

RESOLUTION PLANS IN THEOREM PROVING

C. L. Chang
IBM Research Laboratory
5600 Cottle Road
San Jose, California 95193

Briefly, an example of a resolution plan can be described as follows: Let $L_1 \vee A$ and $L_2 \vee B$ be two clauses, where L_1 and L_2 are literals, and A and B are clauses. If L_1 and L_2 can be made complementary by some substitution, we shall call $A \vee B$ (L_1, L_2) a resolution plan. In general, a resolution plan is represented by

$$C (L_1, M_1) \dots (L_r, M_r) ,$$

where $L_1, M_1, \dots, L_r, M_r$ are literals, and C is a clause. If C is empty, $(L_1, M_1) \dots (L_r, M_r)$ is called a total plan. The total plan corresponds to a proof if there is a unifier which simultaneously make (L_i, M_i) complementary, $i=1, \dots, r$. As shown elsewhere, the total plan approach can eliminate redundancies. In this paper, we shall show that all the strategies developed for resolution can be used to generate total plans.

1. INTRODUCTION

In [Chang and Slagle 1977], the concept of total plans for proving theorems in first order logic was proposed. A total plan is denoted by a sequence of pairs of literals, $(L_1, M_1) \dots (L_r, M_r)$. If there is a substitution θ that can make each of $(L_1, M_1)\theta, \dots, (L_r, M_r)\theta$ a complementary pair of literals, the plan is said to be acceptable, and θ is called a solution. In [Chang and Slagle 1977], a method based upon rewriting rules for generating total plans has been proposed. In this paper, we shall show that we can use resolution strategies [Chang and Lee 1973, Kowalski and Kuhner 1971, Kowalski 1975, Robinson 1965a, 1965b, Wos et al. 1965] to generate total plans. This is done by generating resolution plans. Formal definitions of resolution plans will be given in the sequel. They are defined in terms of notation of literals, instead of literals themselves. Because of this, unification is not actually carried out during generations of resolution plans. This is in contrast to resolution where unification has to be performed. Therefore, it is faster to generate resolution plans than resolvents.

In the following, we shall first give connection graphs, then we shall define various resolution plans such as binary resolution plans, hyper resolution plans, linear resolution plans, etc.

2. CONNECTION GRAPHS

The concept of a connection graph has been used in [Kowalski 1975, Sickel 1976]. Essentially, a connection graph is a data structure for a set of clauses indicating possible refutations. We shall now give a definition of a connection graph.

A pair of literals L_1 and L_2 are called potentially complementary if L_1 and L_2 can be made complementary by applying some substitution, after renaming variables so that L_1 and L_2 share no variables.

A connection graph for a set S of clauses is constructed as follows:

- (1) Every clause in S appears in the graph exactly once;

(2) For every clause, $L_1 \vee \dots \vee L_n$, in S , where $L_1, \dots, L_n$ are literals, it is represented in the graph as

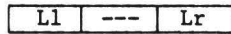


Fig. 1

(3) For every pair of potentially complementary literals, draw an edge connecting the literals. (Note that edges between literals in the same clause are allowed.)

Example 1. Fig. 2 is a connection graph for the set, $\{P(x) \vee Q(y), \neg P(a), \neg Q(b)\}$.

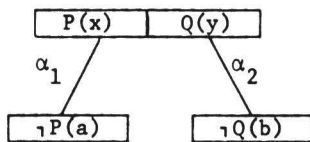


Fig. 2

Example 2. Fig. 3 is a connection graph for the set, $\{\neg P(x) \vee P(f(x)), P(a), \neg P(f(f(a)))\}$. Note that there is an edge connecting literals $\neg P(x)$ and $P(f(x))$ in the same clause.

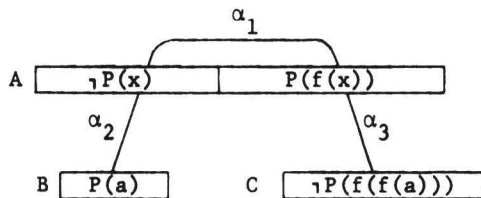


Fig. 3

In the sequel, we shall label a clause by a distinct name and then refer to a literal in the clause by its position in the clause. That is, if C is a clause, then C_n is the n -th literal (counting from the left) of clause C , where n is an integer. By this convention, Fig. 3 will be represented by Fig. 4.

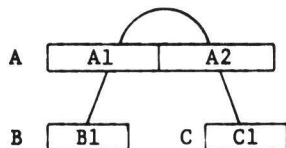


Fig. 4

3. BINARY RESOLUTION PLANS

In terms of a connection graph for a set S of clauses, we now give the following definitions.

A clause plan is represented by a clause followed by a sequence of pairs of literals. A clause plan can be represented as

$$L_1 \vee \dots \vee L_p (M_1, N_1) \dots (M_q, N_q),$$

where L_i, M_j and N_j , $i=1, \dots, p$, $j=1, \dots, q$, are literals. We call $L_1 \vee \dots \vee L_p$ a clause segment, and $(M_1, N_1) \dots (M_q, N_q)$ a plan segment of the clause plan. A clause plan is called a total plan if and only if its clause segment is empty.

Definition. Let two clause plans A and B respectively be denoted by

$$\begin{aligned} A &= C \vee P \\ B &= D \vee Q, \end{aligned}$$

where C and D are the respective clause segments of A and B , and P and Q are the respective plan segments of A and B . If there are literals $L_1, \dots, L_m$ in C , and $M_1, \dots, M_n$ in D such that there are edges in a connection graph connecting every literal of $L_1, \dots, L_m$ to every literal of $M_1, \dots, M_n$, then the clause plan

$$(C - C') \vee (D - D')(L_1, M_1)(L_1, M_2) \dots (L_m, M_n) P \vee Q,$$

where $C' = \{L_1, \dots, L_m\}$ and $D' = \{M_1, \dots, M_n\}$, is called a binary resolution plan of A and B . Note that $(L_1, M_1)(L_1, M_2) \dots (L_m, M_n)$ is a cartesian product of C' and D' , and may be represented by (C', D') .

Example 3. Let $A = P(x) \vee Q(x)$ and $B = \neg P(a) \vee R(x)$. Using the positions of literals in the clause to name the literals, A and B are represented as

$$\begin{aligned} A &= A_1 \vee A_2 \\ B &= B_1 \vee B_2, \end{aligned}$$

where A_1, A_2, B_1 and B_2 are $P(x), Q(x), \neg P(a)$ and $R(x)$, respectively. Note that using the notation in the above definition, $C = \{A_1, A_2\}$ and $D = \{B_1, B_2\}$, and both P and Q are empty. Since A_1 and B_1 are complementarily unifiable, we choose $C' = \{A_1\}$ and $D' = \{B_1\}$. Therefore, a resolution plan is obtained as follows:

$$\begin{aligned} &(C - C') \cup (D - D')(C', D') P \vee Q \\ &= (\{A_1, A_2\} - \{A_1\}) \cup (\{B_1, B_2\} - \{B_1\}) (\{A_1\}, \{B_1\}) \\ &= \{A_2\} \cup \{B_2\} (A_1, B_1) \\ &= \{A_2, B_2\} (A_1, B_1) \\ &= A_2 \vee B_2 (A_1, B_1). \end{aligned}$$

We can use resolution strategies to generate total plans for refutations. That is, given a set S of clauses, we first build a connection

graph for S. Then, to check whether or not a pair of literals are potentially complementary, we need only to look at the connection graph to see if there is an edge connecting the literals. Once we have a connection graph, resolution plans can be generated faster than resolution because unification does not have to be performed. A total plan is a clause plan whose clause segment is empty. We can use resolution strategies to generate total plans. Once a total plan is generated, we then perform unification at the last step to check whether or not the plan is acceptable.

Example 4. Consider the set S of the following clauses,

$\neg T(x,y,u,v) \vee P(x,y,u,v)$
 $\neg P(x,y,u,v) \vee E(x,y,v,u,v,y)$
 $T(a,b,c,d)$
 $\neg E(a,b,d,c,d,b).$

The connection graph for S is shown in Fig. 5.

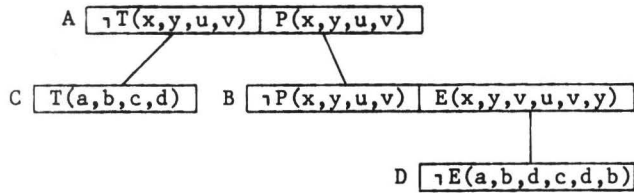


Fig. 5

Using literal names, Fig. 5 can be represented by Fig. 6.

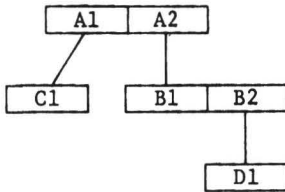


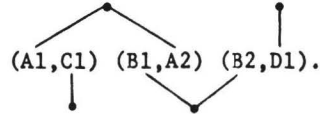
Fig. 6

Now, using Fig. 6, we can generate the following resolution plans:

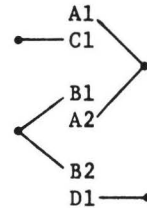
- | | |
|------------------------|----------------------------------|
| (1) A1 $\vee$ A2 | a clause in S |
| (2) B1 $\vee$ B2 | a clause in S |
| (3) C1 | a clause in S |
| (4) D1 | a clause in S |
| (5) B1 (B2,D1) | a resolution plan of (2) and (4) |
| (6) A1 (B1,A2) (B2,D1) | a resolution plan of (1) and (5) |

(7) (A1,C1) (B1,A2) (B2,D1) a resolution plan of (3) and (6).

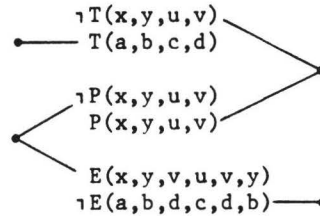
Now, since we want to perform unification, we first need to rename variables. As discussed in [Chang and Slagle 1977], for the total plan (A1,C1)(B1,A2)(B2,D1), all literals belong to a clause are linked to a common dot. Therefore, we obtain the following linked plan,



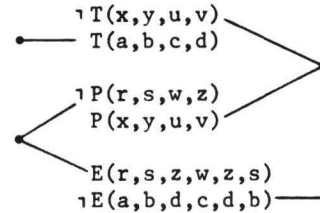
The above linked plan can also be represented as



Substituting the real literals for A1, A2, B1, B2, C1 and D1 in the linked plan, we obtain



Renaming the variables so that no clauses share variables in common, we obtain



Unifying the two expressions,

$(T(x,y,u,v), P(r,s,w,z), E(r,s,z,w,z,s))$ and $(T(a,b,c,d), P(x,y,u,v), E(a,b,d,c,d,b))$,

we obtain a substitution

$\theta = \{a/x, b/y, c/u, d/v, a/r, b/s, c/w, d/z\}.$

This substitution is a solution of the set M of the following clauses,

$$\begin{cases} \neg T(x,y,u,v) \vee P(x,y,u,v) \\ \neg P(r,s,w,z) \vee E(r,s,z,w,z,s) \\ T(a,b,c,d) \\ \neg E(a,b,d,c,d,b) \end{cases}$$

because $M\theta$ is

$$\begin{cases} \neg T(a,b,c,d) \vee P(a,b,c,d) \\ \neg P(a,b,c,d) \vee E(a,b,d,c,d,b) \\ T(a,b,c,d) \\ \neg E(a,b,d,c,d,b) \end{cases}$$

and is truth-functionally unsatisfiable.

4. LINEAR AND SET-OF-SUPPORT RESOLUTION PLANS

Definition. Given a connection graph for a set S of clauses, C_n is called a linear resolution plan if and only if it is obtained as shown in Fig. 7, where C_0 is a clause in S, and for $i=0,1,\dots,n-1$, (a) C_{i+1} is a resolution plan of C_i and B_i , and (b) each B_i is either in S, or is a C_j for some j , $j < i$.

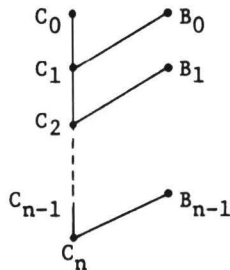


Fig. 7

Example 5. The deduction obtained in Example 4 is a linear deduction as shown in Fig. 8. That is, the total plan, $(A1,C1)(B1,A2)(B2,D1)$, is a linear resolution plan.

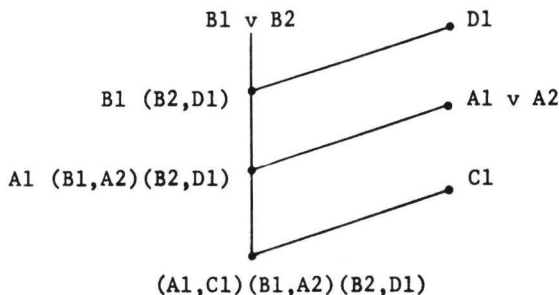


Fig. 8

Definition. A subset T of a set S of clauses is called a set of support of S if $(S-T)$ is satisfiable. A set-of-support resolution plan is a binary resolution plan of two clauses that are not both from $(S-T)$.

Example 6. Consider the set S of clauses given in Example 4. Now, if we let T be the set $\{\neg E(a,b,d,c,d,b)\}$, then the deduction given in Fig. 8 is also a set-of-support deduction. (Note that in general, a set-of-support deduction needs not be a linear deduction.)

5. HYPER RESOLUTION PLANS

In order to define a hyper resolution plan, we shall name atomic formulas in every literal of a clause, instead of naming the literal itself as given in the previous sections. That is, if C is a clause, then the n-th literal of clause C is named as C_n if the literal is positive, and named as $\neg C_n$ if it is negative. For example, if

$$\begin{aligned} A &= P(x) \vee Q(x) \text{ and} \\ B &= \neg P(a) \vee R(x), \end{aligned}$$

then A and B can be respectively represented as

$$\begin{aligned} A &= A1 \vee A2 \text{ and} \\ B &= \neg B1 \vee B2. \end{aligned}$$

We say that a clause plan is positive if its clause segment is positive. Otherwise, it is negative.

Definition. Let C be a negative clause plan, and let $B_0, B_1, \dots, B_{n-1}$ be positive clause plans. Then, C_n is a hyper resolution plan, as shown in Fig. 9, if C_n is a positive clause plan obtained by a deduction shown in Fig. 10, where C_i is a binary resolution plan of C_{i-1} and B_{i-1} , $i=1, \dots, n$.

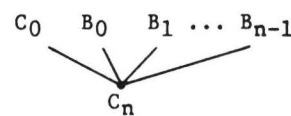


Fig. 9

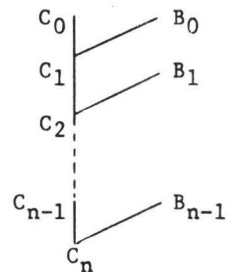


Fig. 10

Example 7. Consider the set S of clauses given in Example 4. Using the clause names in

Fig. 5, we can represent the clauses in S as follows:

- (1) $\neg A1 \vee A2$
- (2) $\neg B1 \vee B2$
- (3) $C1$
- (4) $\neg D1$.

Then, the deduction shown in Fig. 11 is a hyper deduction, while the one shown in Fig. 12 is not.

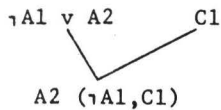


Fig. 11

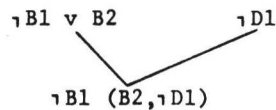


Fig. 12

6. RENAMING VARIABLES AND HANDLING DUPLICATE CLAUSE PLANS

One of the important steps in theorem proving is renaming variables. In ordinary resolution, before a resolution is performed, variables in clauses need to be renamed so that the clauses share no variables in common. Herbrand's Theorem says that a set S of clauses is unsatisfiable if and only if there is a finite unsatisfiable set S' of ground instances of clauses of S. Renaming variables is closely related to Herbrand's Theorem because more than one ground instances in S' may come from the same clause C in S, and the ground instances are essentially obtained by first renaming variables in copies of C.

Given a set S of clauses, we first generate a deduction of a total plan from S, and then check if the total plan is acceptable by renaming variables and by performing unification. In the deduction of the total plan, the same clause plan may appear more than once. In this case, we shall use primes to distinguish the different copies of the clause plan. This is illustrated by the following example.

Example 8. Consider the set S of clauses given in Example 2. That is, S consists of the following clauses

- $$\begin{aligned} &\neg P(x) \vee P(f(x)) \\ &P(a) \\ &\neg P(f(f(a))). \end{aligned}$$

The connection graph of S is shown in Fig. 3. Using literal names, Fig. 3 can be represented by Fig. 13.

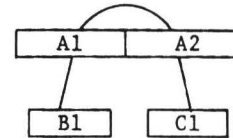


Fig. 13

Using Fig. 13, we can generate a deduction of a total plan shown in Fig. 14.

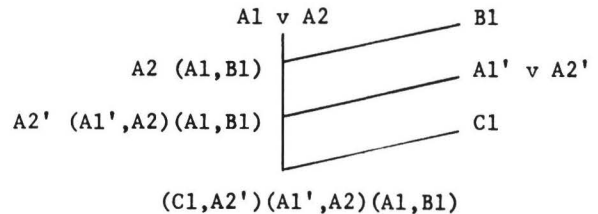
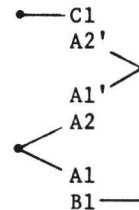
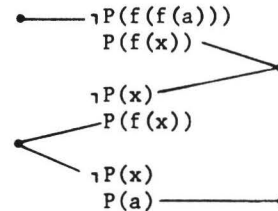


Fig. 14

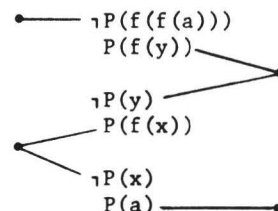
Note that two copies of $(A1 \vee A2)$ are used in the deduction, and we use $(A1' \vee A2')$ to represent the second copy. By linking literals belonging to the same copy of a clause, we obtain



Substituting the real literals for A1, A2, A1', A2', B1 and C1, we obtain



Renaming the variables so that no clauses share variables in common, we obtain



Unifying the two expression,

$(P(f(f(a))), P(y), P(x))$ and
 $(P(f(y)), P(f(x)), P(a)),$

we obtain a substitution

$$\theta = \{f(a)/y, a/x\}.$$

This substitution is a solution of the set M of the following clauses,

$$\begin{cases} \neg P(x) \vee P(f(x)) \\ \neg P(y) \vee P(f(y)) \\ P(a) \\ \neg P(f(f(a))), \end{cases}$$

because $M\theta$ is

$$\begin{cases} \neg P(a) \vee P(f(a)) \\ \neg P(f(a)) \vee P(f(f(a))) \\ P(a) \\ \neg P(f(f(a))), \end{cases}$$

and $M\theta$ is truth-functionally unsatisfiable.

7. CONCLUSIONS

We have defined various resolution plans. These plans are obtained from the connection graph of a set S of clauses. To generate resolution plans, no unification needs to be performed. Therefore, it is faster to generate a resolution plan than to generate a resolvent. Unification is performed only at the end of a proof. One of the advantages of using the resolution plan method is that all the strategies proposed for resolution can be directly applied to generate resolution plans.

ACKNOWLEDGMENTS

The author would like to thank Dr. J. R. Slagle of Naval Research Laboratory, Washington, D.C. for his conversations.

REFERENCES

- Chang, C.L., and Lee, R.C.T. (1973): Symbolic logic and mechanical theorem proving, Academic Press, New York.
- Chang, C.L., and Slagle, J.R. (1977): Using rewriting rules for connection graphs to prove theorems, IBM Research Report RJ2117, San Jose, Ca. (To appear in Artificial Intelligence, 1979.)
- Kowalski, R., and Kuhner, D. (1971): Linear resolution with selection function, Artificial Intelligence 2, 227-260.
- Kowalski, R. (1975): A proof procedure using connection graphs, JACM 22, 4, 572-595.
- Robinson, J.A. (1965a): A machine-oriented logic based on the resolution principle, JACM 12, 1, 23-41.
- Robinson, J.A. (1965b): Automatic deduction with hyper-resolution, Internat. J. Comput. Math. 1, 227-234.
- Sickel, S. (1976): Clause interconnectivity graphs in theorem proving, IEEE Transactions on Computers, C-25, 823-834.
- Wos, L., Robinson, G.A., and Carson, D.F. (1965): Efficiency and completeness of the set of support strategy in theorem proving, JACM 12, 4, 536-541.