A NONCLAUSAL CONNECTION-GRAPH RESOLUTION THEOREM-PROVING PROGRAM

Mark E. Stickel

Artificial Intelligence Center
SRI International, Menlo Park, CA 94025

ABSTRACT

A new theorem-proving program, combining the use of nonclausal resolution and connection graphs, is described. The use of nonclausal resolution as the inference system eliminates some of the redundancy and unreadability of clause-based systems. The use of a connection graph restricts the search space and facilitates graph searching for efficient deduction.

I INTRODUCTION

This paper describes some of the theory and features of a nonclausal connection-graph resolution theorem-proving program being developed as a reasoning component of a natural-language-understanding system.

The most important characteristics of the program are

- Nonclausal resolution is used as the inference system, eliminating some of the redundancy and unreadability of clause-based systems.
- A connection graph is used to represent permitted resolution operations, restricting the search space and facilitating the use of graph searching for efficient deduction.
- Heuristic search and special logical connectives are used for program control.

The following sections will describe these aspects of the program, citing disadvantages and difficulties as well as advantages, and will be followed by a description of the implementation status of the program and future plans for it.

II NONCLAUSAL RESOLUTION

One of the most widely criticized aspects of resolution theorem proving is its use of clause form for wffs. The principal criticisms are

- Conversion of a wff to clause form may eliminate pragmatically useful information encoded in the choice of logical connectives (e.g., $\neg P \vee Q$ may suggest case analysis while the logically equivalent $P \supset Q$ may suggest chaining).
- Use of clause form may result in a large number of clauses being needed to represent a wff, as well as in substantial redundancy in the resolution search space.
- Clause form is difficult to read and not human-oriented.

The clausal resolution rule can be easily extended to general quantifier-free wffs [12,8]. Proofs of soundness and completeness are in [12]. Where clausal resolution resolves on clauses containing complementary literals, nonclausal resolution resolves on general quantifier-free wffs containing atomic wffs (atoms) occurring with opposite *polarity*, which is determined by the parity of the number of explicit or implicit negations in whose scope the atom appears (positive polarity if even, negative polarity if odd). In clausal resolution, resolved-on literals are deleted and remaining literals disjoined to form the resolvent. In nonclausal resolution, all occurrences of the resolved-on atom are replaced by **F**=false (**T**=true) in the wff in which it occurs positively (negatively). The resulting wffs are disjoined and simplified by truth-functional reductions that eliminate embedded occurrences of **T** and **F** and optionally perform simplifications such as $A \wedge \neg A \rightarrow \mathbf{F}$.

Definition 1. If A and B are ground wffs and C is an atom occurring positively in A and negatively in B, then the result of simplifying $A(C \leftarrow \mathbf{F}) \vee B(C \leftarrow \mathbf{T})$, where $X(Y \leftarrow Z)$ is the result of replacing every occurrence of Y in X by Z, is a *ground nonclausal resolvent* of A and B.

It is clear that nonclausal resolution reduces to clausal resolution when the wffs are restricted to be clauses. In the general case, however, nonclausal resolution has some novel characteristics as compared with clausal resolution. It is possible to derive more than one resolvent from the same pair of wffs, even resolving on the same atom, if the atom occurs both positively and negatively in both wffs (e.g., atoms within the scope of an equivalence occur both positively and negatively). Likewise, it is possible to resolve a wff against itself.

The ground nonclausal resolution rule can be lifted to nonground wffs by renaming parent wffs apart and unifying sets of atoms from each parent, one atom of each set occurring positively in the first wff and negatively in the second. As with clausal resolution, only single atoms need be resolved upon if the resolution operation is augmented by a factorization operation that derives a new wff by instantiating a wff by a most general unifier of two or more distinct atoms occurring in the wff (regardless of polarity).

This research was supported by the Defense Advanced Research Projects Agency with the Naval Electronic Systems Command under contract N00039-80-C-0575. The views and conclusions contained in this document are those of the author and should not be interpreted as necessarily representing the official policies, either expressed or implied, of the Defense Advanced Research Projects Agency of the United States Government.

A nonclausal resolution derivation of **F** from a set of wffs demonstrates the unsatisfiability of the set of wffs. Nonclausal resolution is thus, like clausal resolution, a refutation procedure. Variants of the procedure that attempt to affirm rather than refute a wff are possible (e.g., see the variety of resolution rules in [8]), but are isomorphic to this procedure.

Although clause form is often criticized, use of nonclausal form has the disadvantage that most operations on nonclausal form are more complex than the same operations on clause form. The result of a nonclausal resolution operation is less predictable than the result of a clausal resolution operation. Clauses can be represented as lists of literals; sublists are appended to form the resolvent. Pointers can be used to share lists of literals between parent and resolvent [5]. With many simplifications such as $A \wedge \mathbf{T} \rightarrow A$ and $A \wedge \neg A \rightarrow \mathbf{F}$ being applied during the formation of a nonclausal resolvent, the appearance of a resolvent may differ substantially from its parents, making structure sharing more difficult.

For most forms of clausal resolution, an atom does not occur more than once in a clause. In nonclausal resolution, an atom may occur any number of times, with possibly differing polarity. In clausal resolution, every literal in the clause must be resolved upon for the clause to participate in a refutation. Thus if a clause contains a literal that is pure (cannot be resolved with a literal in any other clause), the clause can be deleted. This is not the case with nonclausal resolution; not all atom occurrences are essential in the sense that they must be resolved upon to participate in a refutation. For example, $\{P \wedge Q, \neg Q\}$ is a minimally inconsistent set of wffs, one of which contains the pure atom P. A more complicated definition of purity involving this notion of essential occurrences must be used. The subsumption operation must also be redefined for nonclausal resolution to take account of such facts as the subsumption of A by $A \wedge B$ as well as the clausal subsumption of $A \vee B$ by A.

[12,8] suggest the extension of nonclausal resolution to resolving on nonatomic subwffs of pairs of wffs. For example, $P \vee Q$ and $(P \vee Q) \supset R$ could be resolved to obtain R. Resolving on nonatoms often permits significantly shorter and more readable refutations. However, there are several reasons for not doing this:

- It may be difficult to recognize complementary wffs. For example, $P \vee Q$ occurs positively in $Q \vee R \vee P$ and $\neg P \supset Q$.
- The effect of resolving a pair of wffs on nonatomic subwffs can be achieved by multiple resolution operations on atoms. Resolution on both atomic and nonatomic subwffs could result in redundant derivations.
- A connection-graph procedure would be complicated by the need to attach links to logical subwffs (e.g., $P \vee Q$ in $Q \vee R \vee P$) and link inheritance would be further complicated since subwffs of a resolvent may have no parent subwffs (e.g., when $P \vee Q$ and $\neg P \vee R$ are resolved, the resolvent $Q \vee R$ is a subwff of neither parent). Similar complications arise if equality inferences are used that introduce new structure into the result.

Although the nonclausal resolution rule in general seems adequate as compared with the above proposed extension to matching on nonatomic subwffs, the handling of the equivalence relation in [12] is inadequate. In resolving $P \equiv Q$ and $(P \wedge R) \vee (\neg P \wedge S)$, it is possible to derive $Q \vee S$ and $\neg Q \vee R$, but not the more natural result of simply replacing P by Q. It is questionable whether handling the equivalence relation in nonclausal resolution without further extension is worthwhile in comparison with the representational advantages of negation normal form used in [1,2]. Another difficulty with the equivalence relation is that it sometimes needs to be removed during Skolemization. [9] provides extensions to nonclausal resolution that defer Skolemization and permit equivalence relations to be retained longer.

III CONNECTION GRAPHS

Connection-graph resolution was introduced in [7]. It has the following advantages:

- The connection-graph refinement is quite restrictive. Many resolution operations permitted by other resolution procedures are not permitted by connection-graph resolution.
- The links associated with each wff function partially as indexing of the wffs. Effort is not wasted in the theorem prover examining the entire set of wffs for wffs that can be resolved against newly derived wffs.
- Links can be traversed by a graph-searching algorithm whereby each link traversal denotes a resolution operation. This can be done to plan a deduction without actually constructing it. This graph searching may resemble the searching performed for deduction in knowledge representation languages.

Connection-graph resolution is extended in a natural way to use the nonclausal resolution inference rule.

A connection graph is a set of wffs and a set of links that connect atoms occurring with positive polarity in one wff and negative polarity in the same or another wff. Performing the nonclausal resolution operation indicated by the link results in the production of a new connection graph with the resolved upon link eliminated and the nonclausal resolvent added. Roughly speaking, atoms of the nonclausal resolvent are linked only to atoms to which atoms of the parent wffs were linked.

Definition 2. Let S be a set of ground wffs. Let L be

$$\{\langle C, A, B\rangle \mid A, B \in S,$$
$$\textit{atom } C \textit{ occurs positively in } A \textit{ and negatively in } B\}.$$

Then $\langle S, L\rangle$ is the ***full connection graph*** for S.

Definition 3. Let S be a set of ground wffs and L be its connection graph. Let $\ell = \langle C, A, B\rangle$ be an element of L and C be the nonclausal resolvent $A(C \leftarrow \mathbf{F}) \vee B(C \leftarrow \mathbf{T})$. Let S' be $S \cup \{C\}$. Let L' be

$\mathcal{L} - \{\ell\}$
$\cup\{\langle E, C, D\rangle \mid$ *atom* E *occurs positively in* C *and* $\langle E, A, D\rangle \in \mathcal{L}$ *or* $\langle E, B, D\rangle \in \mathcal{L}\}$
$\cup\{\langle E, D, C\rangle \mid$ *atom* E *occurs negatively in* C *and* $\langle E, D, A\rangle \in \mathcal{L}$ *or* $\langle E, D, B\rangle \in \mathcal{L}\}$
$\cup\{\langle E, C, C\rangle \mid$ *atom* E *occurs positively and negatively in* C *and* $\langle E, A, A\rangle \in \mathcal{L}, \langle E, B, B\rangle \in \mathcal{L}, \langle E, A, B\rangle \in \mathcal{L}$, *or* $\langle E, B, A\rangle \in \mathcal{L}\}$.

Then the connection graph $(\mathcal{S}', \mathcal{L}')$ is derived from $(\mathcal{S}, \mathcal{L})$ by *ground nonclausal connection-graph resolution.*

A nonclausal connection-graph resolution refutation of an input set of wffs is a derivation of a set of wffs including **F** by nonclausal connection-graph resolution from the full connection graph of the input set of wffs.

Ground nonclausal connection-graph resolution can be extended to the nonground case by including in the links the unifier of the atoms they connect, keeping wffs renamed apart, and by including links between variants of the same wff (to allow a wff to directly or indirectly resolve against a variant of itself). Factorization must also be included. Either factors with appropriately inherited links must be added for each wff in the connection graph or special factor links can be used with link inheritance rules for both resolve and factor links after resolution and factorization operations.

The nonclausal connection-graph resolution procedure is sound and there is reason to believe it is complete. However, it has not yet been proved to be complete, and the history of proving completeness of connection-graph procedures for the simpler clausal case (see [2]) suggests it may be difficult.

One reason it is difficult to prove the completeness of the connection-graph procedure is that the link inheritance rules exclude some links that would be present if the connection graph were merely an encoding of all permitted resolution operations for ordinary resolution. Exactly which links are excluded depends on the order in which resolution operations are performed. The effect of connection-graph resolution is to impose the following restriction: if a pair of atoms in a pair of wffs is resolved upon, atoms derived (in later resolution operations) from the resolved-on atoms cannot be resolved against each other. For example, if a set of wffs includes $P \vee Q$ and $\neg P \vee \neg Q$, these two wffs can be resolved upon P and Q—resulting in tautologies that are discarded; after that, neither wff can be resolved with an atom descended from the other, even though doing so would not result in a tautology.

Connection-graph resolution procedures can possibly be incomplete by succeeding in finding refutations when links are resolved upon in some orders, but not others. For example, consider the combination of linear resolution and connection-graph resolution for clauses. Each is complete, but the combination is not. If linear connection-graph resolution is applied to $\{P \vee \neg Q, \neg P \vee \neg Q, Q\}$ with Q as top clause, depth-first search will find a refutation, but breadth-first search will not. This contrasts with the usual situation in which breadth-first search is "safe", always guaranteed to find a refutation if there is one. To see that it fails in this case, observe that after P and $\neg P$ are generated on the first level of breadth-first search, Q and $\neg Q$ have no links—and thus none of the three input clauses can be further resolved upon to lead to a refutation. P and $\neg P$ are linked, but cannot be resolved without violating the linear-resolution restriction.

A set of assertions in a connection graph can to some extent be regarded and treated as a semantic network—more so than the same set of assertions without the connection graph.

For example, the full connection graph for

$$elephant(Clyde)$$
$$elephant(x) \supset mammal(x)$$
$$elephant(y) \supset color(y, gray)$$
$$mammal(z) \supset animal(z)$$

would contain links between the following pairs of atoms

ℓ_1. $\langle elephant(Clyde), elephant(x)\rangle$
ℓ_2. $\langle elephant(Cldye), elephant(y)\rangle$
ℓ_3. $\langle mammal(x), mammal(z)\rangle$.

Answers to such queries as "What color is Clyde?" and "Is Clyde an animal?" can be found by graph searching with minimal analysis of the assertions, by traversing the links in the connection graph. Such searching can be made more efficient by labeling the links (e.g., *isa* for ℓ_1 and ℓ_3, *hascolor* for ℓ_2). The semantic content of the set of assertions is still conveyed by the assertions themselves, but control information is provided to a graph-searching procedure by the link labels.

Similar comments could be made regarding any logical representation. However, the use of a connection graph in which all permissible remaining resolution operations are encoded in explicit links can yield greater efficiency by eliminating traversal of multiple paths to the same goal. For example, suppose ℓ_3 is resolved upon, resulting in the added assertion

$$elephant(w) \supset animal(w)$$

and the added link

ℓ_4. $\langle elephant(Clyde), elephant(w)\rangle$.

The link ℓ_3 is deleted. There is still only one path or proof that Clyde is an animal, since the absence of ℓ_3 blocks the path or proof $elephant(Clyde) \rightarrow mammal(Clyde) \rightarrow animal(Clyde)$.

Graph searching in the connection graph to determine taxonomic relations quickly is a simple illustration of the more general notion, extensively explored in [1,13], of using graph searching to determine the existence of refutations. The ideas and techniques developed there are applicable to nonclausal connection-graph resolution. Connection-graph resolution appears to offer the following advantages over these other schemes:

- Although graph searching can be done in the connection-graph resolution procedure, [1,13] do not allow for the actual formation of resolvents. If their techniques for graph search were adopted as a device for planning or quick refutation, connection-graph resolution could be regarded as a superset of these other methods.
- The actual formation of resolvents and the resulting change in the connection graph are useful for retaining

information during a refutation, as well as for conveying information (about usage of wffs, etc.) from one refutation or assertion to the next. (Here it is assumed that the theorem prover is being used with an assertional database to which queries are posed and assertions occasionally added and deleted, as opposed to the usual situation in theorem proving in which there is no persistent assertional database, all axioms being presented anew for each proof.)

- Connection-graph resolution provides a convenient, albeit unsophisticated, means of interleaving matching complementary literals and adding new instances of assertions (if more than one ground instance of a wff is required), as compared with the separate processes of searching for a mating, and quantifier duplication if the search fails [1].

Of course, the argument in favor of performing only graph searching as in [1,13] is that forming resolvents is expensive compared to traversing links, and the cost of creating and storing inherited links may be high.

A good system will probably have a mixture of resolution and graph searching, as in [4] for clausal connection-graph resolution. Graph searching is used in that system for look-ahead and to determine if a refutation exists within a certain number of steps. Simple graph searching is used (e.g., not looking for refutations in which wffs occur more than once), with the full complexity and completeness of connection-graph resolution in the background.

One problem with graph searching to find refutations is in assessing the effectiveness of the procedure. In ordinary resolution theorem proving, effectiveness can be evaluated in part by examining the number of clauses generated, retained, used in the refutation, and so forth. [4] states "Within this frame of reference it would be easy to design the 'perfect' proof procedure: the supervisor and the look-ahead heuristics would find the proof and then guide the system without any unnecessary steps through the search space." The amount of time used is a good measure for such a program, but should not be used to compare programs as there may be differences in the machines the programs run on and in the efficiency of the programs themselves (as opposed to the algorithms). In general, as [4] states, a measure incorporating both total time and space will be required, adding the further complication of evaluating time-space trade-offs.

IV CONTROL

A link scheduler is used to specify a refutation search strategy. When an assertion is added by the user, it is linked in the connection graph to all previous assertions. When a resolvent is added, it is linked to the other assertions according to the link inheritance rules. All such added links are examined by the link scheduler. Three outcomes are possible:

- The link is deleted. For example, analysis may show that resolving on the link would create a tautology or pure wff that could not be used in a refutation, whereupon the link can be deleted.
- The link is retained, but not scheduled. Thus the link can be inherited, but cannot be resolved upon (though its descendants might be). This is done when combining connection-graph resolution with other refinements of resolution, such as ordering restrictions and the special logical-connective restrictions described below.
- The link is scheduled. It is given a numerical score and placed in the link schedule. The theorem prover operates by repeatedly resolving on the best scored link in the schedule, creating the resolvent, and scheduling the added links.

Scheduling of the links is done after all the new links have been added, so that the link scheduler can act on such important facts as the number of links attached to an atom.

Special logical connectives can be used to impose restrictions on the use of particular assertions. As in [10], the following connectives denote the following procedural interpretations of $A \supset B$:

- $A \rightarrow B$. If literal A is ever asserted, assert B (forward chaining).
- $B \leftarrow A$. To prove literal B, try to prove A (backward chaining). Since a refutation procedure is being used, this is interpreted as "permit the resolution, on literal B, between $A \supset B$ and any wff having support."
- $A \Rightarrow B$. If literal A is ever asserted, also assert B and, to prove $\neg A$, try to prove $\neg B$.
- $B \Leftarrow A$. To prove literal B, try to prove A and, if $\neg B$ is ever asserted, also assert $\neg A$.
- $A \supset B$ and $\neg A \vee B$. Unrestricted and equivalent.

The use of both nonclausal resolution and these special logical connectives gives this program some resemblance to natural deduction [3]. It represents an intermediate point between clausal resolution and natural deduction, with advantages of each. It differs from natural deduction, since, for example, a backward-chaining application of $A \supset B$ to C would result in $\neg A \vee C(B \leftarrow \mathbf{T})$ rather than $C(B \leftarrow A)$ (with perhaps only a single instance of B replaced, requiring additional operations to replace the other occurrences). The latter expression may be more natural, but the former is more concise because all occurrences of B are eliminated and only a single instance of A is added. Heuristic search is used in a manner similar to the way it is employed in a clausal system [14] and in a natural-deduction system [15].

V STATUS AND FUTURE PLANS

The theorem-proving program is implemented as a 4000 line INTERLISP program and is presently being used as the deduction component of the MICROKLAUS natural-language-understanding system. Natural-language assertions and queries are translated by the DIALOGIC system [6] into logical form [11]. This logical form is further translated into predicate calculus for input to the theorem prover. The allowance for predicate variables extends the program slightly beyond ordinary first-order predicate calculus. Future work will expand the range of logical form handled, as not all logical forms that can be generated by DIALOGIC are presently being translated; the range of logical form generated by DIALOGIC is also being expanded).

Besides the unification filtering provided by the connection graph, atoms in assertions are indexed by predicate symbol so as to speed the addition to the connection graph of user input assertions when there is a large number of them. Wffs are also indexed by their propositional structure and predicate symbols to speed checking for alphabetic-commutative variants to be eliminated. More efficient indexing schemes will probably be tried and variant elimination replaced by subsumption.

Factorization has not yet been implemented in the program. When two wffs are resolved upon a pair of atoms, all atoms instantiated to be the same as the instantiated resolved-on atoms are replaced by **F** or **T**, but there is no effort to force additional atoms, by further instantiation, to be the same as the resolved-on atoms. Thus, only "obvious" factors are used. This is incomplete, but effective. Factor links will be added for completeness.

So far, only fairly simple evaluation functions have been used in the search control process. They are similar to those used in [14], being weighted sums of the deduction depth of the wff (a measure of the effort required to derive the wff) and the number of atoms in the wff (a measure of the additional effort that will be needed to complete a refutation using the wff). Performance is generally superior to that in [14]. In ordering restrictions, atoms are also evaluated according to how many links are connected to them, so that atoms with fewer links can be resolved upon preferentially. Not only is the immediate branching factor reduced, but there is also the prospect that the other atoms with more links will be instantiated and inherit fewer links when the resolution operation is performed. Interestingly, as was also noted in [14], there can be negative interactions among individually good ideas on search control. For example, a strong length-preference strategy and the strategy of resolving on an atom with the fewest links are somewhat inconsistent. When there are many assertions about some predicate—some short and specific, others long and general—the atom with the fewest links is likely to be linked only to long and general assertions. Resolving on it thus may result in long resolvents that would be given low preference by a strong-length preference strategy. More work will be done on developing good evaluation functions. The most important extension of the program will be further development of connection-graph searching—both to provide input to evaluation functions and to perform entire deductions without creating any resolvents.

REFERENCES

[1] Andrews, P.B. Theorem proving via general matings. *J. ACM 28,* 2 (April 1981), 193–214.

[2] Bibel, W. On matrices with connections. *J. ACM 28,* 4 (October 1981), 633–645.

[3] Bledsoe, W.W. Non-resolution theorem proving. *Artificial Intelligence 9,* 1 (August 1977), 1–35.

[4] Bläsius, K., Eisinger, N., Siekmann, J., Smolka, G., Herold, A., and Walther, C. The Markgraf Karl refutation procedure (Fall 1981). *Proc. Seventh International Joint Conference on Artificial Intelligence,* Vancouver, B.C., August 1981, 511–518.

[5] Boyer, R.S. and Moore, J S. The sharing of structure in theorem-proving programs. In Meltzer, B. and Michie, D. (eds.). *Machine Intelligence 7.* Edinburgh University Press, 1972.

[6] Grosz, B. Research on natural-language processing at SRI. *Sigart Newsletter* #79 (January 1982), 87–94.

[7] Kowalski, R. A proof procedure using connection graphs. *J. ACM 22,* 4 (October 1975), 572–595.

[8] Manna, Z. and Waldinger, R. A deductive approach to program synthesis. *ACM Transactions on Programming Languages and Systems 2,* 1 (January 1980), 90–121.

[9] Manna, Z. and Waldinger, R. Special relations in program-synthetic deduction. Technical Note 260, SRI Artificial Intelligence Center, March 1982.

[10] Moore, R.C. Reasoning about knowledge and action. Technical Note 191, SRI Artificial Intelligence Center, October 1980.

[11] Moore, R.C. Problems in logical form. Proceedings of the 19th Annual Meeting of the Association for Computational Linguistics, Stanford, June 1981.

[12] Murray, N.V. Completely non-clausal theorem proving. *Artificial Intelligence 18,* 1 (January 1982), 67–85.

[13] Sickel, S. A search technique for clause interconnectivity graphs. *IEEE Transactions of Computers C-25,* 8 (August 1976), 823–835.

[14] Stickel, M.E. The programmable strategy theorem prover: an implementation of the linear MESON procedure. Technical Report, Carnegie-Mellon University Computer Science Department, June 1974.

[15] Tyson, W.M. *APRVR: A Priority-Ordered Agenda Theorem Prover.* Ph.D. Dissertation, University of Texas at Austin, 1981.