in figure 6 produces an illustration of an SLN-refutation of a familiar group theory problem. The sequence of ordered clauses at the nodes is the SLN-refutation.

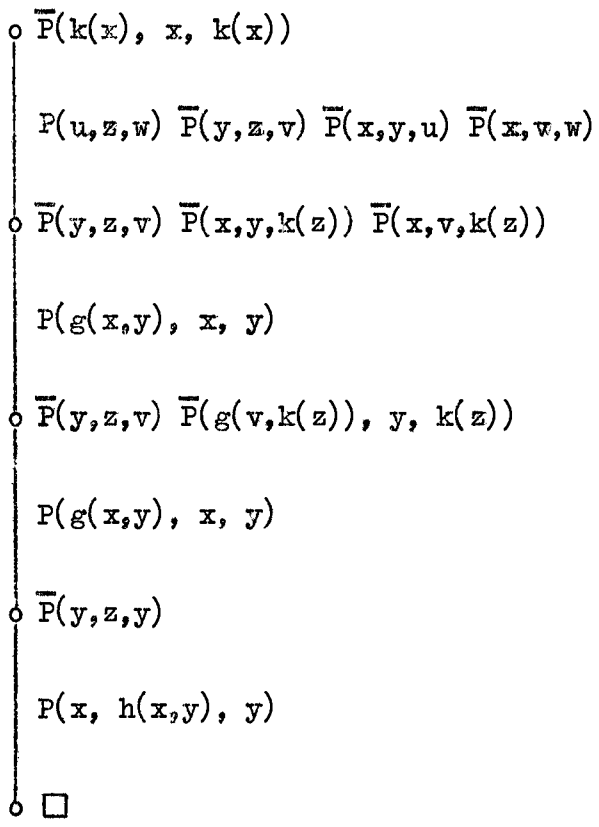


Figure 6.

The implementation of SLN-resolution is more efficient if there is a retroactive ordering of factors of input clauses. Ordered N_1-resolution could be redefined so that the literal resolved on in the negative parent is any literal selected from those literals most recently introduced into the deduction. Thus, if $\overline{L}\ \overline{M}\ \overline{N}$ has near parent $\overline{L}\ \overline{P}$ and input parent $P\ \overline{M}\ \overline{N}$, then $\overline{M}$ and $\overline{N}$ are the literals in $\overline{L}\ \overline{M}\ \overline{N}$ which have been most recently introduced.

If $\overline{L}\ \overline{M}\ \overline{N}$ resolves with M, then $\overline{M}$ is the selected literal of $\overline{L}\ \overline{M}\ \overline{N}$ and $\overline{N}$ is the literal in $\overline{L}\ \overline{N}$ which has been most recently introduced. $\overline{M}$ and $\overline{N}$ are new in $\overline{L}\ \overline{M}\ \overline{N}$, but $\overline{N}$ is not new in $\overline{L}\ \overline{N}$. The selection of $\overline{M}$ can be thought of as rewriting $\overline{L}\ \overline{M}\ \overline{N}$ as $\overline{L}\ \overline{N}\ \overline{M}$ and then performing the previously defined ordered N_1-resolution. Resolving on some selected most recent literal is a retroactive choice of ordered input clause. Thus, resolving on $\overline{M}$ rather than $\overline{N}$ in the preceding example is a retroactive choice of using the input parent $P\ \overline{N}\ \overline{M}$ instead of $P\ \overline{M}\ \overline{N}$. In order to use this dynamic ordering, some marker should be inserted between the residues of the two parents when forming a resolvent. Thus resolvents become sequences of cells of literals separated by markers. Such sequences of cells correspond to the chains of SL-resolution (Chapter 4) and model elimination [16]. However, the use of ordered clauses rather than a selection function simplifies the following discussion.

It should be noted that since every SLN-resolvent is negative, no resolvent can be a tautology. Thus, if tautologies are deleted from the factors of the input set, then no further deletion of tautologies need be done.

A subset S' of a set S of clauses is a <u>support set</u> (Wos et al [31]) for S iff S-S' is satisfiable. In common with other linear resolution systems, the initial clause of an SLN-derivation may be restricted to belong to a given support set of the input set S. In this case, the support set must be a subset of the set of all factors of negative clauses in S.

SLN-resolution can be extended to non-Horn clauses, but it

must then be weakened to SL-resolution. For SL-resolution (Chapter 4) there is no longer a requirement that one parent be negative, and resolution must be allowed with an ancestor C_j^* where $j < i$. It is also possible to extend SLN-resolution to SN-resolution, by no longer requiring linearity, but this weakens the selection function.

2.3 The Completeness and Power of SLN-Resolution

The existence of an SLN-refutation for any unsatisfiable set of Horn clauses is proved by permuting the resolutions of a P_1-refutation of S. In order to do this, it is necessary to examine the structure of a P_1-refutation of a set of Horn clauses.

Let S be any set of Horn clauses. Clearly, any P_1-resolution between members of S must have a Horn clause as a resolvent. Thus, in any P_1-derivation from S, one parent of each resolution must be a positive unit. If the other parent is a mixed Horn clause, then the resolvent is either a shorter mixed Horn clause or a positive unit. If the other parent is a negative Horn clause, then the resolvent is either a shorter negative clause or a null clause. It follows that any P_1-refutation of a set of Horn clauses has one and only one negative input clause.

Since every P_1-resolution has a positive unit clause as one parent, all resolvents are instances of subsets of input clauses. It follows that in every P_1-derivation, the input clauses may be replaced by appropriately ordered Horn clauses, with the positive literal on the left, and with the rightmost literal the literal resolved upon. A P_1-derivation, all of whose clauses are ordered Horn clauses, is an <u>ordered P_1-derivation</u>. Clearly, there is a one-one correspondence between P_1-derivations and isomorphic ordered P_1-derivations.

In order to compare the complexity of P_1-derivations and SLN-derivations, it is necessary to have an appropriate definition

of the complexity. The refutations of figures 4 and 5 are felt to have the same intuitive complexity although the P_1-refutation is of level 6 and the SLN-refutation is of level 8. Thus, for the purpose of comparing linear and non-linear derivations, the measure of complexity to be used is the size of the derivation, the number of resolutions performed in the derivation. In figures 4 and 5, both refutations have size 8.

In calculating the size of either an ordered P_1-derivation or an SLN-derivation, it is assumed that the input clauses of the derivations are ordered factors of the clauses in the input set. This is implemented by constructing the set S* of all ordered clauses constructible from all the factors of the clauses in the input set S. Any clause or variant of a clause is considered to be a factor of itself. Since all ordered P_1-resolvents are instances of subsets of ordered input clauses, P_1-resolution for Horn clauses is complete with no factoring other than the factoring of input clauses. For this reason, ordered P_1-search is more efficient if no parent of an ordered P_1-resolvent has two identical literals.

Lemma 1. Let S be any unsatisfiable set of ground-level Horn clauses. Let $\underset{\sim}{D}$ be any P_1-refutation of S. Then there exists an SLN-refutation $\underset{\sim}{D}^*$ of S which is at least as simple as $\underset{\sim}{D}$.

Proof (by induction on the size of $\underset{\sim}{D}$). Without loss of generality, $\underset{\sim}{D}$ can be assumed to be an ordered P_1-refutation, and S can be assumed to be the set of ordered input clauses of $\underset{\sim}{D}$. The proof is for the stronger lemma which also proves that the ordered

negative input clause of $\underset{\sim}{D}$ is the initial ordered clause of $\underset{\sim}{D}^*$.

If $\underset{\sim}{D}$ has size one, then $\underset{\sim}{D}^*$ and $\underset{\sim}{D}$ are the same derivation, so in this case the theorem is trivially true.

Otherwise, let $\underset{\sim}{D}$ have size $n > 1$, and assume that the theorem holds for all ordered P_1-refutations of size less than n. Let $\overline{L}C^*$ be the ordered negative input clause of $\underset{\sim}{D}$. Then the immediate subderivations of $\underset{\sim}{D}$ derive $\overline{L}$ and L. Let $\underset{\sim}{D}_1$ be obtained by replacing $\overline{L}C^*$ with C^* in the immediate subderivation which derives $\overline{L}$. Then $\underset{\sim}{D}_1$ is an ordered P_1-refutation of the set S_1 of ordered input clauses of $\underset{\sim}{D}_1$. Also $\underset{\sim}{D}_1$ is isomorphic to the derivation of $\overline{L}$ and it has C^* as its negative input clause. Since the size of $\underset{\sim}{D}_1$ is less than n, then by the induction hypothesis there exists an SLN-refutation $\underset{\sim}{D}_1^*$ of S_1 which is at least as simple as $\underset{\sim}{D}_1$, and whose initial ordered clause is C^*.

If $\underset{\sim}{D}_1^*$ is $(C^*, C_1^*, \ldots, C_k^*)$ and C_1^* is the first ordered clause of $\underset{\sim}{D}_1^*$ which contains $\overline{L}$, then let $\underset{\sim}{D}^*$ be obtained from $\underset{\sim}{D}_1^*$ by concatenating $\overline{L}$ onto the left of each of $C^*, C_1^*, \ldots, C_{i-1}^*$, and inserting $\overline{L}C_i^*$ between $\overline{L}C_{i-1}^*$ and C_i^*. Then C_i^* is obtained from $\overline{L}C_i^*$ by merging out $\overline{L}$. ($\overline{L}$ must be new in the first clause of $\underset{\sim}{D}_1^*$ in which it occurs.) Then $\underset{\sim}{D}^*$ is an SLN-refutation of S with size less than n, and initial ordered clause $\overline{L}C^*$.

Otherwise, none of the ordered clauses in $\underset{\sim}{D}_1^*$ contain $\overline{L}$. Let $\underset{\sim}{D}_{11}^*$ be obtained from $\underset{\sim}{D}_1^*$ by concatenating $\overline{L}$ onto the left of each ordered chain of $\underset{\sim}{D}_1^*$. Then $\underset{\sim}{D}_{11}^*$ is an SLN-derivation of $\overline{L}$, of the same size as $\underset{\sim}{D}_1^*$, and with initial ordered clause $\overline{L}C^*$.

Let $\underset{\sim}{D}'$ be the immediate subderivation of $\underset{\sim}{D}$ which derives L. Let LC'* be the ordered input clause of $\underset{\sim}{D}'$ from which L descends. (That is, L occurs in all descendants of LC'* which are ancestors of L.) Let $\underset{\sim}{D}_2$ be obtained by replacing LC'* in $\underset{\sim}{D}'$ by C'*. Then $\underset{\sim}{D}_2$ is an ordered P_1-refutation of the set S_2 of ordered input clauses of $\underset{\sim}{D}_2$. Also $\underset{\sim}{D}_2$ is isomorphic to the derivation of I and has C'* as its negative input clause. Since the size of $\underset{\sim}{D}_2$ is less than n, then by the induction hypothesis there exists an SLN-refutation $\underset{\sim}{D}^*_2$ of S_2 which is at least as simple as $\underset{\sim}{D}_2$, and whose initial ordered clause is C'*.

Let $\underset{\sim}{D}^*_{21}$ be obtained from $\underset{\sim}{D}^*_2$ by adding $\overline{L}$ onto the beginning of the refutation $\underset{\sim}{D}^*_2$. Then the second ordered chain C'* of $\underset{\sim}{D}^*_{21}$ is obtained from $\overline{L}$ by the ordered N_1-resolution with parents $\overline{L}$ and LC'*.

Let $\underset{\sim}{D}^*$ be obtained by identifying the last ordered clause of $\underset{\sim}{D}^*_{11}$ with the initial clause of $\underset{\sim}{D}^*_{12}$ to form a refutation of S, which is at least as simple as $\underset{\sim}{D}$ and which has initial ordered clause $\overline{LC}^*$. Q.E.D.

It should be noted that if there is no merging in $\underset{\sim}{D}^*$ or if the use of the merging operation is suppressed in constructing $\underset{\sim}{D}^*$, then $\underset{\sim}{D}$ and $\underset{\sim}{D}^*$ have exactly the same size.

Since both ordered P_1-resolution and SLN-resolution use ordered clauses, the concept of lifting must be slightly modified. For any ordered clause $C = L_1 \ldots L_n$, and for any substitution θ, the ordered clause $C\theta = L_1\theta, \ldots, L_n\theta$ is an <u>ordered instance</u> of C. $C\theta$ may contain identical literals even though C does not.

The derivation $\underset{\sim}{D}$ lifts the derivation $\underset{\sim}{D}'$ iff

(1) $\underset{\sim}{D}$ is tree isomorphic to $\underset{\sim}{D}'$,

(2) for any ordered clause C at a node of $\underset{\sim}{D}$, the ordered clause C' at the corresponding node of $\underset{\sim}{D}'$ is an ordered instance of C, and

(3) the literal resolved upon in C' is an instance of the literal resolved upon in C and is in the same position in both clauses.

Lemma 2. Let $\underset{\sim}{D}'$ be an SLN-refutation of a set of ground instances of clauses in the set S of Horn clauses. Then there exists an SLN-refutation $\underset{\sim}{D}^*$ which lifts $\underset{\sim}{D}'$ and has the same size as $\underset{\sim}{D}'$.

Proof. Let S' be the set of ordered input clauses of $\underset{\sim}{D}'$, and let C_1' be the ordered negative clause of S'. Let S* be the set of ordered factors of clauses of S such that $C^* \in S^*$ iff S' contains an ordered instance C' of C*.

The initial ordered clause C_1^* of $\underset{\sim}{D}^*$ is the ordered clause in S* which has C_1' as an ordered instance. Assume that the SLN-derivation $(C_1^*, \ldots, C_i^*)$ lifts the subderivation $(C_1', \ldots, C_i')$ of $\underset{\sim}{D}' = (C_1', \ldots, C_i', \ldots, C_n')$. Then C_i^* is an ordered instance of C_i'.

If C_{i+1}' is obtained from C_i' by merging the i-th and j-th literals of C_i', then there exists a most general unifier σ which unites the i-th and j-th literals of C_i^* to produce C_{i+1}^*. If $C_i' = C_i^* \theta$, then $\theta = \sigma\lambda$ for some λ. If the i-th and j-th

literals are the only identical literals of C'_i, then they are the only identical literals of $C^*_i\sigma$. Let C^*_{i+1} be the factor of C^*_i obtainable by deleting the leftmost of the i-th or j-th literals of $C^*_i\sigma$. Clearly C'_{i+1} is an ordered instance of C^*_{i+1}.

Otherwise, let C'_{i+1} be obtained from C'_i by SLN-resolution with $C' \in S'$. Then there exists $C^* \in S^*$ such that C' is an ordered instance of C^*. By the lifting lemma of [29], there exists an SLN-resolvent C^*_{i+1} such that C'_{i+1} is an ordered instance of C^*_{i+1}.

In either case $(C^*_1, \ldots, C^*_i, C^*_{i+1})$ is an SLN-derivation which lifts $(C'_1, \ldots, C'_i, C'_{i+1})$. It follows that there exists an SLN-refutation $\underset{\sim}{D}^*$ which lifts $\underset{\sim}{D}'$ and has the same size as $\underset{\sim}{D}'$. Q.E.D.

Theorem 1. For any unsatisfiable set S of Horn clauses, there is an SLN-refutation $\underset{\sim}{D}^*$ of S such that $\underset{\sim}{D}^*$ is at least as simple as the simplest P_1-refutation of S.

Proof. By Lemma 6 (Section 3.5), there is a simplest P_1-refutation $\underset{\sim}{D}$ of S which lifts and has the same size as an ordered ground P_1-refutation $\underset{\sim}{D}'$ of a set S' of ordered ground instances of clauses in S. By Lemma 1, there is an SLN-refutation $\underset{\sim}{D}'^*$ of S' which is at least as simple as $\underset{\sim}{D}'$. By Lemma 2 there is an SLN-refutation $\underset{\sim}{D}^*$ which lifts $\underset{\sim}{D}'^*$ and which has the same size as $\underset{\sim}{D}'^*$. Therefore $\underset{\sim}{D}^*$ is an SLN-refutation of S which is at least as simple as the simplest P_1-refutation of S. Q.E.D.

2.4 Bi-directional refutations

As was suggested by the table of section 2.1, subderivations of a P_1-refutation and an SLN-refutation may be combined to form a bi-directional refutation.

Let S be an unsatisfiable set of Horn clauses, and let $\underset{\sim}{D}^*$ be an SLN-derivation from S. Let $\underset{\sim}{D}^*_k$ be the k-th subderivation of $\underset{\sim}{D}^*$, and let $\underset{\sim}{D}^*_k$ derive $C_k = \overline{L}_1 \ldots \overline{L}_n$. Let $\underset{\sim}{D}$ be an ordered P_1-derivation from S with subderivations $\underset{\sim}{D}_1, \ldots, \underset{\sim}{D}_n$ of $K_1, \ldots, K_n$ respectively. If $\{L_1, K_1\}, \ldots, \{L_n, K_n\}$ are simultaneously unifiable, then there is a stage k meeting between $\underset{\sim}{D}^*$ and $\underset{\sim}{D}$, where C_k is said to meet $K_1, \ldots, K_n$.

If $\underset{\sim}{D}^*_k$ is considered to be in the derivation format of Chapter 1, then it can be combined with the derivations $\underset{\sim}{D}_1, \ldots, \underset{\sim}{D}_n$ to form a stage k bi-directional refutation of S. Schematically, this bi-directional refutation has the form of figure 7.

In figure 7, each θ_i is the most general simultaneous unifier of $\{L_n, K_n\}, \ldots, \{L_i, K_i\}$. If $\underset{\sim}{D}^*_k, \underset{\sim}{D}_1, \ldots, \underset{\sim}{D}_n$ have sizes $m, m_1, \ldots, m_n$ then the bi-directional refutation has size $m+m_1+ \ldots +m_n+n$.

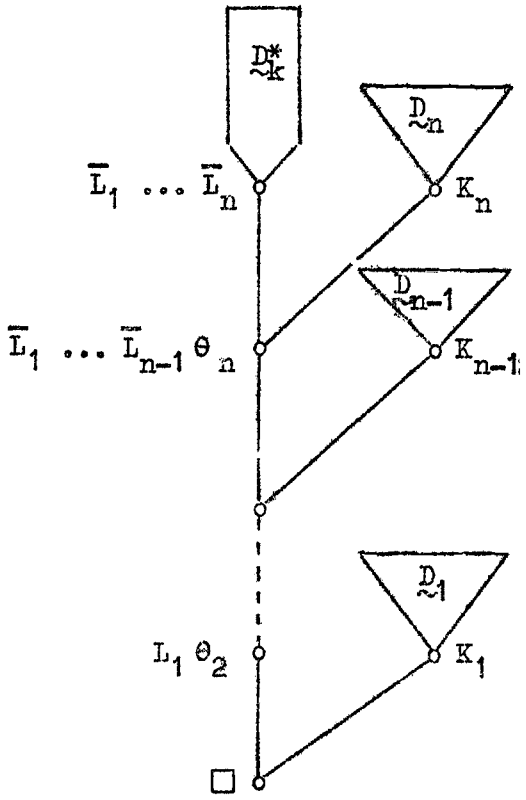


Figure 7.

It should be noted that a bi-directional refutation may be a pseudo-derivation in that C_k may contain identical literals. The following discussion is considerably simplified by allowing this. It is sufficient to note that for any such pseudo-derivation there is a derivation which is at least as simple. The identical literals are merged, and fewer ordered P_1-derivations are used to construct the bi-directional refutation.

Note also that any ordered P_1-refutation is a stage 1 bi-directional refutation, and that any SLN-refutation of size k is a stage k bi-directional refutation.

It is possible to define a bi-directional refutation in which ordered P_1-resolvents other than units may resolve with SLN-

resolvents, or in which units resolve with several SLN-resolvents in the same refutation. However, such extensions of the definition tend to make the search for a meeting more difficult.

Lemma 3. Let $\underset{\sim}{D}$ be any ordered P_1-refutation of the unsatisfiable set S of ground-level Horn clauses, and let $\underset{\sim}{D}$ have size s. Then there exists an SLN-refutation $\underset{\sim}{D}^*$ of S such that, for every k less than or equal to the size of $\underset{\sim}{D}^*$, there is a stage k meeting between $\underset{\sim}{D}$ and $\underset{\sim}{D}^*$. The corresponding stage k bi-directional refutation of S has size less than or equal to s.

Proof. The lemma will be proved by presenting a recursive method for the construction of $\underset{\sim}{D}^*$ from $\underset{\sim}{D}$. That is, the lemma to be proved is that for every positive integer i, either (1) there is an SLN-refutation $\underset{\sim}{D}^*_j = (C_1, \ldots, C_j)$ of S for some $j \leq i$ and there is a k-level meeting between $\underset{\sim}{D}^*_j$ and $\underset{\sim}{D}$ for all $k \leq j$, or (2) there is a derivation $\underset{\sim}{D}^*_i = (C_1, \ldots, C_i)$ from S and there is a stage k meeting between $\underset{\sim}{D}^*_i$ and $\underset{\sim}{D}$ for all $k \leq i$.

If $C_1 = (\overline{L}_1, \ldots, \overline{L}_n)$ is the ordered negative input clause of $\underset{\sim}{D}$, then $\underset{\sim}{D}^*_1 = (C_1)$. Since $\underset{\sim}{D}$ is a refutation, it has subderivations $\underset{\sim}{D}_1, \ldots, \underset{\sim}{D}_n$ of $L_1, \ldots, L_n$. If these subderivations have size $m_1, \ldots, m_n$ then $m_1 + \ldots + m_n + n = s$. Thus there is a stage 1 meeting between $\underset{\sim}{D}^*_1$ and $\underset{\sim}{D}$. The resulting bi-directional refutation is $\underset{\sim}{D}$ itself.

Let i be any positive integer and assume that the lemma holds for i. If (1) holds, then the lemma is proved.

Otherwise, assume that there is a derivation

$\underset{\sim}{D}^*_i = (C_1, \ldots, C_i)$ such that there is a level k meeting between $\underset{\sim}{D}^*_i$ and $\underset{\sim}{D}$ for every $k \leq i$. If C_i is the null clause, then (1) holds, and the lemma is proved.

Otherwise, let $C_i = \overline{L}_1 \ldots \overline{L}_n$. Then $\underset{\sim}{D}$ has subderivations $\underset{\sim}{D}_1, \ldots, \underset{\sim}{D}_n$ of $L_1, \ldots, L_n$ such that if these subderivations have size $m_1, \ldots, m_n$ then, by assumption, $m+m_1,+ \ldots +, m_n+n \leq s$, where m is the size of $\underset{\sim}{D}^*_i$.

If C_i contains two identical literals, then let $\underset{\sim}{D}^*_{i+1} = (C_1, \ldots, C_i, C_{i+1})$ where C_{i+1} is obtained from C_i by merging two identical literals. Since C_i meets $L_1, \ldots, L_n$ it follows that C_{i+1} meets a proper subset of $L_1, \ldots, L_n$ so that the level i+1 bi-directional refutation is at least as simple as the level i bi-directional refutation. Since the latter refutation was assumed to have size less than s, the lemma is proved.

Otherwise, C_i does not contain identical literals and C_{i+1} must be obtained from C_i by SLN-resolution. If L_n is an ordered input clause, then the only clause of $\underset{\sim}{D}_n$ is L_n. In this case, let $\underset{\sim}{D}^*_{i+1} = (C_1, \ldots, C_i, C_{i+1})$ where $C_{i+1} = \overline{L}_1 \ldots \overline{L}_{n-1}$ is obtained from C_i by SLN-refutation with L_n.

Otherwise, let $L_n \overline{K}_1 \ldots \overline{K}_r$ be the ordered input clause of $\underset{\sim}{D}_n$ from which L_n descends. Then $\underset{\sim}{D}_n$ has subderivations $\underset{\sim}{D}'_1, \ldots, \underset{\sim}{D}'_r$ of $K_1, \ldots, K_r$ which have sizes $m'_1, \ldots, m'_r$ where $m'_1+ \ldots +m'_r+r = m_n$. Let $\underset{\sim}{D}^*_{i+1} = (C_1, \ldots, C_i, C_{i+1})$ where $C_{i+1} = (\overline{L}_1, \ldots, \overline{L}_{n-1}, \overline{K}_1, \ldots, \overline{K}_r)$ is obtained from C_i by SLN-resolution with $(L_n, \overline{K}_1, \ldots, \overline{K}_r)$. But $\underset{\sim}{D}$ contains subderivations

$\underset{\sim}{D}_1$, ..., $\underset{\sim}{D}_{n-1}$, $\underset{\sim}{D}'_1$, ..., $\underset{\sim}{D}'_r$ of L_1, ..., L_{n-1}, K_1, ..., K_r whose sizes are m_1, ..., m_{n-1}, m'_1, ..., m'_r. Since $m+m_1+ \ldots +m_n+n \leq s$ and $m'_1+ \ldots +m'_r+r = m_n$, the size of the (stage i+1 bi-directional refutation obtainable from $\underset{\sim}{D}^*_{i+1}$, $\underset{\sim}{D}_1$, ..., $\underset{\sim}{D}_{n-1}$, $\underset{\sim}{D}'_1$, ..., $\underset{\sim}{D}'_r$ is less than or equal to s.

Since it has been shown that $\underset{\sim}{D}^*_{i+1}$ can be constructed for any i when $\underset{\sim}{D}^*_i$ is not a refutation, it follows that $\underset{\sim}{D}^*$ can be constructed and that for every k less than or equal to the size of $\underset{\sim}{D}^*$, there is a stage k meeting between $\underset{\sim}{D}^*$ and $\underset{\sim}{D}$, and that the size of the corresponding stage k bi-directional refutation is less than s. Q.E.L.

The following lemma and theorem follow easily from Lemma 3, using the methods for proving Lemma 2 and Theorem 1.

Lemma 4. Let $\underset{\sim}{D}'$ be a bi-directional refutation of a set of ground instances of clauses in the set S of Horn clauses. Then there exists a bi-directional refutation $\underset{\sim}{D}^*$ of S which lifts $\underset{\sim}{D}'$ and has the same size as $\underset{\sim}{D}'$.

Theorem 2. For any unsatisfiable set S of Horn clauses, there is a bi-directional refutation of S which is at least as simple as the simplest P_1-refutation of S.

2.5 A Bi-directional Search Strategy

In the following discussion of a bi-directional search strategy, it is assumed that the strategy should search so that the first found bi-directional refutation is a simplest bi-directional refutation obtainable. That is, the search strategy should exhaust the possibility of finding a refutation of size k before trying to find a refutation of size $k+1$. Theorem 2 expresses the power of bi-directional refutations in the anticipation of this kind of search strategy.

Let C be an SLN-resolvent of cost g and length h, where the cost of C is the size of the derivation of C, and the length of C is its number of literals. There is a bi-directional refutation of size g^* if C meets h P_1-resolvent units, the sum m of whose costs satisfies $g+h+m = g^*$. To ensure that all such units have been generated, it is necessary that all P_1-resolvent units of cost m or smaller have been generated. It is easy to verify that all ancestors of a unit of cost m have cost g' and length h' where $g'+h' \leq m+1$. Let the merit of an ordered clause be defined to be the sum of its cost and its length. If an SLN-resolvent of merit $g+h$ has been generated and if the search is attempting to generate a refutation of size g^*, then the search should generate all ordered P_1-resolvents of merit $g'+h' \leq g^*+1 - (g+h)$. If all such P_1-resolvents have been generated, then to exhaust the possibility of finding a refutation of size g^*, it is necessary and sufficient to generate all SLN-resolvents of merit $g+h$.

Defining merit in this way layers the SLN- and ordered

P_1-search spaces according to Kowalski's strategy of diagonal search [11]. Most of the strategies of Chapter 6 apply to the searching of these merit levels.

The meeting of SLN- and ordered P_1-derivation has been defined to use only positive unit P_1-resolvents, while any SLN-resolvent may be used to meet these units. There are three searches, the ordered P_1-search for units, the SLN-search, and the search for a meeting between ordered P_1-resolution units and SLN-resolvents. It seems most natural to combine the last two of these searches by using the ordered P_1-resolvent units to augment the SLN-search. Call such units the imported units of the SLN-search. Since the imported units are intended to be used immediately if they are to be used at all, they are treated as input clauses by the SLN-search. Because the purpose of the ordered P_1-search is to produce positive units, no negative clauses should be used as input clauses for the P_1-search.

For any ordered P_1-refutation $\underset{\sim}{D}$ of a set S of Horn clauses there is an SLN-refutation $\underset{\sim}{D}^*$ of S such that $\underset{\sim}{D}$ and $\underset{\sim}{D}^*$ meet at any stage. Since at least one clause of such a $\underset{\sim}{D}^*$ occurs on each merit level up to the size of $\underset{\sim}{D}^*$, each merit level contains at least one clause which meets units of $\underset{\sim}{D}$. Because of this, the SLN-search may be stopped at any merit level, and a meeting will be obtained by generating ordered P_1-resolvents. By generating all ordered P_1-resolvents up to a merit level such that the sum of the merit levels is the size of the refutation, a meeting is generated. A similar argument holds for stopping the ordered P_1-search at any merit level. Thus the number of merit levels saturated by each

search is immaterial as long as the sum of the merit level reaches the size of the refutation.

A bi-directional search strategy is most efficient if it saturates as many merit levels as possible while generating as few clauses as possible. The implementation of a bi-directional search requires an alternation between searching an ordered P_1-search space and searching an SLN-search space. To be most efficient, the alternation should be controlled by the relative number of clauses on the merit levels of the two searches.

On each merit level, the ordered P_1-search and the SLN-search count the number of clauses generated on that level. One search generates clauses until its count exceeds that of the other search. Then the other search begins generating clauses. If search A saturates a merit level before its count exceeds that of search B, then search A begins generating clauses on its next merit level, and its count becomes the number of clauses generated on the new level. In this way, one search may generate all the clauses on one merit level while the other search is inactive. In doing this, the bi-directional search is saturating as many merit levels as possible while generating as few clauses as possible. This also ensures that the difficulty of the bi-directional search is less than or equal to the difficulty of either search by itself. This method resembles the bi-directional search procedure suggested by Pohl [23].

This alternating procedure is interrupted whenever a unit positive clause is generated by the P_1-search and imported to the

SLN-search. The SLN-search uses such a newly imported clause in all possible ways up to its current merit level. This interruption is necessary since the bi-directional search can be terminated only by finding a refutation during the SLN-search.

Apart from the efficiency gained by alternating between search spaces, the efficiency of bi-directional search is based on the assumption that the number of clauses on a merit level increases as the merit increases. It follows that the number of clauses generated in the search up to merit level n is less than half the number generated in the search up to merit level 2n. If an ordered P_1-search and an SLN-search produce a meeting after searching m and n levels respectively, then the bi-directional search generates fewer clauses than if either search were to go to level m+n.

One example of bi-directional search uses the following set of Horn clauses.

$$P(x,e,x)$$

$$P(x,g(x),e)$$

$$\overline{P}(g(a),a,e)$$

$$\overline{P}(x,y,u)\ \overline{P}(y,z,v)\ \overline{P}(u,z,w)\ P(x,v,w)$$

$$\overline{P}(x,y,u)\ \overline{P}(y,z,v)\ \overline{P}(x,w,w)\ P(u,z,w)$$

The ordered P_1-search program of Isobel Smith generated 35 clauses and retained 21. When SLN-search was done by hand, 24 clauses were generated and 16 retained. Hand-done bi-directional search generated 17 clauses and retained 14. In each case, the refutation had size 7.

With some care with the details of the search, the efficiency of bi-directional search can be improved. For instance, if all clauses up to and including merit levels m and n have been generated by the P_1- and SLN-searches, and no meeting has occurred, then there is no bi-directional refutation of size m+n+1 or less. In this case, no imported unit of merit m+1 should be used to meet an SLN-resolvent of merit less than n, for that would be an attempt to find a bi-directional refutation of size m+n+1 or less.

It is interesting to note that Chang [3] proved that for any purely linear refutation, which he calls an input clause refutation, there exists a unit refutation. A unit refutation is one all of whose resolutions have a unit as one parent. Any P_1-refutation with Horn clauses is a unit refutation, and any SLN-refutation with Horn clauses is an input clause refutation. It is an open question whether the class of unsatisfiable sets with purely linear refutations includes more than Horn clauses, or clauses which can be renamed as Horn clauses.

Chapter 3. Linear Resolution Systems

3.1 Introduction

The bi-directional search techniques of Chapter 2 can be extended to input sets which contain non-Horn clauses. But the search for a meeting becomes so difficult that a bi-directional search is less efficient than either of the two searches by itself. However, the search spaces for SLN-resolution are so sparse that an extension of SLN-resolution to non-Horn clauses seems worth investigating.

In order to extend SLN-resolution, its restrictions must be weakened. That is, if every resolution must have one negative parent, one input parent, and have both parents ordered then, for some unsatisfiable sets of non-Horn clauses, no refutation can be constructed. If the restrictions of SLN-resolution are weakened, then the resulting search spaces are not as sparse as the corresponding SLN-search spaces. However, when the restrictions of an inference rule are weakened, its power generally increases. This would be an advantage because it seems difficult to prove that SLN-resolution is any more powerful than P_1-resolution. It is proposed to consider first linear resolution. The linear resolution rule is a weakening of the restrictions of SLN-resolution which is as powerful as the most powerful resolution rule.

This chapter considers linear resolution and certain restrictions to linear resolution which can be applied without decreasing the power of the rule. Chapter 4 considers a further restriction to linear resolution which somewhat decreases its power but for

which there is a great gain in the sparseness of the search spaces.

Most of Chapters 3 and 4 were written with Robert Kowalski and have appeared separately in [15]. I have tried to indicate portions which are due solely to Kowalski, but we worked in such close collaboration that most of these chapters must be considered to be a joint achievement. Kowalski's contribution is particularly evident in the careful reasoning of the proofs, and in the elegance of the very strong admissibility restriction on SL-resolution.

3.2 Linear Derivations

Linear resolution was independently discovered by Loveland [17], Luckham [19] and Zamov and Sharonov [33]. It is a refinement of unrestricted resolution whose search spaces are significantly more sparse than the corresponding search spaces for unrestricted resolution. For certain measures of complexity, such as size, linear resolution can be proved to be as powerful as unrestricted resolution. That is, no form of resolution is more powerful than linear resolution. Linear resolution has the particular advantage that it offers exceptional opportunities for the application of heuristic search because of the relatively uncomplicated structure of its search spaces.

A linear derivation $\underset{\sim}{D}$, from a set S of clauses, 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 initial clause of $\underset{\sim}{D}$ and C_n is the clause derived by $\underset{\sim}{D}$. In case (1), C_{i+1} is obtained by input resolution and, in case (2), by ancestor resolution. If $\underset{\sim}{D}$ derives the null clause from S, then $\underset{\sim}{D}$ is a linear refutation of S.

The sequence $\underset{\sim}{D} = (PQ, Q, R, S, \overline{RT}, T, P, \square)$ is a linear refutation of $S = \{PQ, \overline{P}, \overline{Q}R, \overline{R}S, \overline{RST}, P\overline{T}\}$. Notice that, in this 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

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

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

Note that, in linear derivations, factoring is considered to be part of the resolution rule.

For any set S of input clauses, the linear derivation search space for S is a finite set of disjoint search trees. For each input clause C_1 which is an initial clause for linear derivations, there is a search tree $T = T(C_1)$ satisfying the following.

(1) C_1 is at the root node of T.

(2) If C_n is a clause of T, derived by $(C_1, \ldots, C_n)$, then any clause C_{n+1} is an immediate descendant of C_n in T iff the derivation of C_{n+1} is $(C_1, \ldots, C_n, C_{n+1})$.

Figure 8 illustrates part of a linear derivation search tree. Although this search tree is sparse compared to the corresponding search space for unrestricted resolution, it is evident that there are many redundant derivations which are admitted by linear resolution. In the remainder of this chapter and in Chapter 4, refinements of linear resolution are considered which successively remove most of the redundancies of figure 8.

Input Set

PQ

$\overline{P}$

$\overline{Q}R$

$\overline{R}S$

$\overline{RST}$

$P\overline{T}$

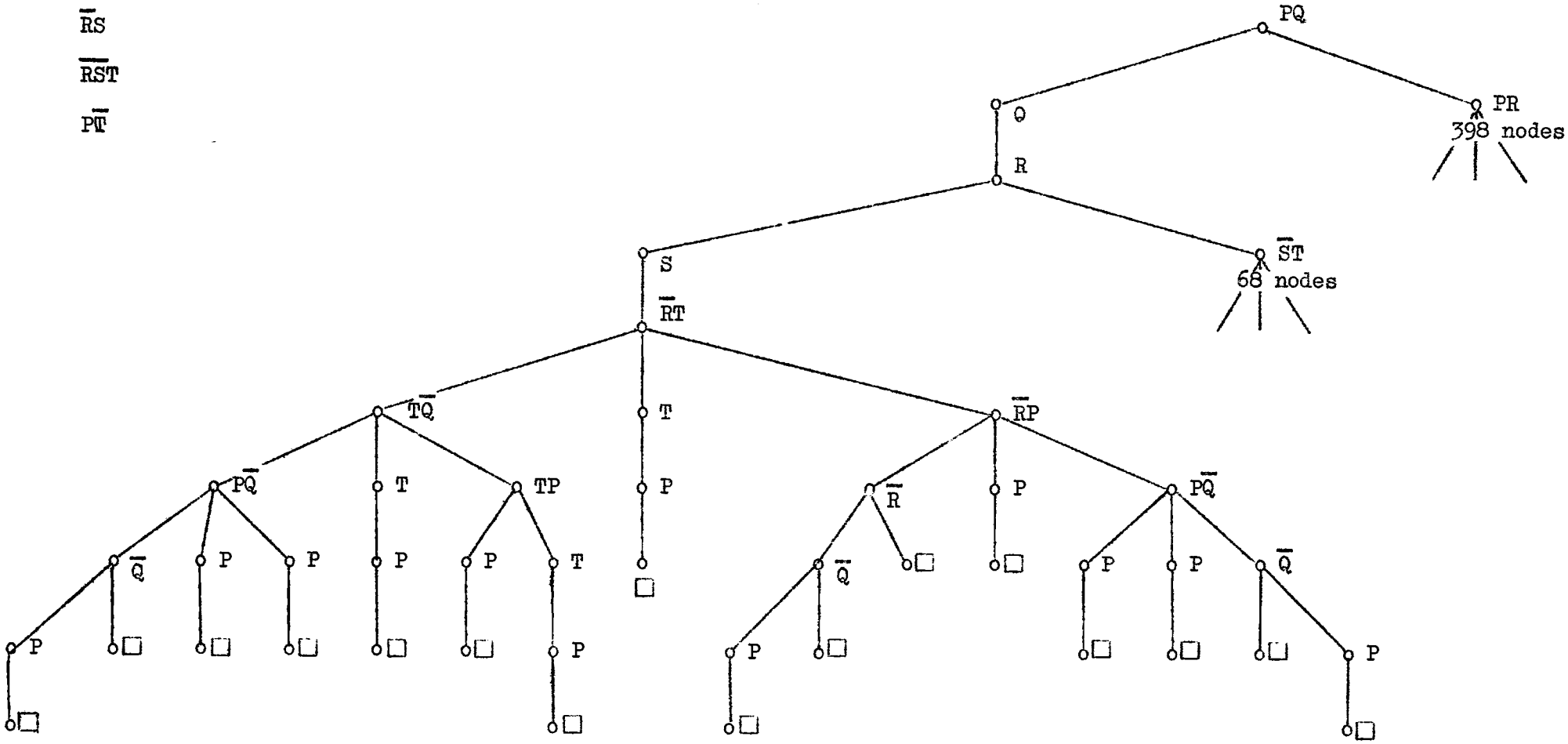


Figure 8. Search tree for linear resolution (513 nodes).

3.3 Refinements of linear resolution

As with SLN-resolution, it is possible to impose on linear resolution the restrictions that no resolvent is a tautology and that the initial clause belongs to a given support set of the input set S. Both restrictions increase the sparseness of search spaces 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). A more detailed discussion of support sets occurs in Chapter 5. For the example of figure 8, the initial clause is the only clause in the support set of positive clauses, so that in this case the search space consists of a single search tree. All of the refinements of linear resolution discussed in this thesis 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 [17] and Zamov and Sharonov [33]) and merging restrictions (Anderson and Bledsoe [1], Yates et al [32], and Kieburtz and Luckham [10]). The t-linear and SL-resolution systems investigated in this and the next chapters 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 is not investigated. 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 figure 8.

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

The combination of linear resolution and the merging restriction defined in [1] is denoted by 'm-linear'; 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 choice of the order of input parents in SLN-resolution. For each choice of order for SLN-resolution, as with each choice of selection function for SL-resolution, there is a different search space.

As is justified by Meltzer in [21], the input clauses of this example can be renamed to be Horn clauses. Rename P to be $\overline{U}$, and $\overline{P}$ to be U. The corresponding SLN-search space consists of the single search tree whose root is labelled by $\overline{U}\ \overline{T}$. The search space in this case contains no redundant or irrelevant clauses, and consists only of a derivation of the null clause. This is an

indication of the extreme sparseness of SLN-search spaces.

The three new restrictions incorporated in t-linear resolution are defined only for derivations from input sets of ground clauses. The extension of the definition to sets of general clauses is not difficult, but the complications involved obscure the discussion.

Let $\underset{\sim}{D} = (C_1, \ldots, C_n)$ 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, $j \leq k \leq i$. 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 $(PQ, Q, R, S, \overline{RT}, T)$ from the input set $\{PQ, \overline{P}, \overline{QR}, \overline{RS}, \overline{RST}\}$, the derived clause C_6 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.

A linear derivation $\underset{\sim}{D}$ 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.

It has already been remarked that the no-tautologies and support set restrictions are compatible with t-linear resolution. Figure 9 illustrates part of the t-linear search space for the example of figure 8.

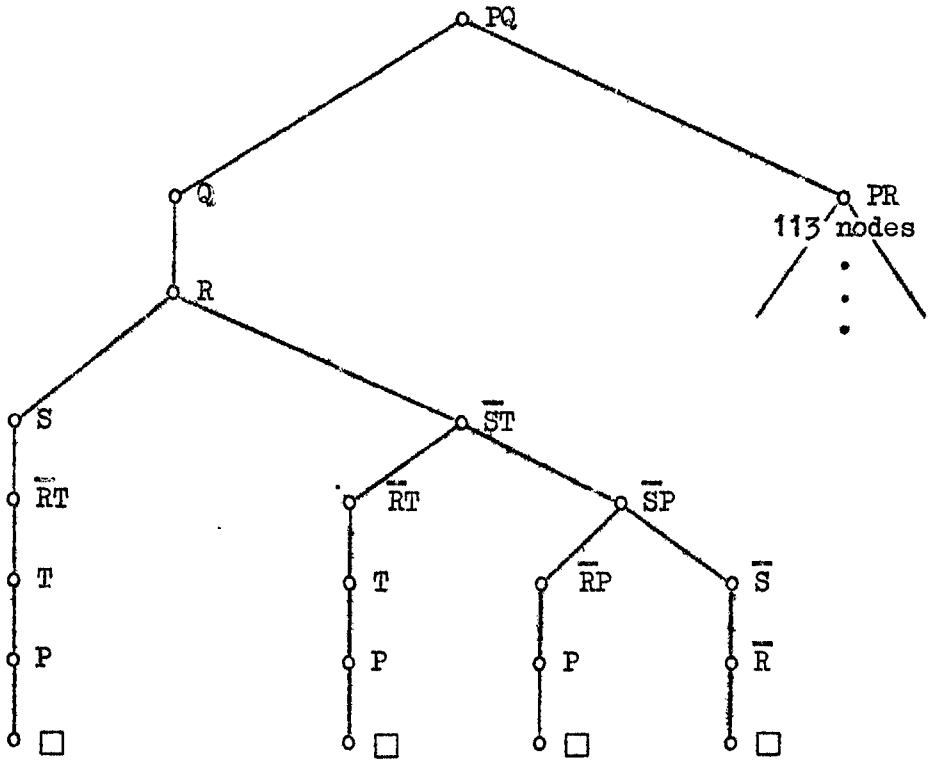


Figure 9. Search tree for t-linear resolution (134 nodes).

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 [17] and Zamov and Sharonov's absorption restriction [33].) The second condition states that ancestor resolution is compulsory in the sense that it must be performed as soon as it can be performed.

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 chapter, there is defined a chain format for SL-derivations which provides just such a way of associating A-literals with clauses.

It is instructive to compare ancestor resolution in t-linear derivations with the implicit merging operation. The merging operation is implicit in the representations of clauses as sets of literals. If clauses were replaced by ordered clauses, 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.

3.4 Minimal Derivations and rm-size

In order to investigate the power of linear and SL-resolution, their refutations will be compared with minimal refutations. Minimal refutations include the simplest obtainable by any resolution rule. Moreover, every minimal refutation (whether simplest or not) can be regarded as reasonably simple for the theorem it proves. It will be shown 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.

A branch of a non-linear derivation D consists of a node labelled by an input clause C, together with all nodes of D labelled by the descendants of C. 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 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. That is, it derives a non-tautology and, for every literal resolved upon at a node, its atom does not occur in any clause at a descendant node. A general derivation is minimal if it lifts a minimal ground derivation. (It is tree-isomorphic, the clause at any node has as an instance the clause at the corresponding node, etc.)

Figure 10 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 refutation has 5 branches, size 4

and level 3. The literal Q is resolved upon twice in the leftmost branch of the non-minimal derivation.

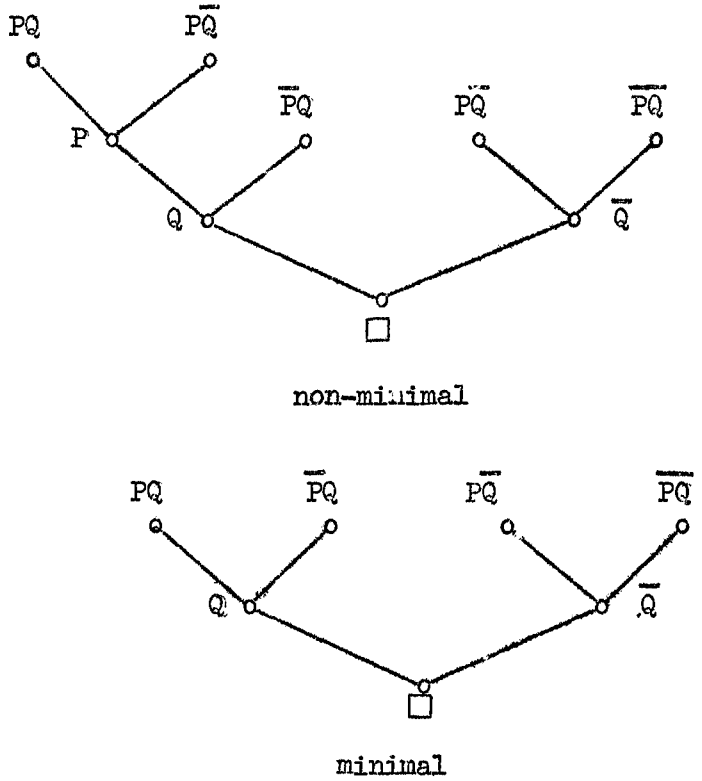


Figure 10. Non-linear refutations of $\{PQ, \overline{P}Q, \overline{PQ}, \overline{\overline{P}Q}\}$

If a set S of ground clauses contains exactly n distinct atoms, then there are only finitely many minimal derivations from S, none of which has size greater than $2^n - 1$ or a branch with more than n+1 nodes. Under quite general conditions on S (which apply, in particular, to the example of figure 10) there are infinitely many non-minimal derivations and refutations of unbounded 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 $\overline{L}$.)

The notion of minimal derivations was introduced by Loveland [17] and investigated independently by Kowalski [12] in conjunction with Pat Hayes. Minimal derivations are just those derivations which can be obtained by the construction of semantic trees (Hayes and Kowalski [6]). 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 3 of the next section).

* * *

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. Ancestor resolution is to be considered a form of factoring. If this is done, then factoring and merging must be considered to be explicit operations in both linear and non-linear derivations if their complexities are to be compared. As with SLN-resolution, Kowalski's m-factoring [12] will be used. That is, all input clauses are initially factored in all possible ways. Subsequently, resolvents and factors may be factored, and must be merged, provided that one of the unified literals is new. Neither parent of a linear or non-linear resolution operation may have two identical literals. These restrictions avoid the redundant refactoring of clauses. Each factoring operation is considered to factor out only one literal.

Define the rm-size of a non-linear derivation to be the pair (r,m) where m is the number of factoring operations performed in the derivation and r the number of resolution (of factors) operations. 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 factoring operations. When factoring is explicitly displayed, then $r+m$ is the number of non-input clauses of the derivation.

In figure 10, the minimal derivation has rm-size $(3,2)$ and the non-minimal derivation has rm-size $(4,2)$.

For both linear and non-linear derivations, m does not include the number of initial factoring operations applied to input clauses. For linear 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.

If complexity is defined as any function of r and m then two derivations, linear 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 $(r_1,m_1) \leq (r_2,m_2)$ 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 $r_1 \leq r_2$ and $m_1 \leq m_2$ imply $(r_1,m_1) \leq (r_2,m_2)$, and that $(r,m) \leq (r+n, m-n)$. Therefore a refutation $\underset{\sim}{D}$ is a simplest refutation of S if its rm-size is less than or equal to the rm-size of all other refutations of S.

3.5 The Completeness and Power of Linear Resolution

Linear resolution will be shown to be as powerful as any resolution rule by proving that there is a linear refutation as simple as the simplest minimal refutation. Thus it is first necessary to prove that there is a minimal refutation which is as simple as the simplest obtainable by unrestricted resolution.

Lemma 5. Let $\underset{\sim}{D}'$ be a non-linear ground refutation of a set of ground instances of clauses in S. Then there exists a refutation $\underset{\sim}{D}$ of S which lifts $\underset{\sim}{D}'$ and has the same rm-size.

The proof is not difficult and is similar to that of Kowalski's Theorem 4.7.1 in [12].

Lemma 6. For any unsatisfiable set S 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 $\underset{\sim}{D}$ be a simplest non-linear refutation of S and assume it lifts a ground refutation $\underset{\sim}{D}'$. Note that $\underset{\sim}{D}$ cannot be simpler than $\underset{\sim}{D}'$. By using Lemma 5 and the fact that $\underset{\sim}{D}$ is simplest and lifts $\underset{\sim}{D}'$, it is easy to verify that $\underset{\sim}{D}$ and $\underset{\sim}{D}'$ have the same rm-size. It follows from a second application of Lemma 5 that $\underset{\sim}{D}'$ is a simplest ground refutation of a set of instances of clauses in S.

If $\underset{\sim}{D}$ 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 construct-

ing a ground 'pseudo-derivation' isomorphic to and having the same rm-size as $\underset{\sim}{D}$. (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 operation than it has fewer resolution operations. Therefore it is at least as simple as the pseudo-derivation. By Lemma 5, 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 $\underset{\sim}{D}$ and is therefore a simplest refutation of S. Q.E.D.

(The definition of a pseudo-derivation is given in [7] and the contraction operation for a pseudo-derivation is the analogue of the contraction operation for derivations studied in [7] and [12].)

Lemma 7. For every unsatisfiable set of ground clauses, there exists a simplest ground refutation which is also minimal.

Proof Outline. Let $\underset{\sim}{D}$ be a simplest ground refutation of the set, S. By Loveland's Corollary 2 [17], if $\underset{\sim}{D}$ is not minimal then it can be 'pruned' to obtain a minimal refutation $\underset{\sim}{D}'$ of S. The pruning operation removes resolution operations and introduces no more merging operations than the resolution operation it removes. Therefore $\underset{\sim}{D}'$ is a simplest refutation of S. Q.E.D.

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

Proof. By Lemma 6, there is a simplest refutation of the

set S which lifts and has the same rm-size as a simplest refutation $\underset{\sim}{D}'$ of a set S' of instances of clauses in S. By Lemma 7, there is a minimal refutation $\underset{\sim}{D}''$ of S' which is as simple as $\underset{\sim}{D}'$. By Lemma 5, there is a refutation $\underset{\sim}{D}$ of S, which lifts $\underset{\sim}{D}''$ and has the same rm-size as $\underset{\sim}{D}''$. Therefore $\underset{\sim}{D}$ is a minimal and simplest refutation of S. Q.E.D.

* * *

The following theorems are proved not for linear resolution but for s-linear resolution. Since s-linear resolution is a refinement of linear resolution, Lemmas 8 and 9, and Theorem 4 apply to linear resolution as well. The proofs for t-linear resolution follow much the same line, but are more complicated.

Lemma 8. Let $\underset{\sim}{D}$ be a minimal ground refutation of a set S of ground clauses. For any input clause C_1 of $\underset{\sim}{D}$, there is an s-linear refutation of S with initial clause C_1 and having the same rm-size as $\underset{\sim}{D}$.

Proof Outline (illustrated in figure 11 and appearing in full detail in [14]). The proof is by induction on the size n of $\underset{\sim}{D}$. 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 $\underset{\sim}{D}$ derive the unit clauses $\{L\}$ and $\{\overline{L}\}$. Because $\underset{\sim}{D}$ is minimal, if we delete from all clauses at nodes of $\underset{\sim}{D}$ the literals L and $\overline{L}$, we obtain minimal refutations $\underset{\sim}{D}_1$ of S_1, and $\underset{\sim}{D}_2$ of S_2, tree-isomorphic respectively to the subderivations of $\{L\}$ and $\{\overline{L}\}$. Suppose that $C_1 - \{L\}$ occurs as

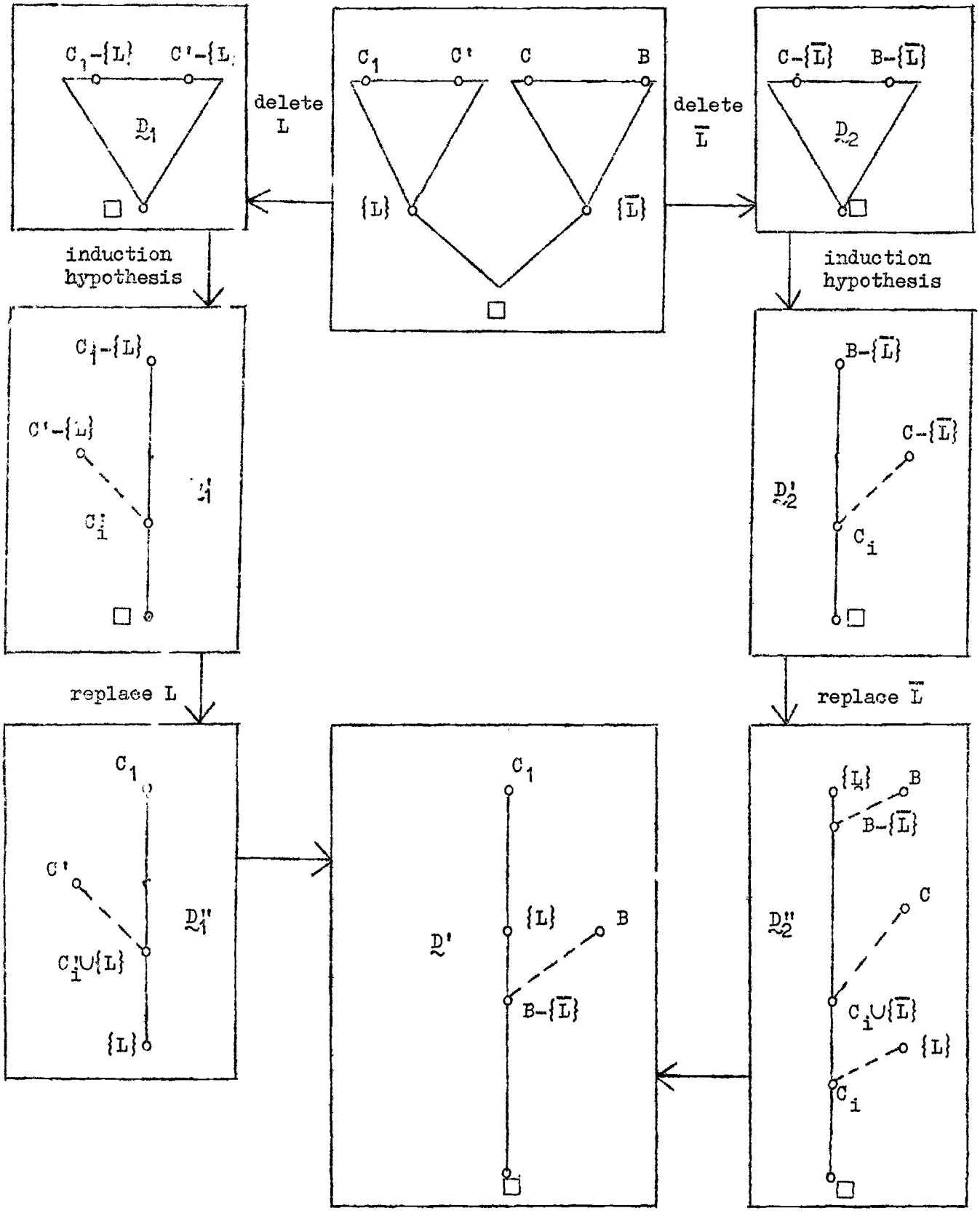


Figure 11. Outline of the proof of Lemma 8. (A broken line, here and in figure 14, connects a resolvent with its input or far parent.)

an input clause of $\underset{\sim}{D}_1$, and B - $\{\overline{L}\}$ (where $\overline{L} \in B$ and $B \in S$) is an input clause of $\underset{\sim}{D}_2$.

By the induction hypothesis, there exist s-linear refutations $\underset{\sim}{D}_1'$ of S_1 with initial clause $C_1 - \{L\}$, and $\underset{\sim}{D}_2'$ of S_2 with initial clause B - $\{\overline{L}\}$. $\underset{\sim}{D}_i$ and $\underset{\sim}{D}_i'$ have the same rm-size.

Let $\underset{\sim}{D}_1''$ be the s-linear derivation of $\{L\}$ from S, isomorphic to $\underset{\sim}{D}_1'$, with initial clause C, 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 $\underset{\sim}{D}_2''$ be obtained from $\underset{\sim}{D}_2'$ by first inserting $\{L\}$ as new initial clause before B - $\{\overline{L}\}$ and by next inserting immediately before any resolvent C_i with near parent of the form C - $\{\overline{L}\}$, where $C \in S$ and $\overline{L} \in S$, the clause $C_i \cup \{\overline{L}\}$. It is easy to check that $\underset{\sim}{D}_2''$ is an s-linear refutation of $S \cup \{\{L\}\}$, where $\{L\}$ occurs only as initial clause. ($\{L\}$ is treated as far parent for resolvents C_i in $\underset{\sim}{D}_2''$ with near parents $C_i \cup \{\overline{L}\}$.)

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

Lemma 9. Let $\underset{\sim}{D}$ be an s-linear refutation of a set of ground instances of clauses in S. Then there exists an s-linear refutation of S which lifts $\underset{\sim}{D}$ and has the same rm-size.

The proof of Lemma 9 is similar to, but much simpler than, the proof of Lemma 11.

Theorem 4. For any unsatisfiable set S and support subset S_0, there exists an s-linear refutation of S with initial clause in S_0 such that no non-linear refutation of S is simpler.

Proof. As in the proof of Theorem 3, there is a simplest non-linear refutation $\underset{\sim}{D}$ of S which lifts and has the same rm-size, as a simplest minimal refutation $\underset{\sim}{D}'$ of a set of ground instances S' of clauses in S. Some input clause C_1' of $\underset{\sim}{D}'$ is an instance of some clause in S_0. By Lemma 8, there is an s-linear refutation $\underset{\sim}{D}''$ of S' with input clause C_1' and having the same rm-size as $\underset{\sim}{D}'$. By Lemma 9, there exists an s-linear refutation of S, with top clause C_1 in S_0, which has the same rm-size as $\underset{\sim}{D}''$ and therefore is as simple as a simplest non-linear refutation of S. Q.E.D.

Chapter 4. SL-Resolution

4.1 Informal Definition

SLN-resolution can be extended and t-linear resolution can be restricted to SL-resolution. The ordering of clauses of SLN-resolution can be considered to be a selection of the literal to be resolved upon. 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 $\underset{\sim}{D}$ 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 C in t-linear search tree than there are in the SL-search tree. If m is the average number of literals in clauses derived by t-linear derivations of size $\leq n$, then there are, on the average, m^n more t-linear derivations of size n than there are SL-derivations of the same size.

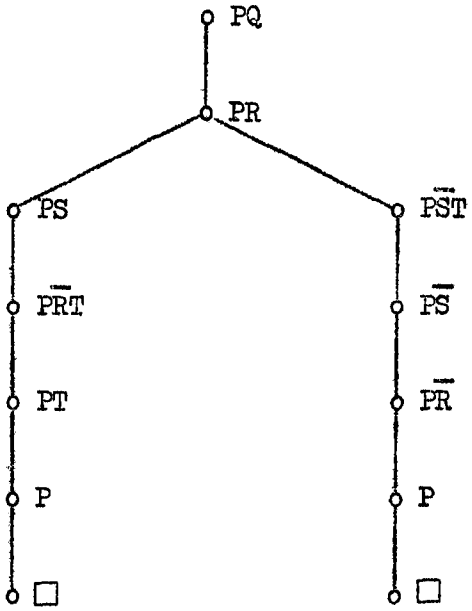


Figure 12. Search tree for SL-resolution (12 nodes)

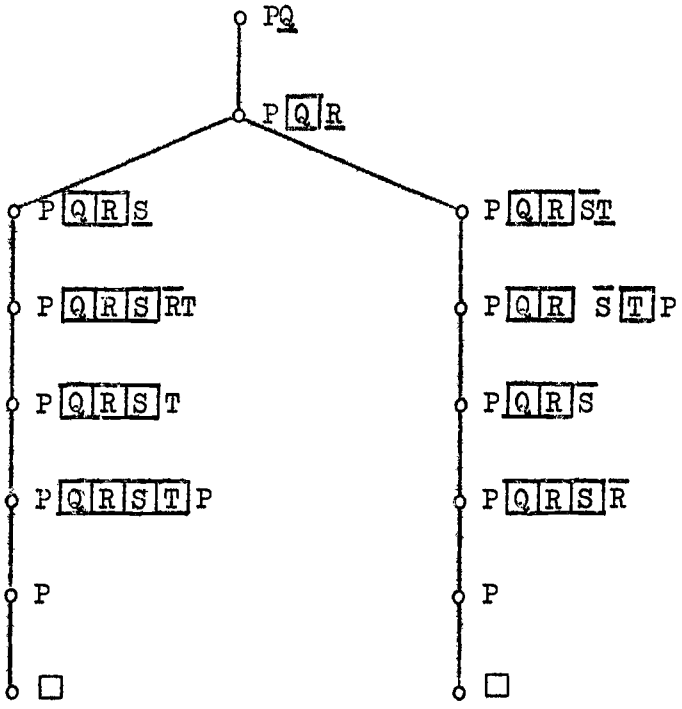


Figure 13. Search tree for SL-resolution in chain format.

Figure 12 illustrates, for the example of figures 8 and 9, the entire search tree for SL-resolution with the selection function which chooses the literal having alphabetically greatest atom. The SL-search tree has only 12 nodes, which compares favourably with the 134 nodes of the t-linear search tree and the 513 nodes of the linear search tree.

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 figure 12, 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 is inserted 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.

Figure 13 illustrates in chain format the SL-search tree of figure 12. A-literals are enclosed in boxes. Merging operations are displayed explicitly. Of two identical literals in a chain,

the rightmost is deleted. Literals resolved upon and literals removed by the merging operation are underlined. The operation of deleting A-literals is not displayed, although defined explicitly in the formal definition.

The definition of SL-resolution treats chains in the same way that separate and explicit rules for factoring and resolution of factors treats 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.

4.2 Formal Definition

Let S be a given set of input 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

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

there is only one corresponding set of input chains. Of the 9 input chains, one is obtained by factoring. For

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

each corresponding set of input chains contains exactly 12 members. Each such set contains exactly one of $\overline{RST}$ and $\overline{R}\overline{T}\overline{S}$, one of $\overline{S}\overline{R}T$ and $\overline{S}T\overline{R}$, and one of $T\overline{R}\overline{S}$ and $TS\overline{R}$. For the purpose 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. Two chains are equivalent if one can be obtained from the other by permuting B-literals within the same cell. Thus the chain $P\boxed{Q}\boxed{R}\overline{S}\ \overline{T}$ has two cells, one containing only the B-literal P and the other, the rightmost cell, containing the B-literals $\overline{S}$ and T. The chains $P\boxed{Q}\boxed{R}\overline{S}\ T$ and $P\boxed{Q}\boxed{R}T\ \overline{S}$ are equivalent.

Let φ be a function defined on non-empty chains, having chains as values. φ is a <u>selection function</u> iff $\varphi(C^*)$ 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 $P\boxed{Q|R}\ \overline{S}\ T$ then $\varphi(C^*)$ is $P\boxed{Q|R}T\ \overline{S}$ or C^* itself. The rightmost literal in $\varphi(C^*)$ is the <u>selected literal</u> in C^*. (The extension operation applied to C^* resolves $\varphi(C^*)$ on its rightmost B-literal with an input chain on its leftmost literal.) We require, further, that equivalent chains have the same selected literals. Thus if $\varphi(P\boxed{Q|R}\overline{S}\ T) = P\boxed{Q|R}T\ \overline{S}$ then $\varphi(P\boxed{Q|R}T\ \overline{S}) = P\boxed{Q|R}T\ \overline{S}$.

For a given set of clauses S, support set S' and selection function φ, an <u>SL-derivation</u> from S is a sequence $\underset{\sim}{D}^* = (C_1^*, \ldots, C_n^*)$ of chains satisfying (1) - (3).

(1) The initial chain C_1^* is an input chain from S'.

(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 (<u>admissibility</u> restriction).

C_{i+1}^* is obtained from C_i^* by <u>truncation</u> 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 <u>reduction</u> 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 $\overline{K}$, 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 $L\theta$ in C^*_{i+1} is the same as the status of the literal L from which it descends in C^*_i.

C^*_{i+1} is obtained from C^*_i by <u>extension</u> 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 $\overline{K}$ in B^* are unifiable with mgu θ.

(d) Let B^{**} be obtained by deleting the leftmost literal $\overline{K}$ from B^*. Then C^*_{i+1} is the chain $(\varphi(C^*_i)B^{**})\theta$ obtained by applying θ to the result of concatenating

$\varphi(C_i^*)$ and B** in that order. The literal $L\theta$ in C_{i+1}^*, descending from the rightmost literal in $\varphi(C_i^*)$ 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**.

* * *

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.

The restrictions (c) (i) and (c) (ii) on reduction are both concerned with restrictions on the factoring operation. If reduction were performed with a B-literal K in the rightmost cell, then the effect of this factoring operation would be to generate a chain already derivable by choosing a different factor for the input chain of the last extension operation. Similarly, if reduction were performed with the rightmost A-literal $\bar{K}$, then a variant chain could 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 (m-factoring) 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.

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.

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 m.g.u.'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-derivations from sets of ground clauses.

* * *

SL-resolution is more closely related to Loveland's model elimination system [16] 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 the power of SL-resolution is compared 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.

4.3 The Completeness and Power of SL-resolution

The following lemmas and theorem establish that the simplest SL-refutation of a set of clauses may be more complex than the simplest obtainable by t-linear resolution. However, the use of a selection function causes such an increase in the spareseness of the search spaces that SL-resolution appears to be considerably more efficient than t-linear resolution. Theorem 5 establishes that the complexity of a most complex minimal refutation is a bound on the complexity of a simplest SL-refutation.

Lemma 10. For every unsatisfiable set S of ground clauses, support set S_0 and selection furction 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 14 and appearing in full detail in [14]). 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 initial 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 initial 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^*.

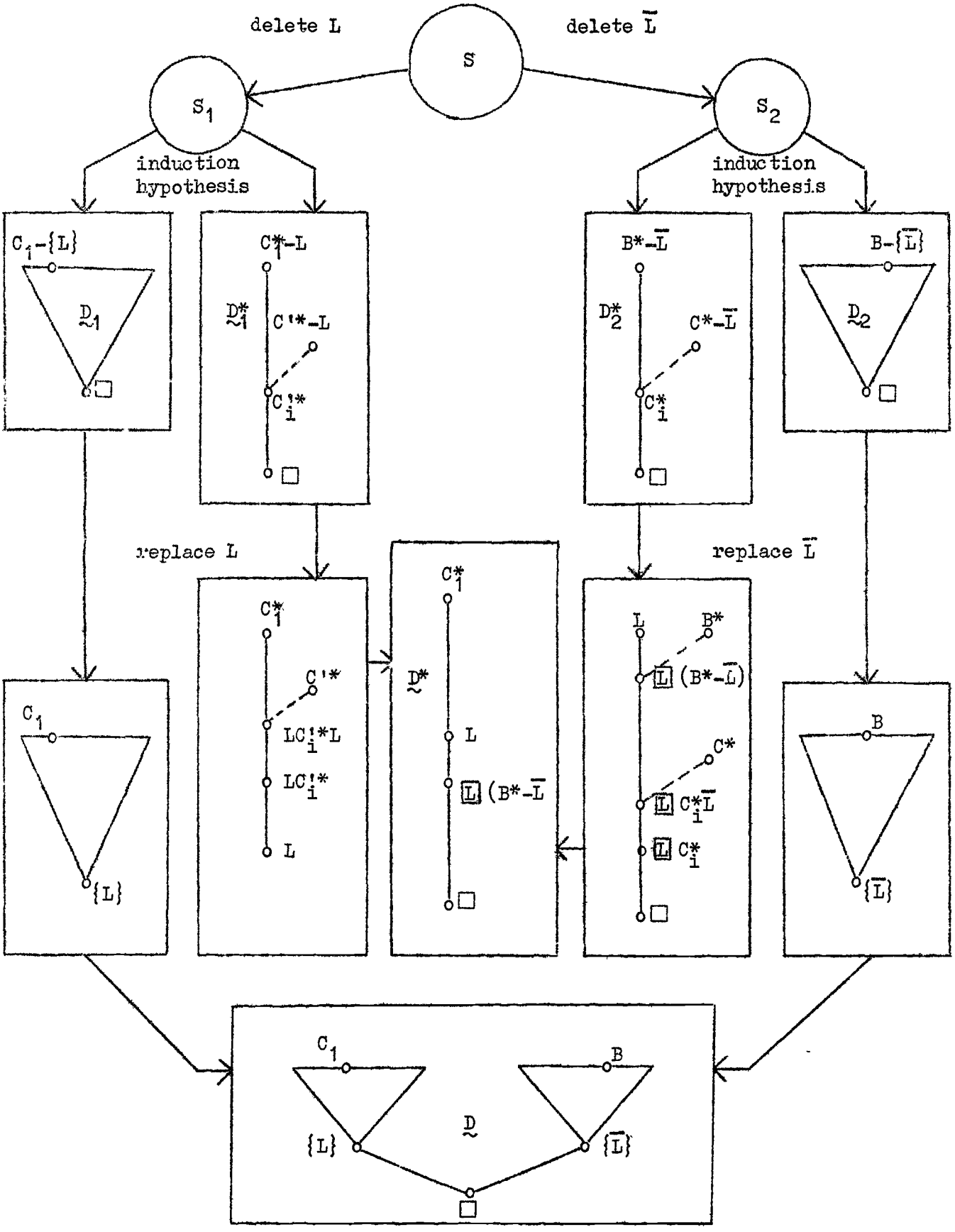


Figure 4 Outline of the proof of Lemma 6

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 therefore contains a minimally unsatisfiable subset S_1. Obtain the corresponding set of chains S_1^* from the set S* of input chains 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 $\bar{L}$.

The induction hypothesis is applied 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 $\varphi_1(C^{**}) = \varphi(C^*)$. If L occurs in C* only as the leftmost A-literal then $\varphi_2(C^{**}) = \varphi(C^*)$. The values of φ_1 and φ_2 for other chains may be defined arbitrarily.

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

The desired SL-refutation $\underset{\sim}{D}^*$ of S can now be obtained from $\underset{\sim}{D}_1^*$ and $\underset{\sim}{D}_2^*$ as in the similar construction of the s-linear refutation of Lemma 8. Introduce L as new B-literal in the leftmost cell of all chains in $\underset{\sim}{D}_1^*$. Introduce L as new initial chain and as a new A-literal to the left of all literals in chains of $\underset{\sim}{D}_2^*$ and insert $C_i^* \bar{L}$ immediately before any chain C_i^* obtained by extension in $\underset{\sim}{D}_2^*$ with a chain $C^* - \bar{L} \in S_2^*$ where $\bar{L} \in C^*$ and $C^* \in S^*$. $\underset{\sim}{D}^*$ 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 $\underset{\sim}{D}^*$ is an SL-refutation of S* for the selection function φ with initial chain C_1^*.

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

Lemma 11. For every unsatisfiable set S, support set S_0 and selection function φ, there exists a set S' of ground instances of clauses in S, a support subset S_0' of S' and a 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 (appearing in full detail in [14]). 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^* be any chain in S* corresponding to C_1 and $C_1'^*$ be the chain in S'* corresponding to C_1', where $C_1'^*$ is an instance of C_1^*. (The chain C'* is an instance of the chain C* iff C'* is an ordered instance of C* and the literals of C'* and C* in corresponding positions have the same status.)

We construct a tree T, each node of which is labelled both by a chain derived by an SL-derivation $\underset{\sim}{D}^*$ from S* for φ with initial chain C_1^* and by a chain derived by a ground SL-derivation $\underset{\sim}{D}'^*$ from S'* with initial chain $C_1'^*$. Both derivations have the same term-size and $\underset{\sim}{D}'^*$ derives an instance of the chain derived by $\underset{\sim}{D}^*$. The root of T is labelled by the chains C_1^* and $C_1'^*$. Suppose that a node N and the SL-derivation $\underset{\sim}{D}^* = (C_1^*, \ldots, C_n^*)$ and $\underset{\sim}{D}'^* = (C_1'^*, \ldots, C_n'^*)$ derive C_n^* and $C_n'^*$ at N have been constructed [illegible] fied to have the desired [illegible] erties. We need to spec[illegible] immediate descendant nodes and the SL-derivations of the clauses labelling them.

If $C_n'^*$ 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 $\underset{\sim}{D}^*$ and $\underset{\sim}{D}'^*$ 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 the chain which results from this reduction. A similar reduction operation can be performed on C_n^* and the result 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 the chain which results from extension. A similar extension operation can be performed on C_n^* with a chain B^* from S^*. The chain 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.

The tree 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 the same ground chain C'^*, but by distinct general chains. The selected literals in the general chains may 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 chains labelling its nodes, constitutes an SL-search tree T for some selection function φ', for the initial chain C'^* and for the input

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

Theorem 5. 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 11. By Lemma 10, there exists an SL-refutation $\underset{\sim}{D}'^*$ of S' for φ', with initial chain in $S_0'^*$ and $\underset{\sim}{D}'^*$ has the same rm-size as some minimal refutation $\underset{\sim}{D}'$ of S'. But, by Lemma 6, there is a minimal refutation $\underset{\sim}{D}$ of S which has the same rm-size as $\underset{\sim}{D}'$ and, by Lemma 11, there is an SL-refutation $\underset{\sim}{D}^*$ for φ with initial chain in S_0^* which has the same rm-size as $\underset{\sim}{D}'^*$. Therefore, the SL-refutation $\underset{\sim}{D}^*$ has the same rm-size as the minimal refutation $\underset{\sim}{D}$. Q.E.D.

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 $S = \{L\overline{M}, \overline{LP}, L\overline{Q}, \overline{LR}, NMQ, NPR, \overline{NT}, T\}$

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 tree 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.

Chapter 5. Selection Function and Support Set

5.1 Introduction

SL-resolution has two parameters which affect the structure of its search spaces. For each choice of selection function and each choice of support set, different search spaces are determined. Various anticipation methods allow some prediction of the structure of the search trees and thus aid the choice of a selection function and a support set.

The choice of a selected literal can be deferred until its chain is generated. Thus heuristic criteria can be employed to select literals from chains at the time of their generation. These heuristics determine the selection function dynamically. Similar heuristics can be used to determine the support set.

The search trees are assumed to be layered from the top down so that the search strategy generates all chains in higher layers before generating any on the next lower layer. Provided that the layering is exhaustive, the search trees for any unsatisfiable set must have a highest layer which contains a null chain. For a given input set, the best search space is one which has fewest chains on all layers above the highest one which contains a null chain. A powerful resolution rule has few layers above the layer containing the null chain. A search space is sparse if it has few chains on each layer. The choice of a selection function and a support set affect both the power of SL-resolution and the sparseness of its search spaces.

Estimating the number of layers above a highest null chain can be done by using the unlimited anticipation methods which will be discussed later. However, if this estimation is done for every choice of a selection function and support set, the difficulty becomes so great that it seems to outweigh the advantages. Granting this, the selection function and support set will be chosen in an attempt to increase the sparseness of the search spaces. Since the choice of support set and selection function does not affect the bound on the complexity of simplest SL-refutations, consistent increase in sparseness results in an overall reduction of the size of the subspace which needs to be generated before finding a first refutation.

5.2 The Uses of a Selection Function

It is assumed that the number of chains on each layer is an increasing function of the branching rates of the immediate ancestors of those chains. (The branching rate of a chain is the number of immediate descendants of that chain.) It follows that choosing a selection function so as to decrease the branching rate should increase the sparseness of the search spaces.

It is convenient to consider each literal of each factor of each input clause as an operator literal. Corresponding to each operator literal is an operator, which is the chain constructed from a factor of an input clause with the operator literal distinguished.

The result of using an operator is the resolvent obtained by using extension with the operator as the input chain, where the distinguished operator literal is the literal resolved upon. A search tree's structure is particularly simple when an operator is considered to be associated with the arc joining the near parent node with the resolvent node.

Let C be a chain and L a literal in C. Any literal $\overline{K}$ such that L and K are unifiable is a mate for L. The branching rate of L is the number of operator literals which are mates for L. (The estimation of branching rates is investigated in the next section.) To reduce the branching rate of a chain, the most recent literal with lowest branching rate should be the selected literal.

Let the most recent literals of C by L_1 and L_2 with branching rates n_1 and n_2, where n_1 is less than n_2. It is not difficult to

verify that if L_1 is selected rather than L_2, then in general there are at least $n_2 - n_1$ fewer chains generated as descendants of C. If a solution is found between the selection of L_1 and L_2, then the advantages of selecting L_1 are even greater.

The branching rate of a literal is defined without reference to the desirability of the chains of the operator literals. For example, operator literals from unit chains are more desirable than operator literals from longer chains. Later, anticipation procedures will be used to estimate the difficulty of 'getting rid of' the literals of an input chain other than the operator literal.

5.3 Estimating Branching Rates

Rather than actually estimating branching rates, it is simpler to estimate which literal of a chain has the lowest branching rate. This may be done by selecting the literal which is most instantiated and is thus likely to resolve with fewer operators. The more general literals which are not selected in the chain will tend to become more highly instantiated in the descendants of the chain, and will therefore tend to produce fewer immediate descendants when selected later. The amount of instantiation might be measured by the number of symbols in its argument places less the number of distinct variables there. Thus the simplest selection function which reduces the branching rate selects the most recent literal with the greatest measure of instantiation.

An upper limit for the branching rate of a literal is the number of operator literals whose predicate letters are appropriate for a mate. The literal $\overline{K}$ is a predicate mate for L if K and L have the same predicate letters.

A lower upper limit for the branching rate of L is the number of predicate mates for L which do not have any outer function clashes with L. Two terms s and t have an outer function clash if they begin with different function letters. (A constant is a function with no arguments.) If s and t do not have an outer function clash, they may still not be unifiable if there is an inner function clash or if they both contain the same variable at different function nestings. Two literals have an outer function clash if they contain terms in corresponding argument places which have an

outer function clash. If two literals are predicate mates and they do not have an outer function clash, then they are outer function mates. The outer function branching rate of a literal is the number of its outer function mates among operator literals.

To avoid repetious calculations of outer function clashes, the operators should be classified so as to make the calculations easy.

5.4 Operator Classification Trees

For any input set, the number of operators is fixed. Thus the work involved in their classification need only be done once. Besides simplifying the estimation of branching rates, the classification of operators according to outer function clashes avoids attempting unifications which are bound to fail. Although alternate and more sophisticated classification trees can be constructed, the following description indicates the method of construction.

Given any literal L, the operator classification tree for a given input set has one and only one branch corresponding to L, and at the tip of that branch is the set of all operators whose distinguished literals are outer function mates for L.

Let $P_1, \ldots, P_m$ and $f_1, \ldots, f_n$ be the predicate and function letters occurring in a set S of input clauses. The operator classification tree for S is the tree T(S) composed of all branches of the form $(r, Q, g_1, \ldots, g_k)$ where r is the root of T(S), $Q \in \{P_1, \overline{P}_1, \ldots, P_m, \overline{P}_m\}$, $g_i \in \{f_1, \ldots, f_n, v\}$ for $1 \leq i \leq k$ and k is the number of argument places of Q. From the root of T(S) one may visualise an arc for each of $P_1, \overline{P}_1, \ldots, P_m, \overline{P}_m$. Each node below the root is either a tip, or from it there is an arc for each of $f_1, \ldots, f_n, v$.

The literal L travels along the branch $(r, Q, g_1 \ldots, g_k)$ if L has predicate letter Q. If $g_i = v$ then the i-th argument of L is a variable. If $g_i \neq v$ then the i-th argument of L is a term beginning with g_i. At the tip of $(r, Q, g_1, \ldots, g_n)$ is the set of all operator literals which are outer function mates for L.

If K is an operator literal at the tip of $(r, Q, g_1, \ldots, g_n)$ then the predicate letter of K is $\overline{Q}$. If $g_i = v$ then the i-th argument of K may be any term. If $g_i \neq v$ then the i-th argument of K is a variable or is a term beginning with g_i.

It may be that several tips of an operator classification tree have the same set of operators. In this case, the tree may be condensed. This is done by labelling an arc with several functions.

Consider the following list of input clauses

$P(x,e,x)$

$P(x,g(x),e)$

$\overline{P}(g(a),a,e)$

$\overline{P}(x,y,u)\ \overline{P}(y,z,v)\ \overline{P}(x,v,w)\ \ P(u,z,w)$

$\overline{P}(x,y,u)\ \overline{P}(y,z,w)\ \overline{P}(u,z,w)\ \ P(x,w,w).$

Figure 15 is a condensed operator classification tree for these eleven literals. The literals are represented by an ordered pair of integers, the first integer being clause number and the second being literal number in the order they are written above. If this operator classification tree were not condensed, it would have 128 tips instead of 5.

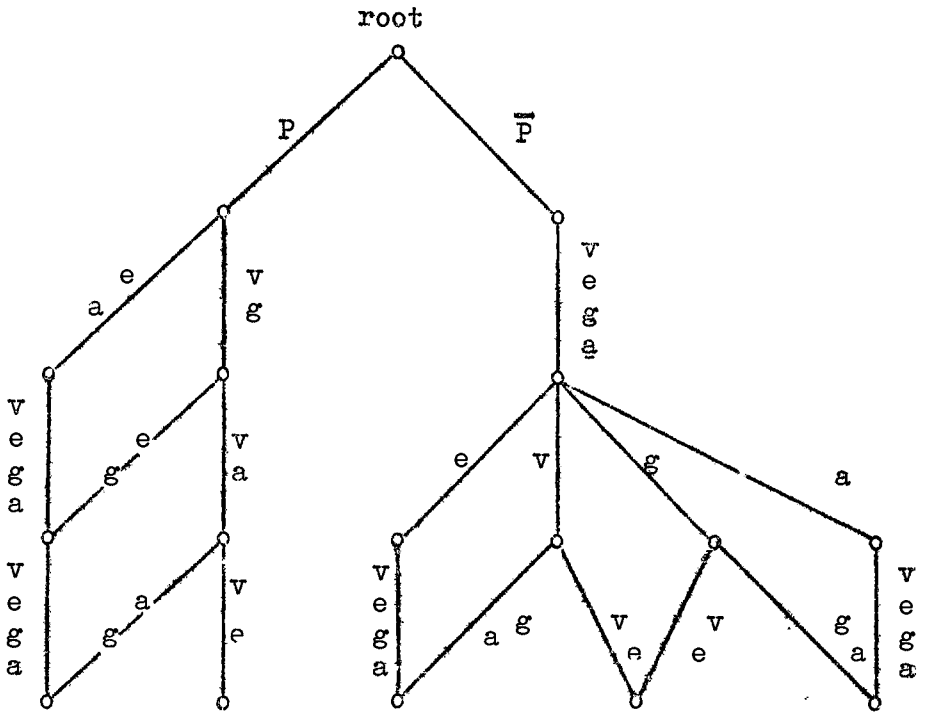


Figure 15.

A more sophisticated operator classification tree than the one described would classify operators by more than the outer function of each term. The length and intricacy of each operator could also be indicated. The following sections indicate various anticipation techniques whose implementation would depend on a sophisticated operator classification tree.

5.5 Functional Agreement for Tie-Breaking

In many cases, there are several most recent literals in a clause which have the same outer function branching rate. The selection function must have some sort of tie-breaking rule. One such rule chooses the literal which has the greatest functional agreement with operator literals.

If two terms are both variables or they begin with the same function letter, then they have functional agreement. The number of functional agreements of two literals which are outer function mates is the number of terms in corresponding argument places which have functional agreement.

A functional agreement tie-breaking rule is the type of heuristic used by mathematicians. To choose to unify literals because of their similarity has a strong appeal for a mathematician. It should also be noted that when functional agreement is high, then the unifying substitution should be simple. The advantages of this are discussed in the next chapter.

The implementation of functional agreement for tie-breaking could be achieved by modifying the operator classification tree. Consider the example of figure 16.

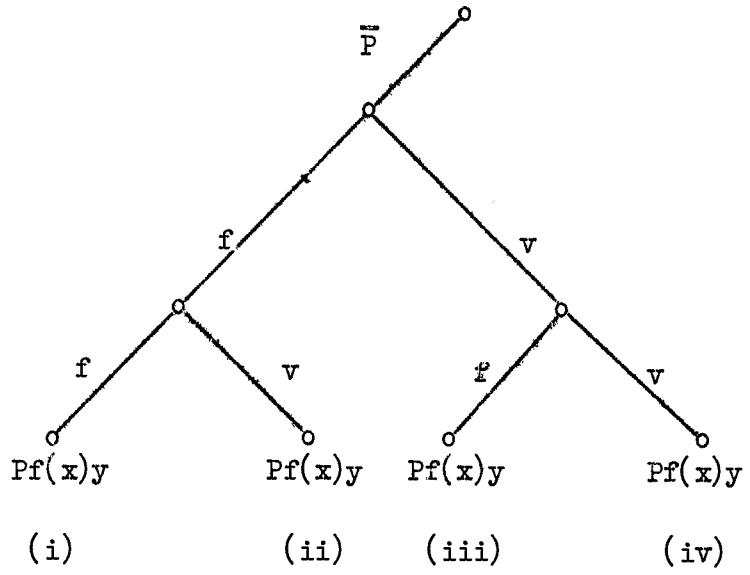


Figure 16.

If a selected literal is of the form $\overline{P}(f(t),z)$, for any term t , then it would follow the tree down to the tip marked (ii). In this case, there are two functional agreements. Any selected literal which follows the tree down to tips (i) or (iv) has one functional agreement. Following the tree down to (iii) indicates no outer function clash, but also no functional agreement.

5.6 A Two Stage Anticipation Procedure

Using an operator classification tree allows the estimation of the number of immediate descendants of a chain. This anticipation can be extended a stage further by considering the branching rates of the immediate descendants of a chain.

Let L and $\overline{K}$ be outer function mates where $\overline{K}$ is the distinguished literal in the operator C. Let the residue of $\overline{K}$ be C', obtained by deleting $\overline{K}$ from C.

The outer function branching rate for each literal in C' can be calculated using the operator classification tree. Let the minimal outer function branching rate of the literals of C' be the residual outer function branching rate of $\overline{K}$. Let the second stage outer function branching rate of L be the sum of the residual outer function branching rates of the outer function mates of L.

Consider the following minimally unsatisfiable set of clauses:

$$P(a,x)\ P(x,f(x)) = LM$$

$$P(a,x)\ P(f(x),x) = NP$$

$$\overline{P}(a,x)\ \overline{P}(x,a) = QR$$

$$\overline{P}(a,a) = S$$

Literal	Literal branching rate	Residue	Residual Rate
L	3	M	1
M	1	L	3
N	3	P	1
P	1	N	3
Q	3	R	3
R	3	Q	3
S	2	none	-

If a selected literal were to resolve with L, then the residue would be the literal M which has a branching rate of 1. Thus the residual outer function branching rate of L is 1. The literal R has outer function mates, L. N and P so that the first stage outer function branch rate of R is 3. But L, N and P have residual outer function branching rates of 1, 1 and 3. Thus the second stage outer function branching rate for R is 1 + 1 + 3 = 5. That is, using outer function branching rates, it is estimated that in selecting R, there will be 3 immediate descendants and 5 second level descendants. In this case, the difficulty of generating all descendants of R two levels below R is 8. Figure 17 illustrates the anticipated descendants of R to two levels below R.

By using a two-stage anticipation method, the first and second level branching rate can be estimated. The selected literal in a chain should be the most recent literal whose estimated first and second level branching rate is smallest.

Second stage anticipation also indicates which operators contribute to the least branching. This can be used by the search strategy.

Clearly, the anticipation method can be extended to estimating third level branching rates. There seems no limit to the number of levels investigated, but the labour involved and the progressive inaccuracy of the results will curb too deep an estimate.

There remains some difficulty when considering the second level branching rate when a chain C resolves with a unit operator. The branching of another literal in C determines the second level branching rate. Although this determination is possible, the amount and differentness of the calculation makes it attractive to define the residual branching rate of a unit operator as zero. From this point of view, the first and second level branching rates are an estimate of the two level difficulty in 'getting rid of' the selected literal. This in turn suggests an extended anticipation method which calculates an upper bound on the difficulty of getting rid of a literal.

5.7 Unlimited Anticipation and Pseudo-Search Trees

The preceding anticipation methods can be extended without limit, but then it becomes difficult to keep track of the structure of the anticipated chains. To overcome this difficulty, it is convenient to construct pseudo-resolvents corresponding to the anticipated chains.

Assume that the chains A and B have an SL-resolvent $(A' \boxed{L} B')\theta$, where L is the literal resolved upon and where A' and B' are the residues of A and B. Then the pseudo-resolvent of A and B is $A' \boxed{L} B'$ where the unifying substitution θ is not applied. Pseudo-factoring, and pseudo-ancestor resolution can be defined in a similar way.

For the example of the preceding section, figure 17 illustrates a pair of 2-stage anticipation trees using pseudo-resolution.

A literal may be selected based on the results of constructing an extended anticipating tree which uses all possible selection functions. If the tree is extended far enough a selection function can be chosen so that the search tree it defines is anticipated to have the fewest chains above the first null chain. In practice, the amount of extension of the anticipation tree depends on the size of the tree. Instead of limiting the number of stages of anticipation as was considered in the preceding section, it is possible to have some limit on the size of the anticipation tree. Then selecting a literal would have some constant difficulty.

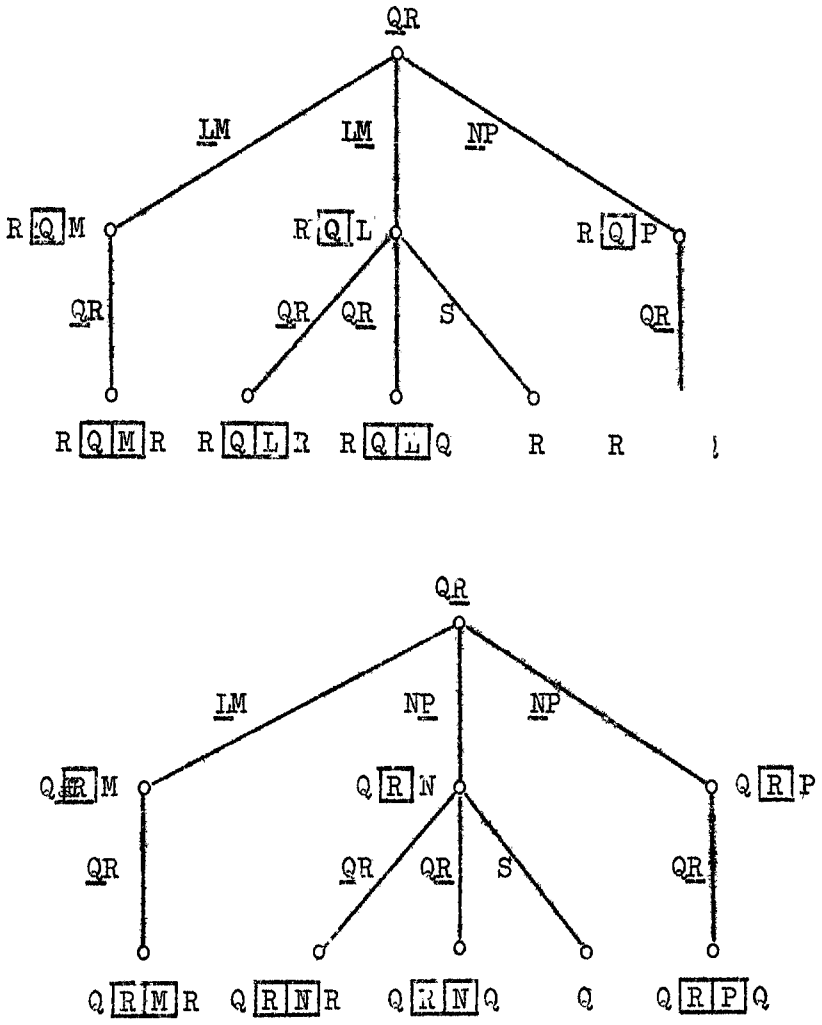


Figure 17.

The possibility of extending an anticipation tree until a null chain is found suggests a new and very attractive method for conducting a search. If only one selection function is used, then the size of the anticipation tree is considerably reduced, so that extending it to a null chain seems more feasible.

For any selection function, a search tree employing pseudo-resolution, factoring and ancestor resolution is a pseudo-search tree. By applying the appropriate unifying substitutions, and deleting branches where unification fails, it is clear that a real

search tree may be obtained from the corresponding pseudo-search tree. Thus any real search tree is an instance of a subtree of the corresponding pseudo-search tree. For ground chains, a pseudo-search tree is a real search tree.

Since SL-resolution is complete, at least one pseudo-refutation of a pseudo-search tree corresponds to a real refutation. Thus the pseudo-search tree indicates which branches of a real search tree are most likely to lead to real refutations. The use of a pseudo-search tree is not itself a search strategy. Because the branching rate of a pseudo-search tree is usually greater than that for a real search tree, a search strategy may be more important in generating a pseudo-search than in generating a real search. However, each pseudo-resolvent is much easier to generate than a real resolvent.

Consider the following example:

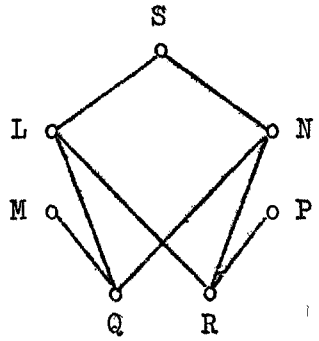
Represent $P(a,x)\ P(x,f(x))$ by LM,

$P(a,x)\ P(f(x),x)$ by NP,

$\overline{P}(a,x)\ \overline{P}(x,a)$ by QR, and

$\overline{P}(a,a)$ by S.

In the following diagram, mate literals are connected by lines.



Using only the relationships represented by this diagram, the pseudo-search of figure 18 can be constructed. In figure 18, {QR,S} is chosen as support set. The parenthesised numbers indicate the order in which chains are generated using upper diagonal search. (A full description of search strategies appears in the next chapter.) The operator appears beside the lines, literals to be resolved upon are underlined, and A-literals appear in boxes.

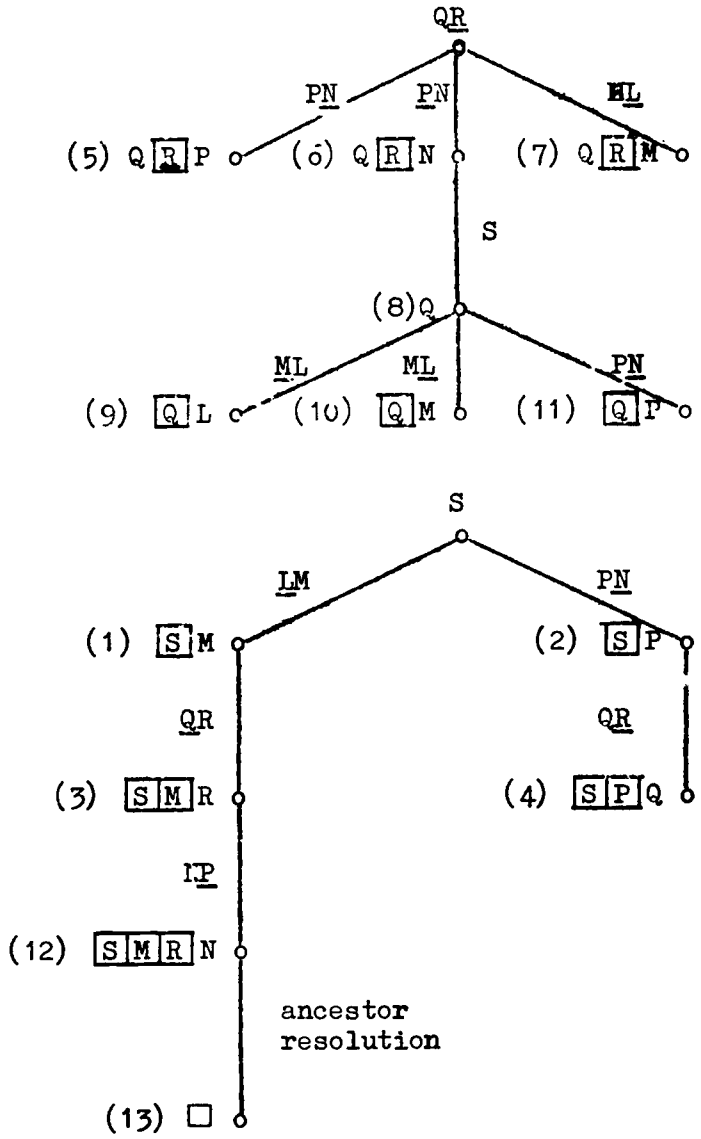


Figure 18.

The pseudo-refutation of the pseudo-search of figure 18 has the refutation of figure 19 as an instance.

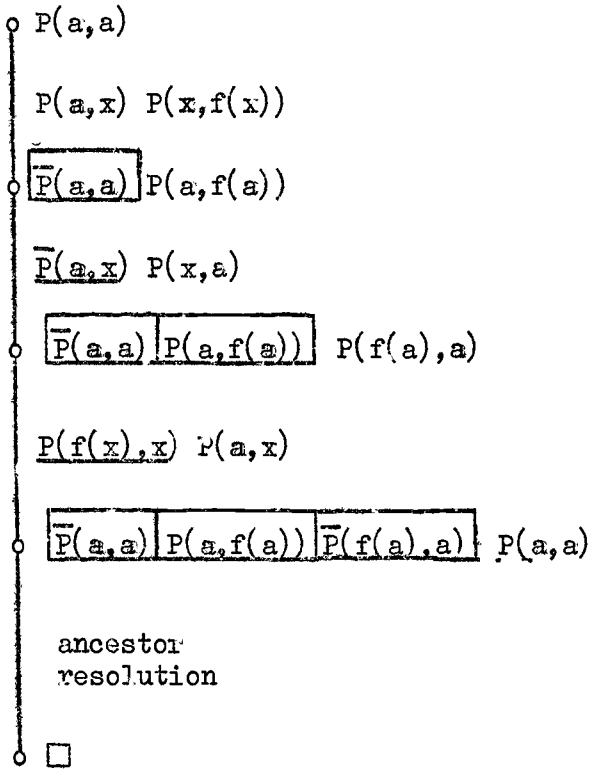


Figure 19.

In more complicated examples, there are pseudo-factorings and pseudo-admissibility clashes (where the admissibility restriction fails to hold). It has been observed in several examples that almost all pseudo admissibility clashes correspond to real admissibility clashes or to failure of unification. It has also been found that almost every branch leading to a null clause in a pseudo-search corresponds to a branch leading to a null clause in a real search. Thus, there are indications that pseudo-search gives fairly accurate information about the corresponding real search.

It should not be difficult to implement a pseudo-search using far less time and space than a real search. If the conjecture

about the accuracy of the information of a pseudo-search is correct, then finding a refutation by using a pseudo-search might be the most efficient method of searching for a refutation.

If the branch corresponding to a pseudo-refutation fails to be a refutation, then the literals on this branch could be treated as the literals in the input set were treated. That is, each is given a name, and the literals which are mates are noted. In this way, a pseudo-search can be partly replaced by a real search. The pseudo-search can be considered to be a look-ahead for the real search. In easy problems, it is possible to look ahead to possible solutions. In more difficult problems, where the pseudo-search becomes very large, the look-ahead might be restricted to some fixed number of levels, or to some fixed degree of difficulty. When the look-ahead is limited to one level, then the pseudo-search is reduced to an estimate of the branching mate.

An alternate unlimited anticipation method has been suggested by Bob Kowalski. The pseudo-search could be limited to single literals of the input set. One advantage of this is that several simpler pseudo-searches are done. Another advantage is that when trying to prove several theorems from the same axiom set, there should be some transferal from one problem to the next.

5.8 The Choice of a Support Set

Let S be any unsatisfiable set of clauses, and let S* be the set of chains corresponding to all factors of clauses of S. Among all the SL-search trees with top chains in S*, at least one and usually several contain a refutation of S. It is clearly advantageous to consider as few of these search trees as possible provided that at least one of those considered contains a refutation. Any subset S'* of S* such that S*-S'* is satisfiable is a support set for S*. At least one search tree with top chain in S'* contains a refutation.

One method for choosing S'* depends only on the sign of the literals in the chains of S. From the completeness of hyper-resolution, it follows that the set of all negative chains in S* is a support set for S*. (A chain is negative if all of its literals are negative.) By renaming every literal in S*, so that every sign is changed, it is evident that the set of all positive chains in S* is also a support set for S*. In fact, for any renaming, the chains of S* which are renamed to become negative are a support set for S*. The ground unsatisfiable set $\{PQ, \overline{P}, \overline{Q}R, \overline{R}S, \overline{RST}, P\overline{T}\}$ has 16 support sets identified by noting which subsets can be renamed to contain negative chains.

A second method of choosing a support set depends upon knowledge of the problem. Many problems are expressed as a number of axioms and a theorem which follows from the axioms. Assuming the axioms and conditions of the theorem to be consistent, the negation of the conclusion of the theorem forms a set of clauses which is a support set for the clauses

of the problem. The support set of chains which can be used as top chains is the set of chains corresponding to factors of the clauses in the support set of clauses. From a mathematician's point of view, this choice of support set is particularly attractive. This is because he can check whether resolvents are false in his model for the axioms. If such checking could be mechanised, then many irrelevant derivations could be deleted from the search.

It is evident that any unsatisfiable set of clauses contains several support sets. The support set should be chosen in an effort to reduce the difficulty of the search. The simplest criterion for the choice of a support set is that it be the one with the fewest chains. In this way, the corresponding search space has fewest search trees. This is a reasonable choice if it is assumed that there is no difference between search trees or that the difference cannot be determined by inspecting the root chains.

The number of chains in a support set can be thought of as its zero level branching rate. The anticipation methods of the preceding sections can be used to estimate the higher level branching rates of a support set. For each chain in a support set, the literal with lowest outer function branching rate is selected. The sum of these branching rates for all chains in the support set is the branching rate of the set. The sparsest search tree should be obtained by choosing the support set with the lowest branching rate. Obviously, this method can be extended to estimate second and higher level branching rates. It should be noted that the chains corresponding to the negation of the conclusion of the theorem tend to be more highly instantiated and should thus have a lower branching rate.

Although choosing the support set with the fewest literals has some intuitive appeal (there are fewer literals to get rid of), it seems to have no theoretical justification, and examples indicate that it has no advantages.

Chapter 6. A Search Strategy for SL-Resolution

6.1 Introduction

All of the preceding development has defined an inference system. For any set of chains, an SL-inference system specifies a search space composed of one or more search trees. It is now necessary to investigate a search strategy which specifies an order in which the chains of the search trees are generated.

There are certain properties of a resolvent which can be precisely predicted from the properties of its parents. The most important of these are the complexity of its derivation, and the number of B-literals it has. Such precise predictions allow a discussion of search strategies which consider properties of chains which have not yet been generated. These candidates for generation can also be distinguished by properties which belong only to their parents. A search strategy compares properties and then selects one chain to be generated from all candidates for generation.

Although the following is conceived of as being used with normal SL-resolution, the search strategies also apply to pseudo-resolution and pseudo-searches.

6.2 An Expediency Tactic

In any search for a refutation, the null chain is the best chain to generate. It should be generated whenever possible and however possible, even if this overrides other tactics of the search strategy.

When a unit chain (with one B-literal) is generated, there should be a search for any other unit chain which will resolve with it. To avoid too much search in the implementation of this tactic, special storage is needed for unit chains. These unit chains need not be on the same branch of a search tree nor even on the same search tree. Thus the resulting refutation may not be SL, nor least complex, but it should have the easiest search.

Although this tactic seems bound to increase efficiency, it is very difficult to find an extension of it. One possibility would be to extend the null chain preference to a unit chain preference. Another is to have unit chains act as operators. However, both of these tactics are likely to cause inefficiency.

6.3 Complexity Saturation and Diagonal Search

As was mentioned in Chapter 1, only exhaustive search strategies are to be considered. Any exhaustive search strategy enumerates the chains of the search space. This enumeration can be partially specified by layering the search space. All chains on the same layer have the same merit. A null chain with a simple derivation is to have a better merit than a null chain with a more complex derivation.

A merit saturation search strategy for SL-resolution first generates all chains of merit zero. After generating all chains with merit k, the search next generates all chains with merit k+1. Such a search will terminate when it finds the refutation of smallest merit obtainable by SL-resolution. If the merit of a derivation is its complexity, then the corresponding saturation search is a complexity saturation search.

The efficiency of a complexity saturation search is much improved by using Kowalski's method of diagonal search [11]. A search tree is layered by complexity of derivation in order that simple refutations may be found before more complex ones. However, all refutations of a given complexity can be generated without generating all derivations of that complexity. For a given complexity, diagonal search generates only those derivations which could be extended to a refutation of that complexity.

Let the cost of a chain C be the complexity of the derivation of C. Let the expectation of C be any lower bound on the cost of deriving a null chain from C. (Kowalski calls the expectation of

C the heuristic value of C.) For instance, if the cost of C is the r+m-size of the derivation of C, then the expectation of C could be the number of B-literals in C, the length of C. (The r+m-size of a derivation of rm-size (r,m) is r+m.) This expectation is clearly a lower bound on the cost of deriving a null chain from C.

Consider all derived chains to be stored in a rectangular array, with all chains of cost g in the g-th row, and all chains with expectation h in the h-th column. Let the n-th diagonal of this array be the set of all cells of the array the sum of whose row number and column number is n. Diagonal search first generates all chains in diagonal 0. If all chains in diagonal k have been generated, then the search next generates all chains in diagonal k+1. Thus diagonal search saturates the diagonals of the storage array.

Since h is a lower bound on the cost of deriving a null chain from C, it follows that g+h is the complexity of the simplest refutation obtainable by extending the derivation of C. Therefore, to find a refutation of complexity k, it is sufficient to generate all chains on diagonals less than or equal to k. Complexity saturation search to complexity k generates all chains generated by diagonal search to diagonal k, but complexity saturation also generates all those chains of cost less than k which are on diagonals greater than k. Thus diagonal search is always more efficient than complexity saturation search.

Diagonal search is a merit saturation search if merit is defined as g+h where g is the cost and h is the expectation.

For the case where complexity is defined as ar+bm for coefficients a and b, the merit should be defined as ar+bm+bh or possibly ar+bm+ah$_1$+bh$_2$ where h$_2$ is the number of B-literals in the rightmost cell, and h$_1$ is the number of B-literals not in the rightmost cell.

In any search, it is often the case that several chains have equal merit. It is then necessary to have a tie-breaking rule since chains are generated one at a time. A good tie-breaking rule for diagonal search is Kowalski's upper diagonal search [11]. If there are several candidates for generation on the same diagonal, then upper diagonal search generates a chain with the smallest expectation. Such a preference for small expectation should decrease the difficulty of the search.

For instance, let the cost of a chain be the r+m-size of its derivation, and let its expectation be the length of the chain. Let C be the chain most recently generated by upper diagonal search, and let C have merit g+h. Then, immediately before the generation of C, all other candidates for generation must have had merit greater than g+h, or if they had merit g+h, then their length must have been greater than or equal to h. If C can be acted upon by the reduction rule or by the extension rule with a unit operator, then the immediate descendant C' has cost g+1 and expectation h-1. This resolvent is on the same diagonal as C, but is shorter than C. It follows that C' will be generated next after C. Thus, with these measures of cost and expectation, when a chain is generated, all of its descendants on the same diagonal are generated before any other chains are generated.

Upper diagonal search is a merit saturation search with merit $h + \frac{1}{2}(g+h)(g+h+1)$. The second term of this merit is the sum of the number of cells in diagonals shorter than the one being searched. This merit ordering can be considered to be a refinement of the diagonal search merit ordering. Each of the diagonal search merit levels is subdivided to provide upper diagonal search merit levels. Most tie-breaking rules can be considered as refinements of the merit ordering for which they break ties. The following table compares the number of derivations generated by complexity saturation and diagonal search for the example and refinements in the table of section 3.3. For both strategies, the complexity is r+m.

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

6.4 Measures of Complexity

With a computer, the difficulty of a search for a refutation is some function of elapsed time and storage facilities used. In order to do theoretical calculations, some simpler estimate of difficulty must be used. The simplest estimate is the size of the search, (the number of chains generated before an empty chain is generated). This estimate is quite inaccurate if some searches attempt many unifications which fail. The use of an operator classification tree should substantially reduce the number of unifications which fail.

The aim of a search strategy is to find a refutation with as little difficulty as possible. An indirect way of attempting to do this is to search for refutations which are least difficult to construct. In constructing a derivation, no chains are generated except those in the derivation. The complexity of a derivation should be a measure of the difficulty of constructing the derivation, and not a measure of elegance or conceptual clarity.

Corresponding to the simplest measure of the difficulty of a search, the simplest measure of the complexity of a derivation is its size. The simplest measure of size would count only the number of extension operations in a derivation. Since both extension and reduction operations make substantial contributions to the difficulty of a search, both should be used when measuring the complexity of a derivation. A size of r+m is the simplest to calculate, but it is somewhat inaccurate since the generation of a chain by extension is more difficult than the generation of a chain by reduction. This

suggests the use of ar+bm where $a \geq b$, $a > o$ and $b \geq o$. Since one extension gets rid of two literals, while reduction gets rid of only one, 2r+m might be a good measure of complexity. It should be noted that a refutation has 2r+m input literals. We have considered values for a and b of (1,0), (1,1) and (2,1). There seems to be little theoretical justification for using any other values.

* * *

A literal such as P(f(x,g(y)), h(z)) is structurally more intricate than P(u,v), and is more difficult to process. A derivation containing more intricate literals is more difficult to construct than an isomorphic derivation whose literals are less intricate. Thus the intricacy of the literals of a derivation contribute to its complexity.

Since a literal with distinct variables in each argument place is in its least intricate form, such a literal should be assigned an intricacy number of zero. The intricacy number of a literal is the number of symbols in its argument places less the number of distinct variables. (This measure is used by Reynolds in [27].) Using this measure, P(x,x), P(x,a) and P(x,f(y)) each have an intricacy number of one, while P(f(x,g(y)), h(z)) has intricacy number three. Measuring intricacy by counting the total number of symbols in a literal is easier, but seems too naive in that it discriminates against literals with many argument places.

If k is the sum of the intricacy numbers of all of the B-literals in a derivation, then ar+bm+ck could be used to measure

the complexity of the derivation. Since k would usually be a larger number than r or m, and because k makes a smaller contribution to the difficulty of a derivation, c should be considerably smaller thar a or b.

A fourth possible contribution to the difficulty of constructing a derivation is the number and intricacy of the substitution components in the unification substitutions. This contribution to the difficulty is most evident when a ground derivation is compared with a general level derivation.

If a substitution component replaces a variable x with a term t and t is composed of n symbols, then define the <u>size</u> of the substitution component to be n-1. The <u>size</u> of a substitution is the sum of the sizes of its components. Let s be the sum of the sizes of the unification substitutions used in constructing a derivation. Then ar+bm+ck+ds could be a measure of the complexity of the derivation. As with c, the coefficient d should be considerably smaller than a or b.

6.5 Heuristic Merit Functions

In a diagonal search, g has been considered to be some measure of the complexity of the derivation of the chain and h to be some measure of the complexity of getting a refutation. However, in the search, the merit is only used to choose which chain to generate next. There are heuristic criteria which could help in the choice of a chain but which are not direct estimates of complexity or expected difficulty.

Alternatively, the cost and expectation of a chain can be considered from a more pragmatic point of view. When a search is only partly completed, the amount of additional search is of more interest than the amount of completed search. For example, if a chain C has a small expectation, and a chain C' has a large expectation, then there should be less search if the descendants of C are generated before the descendants of C'. That is, when a chain has been generated, its cost can be considered to be irrelevant since the work has been done and cannot be undone. One way of implementing this concept would be to use a length saturation strategy. Unfortunately, such a seemingly sensible search strategy is incomplete. Although $\{P(a), \overline{P}(x)\ P(f(x)), \overline{P}(f(f(a)))\ \overline{P}(f(a))\ \overline{P}(a)\}$ is unsatisfiable, the strategy of ignoring the cost of chains can lead along an infinite branch of the search tree. However, the attraction of trying for a simplest search does suggest that some multiple of the expectation should be used so that the cost of a chain contributes less to its merit. For a chain with cost g and expectation h, the merit should be $ag + bh$ where $b > a > o$.

Rather than use the sum of the intricacy numbers of all B-literals in a derivation, it is possible to use only the local intricacy, the intricacy numbers of the B-literals in the near parent of a candidate for generation. For some coefficients a, b and c redefine the merit of a chain as ag+bh+ct where g and h are the cost and expectation of a chain to be generated and t is the sum of the intricacy numbers of the B-literals of its near parent. Using such a merit with a merit saturation search, the generation of chains with intricate literals is delayed. As has been mentioned, this delays the difficulty of manipulating such literals. This delay in generation is preferable to an intricacy upper bound or a function nesting bound.

Rather than use the sum of the sizes of the unification substitutions in a derivation as a measure of its complexity, it is possible to estimate the size of the next substitution and use this as a component of the merit of a candidate for generation. Let f be the number of functional agreements (section 5.5) between the selected literal of the near parent and an operator literal. If the number of functional agreements is large, then the size of the unifying substitution should be small. The merit would then be some ag+bh+ct+df. Because high functional agreement should contribute to a low merit, the coefficient d should be negative. This may result in a negative merit, but this should cause no difficulty.

There is a merit component which experiments have shown to be more useful than g, h or t. This component uses the outer function branching rate of section 5.3. The branching number of a chain is

the sum of the outer function branching rates of the unselected B-literals of its near parent.

A chain with a large branching number is likely to resolve with more operations and so one of its descendants is more likely to be the null chain. On the other hand, many descendants mean more search for a null chain. One method of resolving this dilemma is to test the usefulness of the branching number experimentally.

Let the branching number of a chain be n. Then the merit could be ag+bh+ct+dn. If a large n is helpful to the search then d should be negative. If a small n is most helpful, then d should be positive. For six examples, the searches were least difficult when the merit was g+2h+t-6n, although two examples had smaller searches with g+h+t-5n. For three of these examples a search with heuristic merit of g+2h+t-6n was compared with a merit of g+h. The difficulties of the searches are in the following table:

Problem	g+h	g+2h+t-6n	refutation size
A	26	15	7
B	28	6	4
C	50+	38	7

These results indicate that generating a chain which is likely to resolve with many operators tends to contribute to a less difficult search. Not only is a large branching number useful, but it appears to be more useful than cost, expectation or intricacy.

Some of the components of a many-component merit could be used for tie-breaking as h is used for tie-breaking in diagonal

search. Unfortunately, many of Kowalski's results [12] cannot be used because f, t and n are not properly monotonic. Since all three are zero for the null chain, any or all may be included with h to form the tie-breaking component of an upper diagonal search. Because n seems to have a greater beneficial effect on the search, it might be best to use n and h as the tie-breaking components. However, because there are contributions from several sources, the ag+bh+ct+dn merit numbers are larger so that a great spread of merit numbers usually occurs. This means that tie-breaking values are needed less frequently than with ag+bh.

* * *

The intricacy, functional agreement and branching number of a chain can be treated purely as heuristic values in that they indicate which chain is most likely to lead to a solution. With this treatment, the cost, g, and expectation, h, of a chain are used to calculate its merit, while the intricacy, functional agreement and branching number are used as tie-breaking rules among chains of equal merit. Tie-breaking rules can be considered to define new merit levels within the larger merit levels. But because these heuristic values do not have natural upper bounds, it is difficult to define refinements of a merit using these values.

Since the branching number, n, has proved most useful in experiments, this might be the first tie-breaking rule among chains of equal upper diagonal merit. If there are several of these chains which have the same high branching number, then choose from them the ones with highest function matching number. If there are

still several tied chains, then choose those which are least intricate. Finally, some arbitrary choice must be made if there are still tied chains.

A final possibility for a tie-breaking rule has been suggested by Loveland for model elimination. Of two chains with the same merit, this rule selects the one with greatest number of A-literals. The chain containing more A-literals offers more possibilities for eliminating B-literals by reduction and therefore for eliminating B-literals in the course of generating an empty chain.

6.6 Anticipation Strategies

Although their significance has not yet been thoroughly investigated, the anticipation techniques of the preceding chapter seem to be most promising as tools for efficient search procedures. Following a suggestion of Kowalski, pseudo-search techniques can be used to assign numerical values to chains.

Let L be a literal in a factor of an input chain. Consider the pseudo-search with top chain the unit chain L. Let the anticipated difficulty of L be g, the size of the smallest pseudo-refutation of this search. In order to avoid excessively difficult pseudo-searches, there might be an upper bound on g. Thus, when pseudo-diagonal search exhausts the diagonal corresponding to this upper bound, without finding a refutation, then g is arbitrarily assigned the merit value of the next longer diagonal. The anticipated difficulty of a chain is the sum of the anticipated difficulties of its literals. The anticipated difficulty of a literal in a resolvent is the anticipated difficulty of the literal from which it descends in the near or input parent. This value could be used as the expectation of a chain. The anticipated difficulty of a chain is a lower bound on the difficulty of constructing a derivation of a null chain from it (by extension).

In the case where the anticipated difficulty is the upper bound on g, then the size of the pseudo-search could be useful. This anticipated size could be used either to indicate the likelihood of a good path when the size is large or to indicate the narrowness of the real search when the size is small.

More detailed research should reveal highly sophisticated anticipation methods which will vastly improve the efficiency of search strategies. A more sophisticated use of anticipation would employ pseudo-search to obtain pseudo- solutions of the goals and subgoals of section 6.8.

6.7 A Deletion Strategy

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. A chain C* subsumes another C'* if some instance C*σ 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 be any search strategy for SL-resolution. can be modified to obtain a new strategy Σ' which step by step generates the same chains as Σ, in the same order, but deletes subsumed chains and does not generate chains which are descendants of previously deleted chains.

(1) Both search strategies generate the same first chain.

(2) If Σ generates a chain, then Σ' generates the same chain provided that its near parent has been generated by Σ' and has not been previously deleted.

(3) If Σ' generates a chain C* then

  (a) C* is deleted if it is subsumed by some previously generated and undeleted

chain,

(b) otherwise every previously generated and undeleted 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, there has been no success in searching for 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 Kowalski's thesis [12].

6.8 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 $\mathcal{F}(C^*) = C_0^*L$ 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 $(C_0^*I \rightarrow C_0^*)$ 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

$$(N\boxed{P}QR,\ N\boxed{P}Q\boxed{R}S,\ N\boxed{P}Q\boxed{RS}\overline{R},\ N\boxed{P}Q)$$

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 $\overline{R}$ which can be used as input chain for extension. If the solution to $(N\boxed{P}QR \rightarrow N\boxed{P}Q)$ were

$$(N\boxed{P}QR,\ N\boxed{P}Q\boxed{R}S,\ N\boxed{P}Q\boxed{RS}\overline{P},\ N\boxed{P}Q),$$

then the corresponding lemma would be $\overline{RP}$ 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 $\overline{P}$ as B-literal.

The preceding examples of lemma construction are easy to generalise (see, for instance, Loveland [16]). 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.

Kowalski in [14] and [15] has extended the use of subgoals to the and/or tree representation of a search space for SL-resolution.

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. Chang, C.L., The unit proof and the input proof in theorem proving, Journal of the Association for Computing Machinery, 17, (1970), 698-708.

4. Cohn, P.M., Universal Algebra, Harper and Row, (1965).

5. Davis, M., Eliminating the irrelevant from mechanical proofs. Proceedings of Symposia in Applied Mathematics, 15, (1963), American Mathematical Society.

6. Hayes, P.J., and Kowalski, R.A., Semantic trees in automatic theorem proving. Machine Intelligence 4, Edinburgh University Press (1969), 87-101.

7. Hayes, P.J., and Kowalski, R.A., Lecture notes on automatic theorem-proving. Metamathematics Unit Memo 40, University of Edinburgh, (March, 1971).

8. Herbrand, J., Recherches sur la theorie de la demonstration, Travaux de la Societe des Sciences et des Lettres de Varsovie, III, Vol. 33, (1930), 33-160.

9. Horn, A., On sentences which are true of direct unions of algebras, Journal of Symbolic Logic, 16, (1951), 14-21.

10. Kieburtz, R., and Luckham, D., Compatibility of Refinements of the Resolution Principle, (1969).

11. Kowalski, R.A., Search strategies for theorem-proving. Machine Intelligence 5, Edinburgh University Press (1970), 181-201.

12. Kowalski, R.A., Studies in the completeness and efficiency of theorem-proving by resolution. Ph.D. Thesis, University of Edinburgh, 1970).

13. Kuehner, D.G., A note on the relation between resolution and Maslov's inverse method. Machine Intelligence 6, Edinburgh University Press (1971), 73-76.

14. Kowalski, R.A., and Kuehner, D.G., Linear resolution with selection function. Metamathematics Unit Memo 34, University of Edinburgh, (October, 1970).

15. Kowalski, R.A., and Kuehner, D.G., Linear resolution with selection function. Metamathematics Unit Memo 43, University of Edinburgh, (June, 1971).

16. Loveland, D.W., A simplified format for the model-elimination theorem-proving procedure. J.ACM 16 (July, 1969), 349-363.

17. Loveland, D.W., A linear format for resolution. Symposium on Automatic Demonstration, Lecture Notes in Mathematics 125, Springer-Verlag, Berlin and New York (1970), 147-163.

18. Loveland, D.W., Some linear Herbrand proof procedures: an analysis. Department of Computer Science, Carnegie-Mellon University (December, 1970).

19. Luckham, D., Refinement theorems in resolution theory. Symposium on Automatic Demonstration, Lecture Notes in Mathematics 125, Springer-Verlag, Berlin and New York (1970), 163-191.

20. Maslov, S.J., Proof-search strategies for methods of the resolution type. Machine Intelligence 6, Edinburgh University Press (1971), 77-90.

21. Meltzer, B., Theorem-Proving for computers: some results on resolution and renaming, Computer Journal, 8, (1966), 341-343.

22. 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), 15-33.

23. Pohl, I., Bi-directional search. Machine Intelligence 6, Edinburgh University Press (1971).

24. Prawitz, D., An improved proof procedure, Theoria 26, (1960).

25. Prawitz, D., Advances and problems in mechanical proof procedures. Machine Intelligence 4, Edinburgh University Press, (1969), 59-71.

26. Reiter, R., Two results on ordering for resolution with merging and linear format. Department of Computer Science, University of British Columbia (July, 1970).

27. Reynolds, J.C., Transformational systems and the algebraic structure of atomic formulas, Machine Intelligence 5, Edinburgh University Press (1969, 135-152.

28. Robinson, J.A., A machine oriented logic based on the resolution principle. Journal of the Association for Computing Machinery, 12, (1965).

29. Robinson, J.A., Automatic deduction with hyper-resolution, International Journal of Computer Mathematics 1, (1965).

30. Robinson, J.A., A review of automatic theorem-proving. Proceedings of Symposia in Applied Mathematics, 19, (1967), 1-18.

31. 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.

32. Yates, R.A., Raphael, B., and Hart, J.P., Resolution Graphs. Artificial Intelligence 1, (1970), 257-289.

33. Zamov, N.K., and Sharonov, 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 matematicheskoic logikve III, 16 (1969), 54-64. (National Lending Library, Russian Translating Program 5857, Boston Spa, Yorkshire.