10

Search Strategies for Theorem-Proving

Robert Kowalski
Metamathematics Unit
University of Edinburgh

We will define the notions of abstract theorem-proving graph, abstract theorem-proving problem $\mathscr{P}$ and search strategy Σ for $\mathscr{P}$. These concepts generalize the usual tree (or graph) searching problem and admit Hart, Nilsson and Raphael (1968) and Pohl (1969) theories of heuristic search. In particular the admissibility and optimality theorems of Hart, Nilsson and Raphael generalize for the classes $\mathscr{D}$ and $\mathscr{D}^u$ of diagonal search strategies for abstract theorem-proving problems. In addition the subclass $\mathscr{D}^u$ of $\mathscr{D}$ is shown to be optimal for $\mathscr{D}$. Implementation of diagonal search is treated in some detail for theorem-proving by resolution rules (Robinson 1965).

SEARCH STRATEGIES, COMPLETENESS AND EFFICIENCY

Completeness and efficiency of proof procedures can be studied only in the context of search strategies. A system T of inference rules and axioms can be complete or incomplete for a given class of intended interpretations. Similarly a search strategy Σ for T may or may not be complete for obtaining proofs constructible in T – independently of the completeness of T. A proof procedure (T, Σ) consists of a system of inference rules and axioms T together with a search strategy Σ for T. The procedure (T, Σ) can be complete in several distinct senses depending upon the completeness of T and Σ. These distinct notions of completeness are often confused and this confusion results in confused discussion regarding the value of complete versus heuristic methods in automatic theorem-proving.

The situation is no better with regard to discussions of efficiency. Proposals have been put forward both for increasing the strength of inference systems and for restricting the application of inference rules. Thus, for instance, ω-order logic is a strong inference system, whereas set-of-support resolution (Wos, Carson and Robinson 1965) is a restricted inference rule. However it is only proof procedures (T, Σ) which can be efficient for proving theorems.

A system of inference rules and axioms T is potentially efficient only if it admits a search strategy Σ which yields an efficient proof procedure (T, Σ). On the other hand, a search strategy Σ can be efficient for obtaining proofs constructible within T regardless of the efficiency of the resulting proof procedure (T, Σ), i.e., Σ may do a best possible job for an impossible T.

It is interesting that certain inference-related rules can be defined only in the context of search strategies. Deletion of subsumed clauses is an important example. The completeness of a deletion strategy for a proof procedure (T, Σ) is relative to the completeness of (T, Σ) and might be better termed 'compatibility with (T, Σ)'. Our own proof for the compatibility of deleting subsumed clauses (Kowalski and Hayes 1969) fails because no regard is taken of this relativity to search strategies. The compatibility with given (T, Σ) of deleting subsumed clauses has been proved for the case where Σ is a level saturation search, T is ordinary binary resolution (Robinson 1965) or AM-clash resolution (Sibert 1969) and deletion is done only after each level is saturated. Compatibility of the usual deletion rule for most complete T and for arbitrary Σ is proved in Kowalski (1970) where counterexamples are also exhibited for the compatibility and efficiency of alternative rules for deleting subsumed clauses. The compatibility with given (T, Σ) of deleting tautologies is a much simpler matter – but first proved explicitly for arbitrary Σ and most resolution systems T in Kowalski (1970).

Despite the importance of search strategies, most research in automatic theorem-proving has concentrated on developing new inference systems which are either more powerful or more restricted than ones already existing. The Unit Preference strategy of Wos, Carson and Robinson (1964) seems to be the basic search strategy employed by most computer programs which implement resolution rules in proof procedures. Slagle's Fewest Components strategy (*see* Sandewall 1969), Green's (1969a) partitioning of clauses into active and passive clauses, and Burstall's (1968) indexing scheme seem to be the only other reported proposals for improving search strategies.

THEOREM-PROVING GRAPHS

It is disconcerting that none of the research in tree searching techniques has yielded improved search strategies for theorem-proving. We wholly agree with Sandewall's (1967) assessment that searching for paths in trees is not general enough to represent the searches needed in automatic theorem-proving. A similar situation exists with respect to and/or trees where searching for subtrees cannot be represented helpfully by searching for paths in other trees. The notion of theorem-proving graph, defined below, is intended to extend the usual notion of tree (or graph) and to apply to the theorem-proving problem without encompassing the notion of and/or tree.

In the theorem-proving problem we begin with an initial non-empty set of sentences S_0 and with a set of inference rules Γ. If $\varphi \in \Gamma$ and S is a set of sentences then $\varphi(S)$ is another set of sentences. $\varphi(S) = \varnothing$ if φ is not applicable

to S. In particular $\varphi(S)=\varnothing$ if S is not finite. In applications to resolution systems, S_0 is a set of clauses and Γ consists of a single resolution rule or of a factoring rule and a separate rule for resolving factors. If φ is binary resolution of factors then $\varphi(S)=S'\neq\varnothing$ if S contains two factors which resolve or one factor which resolves with itself and each $C'\in S'$ is a resolvent of the clauses in S. If φ is the operation of unifying literals in a single clause then $\varphi(S)=S'\neq\varnothing$ if S is a singleton, $S=\{C\}$, and each $C'\in S'$ is a factor of C.

Given an initial set of sentences S_0 and a set of inference rules Γ let S^* be the set of all sentences which can be derived from S_0 by iterated application of the rules in Γ. Then each $\varphi\in\Gamma$ is a function $\varphi: 2^{S^*}\rightarrow 2^{S^*}$ defined on subsets of S^* taking subsets of S^* as values. Each sentence $C\in S^*$ can be assigned a level: if $C\in S_0$ then the level of C is zero, otherwise $C\in\varphi(S)$ for some $\varphi\in\Gamma$ and for some $S\subseteq S^*$ and the level of C is one greater than the maximum of the levels of the sentences $D\in S$. If S_i is the set of all sentences of level i then $S^*=\bigcup_{0\leqslant i} S_i$. Since a sentence $C\in S^*$ may have several distinct derivations, the level of C need not be unique. Since $\varphi(S)\neq\varnothing$ only if S is finite, the set of sentences which occur in a given derivation of a sentence $C\in S^*$ is always finite. The theorem-proving problem for a triple (S_0, Γ, F), $F\subseteq S^*$, is that of generating by means of a search strategy Σ some $C^*\in F$ by iterated application of the rules in Γ beginning with the sentences in S_0. For certain applications it may be required to derive a sentence $C^*\in F$ having minimum level in F or, more generally, having minimum cost in F, where cost is determined by some 'costing function' defined on the sentences in S^*. The tree (or graph) searching problem (Doran and Michie 1966, Sandewall 1969) can be interpreted as a theorem-proving problem (S_0, Γ, F) where each operator $\varnothing\in\Gamma$ has the property that $\varphi(S)=\varnothing$ whenever S is not a singleton.

A triple (S_0, Γ, F) determines a directed graph whose nodes are single sentences $C\in S^*$. C' is an immediate successor of C (i.e., C' is connected to C by an arc directed from C to C') if for some $S\subseteq S^*$ and $\varphi\in\Gamma$, $C\in S$ and $C'\in\varphi(S)$. The situation is similar to that which exists for ordinary graph searching problems as distinguished from tree searching problems. Searching in a directed graph for a path from a node a to a node b can be interpreted as searching in a directed labelled tree for a path from a node n_1, with label $c(n_1)=a$, to a node n_2, with label $c(n_2)=b$. The tree search interpretation of graph searching has the property of representing a single node c in a graph as distinct nodes $n_1, \ldots, n_k$ in a tree when the node c can be generated in k different ways as the end node of k different paths from the initial node a. This property of the tree search representation is one which we find useful when extended to deal with the more general theorem-proving problem. In particular the extended tree search representation associates distinct nodes with distinct derivations. This $1-1$ correspondence between nodes and derivations allows the number of nodes generated by a search

strategy Σ in the course of obtaining a terminal node to be treated as a measure of the efficiency of Σ for the given problem.

We define the notion of an *abstract theorem-proving graph* ('abstract graph' or simply 'graph') (G, s). The extended tree representation of an *interpreted theorem-proving graph* (S_0, Γ) can be obtained from (G, s) by labelling the nodes $n \in G$ by use of a labelling function $c: G \to S^*$, and by interpreting each application of the function s to a subset $G' \subseteq G$ as an application of a function $\varphi \in \Gamma$ to the subset $\{c(n) | n \in G'\}$. An abstract theorem-proving graph is a pair (G, s) where G is a set of nodes, $s: 2^G \to 2^G$ is a successor function defined on subsets of G taking subsets of G as values. G and s satisfy the following conditions:

(1) $s(\emptyset) = \emptyset$.
(2) $s(G') \neq \emptyset$ implies that G' is finite.
(3) $G' \neq G''$ implies that $s(G') \cap s(G'') = \emptyset$.
(4) Let $\mathscr{S}_0 = \{n \in G | n \notin s(G') \text{ for any } G' \subseteq G\}$,
let $\mathscr{S}_{k+1} = \{n \in G | n \in s(G') \text{ for some } G' \subseteq \bigcup_{i \leqslant k} \mathscr{S}_i,\ G' \cap \mathscr{S}_k \neq \emptyset\}$.

Then

(a) $\mathscr{S}_0 \neq \emptyset$,
(b) $G = \bigcup_{0 \leqslant i} \mathscr{S}_i$,
(c) $\mathscr{S}_i \cap \mathscr{S}_j = \emptyset$ for $i \neq j$.

The graph (G, s) reduces to an ordinary tree if $s(G') \neq \emptyset$ implies that G' is a singleton. For this case condition (3) states that distinct nodes have distinct sets of successors. More generally, (3) states that distinct sets of nodes have distinct sets of successors. It is precisely this condition which ensures that the graphs (G, s) extend the ordinary tree representation of search spaces. Condition (4) states that (G, s) is a levelled acyclic directed graph. In other words each $n \in G$ can be assigned a unique *level* i where $n \in \mathscr{S}_i$ and $n \notin \mathscr{S}_j$ for all $j \neq i$. If (S_0, Γ) is an interpretation of (G, s) with labelling function c: $G \to S^*$ then $S_i = \{c(n) | n \in \mathscr{S}_i\}$ is just the set of labelled nodes of level i. Condition (3) guarantees that for each $C \in S^*$ and for each distinct derivation of C from S_0 there is a distinct node $n \in G$ such that $C = c(n)$. There is no restriction that $\mathscr{S}_0$ or S_0 be finite. The case where $\mathscr{S}_0$ is infinite allows us to deal with axiom schemes in theorem-proving and more generally with potentially infinite sets of initial nodes $\mathscr{S}_0$.

The successor function s of (G, s) determines a partial ordering of the nodes in G: n' is an *immediate successor* of n (and n an *immediate ancestor* of n') if $n' \in s(G')$ and $n \in G'$ for some $G' \subseteq G$. A node n' is a *successor* of n (and n an ancestor of n'), written $n' > n$, if n' is an immediate successor of n or if n' is a successor of an immediate successor of n. We write $n \leqslant n'$ if $n < n'$ or $n = n'$. The definition of (G, s) guarantees that for all $n \in G$ the set $\{n' | n' \leqslant n\}$ is finite, although the set $\{n' | n' \geqslant n\}$ may be infinite. Notice that in the theorem-proving interpretation of graphs (G, s), a derivation of a sentence

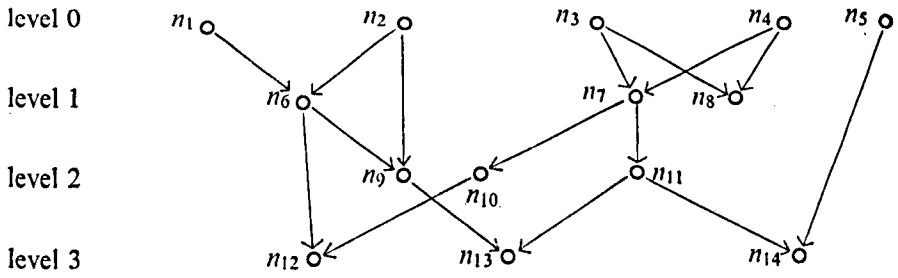

Figure 1

$c(n)$ consists of all the sentences $c(n')$ where $n' \leqslant n$. Each such derivation contains only finitely many sentences $c(n')$.

Figure 1 illustrates a graph (G, s) where nodes are represented as points and where points n and n' are connected by a directed line from n to n' if n' is an immediate successor of n. In general it is convenient to picture graphs as directed downward, so that n lies above n' if n' is a successor of n. To determine in figure 1 if $n \in s(G')$ it suffices to verify that G' is the set of all nodes connected to n by an arc directed to n. Thus, for example,

$$s(n_1, n_2) = \{n_6\},$$
$$s(n_2, n_6) = \{n_9\},$$
$$s(n_3, n_4) = \{n_7, n_8\},$$
$$s(n_7) = \{n_{10}, n_{11}\},$$
$$s(n_2) = s(n_5) = s(n_8) = s(n_1, n_2, n_6) = \emptyset.$$

If the graph of figure 1 is interpreted as a resolution graph by a labelling function c: $G \to S^*$ then the two clauses $c(n_7)$ and $c(n_8)$ must be all the resolvents of the pair $c(n_3)$, $c(n_4)$. The clause $c(n_8)$ resolves with none of the clauses $c(n_i)$, $1 \leqslant i \leqslant 14$. The clauses $c(n_{10})$ and $c(n_{11})$ are either factors of $c(n_7)$ or are obtained from $c(n_7)$ by resolving $c(n_7)$ with itself. If $C = c(n_6) = c(n_7) = c(n_{14})$ then C has three derivations, two of level one and one of level three. Derivations are not necessarily represented by derivation trees. For instance the derivation of $c(n_{13})$ consists of the clauses $c(n_1)$, $c(n_2)$, $c(n_3)$, $c(n_4)$, $c(n_6)$, $c(n_7)$, $c(n_9)$, $c(n_{11})$, $c(n_{13})$. The clause $c(n_2)$ is used twice in the derivation of $c(n_{13})$ but is represented by only one node in G.

An *abstract theorem-proving problem with non-negative costs* ('abstract problem with costs' or simply 'problem') is a tuple $\mathscr{P} = (G, s\ \mathscr{F}, g)$ where $\mathscr{F} \subseteq G$, the set of *terminal nodes* for $\mathscr{P}$ (or solution nodes), and g: $G \to \mathbb{R}$, the *costing function* of $\mathscr{P}$, ($\mathbb{R}$, the set of real numbers) are such that

(1) $n \in \mathscr{F}$ implies that $s(G') = \emptyset$ whenever $n \in G' \subseteq G$,

(2) (a) $g(n) \geqslant 0$ for all $n \in G$,

(b) if $n \in s(n_1, \ldots, n_k)$ (we write $s(n_1, \ldots, n_k)$ instead of $s(\{n_1, \ldots n_k\})$) then $g(n) \geqslant \max_{1 \leqslant i \leqslant k} g(n_i)$.

A solution to the problem $\mathscr{P}$ is obtained by constructing an algorithm Σ which generates from $\mathscr{S}_0$ a node $n \in \mathscr{F}$. Each node $n \in \mathscr{F}$ is assigned a cost and it is often required to solve $\mathscr{P}$ by generating a node $n \in \mathscr{F}$ having minimal cost in $\mathscr{F}$. If $g(n) = 0$ for all $n \in G$ then in effect we have a problem without costs. Alternatively $g(n)$ may be taken to be the level of n, the number of nodes $n' \leqslant n$ or any other value which satisfies (3) above. In applications to resolution theory $g(n)$ is usually taken to be the level of the clause $c(n)$. For $n \in \mathscr{S}_0$ we do not require that $g(n) = 0$. This freedom allows us to assign different costs to distinct nodes in $\mathscr{S}_0$ and is especially useful when $\mathscr{S}_0$ is infinite. The set $\mathscr{F}$ may be empty in which case the problem has no solution. In resolution applications when $\mathscr{F} = \{n \in G | c(n) = \square\}$ then $\mathscr{F}$ is empty if the set $S_0 = \{c(n) | n \in \mathscr{S}_0\}$ is satisfiable. The general problem $\mathscr{P} = (G, s, \mathscr{F}, g)$ reduces to the ordinary tree searching problem when (G, s) is a tree.

SEARCH STRATEGIES FOR ABSTRACT THEOREM-PROVING PROBLEMS

A *search strategy* Σ for $\mathscr{P}$ is a function $\Sigma: 2^G \to 2^G$ which generates subsets of G from other subsets of G. Given such a function Σ for $\mathscr{P}$ we define the sets Σ_i of nodes already *generated by* Σ *before the* $(i+1)$th *stage* and $\tilde{\Sigma}_i$ of nodes which are *candidates for generation by* Σ *at the* $(i+1)$th *stage*:

$$\Sigma_0 = \varnothing, \ \tilde{\Sigma}_0 = \mathscr{S}_0, \tag{1}$$

$$\Sigma_{i+1} = \Sigma_i \cup \Sigma(\Sigma_i),$$

$$\tilde{\Sigma}_{i+1} = (\{n | n \in s(G'), \ G' \subseteq \Sigma_{i+1}\} \cup \tilde{\Sigma}_i) - \Sigma_{i+1}. \tag{2}$$

We require that Σ satisfy

$$\Sigma(\Sigma_i) \subseteq \tilde{\Sigma}_i. \tag{3}$$

The set of nodes $\Sigma(\Sigma_i)$, chosen from the set of candidates $\tilde{\Sigma}_i$, is the set of nodes *newly generated by* Σ *at the* $(i+1)$th *stage*. (It is easy to verify that $\Sigma_i \cap \tilde{\Sigma}_i = \varnothing$ for all $i \geqslant 0$.) The function Σ should be interpreted as selecting subsets G' of Σ_i and generating nodes $n \in s(G')$ which have not been previously generated. The definitions above only partially formalize the intuitive notion of search strategy for P. In particular the search strategies Σ are never allowed to display any redundancy, i.e., generate the same node twice. This restriction is not essential because given any concrete, possibly redundant, algorithm for generating nodes in G there corresponds a unique search strategy Σ which, except for redundancies, generates the same nodes in the same order.

Notice that $\Sigma(\Sigma_i)$ may contain more than one node – as is common with resolution strategies which simultaneously generate several resolvents of a single clash or several factors of a single clause. Notice too that nodes in $\mathscr{S}_0$ can be generated at any stage. We do not require that $\Sigma(\Sigma_i)$ contain a node $n \in \mathscr{F}$ when $\tilde{\Sigma}_i \cap \mathscr{F} \neq \varnothing$. If $\mathscr{P}$ is an ordinary tree search problem then the definition of search strategy for $\mathscr{P}$ provides a formal notion which applies to the usual strategies employed in searching for nodes in trees.

A search strategy Σ for $\mathscr{P}=(G, s, \mathscr{F}, g)$ is *complete* for $\mathscr{P}$ if for all $n \in G$ there exists an $i>0$ such that $n \in \Sigma_i$. It is possible to define completeness in this way since we do not insist that Σ generates no additional nodes after generating a first node $n^* \in \mathscr{F}$. We say that Σ *terminates* at stage $i>0$ if $\mathscr{F} \cap \Sigma_{i-1}=\varnothing$ and either (1) $\mathscr{F} \cap \Sigma_i \neq \varnothing$ or (2) $\Sigma_i=\Sigma_{i-1}$. In the first case Σ terminates with a solution and without a solution in the second case.

In the terminology of Hart, Nilsson and Raphael (1968), a search strategy Σ is said to be *admissible* for $\mathscr{P}$ if Σ is complete for $\mathscr{P}$ and terminates with a solution having least cost in $\mathscr{F}$ if $\mathscr{F} \neq \varnothing$, i.e., $n^* \in \mathscr{F} \cap \Sigma_i$, $\mathscr{F} \cap \Sigma_{i-1}=\varnothing$ implies that $g(n^*) \leqslant g(n)$ for all $n \in \mathscr{F}$. In resolution applications admissible search strategies are of special interest for robot control and automatic program writing (Green 1969b) where minimal cost solutions are related to simplest strategies and most efficient programs. More generally intuition suggests that, in the absence of special information about the location of non-minimal solutions, admissible search strategies will tend to be more efficient than non-admissible strategies for finding arbitrary solutions. An important step towards formalizing this intuition has already been made in the optimality theorems of Hart, Nilsson and Raphael (1968).

We define the notion of a search strategy Σ for a problem $\mathscr{P}=(G, s, \mathscr{F}, g)$ being compatible with a *merit ordering* $\preccurlyeq$ defined on the nodes of G. For the moment we require only that $\preccurlyeq$ be reflexive and defined for all pairs of nodes in G. We write $n_1 \prec n_2$ (n_1 has *better merit* than n_2) when $n_1 \preccurlyeq n_2$ and not $n_2 \preccurlyeq n_1$. We write $n_1 \approx n_2$ (n_1 and n_2 have *equal merit*) if $n_1 \preccurlyeq n_2$ and $n_2 \preccurlyeq n_1$. A search strategy Σ for $\mathscr{P}$ is compatible with a merit ordering $\preccurlyeq$ if for all $i \leqslant 0$,

(1) $\tilde{\Sigma}_i \neq \varnothing$ implies that $\Sigma(\Sigma_i) \neq \phi$,

(2) $n \in \Sigma(\Sigma_i)$ implies that $n \preccurlyeq n'$ for all $n' \in \tilde{\Sigma}_i$.

In other words Σ always generates, from a non-empty set $\tilde{\Sigma}_i$, at least one node $n \in \tilde{\Sigma}_i$, and no node $n' \in \tilde{\Sigma}_i$ which is not generated by Σ has better merit than any node $n \in \tilde{\Sigma}_i$ which is generated by Σ. Since a node n may have better merit than an ancestor $n' < n$, Σ need not generate nodes in order of merit. Distinct strategies Σ and Σ' for the same $\mathscr{P}$ compatible with the same merit ordering $\preccurlyeq$ differ only with regard to tie breaking rules for choosing which nodes to generate from a set of candidates having equal merit. If $\preccurlyeq$ is the trivial ordering, $n \preccurlyeq n'$ for all $n, n' \in G$, then $\preccurlyeq$ is a merit ordering for G and any search strategy Σ for $\mathscr{P}$ is compatible with $\preccurlyeq$. If $\preccurlyeq$ is the ordering by levels, $n \preccurlyeq n'$ if and only if $n \in \mathscr{S}_i$, $n' \in \mathscr{S}_{i'}$ and $i \leqslant i'$, then any search strategy for $\mathscr{P}$ compatible with $\preccurlyeq$ is a level saturation (or breadth first) strategy for $\mathscr{P}$. If $\preccurlyeq$ is the ordering by costs, $n \preccurlyeq n'$ if and only if $g(n) \leqslant g(n')$, then Σ compatible with $\preccurlyeq$ is a cost saturation strategy for $\mathscr{P}$. If $\preccurlyeq$ is the inverse ordering by levels, $n \preccurlyeq n'$ if and only if $n \in \mathscr{S}_i$, $n' \in \mathscr{S}_{i'}$ and $i \geqslant i'$, then Σ compatible with $\preccurlyeq$ is a depth first strategy for $\mathscr{P}$.

Lemma 1 states the fundamental properties of search strategies Σ compatible with merit orderings: (a) any node $n_2 \in G$ is generated by Σ before

any node n_1 which has worse merit than n_2 and than all the ancestors of n_2, (b) if n_1 is generated before n_2 then n_2 or some ancestor of n_2 has worse or equal merit to n_1.

Lemma 1

Let $\mathscr{P}=(G, s, \mathscr{F}, g)$ be a problem, $\preccurlyeq$ a merit ordering for G and Σ a search strategy for $\mathscr{P}$ compatible with $\preccurlyeq$.

(a) If $n_1 \in \Sigma_i$ and $n_2 \in G$ are such that $n \prec n_1$ for all $n \leqslant n_2$ then $n_2 \in \Sigma_{1-i}$.

(b) If $n_1 \in \Sigma_i$ and $n_2 \in \Sigma(\Sigma_i)$ then $n_1 \preccurlyeq n$ for some $n \leqslant n_2$.

Proof. (a) Let n_1 be generated at the $(j+1)$th stage, i.e., $n_1 \in \Sigma(\Sigma_j)$, $n_1 \notin \Sigma_j$ and $j<i$. If $n_2 \notin \Sigma_j$ then for some $n \leqslant n_2$, $n \notin \Sigma_j$ and $n \in \tilde{\Sigma}_j$. But $n \prec n_1$ and therefore Σ is not compatible with $\preccurlyeq$ since it generates n_1 instead of n at the $(j+1)$th stage. Therefore $n_2 \in \Sigma_j$ and $n_2 \in \Sigma_{i-1}$ since $j<i$.

(b) Suppose $n \prec n_1$ for all $n \leqslant n_2$. Then by (a), $n_2 \in \Sigma_{i-1}$ and therefore $n_2 \notin \Sigma(\Sigma_i)$.

A merit ordering $\preccurlyeq$ for G is *δ-finite* if for all $n \in G$ the set $\{n' \in G \mid n' \preccurlyeq n\}$ is finite (compare Hart, Nilsson and Raphael 1968). The importance of δ-finite merit orderings is a consequence of Theorem 1: any search strategy compatible with a δ-finite merit ordering is complete. Any merit ordering for a finite set G is δ-finite. Ordering by levels is δ-finite if $\mathscr{S}_0$ is finite and $s(G')$ is finite for all $G' \subseteq G$, under the same conditions inverse ordering by levels is not δ-finite if G is infinite (by König's Lemma).

Theorem 1

If $\mathscr{P}=(G, s, \mathscr{F}, g)$ is a problem, $\preccurlyeq$ a δ-finite merit ordering for G and Σ a search strategy for $\mathscr{P}$ compatible with $\preccurlyeq$, then Σ is complete for $\mathscr{P}$.

Proof. Let $n^* \in G$ be given. We need to show that $n^* \in \Sigma_j$ for some $j>0$. If G is finite then $G=\bigcup_{i>0} \Sigma_i$ since $\tilde{\Sigma}_i \neq \emptyset$ implies that $\Sigma(\Sigma_i) \neq \emptyset$ and since $\Sigma(\Sigma_i) \cap \Sigma_i = \emptyset$.

Otherwise if G is infinite let $n' \leqslant n^*$ be a node such that $n \preccurlyeq n'$ for all $n \leqslant n^*$. Since $\preccurlyeq$ is δ-finite, since $\tilde{\Sigma}_i \neq \emptyset$ implies that $\Sigma(\Sigma_i) \neq \emptyset$ and since $\Sigma(\Sigma_i) \subseteq \tilde{\Sigma}_i$ it follows that for some $j>0$ and for some $n_1 \in \Sigma_j$, $n' \prec n_1$, and therefore $n \prec n_1$ for all $n \leqslant n^*$ and by Lemma 1 (a), $n^* \in \Sigma_j$.

HEURISTIC FUNCTIONS AND DIAGONAL SEARCH

There is special interest in merit orderings which can be expressed in terms of the cost function g of $\mathscr{P}=(G, s, \mathscr{F}, g)$ and of an additional heuristic function h (Hart, Nilsson and Raphael 1968, Nilsson 1968, Pohl 1969). A *heuristic function h* for $\mathscr{P}$ is a function $h: G \to \mathbb{R}$ such that $h(n) \geqslant 0$, for all $n \in G$. Let $f(n)=g(n)+h(n)$ for all $n \in G$. The intended interpretation of the heuristic function h is that $f(n)=g(n)+h(n)$ is an estimate of the cost $g(n^*)$ of a terminal node $n^* \in \mathscr{F}$, such that $n \leqslant n^*$, i.e., $h(n)$ is an estimate of

$g(n^*)-g(n)$. If it is desired that Σ be admissible then $h(n)$ is intended to estimate the minimum value of $g(n^*)-g(n)$ for $n^* \in \mathscr{F}$ such that $n \leqslant n^*$.

Suppose, for example, that we know of a given problem $\mathscr{P}_0 = (G_0, s_0, \mathscr{F}_0, g_0)$ that if it has a solution then its minimum cost is k. Suppose for simplicity that no $n \in G_0$ has cost $g_0(n)$ greater than k. Given only this information then an appropriate definition of a heuristic function h_0 for $\mathscr{P}_0$ is $h_0(n) = k - g_0(n)$ for all $n \in G_0$.

Suppose that a given problem $\mathscr{P}_1 = (G_1, s_1, \mathscr{F}_1, g_1)$ is interpreted as a resolution problem by a labelling function $c: G_1 \to S^*$. Suppose that the inference rules Γ consist of a factoring operation for unifying two literals in a clause and of a separate resolution rule for resolving at most two factors. Let $g_1(n)$ be the level of n and $\mathscr{F}_1 = \{n \mid c(n) = \square\}$. For $n \in G_1$ let $l(c(n))$ be the length of $c(n)$ (number of literals in $c(n)$). The heuristic function h_1 for $\mathscr{P}_1$ is defined by letting $h_1(n)$ be the expected length of $c(n)$:

(1) for $n \in \mathscr{S}_0$, $h_1(n) = l(c(n))$,
(2) for $c(n)$ a resolvent of $c(n_1)$ and $c(n_2)$, $h_1(n) = l(c(n_1)) + l(c(n_2)) - 2$,
(3) for $c(n)$ a factor of $c(n')$ (the result of unifying two literals in $c(n')$) $h(n) = l(c(n')) - 1$.

To the extent that merging does not occur (i.e., so long as $h_1(n) = l(c(n))$), $h_1(n)$ is a lower bound on the value of $g_1(n^*) - g_1(n)$ for $c(n^*) = \square$ when $c(n)$ occurs in a derivation of $\square$.

The costing function g and heuristic function h allow us to define two important classes of search strategies for $\mathscr{P}$. Given $\mathscr{P} = (G, s, \mathscr{F}, g)$ and h a heuristic function for $\mathscr{P}$. Let the merit orderings $\preccurlyeq_d$ and $\preccurlyeq_{du}$ for G be defined for all $n_1, n_2 \in G$, by

(1) $n_1 \preccurlyeq_d n_2$ if and only if $f(n_1) \leqslant f(n_2)$,
(2) $n_1 \preccurlyeq_{du} n_2$ if and only if $f(n_1) \leqslant f(n_2)$ and $h(n_1) \leqslant h(n_2)$ when $f(n_1) = f(n_2)$.

A search strategy Σ for $\mathscr{P}$ is a *diagonal search strategy* for $\mathscr{P}$ and h (written $\Sigma \in \mathscr{D}(\mathscr{P}, h)$) if and only if Σ is compatible with the merit ordering $\preccurlyeq_d$. Σ is an *upwards diagonal search strategy* for $\mathscr{P}$ and h ($\Sigma \in \mathscr{D}^u(\mathscr{P}, h)$) if and only if Σ is compatible with the merit ordering $\preccurlyeq_{du}$. Notice that $\mathscr{D}^u(\mathscr{P}, h) \subseteq \mathscr{D}(\mathscr{P}, h)$ and that $\mathscr{D}^u(\mathscr{P}, h) = \mathscr{D}(\mathscr{P}, h)$ if $h(n) = 0$ for all $n \in G$.

Except for minor differences, the search strategies $\Sigma \in \mathscr{D}(\mathscr{P}, h)$ coincide with those investigated in Hart, Nilsson and Raphael (1968) for the case of ordinary tree search. The search strategies $\Sigma \in \mathscr{D}^u(\mathscr{P}, h)$ differ from those in $\mathscr{D}(\mathscr{P}, h)$ by generating, from among candidate nodes which have equal merit for $\preccurlyeq_d$, those nodes whose cost is estimated to be closest to the cost of a solution node. In the case of the problem $\mathscr{P}_0$ and heuristic function h_0, defined above, $f_0(n) = g_0(n) + h_0(n) = k$ for all $n \in G_0$. All nodes in G have equal merit for search strategies $\Sigma \in \mathscr{D}(\mathscr{P}_0, h_0)$. For $\Sigma \in \mathscr{D}^u(\mathscr{P}_0, h_0)$ nodes

which have cost closer to k have better merit than nodes which have smaller cost. In case $g_0(n)$ is the level of n for all $n \in G_0$ then $\Sigma \in \mathscr{D}^u(\mathscr{P}_0, h_0)$ is a depth-first search strategy, which intuitively seems the most efficient search strategy for $\mathscr{P}_0$, given only the information that a minimal solution of $\mathscr{P}_0$ must have level k. Concrete search algorithms for $\Sigma \in \mathscr{D}^u(\mathscr{P}_1, h_1)$ are discussed in the next section.

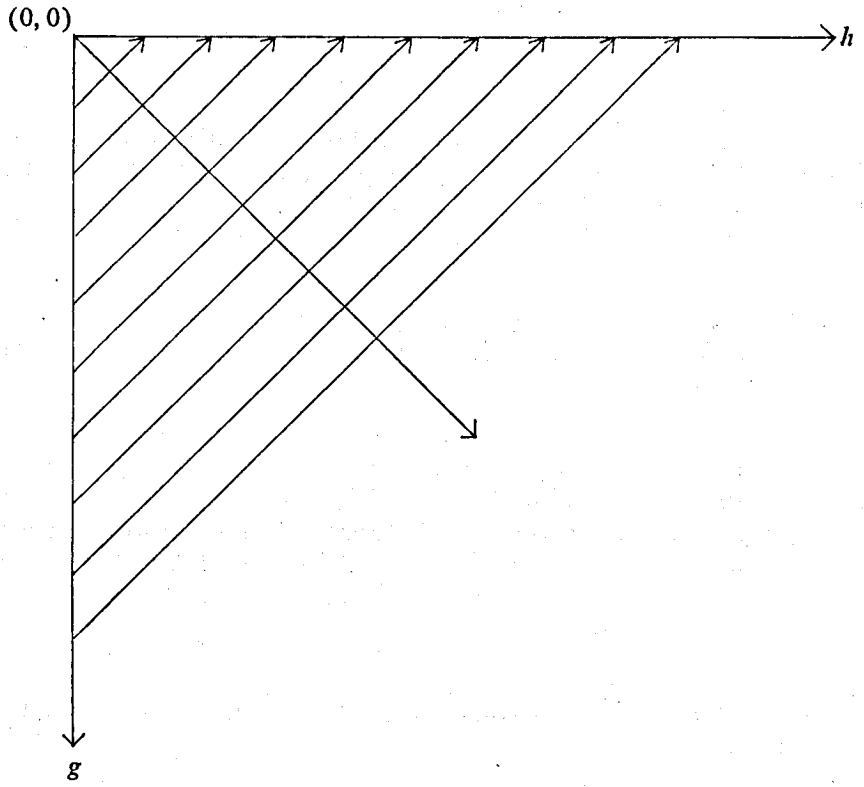

Figure 2

The terminology, diagonal and upwards diagonal search, is suggested by representing nodes $n \in G$ as occupying positions in the plane with coordinates $(h(n), g(n))$, where h increases rightwards away from the origin and g increases downwards away from the origin (see figure 2). $\Sigma \in \mathscr{D}(\mathscr{P}, h)$ attempts to generate nodes on consecutive diagonals in order of increasing distance from the origin (0, 0). In addition if $\Sigma \in \mathscr{D}^u(\mathscr{P}, h)$ then Σ attempts to generate nodes, lying on a given diagonal d, upwards in order of increasing h. If $\preccurlyeq_d$ or $\preccurlyeq_{du}$ are δ-finite then each diagonal contains only finitely many nodes $n \in G$ and for every diagonal d there are only finitely many diagonals which contain nodes $n \in G$ and which are closer than d to (0, 0).

Figure 3 illustrates Lemma 1 and Theorem 1 for a problem $\mathscr{P}$ and for a search strategy $\Sigma \in \mathscr{D}^u(\mathscr{P}, h)$ where $\preccurlyeq_{du}$ is assumed to be δ-finite. The node $n^* \in \mathscr{T}$ has minimum cost in $\mathscr{T}$ and $n' \leqslant n^*$ is a node having worst merit in the set consisting of n^* and all ancestors of n^*. The node $n \in G$ has better merit than n^* and $n'' \leqslant n$ has worst merit in the set consisting of n and all ancestors of n. Dots represent nodes, lying on diagonals, generated by Σ before the generation of n^*. The small circles represent nodes generated by Σ after the generation of n^*. The diagonal d contains the node n'. By Lemma 1, Σ generates n^* before generating nodes having worse merit than n', i.e., before generating nodes lying above n' on d and before generating nodes lying on diagonals to the right of d.

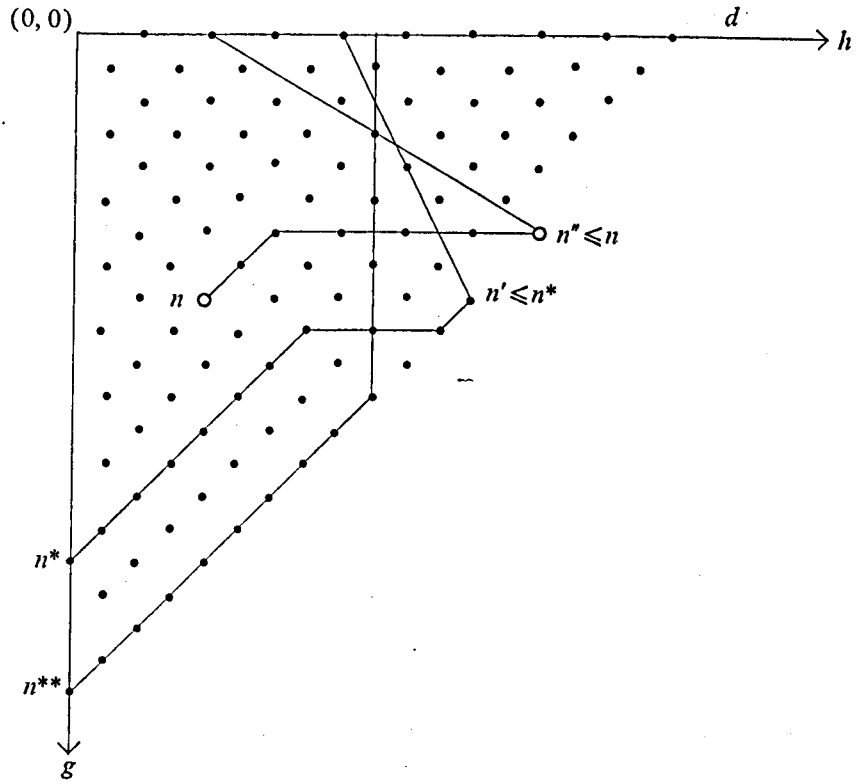

Figure 3

The heuristic function h satisfies no conditions other than $h(n^*) = h(n^{**}) = 0$ and those imposed by the δ-finiteness of $\preccurlyeq_{du}$. Σ may fail to be admissible because some $n^{**} \in \mathscr{T}$ having worse merit than n^* will be generated before n^* if n^{**} and all ancestors of n^{**} have better merit than n'. The node $n \in G$ will not be generated before n^* if n'' lies to the right of d or above n' on d.

UPWARDS DIAGONAL SEARCH STRATEGIES FOR RESOLUTION

The algorithm Σ^* defined below approximates an upwards diagonal search strategy for the resolution problem $\mathscr{P}_1$ and heuristic function h_1. The same algorithm Σ^* when applied to the resolution problem $\mathscr{P}_2$ and heuristic function h_2 defined below is a pure upwards diagonal search strategy for $\mathscr{P}_2$ and h_2. The admissibility and optimality theorems of the next section apply to Σ^* for $\mathscr{P}_2$ and h_2 and to Σ^* for $\mathscr{P}_1$ and h_1, except when merging occurs in $\mathscr{P}_1$. A search strategy which differs inessentially from Σ^* has been implemented in POP-2 by Miss Isobel Smith for a problem and heuristic function similar to $\mathscr{P}_1$ and h_1.

The definition and identification of the problem $\mathscr{P}_2$ was motivated by a suggestion of Mr Donald Kuehner. $\mathscr{P}_2=(G_2, s_2, \mathscr{F}_2, g_2)$ differs from $\mathscr{P}_1$ by interpreting clauses $c(n)$ as lists of literals and by explicitly exhibiting and assigning cost to the operation (treated as a special case of factoring) of identifying two copies of the same literal within a clause. The length $l(c(n))$ of $c(n)$ is defined as the number of literals in the clause $c(n)$, counting duplications. $g_2(n)$ and $h_2(n)$ are still defined respectively as the level of n and expected length of $c(n)$. $h_2(n)=l(c(n))$ for all $n\in G_2$ and $h_2(n)$ is always a lower bound on the value of $g_2(n^*)-g_2(n)$ when $n\leqslant n^*$ and $n^*\in\mathscr{F}_2=\{n\in G_2 \mid c(n)=\square\}$.

Throughout the remainder of this section, $\mathscr{P}=(G, s, \mathscr{F}, g)$ and h are either $\mathscr{P}_1$ and h_1 or $\mathscr{P}_2$ and h_2. The definition of Σ^* for $\mathscr{P}$ and h is the same for both of these cases except for the details remarked upon at the end of this section.

Clauses $c(n)$ are stored upon the generation of n in cells $A(i,j)$ of a two-dimensional array A. $c(n)$ is stored in $A(i,j)$ if $l(c(n))=i$ and $g(n)=j$. Although it is natural to represent cells $A(i,j)$ as lists of clauses, we write $c(n)\in A(i,j)$ if $c(n)$ is stored in $A(i,j)$ when n is generated. The *merit* of a node $n\in G$ is defined to be the cell $A(h(n), g(n))$. The cell $A(i,j)$ is said to be *better* than $A(i',j')$ (written $A(i,j)\prec A(i',j')$) if

(1) $i+j<i'+j'$ or
(2) $i+j=i'+j'$ and $i<i'$.

Thus *a node $n\in G$ has better upwards diagonal merit than a node $n'\in G$ if and only if the merit of n is better than the merit of n'*, equivalently $n\leqslant_{d^u} n'$ if and only if $A(h(n), g(n))\prec A(h(n'), g(n'))$. Notice that for $\mathscr{P}_2$ and h_2, $n\in G_2$ has merit $A(i,j)$ if and only if $c(n)\in A(i,j)$. For $\mathscr{P}_1$ and h_1, if $n\in G_1$ has merit $A(i,j)$ then $c(n)\in A(i',j)$ where $i'=l(c(n))\leqslant h(n)=i$. Σ^*, on the whole, attempts to generate nodes of merit $A(i,j)$ before attempting to generate nodes of worse merit $A(i',j')\succ A(i,j)$. A node of merit $A(i,j)$ is generated either

(0) by inserting into $A(i,0)$, when $j=0$, a clause $c(n)$ where $l(c(n))=i$ and $g(n)=0$,

(1) by unifying two literals within a clause $c(n) \in A(i+1, j-1)$ or

(2) by resolving a factor $c(n_1) \in A(i_1, j_1)$ with a factor $c(n_2) \in A(i_2, j_2)$ where n_1 may be identical to n_2 and where

$i = i_1 + i_2 - 2$ and

$j = \max(j_1, j_2) + 1$.

Σ^* employs two subalgorithms for generating nodes $n \in G$. The principal subalgorithm, *Fill*(i, j), generates in all possible ways, from nodes already generated, nodes n of merit $A(i, j)$ which have worse merit than all their ancestors. *Fill*(i, j) terminates when all such nodes have been generated. *Fill*(i', j'), where $A(i', j')$ is the next cell after $A(i, j)$, begins when *Fill*(i, j) terminates. Σ^* begins by invoking *Fill*$(0, 0)$.

Whenever a node n_0 is generated by *Fill*(i, j) the second subalgorithm *Recurse*$(c(n_0))$ interrupts *Fill*(i, j) and generates in all possible ways, from nodes already generated, nodes n which are immediate successors of n_0 and which are of merit $A(i_1, j_1)$ better than $A(i, j)$. In general whenever a node n is generated, either directly by *Fill*(i, j) or by some call of *Recurse*$(c(n'))$ which is local to *Fill*(i, j), *Recurse*$(c(n))$ generates, from nodes already generated, immediate successors of n which are of better merit than $A(i, j)$. Notice that if n is generated by *Recurse*$(c(n'))$ during *Fill*(i, j) then n has better merit than some ancestor of merit $A(i, j)$. Notice too that the depth of recursion involved in *Recurse*$(c(n'))$ is bounded by the sum $i+j$.

REMARKS

(1) If $\mathscr{P}$ and h are $\mathscr{P}_2$ and h_2 and if $c(n_0)$ is generated directly by *Fill*(i, j) then $c(n_0) \in A(i, j)$ and the only immediate successors of n_0 which are of better merit than $A(i, j)$ are nodes $n_1 \in A(i-1, j+1)$. Any such n_1 generated by *Recurse*$(c(n_0))$ is obtained either by factoring $c(n_0)$ or by resolving $c(n_0)$ with a unit clause $c(n)$ of level $g(n) \leqslant j$. More generally if n_0 is generated by *Recurse* during *Fill*(i, j) and if $c(n_0) \in A(i_0, j_0)$ then it is easy to verify that $i_0 + j_0 = i + j$ and any immediate successor of n_0 of better merit than $A(i, j)$ is of merit $A(i_0 - 1, j_0 + 1)$.

(2) If $\mathscr{P}$ and h are $\mathscr{P}_1$ and h_1 then Σ^* may fail to do upwards diagonal search because of merging, i.e., nodes may be generated by *Recurse* which have worse merit than other candidates for generation. Suppose that n_0 is generated by *Fill*(i, j) and that $c(n_0) \in A(i', j)$ where $i' < i$. Suppose that n_1 and n_2 of merit $A(i'-1, j+1)$ are generated by *Recurse*$(c(n_0))$, n_1 before n_2. Suppose that n_3 of merit $A(i'-1, j+2) \prec A(i, j)$ is generated by *Recurse*$(c(n_1))$. Then n_2 has better merit than n_3 but n_3 is generated before n_2 since *Recurse* $(c(n_1))$ must terminate before *Recurse* $c(n_0))$ generates n_2.

(3) For both $\mathscr{P}_1, h_1$ and $\mathscr{P}_2, h_2$, Σ^* has the desirable property of attempting to resolve every unit clause $c(n_0)$ with all previously generated units $c(n_1)$ as soon as $c(n_0)$ is generated. If n_0 is generated during *Fill*(i, j) and if $c(n_0) \in A(1, j_0)$ and $c(n_1) \in A(1, j_1)$ then $A(0, max(j_0, j_1) + 1) \prec A(i, j)$ and an attempt will be made to resolve $c(n_0)$ with $c(n_1)$ during *Recurse*$(c(n_0))$.

(4) Suppose that $Fill(i,j)$ has just begun, then Σ^* has not yet generated any nodes of merit worse than $A(i,j)$. Thus if n has merit $A(i,j)$ then either $j=0$ and $g(n)=0$ or $c(n)$ is a resolvent of factors $c(n_1)$ and $c(n_2)$ and both n_1 and n_2 are of merit better than $A(i,j)$. In order to generate all such nodes n it suffices to attempt to resolve all clauses $c(n_1)$ with clauses $c(n_2)$ where

$$c_1 \in A(l,k),\ c_2 \in A(i-l+2, j-1)$$

for $0 \leqslant k \leqslant j-1$ and $1 \leqslant l \leqslant \frac{i}{2}$ if i is even or $1 \leqslant l \leqslant \frac{i+1}{2}$ if i is odd.

(5) The details for generating nodes during $Recurse(c(n))$ have already been discussed for $\mathscr{P}_2$ and h_2 in remark (1). For $\mathscr{P}_1$ and h_1 these details are more complicated. Suppose that n has been generated during $Fill(i^*,j^*)$ and that $c(n) \in A(i,j)$. The following procedure will generate, without redundancy, from nodes generated before n, immediate successors of n which are of better merit than $A(i^*,j^*)$:

(a) First resolve $c(n)$ with clauses in $A(i',j')$ where $j-1 \leqslant j' \leqslant i^*+j^*-i+2$, in order of decreasing j', and for given j', where $1 \leqslant i' \leqslant i^*+j^*-j'-i+1$ in arbitrary order but preferably in order of increasing i'.

(b) Next generate factors of $c(n)$ by attempting to unify, in all possible ways, two literals in $c(n)$.

(c) Finally resolve $c(n)$ with clauses in $A(i',j')$ where $1 \leqslant i' \leqslant i^*+j^*-i-j+1$; $0 \leqslant j' \leqslant j$ in arbitrary order but preferably in order of increasing i'.

ADMISSIBILITY AND OPTIMALITY OF $\mathscr{D}$ AND $\mathscr{D}^u$

Let $\mathscr{P}=(G, s, \mathscr{F}, g)$ be an abstract theorem-proving problem. For $n \in G$ let

$$H(n)=\{g(n^*)-g(n) \mid n^* \in \mathscr{F},\ n \leqslant n^*\},$$
$$h^*(n)=inf\, H(n) \text{ when } H(n) \neq \varnothing,$$
$$h^*(n)=\infty \text{ when } H(n)=\varnothing.$$

Then when $n \leqslant n^*$, for some $n^* \in \mathscr{F}$, $h^*(n)$ is the greatest lower bound on the additional cost over $g(n)$ of $g(n^*)$. The heuristic function h is intended to be an estimate of h^*. The only property of ∞ needed below is that $k<\infty$ for all real numbers k. Since we do not allow $h(n)=\infty$, it is often impossible to construct a heuristic function h which gives a perfect estimate of h^*. In particular it is impossible to incorporate into a definition of h any information that a node n is not an ancestor of a node $n^* \in \mathscr{F}$. However such heuristic information can be applied to a problem $\mathscr{P}$ by defining a new problem $\mathscr{P}'$ which differs from $\mathscr{P}$ by containing no such nodes n. Alternatively it is possible to allow $h(n)=\infty$ in which case several complexities need to be introduced in preceding definitions (e.g., in the definition of δ-finiteness).

A heuristic function h for $\mathscr{P}$ satisfies the *lower bound condition* for $\mathscr{P}$ if

$$h(n) \leqslant h^*(n) \text{ for all } n \in G,$$

i.e., if $h(n) \leqslant g(n^*)-g(n)$ whenever $n^* \in \mathscr{F}$ and $n \leqslant n^*$. Thus the lower bound condition constrains in effect only the value of $h(n)$ when n is an

ancestor of some solution node. Recall that h_2 satisfies the lower bound condition for $\mathscr{P}_2$ while h_1 does the same for $\mathscr{P}_1$ except for merging.

Lemma 2 states certain fundamental properties of heuristic functions h satisfying the lower bound condition: (a) $h(n^*)=0$ for $n^* \in \mathscr{F}$, (b) no ancestor of a solution node $n^* \in \mathscr{F}$ has worse diagonal merit than n^*, (c) there exists a solution node $n^* \in \mathscr{F}$ having minimum cost in $\mathscr{F}$ if diagonal merit is δ-finite.

Lemma 2

Let $\mathscr{P}=(G, s, \mathscr{F}, g)$ be an abstract theorem-proving problem and let the heuristic function h for $\mathscr{P}$ satisfy the lower bound condition.

(a) If $n^* \in \mathscr{F}$ then $h(n^*)=h^*(n^*)=0$ and therefore $f(n^*)=g(n^*)$.
(b) If $n^* \in \mathscr{F}$ and $n \leqslant n^*$ then $f(n) \leqslant f(n^*)$.
(c) If $\preccurlyeq_d$ is δ-finite then for some $n^* \in \mathscr{F}$ $g(n^*) \leqslant g(n)$ for all $n \in \mathscr{F}$, provided $\mathscr{F} \neq \varnothing$.

Proof. (a) is obvious, since $H(n^*)=\{0\}$ and $h^*(n^*)=0$.
(b) If $n^* \in \mathscr{F}$ and $n \leqslant n^*$ then $h(n) \leqslant g(n^*)-g(n)$.
But then $f(n)=g(n)+h(n) \leqslant g(n^*)=f(n^*)$.
(c) If $\preccurlyeq_d$ is δ-finite then for all $n \in G$, the set $\{n' \mid f(n') \leqslant f(n), n' \in G\}$ is finite. In particular for $n \in \mathscr{F}$ the set $\{n' \mid g(n') \leqslant g(n), n' \in \mathscr{F}\}$ is finite and therefore contains an element n^* such that $g(n^*)$ is minimal. But then $g(n^*) \leqslant g(n')$ for all $n' \in \mathscr{F}$.

Theorem 2

If $\preccurlyeq_d$ is δ-finite for $\mathscr{P}=(G, s, \mathscr{F}, g)$ and if h satisfies the lower bound condition for $\mathscr{P}$ then $\Sigma \in \mathscr{D}(\mathscr{P}, h)$ is admissible for $\mathscr{P}$.

Proof. Assume that $\mathscr{F} \neq \phi$. Let $n^* \in \mathscr{F}$ be such that $g(n^*) \leqslant g(n)$ for all $n \in \mathscr{F}$ (such an $n^* \in \mathscr{F}$ exists by Lemma 2(c)). By Theorem 1, Σ is complete and therefore there is a stage i such that for some n,

$$n \in \mathscr{F} \cap \Sigma_i \quad \text{and} \quad \mathscr{F} \cap \Sigma_{i-1} = \varnothing.$$

Suppose that Σ is not admissible for $\mathscr{P}$. Then $g(n^*) < g(n)$. But, by Lemma 2, for all $n' \leqslant n^*$, $f(n') \leqslant f(n^*) = g(n^*) < f(n)$. So $f(n') < f(n)$ for all $n' \leqslant n^*$. But then $n' \prec n$ for all $n' \leqslant n^*$. By Lemma 1 (a), $n^* \in \Sigma_{i-1}$ and therefore $\mathscr{F} \cap \Sigma_{i-1} \neq \varnothing$, contrary to assumption.

Theorem 2 specializes to a generalization of Theorem 1 in Hart, Nilsson and Raphael (1968) when $s(G')=\varnothing$ for all $G' \subseteq G$ which are not singletons. In particular it is not necessary to require that $\mathscr{S}_0$ be finite or that $g(n)$ be strictly greater than $g(n')$ whenever $n' < n$. Since the specialization yields a tree representation of graph search, it is unnecessary to distinguish between the cost $g(n)$ and the total cost along some minimal path to n.

Figure 4 illustrates Lemma 2 and Theorem 2. $\mathscr{P}$, Σ, n^*, n', n and n'' are as in figure 3, but h satisfies the lower bound condition. By Lemma 2, n' lies on the same diagonal d as does n^*. Σ is admissible since any $n^{**} \in \mathscr{F}$ having

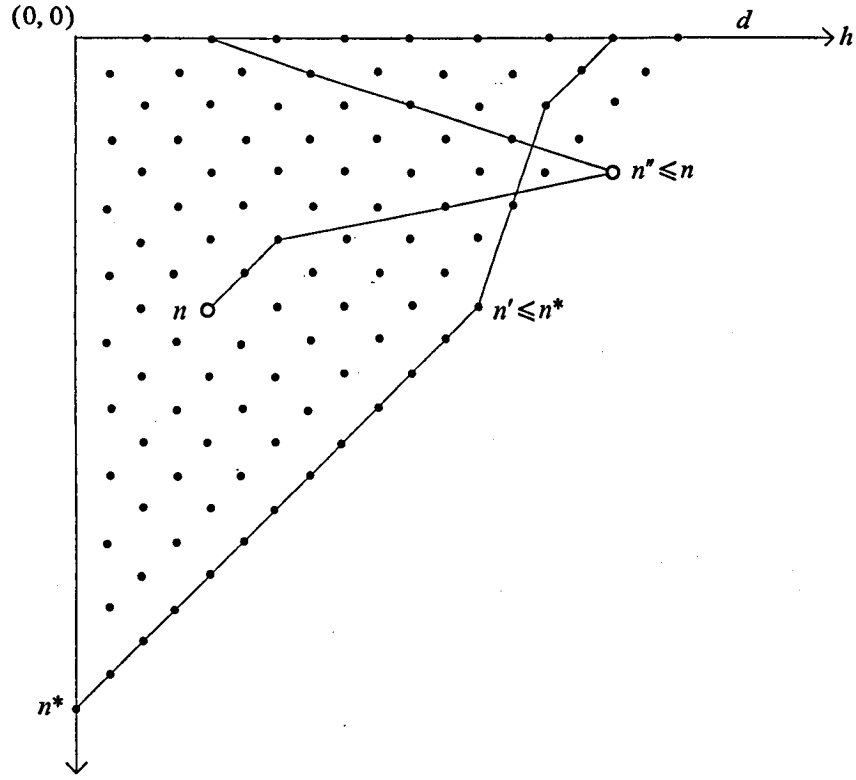

Figure 4

worse merit than n^* lies on a diagonal to the right of d and is not generated before n^*. It is still possible for a node $n \in G$ to have better merit than n^* and not be generated before n^* because n'' has worse merit than n'.

To prove the appropriate extension of the Hart–Nilsson–Raphael Theorem on the optimality of $\Sigma \in \mathscr{D}^u$, we need to formulate an assumption equivalent to their 'consistency assumption'. The reader familiar with Hart, Nilsson and Raphael (1968) will easily convince himself that the following condition is equivalent to the consistency assumption. We say that the evaluation function f satisfies the *monotonicity condition* if

$$f(n') \leqslant f(n) \text{ for } n' \leqslant n \text{ and}$$
$$f(n^*) = g(n^*) \text{ for } n^* \in \mathscr{F}.$$

(The first condition is equivalent to

$$h(n) \geqslant h(n') - (g(n) - g(n')) \text{ for } n' \leqslant n.)$$

Notice that for $\mathscr{P}_2$ the evaluation function $f_2 = g_2 + h_2$ satisfies the monotonicity condition whereas for $\mathscr{P}_1$ the function $f_1 = g_1 + h_1$ is monotonic except for merging.

Figure 5 illustrates upwards diagonal search when the function f satisfies the monotonicity condition. $\mathscr{P}$, Σ, n^*, n', n and n'' are as in figures 3 and 4. By Lemma 3, h satisfies the lower bound condition and therefore Σ is admissible and n' lies on the same diagonal as n^*. The monotonicity condition implies that if n has better diagonal merit than n^* then all ancestors of n have better merit than n^* and therefore, by Lemma 1, n is generated before n^*.

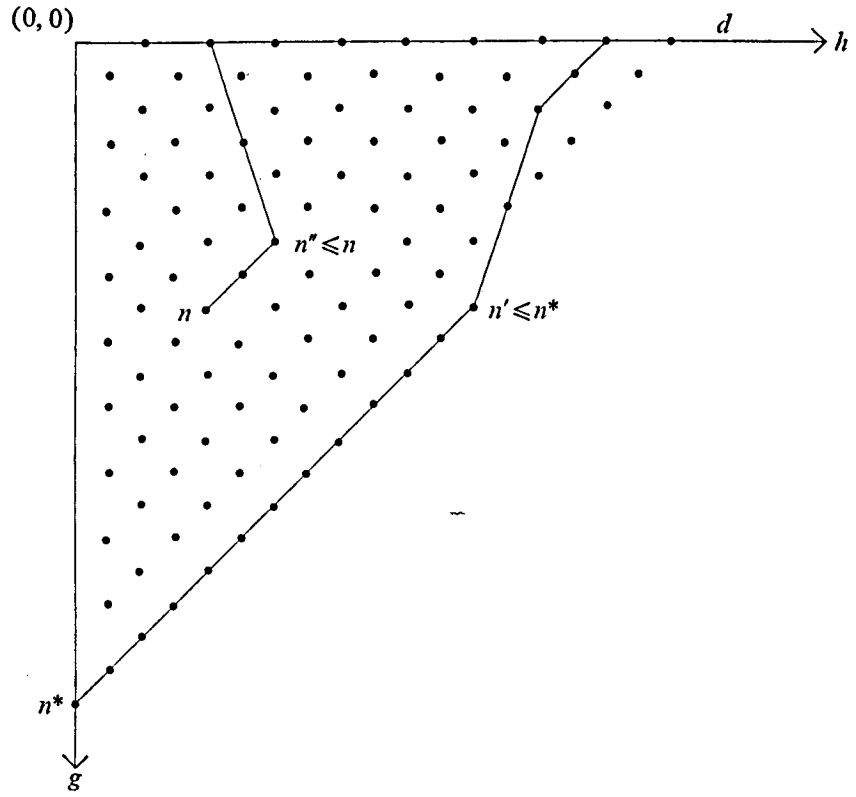

Figure 5

Lemma 3

Let $\mathscr{P}=(G, s, \mathscr{T}, g)$ be an abstract theorem-proving problem, let h be a heuristic function for $\mathscr{P}$, and let f satisfy the monotonicity condition, where $f(n)=g(n)+h(n)$, $n \in G$. Then

(a) h satisfies the lower bound condition,
(b) if $\Sigma \in \mathscr{D}(\mathscr{P}, h)$, $n_1 \in \Sigma_i$ and $n_2 \in \Sigma(\Sigma_i)$ then $f(n_1) \leqslant f(n_2)$.

Proof. (a) h satisfies the lower bound condition if $h(n) \leqslant g(n^*)-g(n)$ whenever $n^* \in \mathscr{T}$ and $n \leqslant n^*$.

But monotonicity of f implies that

$$f(n)=g(n)+h(n)\leqslant f(n^*)=g(n^*).$$

So $\quad h(n)\leqslant g(n^*)-g(n).$

(b) Suppose the contrary, namely that $n_1\in\Sigma_i$, $n_2\in\Sigma(\Sigma_i)$ and $f(n_1)>f(n_2)$. But then, since $f(n')\leqslant f(n_2)<f(n_1)$ for all $n'\leqslant n_2$, it follows that $n'\prec n_1$ for all $n'\leqslant n_2$. By Lemma 1(a), $n_2\in\Sigma_{i-1}$, contradicting the assumption that $n_2\in\Sigma(\Sigma_i)$.

For the case of ordinary graphs, the optimality theorem (Theorem 2) of Hart, Nilsson and Raphael (1968) compares, in effect, search strategies $\Sigma\in\mathscr{D}(\mathscr{P},h)$ with strategies $\Sigma'\in\mathscr{D}(\mathscr{P},h')$ where $h'(n)\leqslant h(n)$ for all $n\in G$ and where $f=g+h$ is monotonic. [In Hart, Nilsson and Raphael (1968) the search strategy Σ' is assumed only to be 'no better informed' than Σ – we interpret this to mean that $h'(n)\leqslant h(n)$ and $\Sigma'\in\mathscr{D}(\mathscr{P},h)$.] If Σ_i and $\Sigma'_{i'}$ are the first sets which contain nodes $n^*\in\mathscr{F}$ then $\Sigma_i\subseteq\Sigma'_{i'}\cup G'$ where G' is the set of nodes $n\in\Sigma_i$ which have diagonal merit equal to $n^*\in\Sigma_i\cap\mathscr{F}$, i.e., before termination Σ' generates all the nodes generated by Σ except possibly for unlucky choices by Σ of nodes tied for merit with the solution node $n^*\in\Sigma_i$. Theorem 3 below generalizes Theorem 2 of Hart, Nilsson and Raphael (1968) and implies in addition that $\mathscr{D}^u$ is an optimal subclass of $\mathscr{D}$.

It should be noted that the monotonicity condition on f in Theorem 3 can be replaced by the lower bound condition on h with the result that Σ' may now fail to generate nodes in the larger set G' of nodes $n\in\Sigma_i$ where some $n''\leqslant n$ has diagonal merit tied with the solution node $n^*\in\Sigma_i$. A special case of this modification of Theorem 3 is illustrated by the example of figure 6, following the proof of Theorem 3.

Theorem 3

Let $\mathscr{P}=(G,s,\mathscr{F},g)$ and let h and h' be heuristic functions for $\mathscr{P}$ such that

$$h'(n)\leqslant h(n) \text{ for } n\in G.$$

Let $f(n)=g(n)+h(n)$ and $f'(n)=g(n)+h'(n)$. Suppose that f is monotonic. Given $\Sigma\in\mathscr{D}^u(\mathscr{P},h)$ and $\Sigma'\in\mathscr{D}(\mathscr{P},h')$, suppose that

$$n_1\in\mathscr{F}\cap\Sigma_i,\ \mathscr{F}\cap\Sigma_{i-1}=\varnothing,$$
$$n_2\in\mathscr{F}\cap\Sigma'_{i'} \text{ and } \mathscr{F}\cap\Sigma'_{i'-1}=\varnothing.$$

Then $\Sigma_i\subseteq\Sigma'_{i'}\cup G^*$ where

$$G^*=\{n\mid n\in\Sigma_i \text{ and for some } n'\leqslant n_1,\ f(n)=f(n')=f(n_1) \text{ and } h(n)\leqslant h(n')\}.$$

Proof. Σ' satisfies the lower bound condition since $h'(n)\leqslant h(n)$ for all $n\in G$ and since Σ satisfies the lower bound condition. Therefore both Σ and Σ' are admissible and $g(n_1)=g(n_2)$, $f(n_1)=f(n_2)$. Suppose that $n\in\Sigma_i$ and that $n\notin\Sigma'_{i'}$. It suffices to show that $n\in G^*$.

By Lemma 1(b), $n\in\Sigma_i$ implies that $n\leqslant_{dw}n'$ for some $n'\leqslant n_1$. But by Lemma 3(b), since f is monotonic

$f(n) \leqslant f(n_1)$,
$f(n') \leqslant f(n_1)$,
$f(n'') \leqslant f(n)$ for all $n'' \leqslant n$.

But $h'(n'') \leqslant h(n'')$ implies

$f'(n'') \leqslant f(n'')$. So
$f'(n'') \leqslant f(n)$ for all $n'' \leqslant n$.

Also $n \notin \Sigma_{i'}$ and $n_2 \in \Sigma_{i'}$ imply by Lemma 1(a) that for some $n'' \leqslant n$, $n'' \geqslant_d n_2$, i.e.,

$f'(n'') \geqslant f'(n_2) = f(n_1)$. So
$f(n) \geqslant f(n_1)$ and
$f(n) = f(n_1)$.

$n \leqslant_{au} n'$ implies

$f(n) \leqslant f(n') \leqslant f(n_1)$. So
$f(n) = f(n') = f(n_1)$ and
$h(n) \leqslant h(n')$, i.e.,
$n \in G^*$.

Figure 6 compares nodes generated, before the generation of a given $n^* \in \mathscr{F}$, by different search strategies $\Sigma_i \in \mathscr{D}(\mathscr{P}, h_i)$ for a fixed problem $\mathscr{P} = (G, s, \mathscr{F}, g)$ and for different heuristic functions h_i. $h_1(n)$ is assumed to be a greatest lower bound on the value of $h^*(n)$ when $n \leqslant n^*$, where n^* has least cost in $\mathscr{F}$. Nodes $n \in G$ are represented as points with co-ordinates $(h_1(n), g(n))$. The node n' has worst upwards diagonal merit in the set consisting

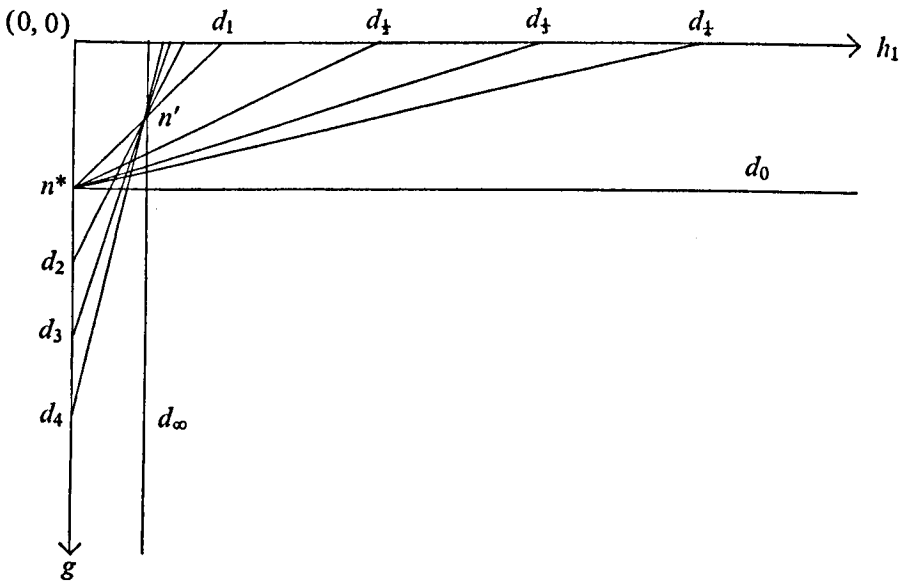

Figure 6

of n^* and the ancestors of n^*. The functions h_i are defined by $h_i(n) = ih_1(n)$ for all $n \in G$, $0 \leqslant i \in \mathbb{R}$.

For $0 \leqslant i \leqslant 1$, h_i satisfies the lower bound condition for $\mathscr{P}$ and Σ_i is admissible for $\mathscr{P}$. Σ_i need not be admissible for $\mathscr{P}$ when $i > 1$. The area to the left of the line d_i contains nodes generated by Σ_i before the generation of n^*. For $0 \leqslant i \leqslant 1$, Σ_i generates all the nodes generated by Σ_1. For $i > 1$, Σ_i generates all the nodes left of d_i which have been generated by Σ_1. No Σ_i is more efficient than Σ_1, if $i > 1$. Some Σ_i may generate fewer nodes than Σ_1, if $i > 1$, but this possibility becomes more remote as i increases. However even for large i, Σ_i may be more efficient than Σ_1 for generating solution nodes of arbitrary cost. A more thorough analysis of relationships similar to those discussed here has been made by Ira Pohl (1969, 1970).

Acknowledgements

The author wishes to acknowledge helpful discussions with Dr Bernard Meltzer, Dr Ira Pohl, Miss Isobel Smith, Mr Pat Hayes and Mr Donald Kuehner. Special thanks are due to Miss Isobel Smith for implementing diagonal search for resolution problems and to Dr Nils Nilsson and Mr Donald Kuehner for suggestions made for improving an earlier draft of this paper.

This research was supported by an IBM fellowship and grant from Imperial College, and more recently by the Science Research Council.

REFERENCES

Burstall, R.M. (1968) A scheme for indexing and retrieving clauses for a resolution theorem-prover. *Memorandum* MIP-R-45. University of Edinburgh: Department of Machine Intelligence and Perception.

Doran, J. & Michie, D. (1966) Experiments with the graph traverser program. *Proceedings of the Royal Society (A)*, **294**, 235–59.

Green, C. (1969a) Theorem-proving by resolution as a basis for question-answering systems. *Machine Intelligence 4*, pp. 183–205 (eds Meltzer, B. & Michie, D.). Edinburgh: Edinburgh University Press.

Green, C. (1969b) The application of theorem-proving to question-answering systems Ph.D. thesis. Stanford University.

Hart, P.E., Nilsson, N.J. & Raphael, B. (1968) A formal basis for the heuristic determination of minimum cost paths. *IEEE Trans. on Sys. Sci. and Cybernetics*, July 1968.

Kowalski, R. & Hayes, P.J. (1969) Semantic trees in automatic theorem-proving. *Machine Intelligence 4*, pp. 87–101 (eds Meltzer, B. & Michie, D.). Edinburgh: Edinburgh University Press.

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

Nilsson, N.J. (1968) Searching problem-solving and game-playing trees for minimal cost solutions. *IFIPS Congress preprints*, H125–H130.

Pohl, I. (1969) Bi-directional and heuristic search in path problems. Ph.D. thesis. Stanford University.

Pohl, I. (1970) First results on the effect of error in heuristic search. *Machine Intelligence 5*, pp. 219–36 (eds Meltzer, B. & Michie, D.). Edinburgh: Edinburgh University Press

Robinson, J.A. (1965) A machine-oriented logic based on the resolution principle. *J. Ass. comput. Mach.*, **12**, 23–41.

Robinson, J.A. (1967) A review of automatic theorem-proving. *Annual symposia in applied mathematics XIX*. Providence, Rhode Island: American Mathematical Society.

Sandewall, E. (1969) Concepts and methods for heuristic search. *Proc. of the International Joint Conference on Artificial Intelligence*, pp. 199–218 (eds. Walker, D. E. & Norton, L. N.).

Sibert, E.E. (1969) A machine-oriented logic incorporating the equality relation. *Machine Intelligence 4*, pp .103–33 (eds Meltzer, B. & Michie, D.). Edinburgh: Edinburgh University Press.

Wos, L.T., Carson, D.F. & Robinson, G.A. (1964) The unit preference strategy in theorem-proving. *AFIPS*. **25**, 615–21, Fall, J.C.C. Washington, D.C.: Spartan Books.

Wos, L.T., Carson, D.F. & Robinson, G.A. (1965) Efficiency and completeness of the set-of-support strategy in theorem-proving. *J. Ass. comput. Mach.*, **12**, 536–41.