Consolution and its Relation with Resolution*

Elmar Eder
Fachbereich Mathematik/Informatik
Universitat Marburg
D-3550 Marburg
Germany

Abstract

In this paper the method of consolution for clause form theorem proving is introduced. Consolution is based on the connection method. This means that in a consolution derivation the paths through the input formula are checked for complementarity. In contrast to resolution this checking can be done in a systematic way as in any connection calculus. It is proved that the consolution calculus presented here is sound and complete and that it can simulate resolution step by step and is a generalization of resolution. It combines the advantages of the connection method such as the directedness of search with the advantages of resolution such as the possibility of the use of lemmata.

1 Introduction

In the field of automated theorem proving, there are a number of calculi that have been proposed and implemented for deducing the validity or, equivalently, the unsatisfiability of a formula of first order predicate logic. The best known such calculus is resolution [Robinson, 1965] which is (usually) restricted to formulas in clause form. Another major method for automated deduction is the connection method (see [Bibel, 1987]). We have to distinguish between the concepts of a calculus and a method. A *calculus* has fixed rules which are given once and for all. On the other hand, a *method* for deduction is a design philosophy for designing calculi. A method determines a class of calculi by providing means and concepts for the specification of calculi, and, usually, theorems which are useful for proving the adequacy of calculi specified with these means. Resolution is a calculus whereas the connection method is a method. A number of calculi, called connection calculi, have been developed on the basis of the connection method.

There has been a long discussion on the advantages and disadvantages of the connection method versus resolution. Some clear advantages of the connection method are the greater directedness of its proof search and the fact that its calculi do not disrupt the structure of the input formula as resolution does. This means it provides more information that can be used by strategies for directing the proof search. On the other hand, the connection calculi developed so far lack an important feature that is present in resolution, namely the use of lemmata. In resolution, a clause that has been derived can be reused for resolution an arbitrary number of times. The lack of this feature makes a derivation in these connection calculi for some classes of formulas considerably longer than in resolution (see [Eder, 1989]).

In this paper we introduce the method of consolution which is based on the connection method. We also specify a particular consolution calculus as a basic calculus which may still be enhanced. Consolution has a very close relationship to resolution. In fact, resolution is a restriction of consolution in the sense that a strategy can be imposed upon the consolution calculus to make it identical to resolution. Thus consolution provides a bridge between the connection method and resolution. It allows to combine the directedness of the proof search present in the connection method with the powerful tool of the use of lemmata present in resolution.

2 The connection method

We give here only a brief description of those concepts underlying the connection method which are relevant for consolution. As in resolution, a formula in normal form is represented by a set of clauses in the connection method.[1] But since the connection method is formulated as a method for proving validity rather than unsatisfiability of formulas, we consider here the disjunctive normal form rather than the conjunctive normal form. So, a clause represents the existential closure of the conjunction of its literals, and a set of clauses represents the disjunction of the formulas represented by its clauses. A set of clauses is also called a *matrix.* A clause is usually

*Most of the research and work leading to this paper was done during a stay at ICOT in Tokyo. I want to thank Wolfgang Bibel for his help in the formulation of the method and for valuable hints and discussions.

[1] Actually, the connection method is not restricted to formulas in clause form. Since, however, any arbitrary formula can be transformed to clause form by a fast structure-preserving transformation (see [Greenbaum *et al.*, 1982; Eder, 1985; Plaisted and Greenbaum, 1986]), we shall for simplicity restrict ourselves to clause form formulas in this paper.

depicted as a vertical column consisting of its literals. A matrix is depicted in two-dimensional representation as the columns depicting its clauses, written next to each other. For example, the formula

$$(P \wedge Q) \vee (\neg P \wedge Q) \vee \neg Q$$

is represented by the matrix

$$\{\{P, Q\}, \{\neg P, Q\}, \{\neg Q\}\}$$

which is depicted as

$$\begin{array}{ccc} P & \neg P & \neg Q \\ Q & Q & \end{array}$$

A *path through* a matrix is a set of literals, exactly one taken from each clause. In our example there are four paths through the matrix, and $\{P, \neg P, \neg Q\}$ is one of them. Two literals are *complementary* if one of them is tiie negation of the other. A *connection* in a set of clauses is a pair of literals which can be made complementary by instantiation, ie., by application of substitutions. A path is *complementary* if it contains two complementary literals. The connection method is based on the following theorem.

A formula in disjunctive normal form is valid if and only if there is a finite set of instances of its clauses through which each path is complementary.

A theorem prover implementing the connection method takes the clauses of the input formula and checks all paths through this set of clauses systematically for complementarity. At each step, a connection is chosen in a path that has not yet been checked for complementarity. If necessary, this connection has to be made complementary by unification, thus instantiating the involved clauses in the proper way. Moreover, it may be necessary to take into account more than one instance of a clause. In this case a new variant of the involved clause has to be considered. A detailed introduction to the connection method can be found in [Bibel, 1987].

3 Basic definitions

By a *path in* a matrix we mean a subset of a path through a matrix. Thus, a path in a matrix is a set of literals, at most one chosen from each clause. We shall also use the term *partial path* for a path in a matrix. An *extension* of a path p in a matrix M is a path q in M such that $p \subseteq q$. If c is a clause, p and q are finite sets of literals and $\mathcal{P}$ and $\mathcal{Q}$ are finite sets of finite sets of literals then we define

$$\mathcal{P}_c := \{\{L\} \mid L \in c\}.$$
$$pq := p \cup q.$$
$$\mathcal{P}\mathcal{Q} := \{pq \mid p \in \mathcal{P} \text{ and } q \in \mathcal{Q}\}.$$

$\mathcal{P}_c$ is the set of (one-element) paths through the (one-clause) matrix $\{c\}$. If M and N are disjoint matrices, ie., disjoint finite sets of clauses, then pq is a path in $M \cup N$ for every path p in M and for every path q in N. Similarly, $\mathcal{V}\mathcal{Q}$ is a set of paths *in* $M \cup N$ for every set $\mathcal{V}$ of paths in M and for every set $\mathcal{Q}$ of paths in N. We call the set $\mathcal{V}\mathcal{Q}$ the *product* of the sets $\mathcal{V}$ and $\mathcal{Q}$.

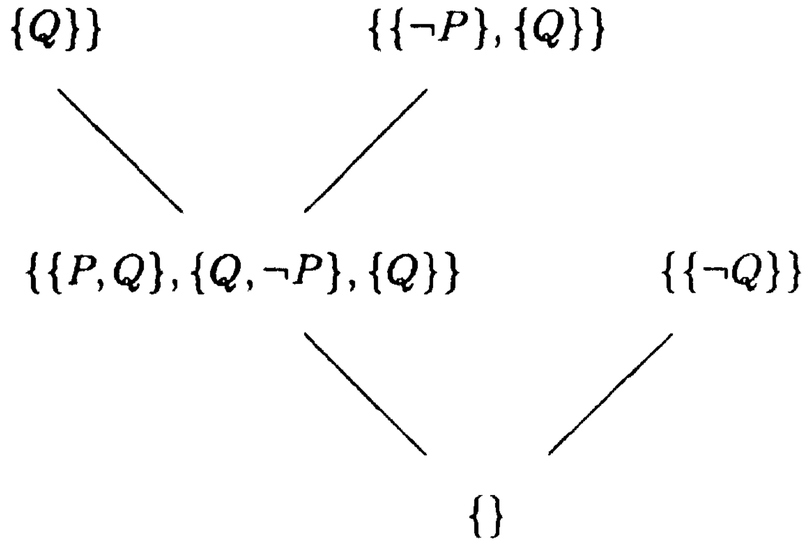
4 Consolution in propositional logic

The idea behind consolution is the following. In order to prove the validity of a given formula using the connection method, all paths through its matrix have to be checked for complementarity. At each stage of a connection proof a certain set of paths has already been checked for complementarity whereas all the remaining paths still have to be processed this way. In consolution at each stage of a proof process, this remaining set of paths yet to be checked for complementarity is represented. It would be inefficient, however, to explicitly represent each path of this set since the number of paths in general increases exponentially with the number of clause instances. Instead, this set of paths is coded in the form of a set of partial paths. Each partial path encodes the set of all its extensions through the whole matrix. So one partial path may encode many paths through the matrix. Let us consider an example which for simplicity is taken from propositional logic.

Suppose we want to prove the validity of the formula

$$(P \wedge Q) \vee (\neg P \wedge Q) \vee \neg Q.$$

The following tree is a proof tree of this formula by consolution.



Each node of the tree is marked with a set of (partial) paths in the matrix M

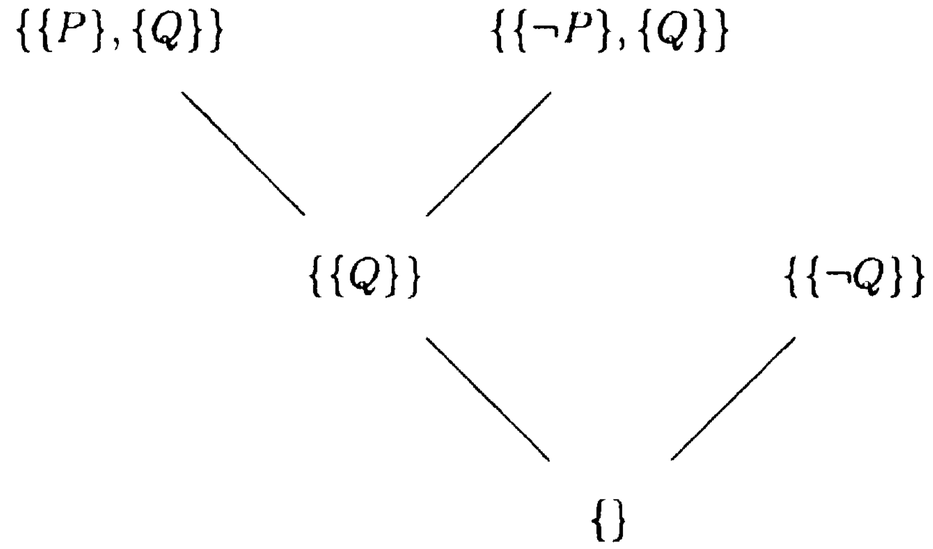
$$\begin{array}{ccc} P & \neg P & \neg Q \\ Q & Q & \end{array}$$

representing the given formula. A leaf of the proof tree is marked with the set $\mathcal{P}_c$ of all one-element paths through some clause c of the formula, ie. $\mathcal{P}_c = \{\{L\} \mid L \in c\}$. Since every path through the given matrix M is an extension of a path through c for every $c \in M$, the set $\mathcal{P}_c$ encodes the set of all paths through M. Thus the marks $\mathcal{P}_c$ of the leaves reflect the fact that at the beginning of the proof process all paths through M still have to be checked for complementarity.

The inference rule, also called *consolution,* takes the set of partial paths from each premise and combines these sets into a new set, called the *consolvent.* This combination can be regarded to consist of two parts. The first part consists of building the product *VQ* of *V* and *Q.* For example, if *v = {{P}AQ}} and Q = {{¬P}, {Q}* then PQ = {{P, ¬P},{P,Q},{Q, ¬*P},{Q}}-* The second part of the inference rule consists in simplifications of P*Q.* One such simplification is the elimination of complementary paths. To continue the illustration of our example, this means that the complementary path *{P,P}* is removed from *VQ.* Actually, both parts are to be seen as a single operation, a remark which bears its relevance on the efficiency of implementation. But for the ease of the reader's understanding we will continue to make the distinction.

The proof in our example is completed by applying consolution once again to the result of the previously illustrated step and the remaining leaf $\{\{\neg Q\}\}$. The resulting empty set is the criterion of a successful derivation (as in resolution). To summarize, for a formula *F* in disjunctive normal form, a *proof* of *F* is a derivation of the empty set from the sets P_c *{c ∈ M)* with the consolution rule, where *M* is the set of clauses of *F.*

While this explains the essentials of the calculus, the following additional details complete its description. Above we have used a single simplification which is elimination of complementary paths. This is all needed for completeness and soundness of the calculus. For efficiency, it is necessary to incorporate at least the following simplification (since otherwise a huge number of paths will quickly be generated in practice). Any path *p* in PQ may be replaced by any subset of *p* . Both simplifications may be applied simultaneously. So in our present example, the first consolvent {{P, *Q},{Q, ¬P},{Q}}* may be further reduced, for instance to {{Q}} • Note that three distinct paths have been replaced by a single one in this case. In general, the number of paths may be reduced considerably this way. If we incorporate this simplification into the derivation shown above, the following proof tree of the same formula results.



In summary, any consolution calculus must, among its simplifications within the consolution rule, include the elimination of complementary paths. While it is not absolutely necessary, the shortening of paths is understood to be always included in consolution. As an aside we mention that such an inclusion amounts to an extension of the basic calculus. A further extension of consolution might include Prawitz' matrix reduction, but no investigation has been made yet into this possibility.

Note that consolution allows a systematic checking of paths. If we enumerate the paths through a matrix in any given order then a consolution derivation can check them one after the other in this order. A suitable application of shortening of paths will make this systematic checking more efficient by allowing to check more than one path in one step. Systematic checking of paths is not possible with resolution since there at each step the paths are shortened to length 1.

5 Consolution in first oder logic

The lifting of consolution to first order logic is done in much the same way as it is done for resolution. In this section we give a formal description of consolution for full first order logic.

By a *path set* we mean a finite set of finite sets of literals. We shall use this term even if these sets of literals are not paths in some particular matrix.

Definition 5.1
A path set *Q* is obtained from a path set *V* by *elimination of complementary paths* if there is a set of connections in elements of *V* and a most general unifier *a* of this set of connections such that *Q* is the set of non-complementary elements of Po.

Definition 5.2
A path set $\mathcal{Q}$ is obtained from a path set $\mathcal{P}$ by *shortening of paths* if there is a surjective mapping $f : \mathcal{P} \to \mathcal{Q}$ such that $f(p) \subseteq p$ holds for all $p \in \mathcal{P}$.

Definition 5.3
A path set $\mathcal{R}$ is obtained from a path set $\mathcal{P}$ by *simplification* if there is a path set $\mathcal{Q}$ such that $\mathcal{Q}$ is obtained from $\mathcal{P}$ by elimination of complementary paths and such that $\mathcal{R}$ is obtained from $\mathcal{Q}$ by shortening of paths.

The inference rule

$$\frac{\mathcal{P} \qquad \mathcal{Q}}{\mathcal{R}}$$

if there exists a variant $\mathcal{Q}'$ of $\mathcal{Q}$ which does not have any variables in common with $\mathcal{P}$ such that $\mathcal{R}$ is obtained from the product $\mathcal{P}\mathcal{Q}'$ by simplification.

We say then that the path set $\mathcal{R}$ is a *consolvent* of the path sets $\mathcal{P}$ and $\mathcal{Q}$.

Definition 5.4
A *derivation* of a matrix M is a finite sequence $(\mathcal{P}_0, \ldots, \mathcal{P}_n)$ of path sets such that the following conditions hold.

1. **For all $k = 1, \ldots, n$, the set $\mathcal{P}_k$ equals $\mathcal{P}_c$ for some $c \in M$, or $\mathcal{P}_k$ is a consolvent of $\mathcal{P}_i$ and $\mathcal{P}_j$ for some $i, j < k$.**
2. **$\mathcal{P}_n = \emptyset$.**

Theorem 5.1
A formula in disjunctive normal form is valid if and only if there is a derivation of its matrix by consolution.

Proof: In order to prove correctness, let us assume that F is a formula and M its matrix. Further assume that $(\mathcal{P}_0, \ldots, \mathcal{P}_n)$ is a derivation of M by consolution. To each finite set p of literals we attribute a formula F_p as follows. Let F_p be the disjunction of all literals of p. To each path set $\mathcal{P}$ we attribute a formula $F_{\mathcal{P}}$ as follows. Let $F_{\mathcal{P}}$ be the existential closure of the conjunction of all F_p with $p \in \mathcal{P}$. It then holds that $\{\neg F_{\mathcal{P}}, \neg F_{\mathcal{Q}}\} \models \neg F_{\mathcal{R}}$ if $\mathcal{R}$ is a consolvent of $\mathcal{P}$ and $\mathcal{Q}$ (The proof is exactly the same as the proof for the soundness of the resolution rule). Moreover, $\mathcal{P}_c \models F$ for all $c \in M$. By induction on j it follows that $F_{\mathcal{P}_j} \models F$ for all $j = 0, \ldots, n$. In particular for $j = n$, the set $\mathcal{P}_j$ is the empty set and therefore $F_{\mathcal{P}_j}$ is the empty conjunction, ie., the verum $\top$. So $\top \models F$ which means that F is valid.

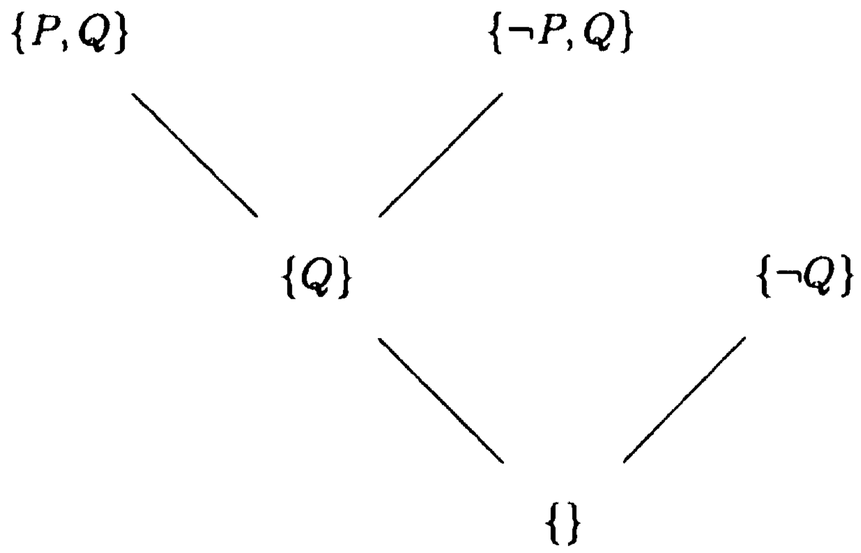
For the proof of completeness we shall see in the next section that every set of clauses for which there is a resolution refutation, also has a consolution derivation. From the completeness of resolution it then follows that also consolution is complete.

q.e.d.

In the completeness proof note the duality between proving validity of a formula in disjunctive normal form and proving unsatisfiability of a formula in conjunctive normal form. On the level of clauses and matrices there is no difference between affirmative and refutational proving. In fact any refutational calculus can just as well be formulated in an affirmative way and vice versa, and it would not even make a difference in the codes of implementations.

6 Relations to resolution

If we look at the last proof tree shown in Section 4 then we see that its path sets contain only one-element paths. If we replace each path with the single literal which it contains then we obtain the tree



Note that this tree is a resolution refutation of the given clause set. In this way any resolution refutation of a set of clauses can be obtained by consolution. To be more specific, the following holds.

Theorem 6.1
Let M be a matrix and let (co,...,c„) *be a resolution refutation of M. Then {V*$_{Co}$*,... ,V*$_{Cn}$*) is a consolution derivation of M.*

Proof: The only non-trivial part of the proof is to show that P_e is a consolvent of P_e and P_d if e is a resolvent of *c* and *d.* So let e be a resolvent of c and *d* and let C_o U d_o be the set of literals resolved upon where CQCC and *do* C *d.*[2]. Let *d'* be the clause obtained from *d* by seperating apart the variables of *c* and *d.* So, *d'* is a variant of *d.* Let *d'*$_0$ be the corresponding variant of P_d'- Then P*d'* is a variant of Pd, and P*cPd'* is the set of all {I, *J)* such that *I E c* and *J* £ *d!.* Let *K* be the set of all pairs (J, *J)* such that *I E CQ* and *J* € *d'*$_0$. Then the set *Q* defined as

{{la, Ja) | *I* £ *c* and *J* € *a** and not (*/*, *J)* £ *K}*

is obtained from *V*$_c$*Vd>* by elimination of complementary paths (choosing as connections the elements of *K).* For each path {*/*, *J)* 6 *Q* we define

$$mi,j\})-= \begin{cases} \{1\} \\ \{J\} \end{cases} \quad \begin{array}{l} \text{if } /\hat{}cp \\ \text{otherwise} \end{array}$$

Then the range of / is TV So V_e is obtained from *Q* by shortening of paths. Thus V_t is a consolvent of V_c and V_d.

q.e.d.

From this theorem it follows that consolution can simulate resolution step by step (and therefore consolution is complete as indicated in the last section). On the other hand, consolution is much more general than resolution because it can handle paths of arbitrary lengths. In fact, resolution can be seen as one consolution strategy where all paths are shortened to length 1 in a certain way at each step. Other strategies of consolution are given by the connection calculi that have been developed so far. They involve paths of arbitrary lengths.

As a further remark we point out that the sets resulting by consolution from paths in a given matrix are not always paths in the matrix as in the example shown above. Rather they may also be the union of such paths even in propositional logic. For illustration, the reader may also think of paths in a matrix that has multiple occurrences of clauses.

7 Conclusion

Consolution is a method for clause form theorem proving. We have proved that the consolution calculus presented here is complete and sound and that it can simulate resolution step by step. It is more general than resolution because resolution allows only paths of length 1 whereas consolution allows paths of arbitrary length. This advantage becomes most apparent from the fact that consolution allows a systematic checking of paths as any connection calculus does. Such a systematic checking of paths is not possible in resolution since there the paths are shortened to length 1 in every step. On the other hand, consolution provides the powerful tool of the use of lemmata which is present in resolution but lacking in previous connection calculi. A consolvent that has once been derived can be used as a parent in any number of further consolution steps.

[2] We assume that factorization is included in the resolution rule as in [Robinson, 1965]

References

[Bibel, 1987] Wolfgang Bibel. *Automated Theorem Proving.* Artificial Intelligence. Vieweg, Braunschweig/Wiesbaden, second edition, 1987.

[Eder, 1985] Elrnar Eder. An implementation of a theorem prover based on the connection method. In W. Bibel and B. Petkoff, editors, *Artificial Intelligence, Methodology, Systems, Applications (AIMSA '84),* pages 121-128, Amsterdam, New York, Oxford, 1985. European Coordinating Committee for Artificial Intelligence, North-Holland.

[Eder, 1989] Elmar Eder. A comparison of the resolution calculus and the connection method, and a new calculus generalizing both methods. In E. Borger, H. Kleine Burning, and M. M. Richter, editors, *CSL '88, 2nd Workshop on Computer Science Logic, Duisburg, FRG, October 1988, Proceedings, Lecture Notes in Computer Science 885,* pages 80-98, Berlin, Heidelberg, New York, 1989. Springer-Verlag.

[Greenbaum *et al,* 1982] S. Greenbaum, A. Nagasaka, P. O'Rorke, and D. A. Plaisted. Comparison of natural deduction and locking resolution implementations. In D. Loveland, editor, *Proceedings of the 6th Conference of Automated Deduction,* pages 159-171. Springer Lecture Notes in Computer Science 138, 1982.

[Plaisted and Greenbaum, 1986] David Plaisted and Steven Greenbaum. A structure-preserving clause form translation. *Journal of Symbolic Computation,* 2:293-304, 1986.

[Robinson, 1965] J. A. Robinson. A machine-oriented logic based on the resolution principle. *Journal of the ACM,* 12:23-41, 1965.