Fachbereich Informatik
Universität Kaiserslautern
Postfach 3049
D-6750 Kaiserslautern

SEKI - REPORT

Reducing the Derivation of Redundant Clauses in Reasoning Systems

Rolf Socher-Ambrosius
SEKI Report SR-89-04

Reducing the Derivation of Redundant Clauses in Reasoning Systems

ROLF SOCHER-AMBROSIUS
Fachbereich Informatik, Universität Kaiserslautern
Postfach 3049, D-6750 Kaiserslautern, W.-Germany

Abstract: This paper addresses two problems concerning the issue of redundant information in resolution based reasoning systems. The first one deals with the question how the derivation of redundant clauses, such as duplicates or instances of already retained clauses, can be substantially reduced. The second one asks for a criterion to decide, which clauses need not be tested for redundancy.
In this paper we consider a particular kind of reduncancy, which we call *ancestor subsumption*, that is the subsumption of a resolvent by one of its ancestors. We give a complete syntactic characterization of clause sets producing ancestor subsumed clauses. This characterization partially answers the two questions. First, if a clause set is known to exclude ancestor subsumption, linear resolution turns out to be a preferable strategy in order to reduce the generation of subsumed clauses. Concerning the second question, this result allows a suitable restriction of the (usually very expensive) subsumption test.
Finally, we show that in particular cases those clauses that account for the occurrence of ancestor subsumption can be excluded from the resolution process. SAM's lemma will serve as an example for demonstrating various possibilities to remove reduncancy-generating clauses.

1 Introduction

The derivation of redundant information is one of the greatest obstacles to the efficiency of reasoning programs. Wos (1988) reports an attempt to prove SAM's lemma (see Guard 1969) using hyperresolution, where 6000 clauses identical to retained clauses and 5000 clauses being proper instances of retained clauses were generated. Even if these redundant clauses can be removed after their generation, they must be processed with demodulation, subsumption, and other procedures. Moreover, the test on subsumption, being a very useful means for removing reduncancies, is rather expensive as Chang & Lee (1973), Eisinger (1981), and some others remark. A strategy to prevent the generation of redundant clauses, or at least to reduce the amount of newly generated unneeded clauses, would thus prove very useful for increasing the power of reasoning systems.

On closer inspection of the proof of SAM's lemma it turns out that many of the 6000 duplicates are generated by double resolution with a clause of the form $Pxy \vee \neg Pyx$. Such a duplicate is identical to its own "grandfather" in this resolution derivation. More general, we will deal with the situation that a resolvent is subsumed by one of its own ancestors, which we will call *ancestor subsumption.* Ancestor subsumption is a particular kind of *forward subsumption* (Overbeek 1975), that is the subsumption of a newly deduced clause by a given clause. One of the paper's objectives is to characterize clause sets that admit ancestor subsumption. This approach is based on the following observation: A resolvent of two ground clauses cannot be

subsumed by one of its parent clauses (a situation, which could be called *parent subsumption*), unless the other parent is self-resolving. This can be easily seen: let $C=\{L_1,L_2,..,L_n\}$ and $D=\{\neg L_1,K_2,..,K_n\}$ be ground clauses and assume, C subsumes the resolvent $R=\{K_2,..,K_n,L_2,..,L_n\}$. Then $L_1 \in R$ must hold and from $L_1 \notin \{L_2,..,L_n\}$ now follows $L_1 \in \{K_2,..,K_n\}$. Hence D is a tautology. We will generalize this observation in the following way: A resolvent R cannot be subsumed by an ancestor C, unless the set of ancestors of R contains a cycle (the notion of a cycle was introduced by Shostak (1976)). Noncyclic clause sets thus have the nice property of excluding ancestor subsumption. A prominent example for this class of clause sets is *Schubert's Steamroller* (see Stickel 1986).

Another question addressed by this paper is closely related to the first problem. The characterization mentioned above provides a means to avoid the generation of subsumed clauses only for noncyclic clause sets. But for other clause sets we can at least give a criterion to decide, which clauses have to be considered as potential subsumers in the subsumption test. A non-self-resolving clause, for example, can always be excluded from being subsumer of its own "child".

As cycles in clause sets are heavily responsible for the generation of redundant information, a technique to remove such cycles would prove very useful. Two approaches are considered in this paper, both of which are based on the observation that cycles correspond to equivalences and thus equality reasoning methods apply to cycles. To some extent cycles can be made harmless by using them only as demodulators. Consider for example a clause set containing the two clauses $\neg P \vee Q$ and $\neg Q \vee P$, which form a cycle. These two clauses express that P and Q are equivalent literals, which therefore can be substituted for each other without changing the truth value of the whole clause set. Replacing for instance each occurrence of P in the clause set by Q yields an equivalent clause set, where the two cycle clauses have become tautologies. This approach in fact amounts to a demodulation on literals instead of terms. The well-known problems with term rewriting, however, arise with literal demodulation, too. Directing equations to rewrite rules requires the existence of a well founded ordering on terms. Thus, according to the same reason why the equation $fxy = fyx$ cannot be directed to a rewrite rule, the equivalence $Pxy \equiv Pyx$ cannot be used as a demodulator. The second approach, which overcomes the problem with directing equivalences, consists in using cycles as the basic theory for performing *theory resolution*. The cyclic clauses, for instance the symmetry clause $Pxy \vee \neg Pyx$, disappear in a *theory box*, enabling in this case resolution between the clauses Pab and $\neg Pba$.

2 Basic Notions

In the following we assume the reader to be familiar with the standard terminology of First Order Logic. The few basic notions of clause graphs used in this paper can be found for instance in Eisinger's (1988) thesis. Clauses are always considered as sets of literals, but written without set braces. We do not distinguish between a unit clause and its single literal. The empty clause is denoted by $\emptyset$.

Let $\mathcal{V}$ be a denumerable set of variables. A *substitution* σ is an endomorphism on the term algebra, which is identical almost everywhere on $\mathcal{V}$ and thus can be represented as a finite set σ

$= \{x_1 \rightarrow t_1, \ldots, x_n \rightarrow t_n\}$. A substitution ρ is called a *renaming* (substitution), iff ρ is injective on its domain and $\mathcal{V}\rho \subseteq \mathcal{V}$. A literal or clause L is called a *variant* (or a *copy*) of the literal or clause K, if there exists a renaming substitution ρ, such that $L\rho = K$. Two literals L and K are *weakly unifiable*, if L and a copy of K are unifiable. Two substitutions σ, τ are *compatible*, if there exists some substitution λ with $\sigma\lambda = \tau\lambda$.

3 Cycles in Clause Sets

This chapter provides a syntactic characterization of clause sets admitting ancestor subsumption. Our main result is as follows: Clause sets admitting ancestor subsumption possess *cycles*, whose elements are the *far parents* of the subsumed clause. First, we establish some results for the particular case of *parent subsumption.*

3.1 Definition:

A clause is **self-resolving**, if it resolves with a copy of itself.

The following lemma determines those clauses that possibly produce subsumed resolvents. For any clause D we define the deduction relation $\rightarrow_D$ between clauses C and R by $C \rightarrow_D R$, iff R is a resolvent of C and D.

3.2 Lemma:

Let C be a unit clause and D , R be clauses with $C \rightarrow_D R$.

a) If R is a variant of C, then D is self-resolving.

b) If R is an instance of C, that is, $C\mu = R$ for some substitution μ, then $D = LK_1 \ldots K_n$, and L, K_i have the same predicate symbol, but different polarity for all $i \in \{1..n\}$.

c) If C subsumes R, that is, $C\mu \subseteq R$ for some substitution μ, then $D = LK_1 \ldots K_m M_1 \ldots M_n$, and L, K_i have the same predicate symbol, but different polarity for all $i \in \{1..n\}$.

Proof: a) Let $C = M$ and $D = LK_1 \ldots K_n$ and let σ be a unifier of M and $\neg L$. Then $(K_1 \ldots K_n)\sigma\rho = M$, hence $(K_1 \ldots K_n)\sigma\rho\sigma = M\sigma$, i.e. $K\sigma\rho\sigma = M\sigma = \neg L\sigma$ for each $K \in \{K_1, .., K_n\}$. Let $K \in \{K_1, .., K_n\}$ and $\varphi = \sigma\rho\sigma$. We show that there is a renaming substitution ρ' and a substitution ψ, such that $K\rho'\psi = \neg L\psi$. Let ρ' be a renaming substitution with $dom(\rho') = dom(\sigma) \cap dom(\varphi)$ and $cod(\rho') \cap \mathcal{V}(L,K) = \emptyset$. Define the substitution ψ with $dom(\psi) = dom(\sigma) \cup dom(\varphi) \cup cod(\rho')$ by $\psi|_{dom(\sigma)} = \sigma$ and $\psi|_{dom(\varphi) \setminus dom(\sigma)} = \varphi$ and $(x\rho')\psi = x\varphi$ for $x\rho' \in cod(\rho')$. Then we have $x\psi = x\sigma$ for $x \in \mathcal{V}(L)$ and $y\rho'\psi = y\varphi$ for $y \in \mathcal{V}(K)$, hence $L\psi = L\sigma = K\varphi = K\rho'\psi$.

b) and c) are obvious. ■

The next lemma shows that clauses, which only produce parent subsumed clauses, are tautologies.

3.3 Lemma:

Let D be a clause.

a) Suppose for all clauses C the following holds: $C \rightarrow_D R$ implies that C and R are variants. Then D is a tautology, $|D| = 2$, and D is function and constant free.

b) Suppose for all clauses C the following holds: $C \rightarrow_D R$ implies that R is an instance of C. Then D is a tautology and $|D| = 2$.

c) Suppose for all clauses C the following holds: $C \rightarrow_D R$ implies that C subsumes R. Then D is a tautology.

Proof: a) Let $L = Pt_1...t_n$ be an arbitrary literal of D. Let C be the unit clause consisting of the literal $M = \neg L\rho$, for some variable renaming substitution ρ. Then there is a resolvent R of C and D and R is a subset of D. From the assumption follows that C is a variant of the resolvent R, that is, there is a renaming substitution σ, such that $R = M\sigma = \neg L\rho\sigma$. Thus C is the binary clause LR. Let μ be the renaming substitution $\rho\sigma$. If μ is not the identity, then there is some $x \in \mathcal{V}(L)$ such that $x\mu = x'$ with $x \neq x'$. Let a be an arbitrary constant not occurring in C nor D and let C' be the clause consisting of the literal $M' = \neg L\{x \rightarrow a\}$. Let R' be the resolvent of C' and D. Then, as a occurs in M', but not in $R' = \neg L\mu\{x \rightarrow a\} = \neg L\mu$, we obtain $R' \neq M'$, which is a contradiction. Hence μ is the identity and $R = \neg L$. If D contains a constant or function symbol g, then g occurs in both literals of D, hence also in any resolvent of D. Let C_1 be the unit clause consisting of the literal $Px_1..x_n$, with $x_i \notin \mathcal{V}(D)$ for all $i \in \{1..n\}$. Then C_1 has a resolvent R_1 with D, however, as g occurs in R_1, but not in C_1, the clauses R_1 and C_1 cannot be variants, which is a contradiction to the assumption. Thus D cannot contain function nor constant symbols.

b) The same argumentation as for the first part of a) holds, except that μ is now an arbitrary substitution, and x' must be replaced by a term t.

c) If C is any clause resolving with D to some resolvent R, then C subsumes R, that is, there exists some subset R' of R, which is an instance of C. The assertion now follows from part b).

■

In the following we will generalize lemmata 3.2 and 3.3 to ancestor subsumption instead of parent subsumption, that is, we want to determine those clause sets $\mathcal{D}$, which - possibly or only - produce (ancestor) subsumed clauses. It will turn out that the appropriate generalization of self-resolving and tautologous clauses are *cyclic* clause sets.

We generalize the deduction relation to clause sets $\mathcal{D}$, by $C \rightarrow_{\mathcal{D}} R$, iff there is a sequence $C \rightarrow_{D_1} C_1 \rightarrow_{D_2} ... \rightarrow_{D_n} R$ such that $\{D_1,..,D_n\} = \mathcal{D}$.

In the following we deal with finite *directed graphs* $G = (\mathcal{N}, \Lambda)$, whose nodes are labelled with clauses and whose links are *R-links* (i.e. links joining resolvable literals) labelled with substitutions. We do not distinguish between a node and its label.

A **path** from node N_1 to N_n in a directed graph is an alternating sequence $(N_1, l_1, ...,N_{n-1}, l_{n-1}, N_n)$ of nodes and links, such that each node N_i is of the following form (figure 1), and there is a common instance σ of all substitutions.

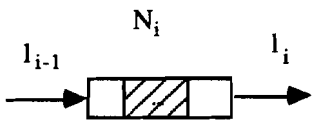


fig. 1

The path from N_1 to N_n is called **weakly cyclic**, if there is also a link from N_n to N_1 with substitution τ, a weakly cyclic path is called **cyclic**, if σ is compatible with τ. A directed graph is called cyclic, if it contains a cyclic path.

A **branching node** in a directed graph is a node $N = LK_1..K_n$ with $n \geq 0$ of the following form (figure 2)

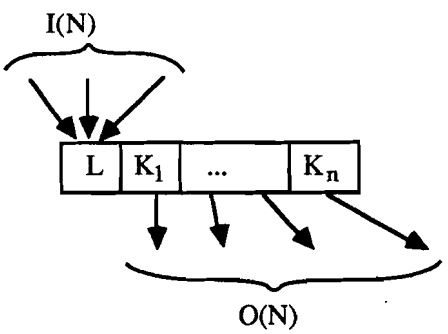


fig. 2

Each literal K_j possesses exactly one outgoing link. The **outdegree** $O(N)$ of a branching node N is the number of outgoing links, the **indegree** $I(N)$ is the number of incoming links and $I(N) \geq 0$ and $O(N) \geq 0$ hold. L is called the **I-literal**, each K_j is an **O-literal** of N.
A **semicycle** G is a directed graph, which satisfies the following conditions.

a) Each node N of G is a branching node with $I(N) > 0$ and $O(N) \geq 0$.
b) There is a node N_0 such that each cyclic path of G passes N_0.
c) All occurring substitutions are compatible, with common instance χ. The substitution χ is called the cycle substitution of G.

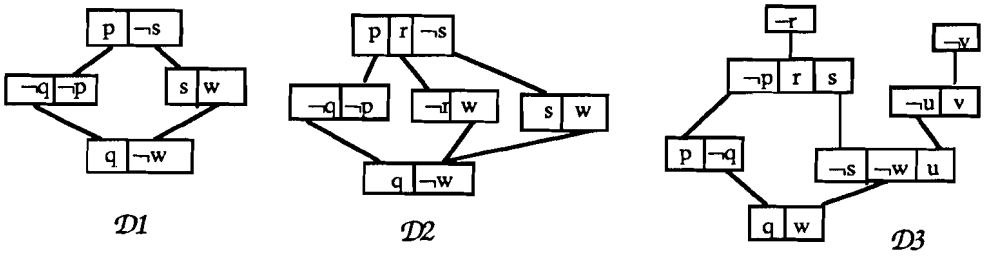
The fact that each node N of a semicycle has an incoming link guarantees the existence of a cyclic path in a semicycle. Let $\mathcal{D}$ be a semicycle consisting of only one clause C. Then σ is the identity substitution and C is self-resolving. $\mathcal{D}$ is called a **cycle**, if each node has exactly one successor.

Usually, clause graphs are *undirected* graphs. We have chosen the representation with directed graphs because this representation allows a much more concise definition of semicycles than the undirected version. It should be noted, however, that the orientation of the links can be chosen arbitrarily. Any assertion "There is a semicycle $(\mathcal{D},\Lambda)$..." should be read as "There is a clause set $\mathcal{D}$, and a set Λ of R-links between clauses of $\mathcal{D}$ which can be directed in a way, such that $(\mathcal{D},\Lambda)$ is a semicycle..."

A cycle is just what Shostak (1976) and (1979) calls a *loop*. This notion also corresponds to the notion of *recursive predicates* in the terminology of deductive databases (Vieille 1987, Ohlbach 1988) and logic programming.

3.4 Example:

The clause sets $\mathcal{D}_1$, $\mathcal{D}_2$, and $\mathcal{D}_3$, which are shown in figure 3, are semicycles; $\mathcal{D}_1$ is a cycle. It can be seen that each node of a cycle may be chosen to be the special clause N_0. As to semicycles, still several, but in general not all nodes have this property. For instance the clauses $\neg rw$ and sw cannot be chosen to be N_0.



$\mathcal{D}1$ $\mathcal{D}2$ $\mathcal{D}3$

fig. 3

The previous examples illustrated that in general there are several possibilities to choose a clause of a semicycle to be the special clause N_0.

Next we define the projection of a derivation onto some literal. This notion is a purely technical one, it is needed for the proof of our main result.

3.5 Definition:

Let $\Gamma = C_1 \rightarrow_{D_1} \ldots \rightarrow_{D_n} C_n$ and let $L \in C_1$. The sequence $(\Gamma)_L = (C_1)_L \rightarrow \ldots \rightarrow (C_n)_L$ defined by

$(C_1)_L = L$
$(C_i)_L = (C_{i-1})_L$, if $(C_{i-1})_L$ does not resolve with D_{i-1},
the resolvent of $(C_{i-1})_L$ with D_{i-1} otherwise

is called the **projection** of Γ onto L. A step $(C_i)_L \rightarrow (C_{i+1})_L$ is called trivial, iff $(C_i)_L = (C_{i+1})_L$.
A deduction $(C_1)_L \rightarrow \ldots \rightarrow (C_n)_L$ is obtained in a canonical way from $(\Gamma)_L$ by omitting those elements $\rightarrow_{D_i} (C_i)_L$ which satisfy $(C_i)_L = (C_{i-1})_L$.

3.6 Example:

Let $\Gamma = PQ \rightarrow_{\neg PR} QR \rightarrow_{\neg QS} RS$. Projecting Γ on the literals P and Q, respectively, yields the sequences

$$(\Gamma)_P = P \rightarrow_{\neg PR} R \rightarrow_{\neg QS} R \text{ and } (\Gamma)_Q = Q \rightarrow_{\neg PR} Q \rightarrow_{\neg QS} S$$

3.7 Lemma:

Let $\Gamma = C \rightarrow C_1 \rightarrow \ldots \rightarrow C_n$ and let $L \in C$. Then

$$\bigcup_{L \in C} (C_i)_L = C_i$$

holds for each $i \in \{1,..,n\}$.

Proof: The lemma is proved easily by induction on n. ■

3.8 Lemma:

Let $C = \{P\}$ and $R = \{Q\}$ be ground unit clauses, let $\mathcal{D} = \{D_1,..,D_n\}$ be a set of ground clauses such that $\Gamma = C \rightarrow_{D_1} ... \rightarrow_{D_n} R$ holds. Let $D_{n+1} := \neg R$ and $\mathcal{D}' = \mathcal{D} \cup \{D_{n+1}\}$.
Then there is an acyclic directed graph $G = (\mathcal{D}', \Lambda)$, which satisfies:

(i) Each element of $\mathcal{D}'$ is a branching node.

(ii) $I(D_1)=0$, $O(D_1)>0$, $O(D_{n+1})=0$ and $I(D_{n+1})>0$ holds.

(iii) $I(D_i)>0$ holds for $i \in \{2,..,n\}$.

Proof: By induction on n. Let $D_1 = \neg P Q_1..Q_m$ and let $\Delta = Q_1..Q_m \rightarrow_{D_2} ... \rightarrow_{D_n} R$. Take any $i \in \{1,..,m\}$ and consider the projection $(\Delta)_{Q_i} = Q_i \rightarrow \cdots \rightarrow S$. Then, according to lemma 3.7, either S=R or S is the empty clause. If S is the empty clause, then the last not empty clause in the sequence must be a unit. In either case, we obtain a deduction $Q_i \rightarrow_{D_{i,1}} ... \rightarrow_{D_{i,m}} S_i$ with unit clauses Q_i and S_i and $\{D_1',..,D_m'\} \subseteq \{D_2,..,D_n\}$. This deduction satisfies the conditions of the lemma and has length smaller than n. Hence there is a acyclic directed graph $G_i = (\mathcal{D}_i, \Lambda_i)$ which satisfies (i) to (iii). Then we obtain a directed acyclic graph G from the union of the G_i's by adding the node D_1 and the links from the literal Q_i in the clause D_1 to the literal $\neg Q_i$ in the clause $D_{i,1}$. This implies $I(D_{i,1})>0$ and $O(D_1)>0$, which shows that (iii) and the first half of (ii) holds for G. The assertion (i) holds trivially for G. According to lemma 3.7, there is at least one j such that $(\Delta)_{Q_j} = Q_j \rightarrow ... \rightarrow R$ holds. Hence the induction hypothesis implies $O(R)=0$ and $I(R)>0$, which implies the second half of (ii). ■

3.9 Lemma:

Let C be a ground unit clause and $\mathcal{D}=\{D_1,..,D_n\}$ be a set of ground clauses with $C \rightarrow_{D_1} C_1 \rightarrow_{D_2} ... \rightarrow_{D_n} D$. Then there is a graph $G=(\mathcal{D},\Lambda)$ that contains a path from D_1 to D_n.

Proof: By induction on n. Let Q be any literal of D_1 with $Q \neq \neg C$. Consider the projection $(\Gamma)_Q = Q \rightarrow ... \rightarrow (D)_Q$ of the deduction $\Gamma = C_1 \rightarrow_{D_2} ... \rightarrow_{D_n} D$. Then by the induction hypothesis there is a graph $G' = (\{D_2,..,D_n\}, \Lambda')$ containing a path from D_2 to D_n. By adding the node D_1 and the link from the literal Q in D_1 to the literal $\neg Q$ in D_2 we obtain a path from D_1 to D_n. ■

The following theorem provides the characterization of ancestor subsumption for the ground case.

3.10 Theorem:

Let C be a ground unit clause, let $\mathcal{D}=\{D_1,..,D_n\}$ be a set of ground clauses such that $C \rightarrow_{D_1} C_1 \rightarrow_{D_2} ... \rightarrow_{D_n} R$ holds.

a) If $R = C$, then there is a semicycle $G = (\mathcal{D},\Lambda)$ and D_1 can be chosen to be the special node N_0 of G.

b) If R is subsumed by C, then there is a cyclic directed graph $G = (\mathcal{D},\Lambda)$.

Proof: a) By adding the node $D_{n+1} = \neg R$ to $\mathcal{D}$, we obtain an acyclic directed graph G' satisfying conditions (i) to (iii) of lemma 3.8. In particular, D_1 is the only node N of G' with $I(N)=0$. Moreover, the nodes D_1 and D_{n+1} have the same I-literal $\neg C = \neg R$. By taking all the

links that go into D_{n+1} and directing them instead to D_1, we obtain a cyclic directed graph $G=(\mathcal{D},\Lambda)$, with $I(D)>0$ for all $D\in\mathcal{D}$. Obviously, each cyclic path of G passes the node D_1. Therefore conditions a) to c) of the definition of a semicycle are satisfied (with identity substitutions).

b) There is a deduction $C\rightarrow_{D_1}C_1\rightarrow_{D_2}...\rightarrow_{D_n}C_n = R$, and C is a literal of R. Let $D = \neg C$, then there is a deduction $C\rightarrow_{D_1}C_1\rightarrow_{D_2}...\rightarrow_{D_n}C_n\rightarrow_{D}R'$ and according to lemma 3.9 there is a graph G' containing a path from D_1 to D. By taking the link that goes to the literal $\neg C$ in the clause D and directing it to the literal $\neg C$ in D_1 we obtain a graph G with a cyclic path. ■

The following theorems 3.12 and 3.13 provide the appropriate generalizations of theorem 3.10 to the non-ground case. It turns out that those clause sets have a "cyclic" structure, varying from the weakest form for clause sets producing some ancestor subsumed resolvent to the strongest form of a cycle for clause sets producing only copies as resolvents.

Let $G=(\mathcal{D},\Lambda)$ be a semicycle with special node N_0 and let n be the length (i.e. the number of nodes) of the longest cyclic path in G. Let the clauses $C_0,..,C_{n-1}$ be defined as follows: Let $C_0 = N_0$ and for $i\in\{1,..,n-1\}$ let C_i be the clause resulting from resolving on all the links outgoing from C_{i-1} except those links going into N_0. Let L be the I-literal of N_0 and let $K_1,..,K_m$ be the O-literals of the predecessors of N_0. Then $D:=C_{n-1}$ can be written as $(LK_1..K_m)\sigma$ where σ is a common instance of the substitutions that were used in the resolution steps (see fig. 4). Obviously, the clause D resulting from this construction depends on the choice of the special node. We call D the *C_0-reduct* of G.

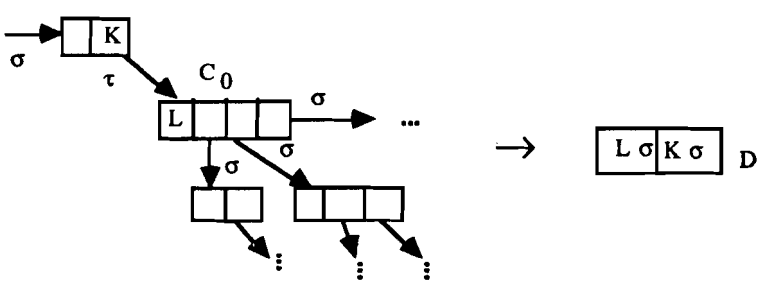


fig. 4

3.11 Lemma:

Let L and K be literals and σ be a substitution.

a) If $L\sigma$ and $K\sigma$ are weakly unifiable, then also L and K. Moreover, there is a substitution θ and a renaming ρ, such that $L\sigma\theta = K\rho\sigma\theta$ and $\mathrm{dom}(\sigma)\cap\mathcal{V}(\mathrm{cod}(\rho)) = \mathrm{dom}(\rho)\cap\mathcal{V}(\mathrm{cod}(\sigma)) = \emptyset$.

b) If there is a substitution φ and a renaming ρ, such that $L\rho\varphi = K\varphi$ holds, then $L\rho'\varphi = K\rho'\varphi$ for some renaming ρ'.

Proof: a) see (Herold 1983), lemma III.9.

b) see (Herold 1983), lemma III.8. ■

3.12 Theorem:

Let C be a unit clause and let $\mathcal{D}= \{D_1,..,D_n\}$ be a set of clauses such that $C \rightarrow_{D_1} C_1 \rightarrow_{D_2} ... \rightarrow_{D_n} R$ holds.

a) If R is a variant of C, then there is a semicycle G = ($\mathcal{D}$,Λ).

b) If R is subsumed by C, then there is a weakly cyclic graph G = ($\mathcal{D}$,Λ).

Proof: a) Let C' be a common ground instance of C and R, $\mathcal{D}'$ a set of ground instances of the clauses in $\mathcal{D}$ such that $C' \rightarrow_{\mathcal{D}'} C'$ holds. Then theorem 3.10 determines the "cyclic" structure of $\mathcal{D}$, and the clause D_1 represents the special node N_0. What remains to show is the compatibility of the substitutions.

Let the unifier σ_i belong to the step $C_{i-1} \rightarrow_{D_i} C_i$. Then $\{\sigma_1,\sigma_2,..,\sigma_n\}$ is a set of compatible substitutions (see Herold 1983). In particular, the substitutions $\sigma_2,..,\sigma_n$ are compatible, with some common instance σ. Let D be the D_1-reduct of G. As the result of a sequence of resolution steps does not depend on their order, the relation $C \rightarrow_D R$ must hold (see figure 5). Thus, according to lemma 3.2, D is a self-resolving clause of the form $L'K_1'..K_n'$, with a renaming ρ and a substitution τ, such that $L'\tau = \neg K_i'\rho\tau$ for each $i \in \{1,..,n\}$ holds. Moreover, we have $L'=L\sigma$ and $K_i'=K_i\sigma$, where L is the I-literal of D_1 and the K_i are the O-literals of the predecessors of D_1. Take any $K \in \{K_1,..,K_n\}$. We have $L\sigma\tau = \neg K\sigma\rho\tau$ and from lemma 3.11 a) follows that there is some substitution θ with $L\sigma\theta = \neg K\rho'\sigma\theta$ for some renaming ρ', and from part b) of the same lemma follows that $L\rho''\sigma\theta = \neg K\rho''\sigma\theta$ for some renaming ρ''. Let $\lambda = \rho''\sigma\theta$, then λ is a unifier of L and $\neg K$, hence λ is an instance of the most general unifier φ of L and $\neg K$. We show that λ and φ are compatible substitutions. Let $\lambda = \varphi\lambda'$. The renaming ρ'' can be chosen, such that $dom(\sigma) \cap \mathcal{V}(cod(\rho'')) = dom(\rho'') \cap \mathcal{V}(cod(\sigma)) = \emptyset$, that is, $\sigma\rho = \rho\sigma$.This implies $\sigma\lambda = \lambda = \varphi\lambda'$, hence φ and σ have a common instance, that is, they are compatible. We have shown that σ is compatible with each unifier of a link going into D_1, that is, all occurring unifiers are compatible and condition c) of the definition of a semicycle is satisfied.

b) is proved analogously to a) ■

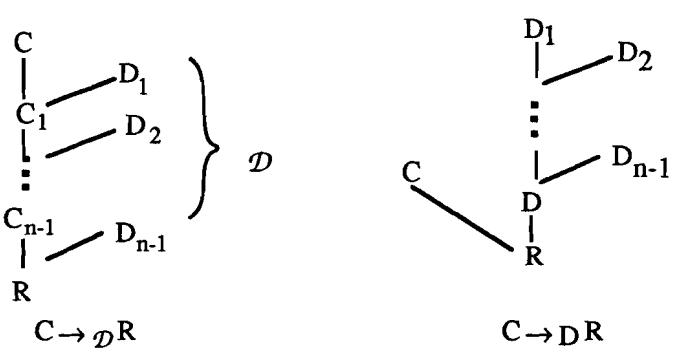


fig. 5

3.13 Theorem:

Let $\mathcal{D}$ be a set of ground clauses.

Suppose for all clauses C the following holds: $C \rightarrow_{\mathcal{D}} R$ implies that $C = R$. Then there is a cycle $G = (\mathcal{D},\Lambda)$.

Proof: We proceed by induction on the length of $\mathcal{D}$. As a tautology is a particular cycle, the case $|\mathcal{D}|=1$ follows from lemma 3.3. Let $|\mathcal{D}|>1$. Theorem 3.10 implies that there is a semicycle $G = (\mathcal{D},\Lambda)$. We show that G is a cycle. Let $D,D' \in \mathcal{D}$ such that D' is a successor of D. According to the definition of a semicycle, the clauses D and D' are resolvable. Let R be their resolvent and let $\mathcal{D}' = \mathcal{D} \setminus \{D,D'\} \cup \{R\}$, which is illustrated by figure 6.

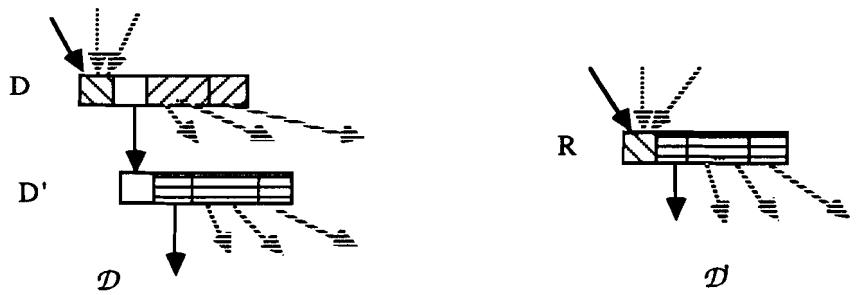


fig. 6

Since $C \rightarrow_{\mathcal{D}'} R$ implies $C \rightarrow_{\mathcal{D}} R$, the set $\mathcal{D}'$ satisfies the assumptions of the theorem and from the induction hypothesis follows that $\mathcal{D}'$ is the set of nodes of a cycle. This implies that $|C|=2$ for each $C \in \mathcal{D}$, hence also D and D' are clauses of length two, which implies that $\mathcal{D}$ is a cycle. ■

The proof of theorem 3.12 showed that semicycles are structures that can be "reduced" to self-resolving clauses. Those particular semicycles that can be reduced to tautologies, are characterized by

3.14 Definition:

Let $G=(\mathcal{D},\Lambda)$ be a semicycle with special node C_0. Let τ be the most general common instance of all the substitutions of the R-links going into C_0 and let σ be a common substitution for all other links of Λ. The semicycle G is called **complete**, if σ is an instance of τ, that is $L\sigma = K\sigma$ for all literals L and K joined by a link going into C_0.

Let G be a complete semicycle as in definition 3.14. Let $D := (LK_1..K_m)\sigma$ be the C_0-reduct of G. As for each $i \in \{1,..,m\}$, K_i and L are complementary unifiable under τ, and σ is an instance of τ, D must be a tautology.

3.15 Theorem:

Let $\mathcal{D}$ be a set of clauses.

a) Suppose for all clauses C the following holds: $C \rightarrow_{\mathcal{D}} R$ implies that C and R are variants. Then there is a complete cycle $G = (\mathcal{D},\Lambda)$, all clauses of which are function and constant free.

b) Suppose for all clauses C the following holds: $C \rightarrow_{\mathcal{D}} R$ implies that R is an instance of C. Then there is a complete cycle $G = (\mathcal{D},\Lambda)$.

c) Suppose for all clauses C the following holds: $C \rightarrow_{\mathcal{D}} R$ implies that C subsumes R. Then there is a complete semicycle $G = (\mathcal{D},\Lambda)$.

Proof: a) The cyclic structure of $G=(\mathcal{D},\Lambda)$ is provided by theorem 3.13. Let $\mathcal{D}=\{D_1,..,D_n\}$ and choose any D_k to be the special node of G. As in the proof of theorem 3.12, let $D = (L_kK_{k-1})\sigma$ be the D_k-reduct of G. Then the relation $C \rightarrow_D C$ must hold for each clause C that resolves with D. From theorem 3.3 follows that D is function and constant free, which shows that the same holds for L_k and K_{k-1}. As k was chosen arbitrary, all elements of D are function and constant free. As D is a tautology, σ is a unifier of L_k and K_{k-1}. The link from K_{k-1} to L_k has most general unifier τ, hence σ is an instance of τ, that is, the cycle G is complete.
b) and c) are proved analogously. ■

Note that the definition of the deduction relation $\rightarrow_{\mathcal{D}}$ does not capture ancestor resolution, which is part of complete linear strategies.

3.16 Example:

Let $\mathcal{D}$ be the clause set that is shown in figure 7.

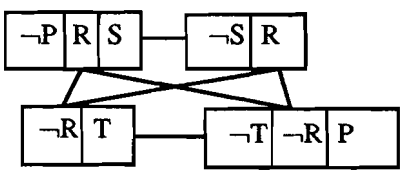


fig. 7

$\mathcal{D}$ allows a linear deduction with ancestor resolution as follows:

$$P \rightarrow_{\neg PRS} RS \rightarrow_{\neg SR} R \rightarrow_{\neg RT} T \rightarrow_{\neg T \neg RP} \neg RP \rightarrow_R P$$

where the last step is an ancestor resolution step. This is an ancestor resolution situation, however, $\mathcal{D}$ is not a semicycle.

The following lemma shows that it is sufficient to look for cycles in an initial clause set during a resolution deduction, as "new" cycles cannot be generated by resolution. For any clause set S, $\Re(S)$ denotes the resolution closure of S, that is the smallest clause set containing S and closed under the resolution operation.

3.17 Lemma:

Let S be a set of clauses. Then S contains a semicycle, iff $\Re(S)$ contains a semicycle.

Proof: Let R be a resolvent of clauses C and D with $C, D \in S$ and assume that R is a node of a semicycle $G=(\mathcal{D},\Lambda)$. We show that C and D are also nodes of a semicycle.

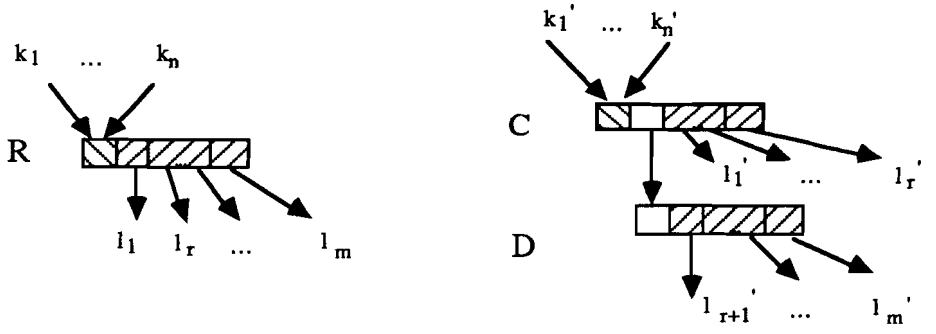


fig. 8

Each link k_i and l_j of G going into or coming from the node R must be inherited from some link k_i' and l_j', respectively, as shown in figure 8. The diagram illustrates that these links are links of a semicycle G', which contains nodes C and D. ■

4 Removing cycles from clause sets

Besides of being the very reason for the undecidability of first order logic, cycles in clause sets also turned out to be a main source of reduncancy for clause set resolution. The question thus arises as to what extent cycles can be excluded from producing lots of unneeded clauses. A first approach to this question is due to Bibel (1981). He showed that under certain conditions, similar to those allowing the deletion of tautologies in clause graphs, cycles can be removed from clause sets. We will consider here two approaches, both restricted to cycles. However, we will also show by means of SAM's lemma, that in particular cases there exists also an appropriate treatment of semicycles. Both approaches are based on the observation, that a cycle $\{\neg L_1L_2, \neg L_2L_3,...,\neg L_nL_1\}$ represents a set $\{L_1 \Rightarrow L_2, L_2 \Rightarrow L_3,..,L_n \Rightarrow L_1\}$ of implications, and thus expresses that the literals L_i are pairwise logically equivalent. Logical equivalence can be treated in a very similar way to equality, hence the well-known methods of equational reasoning, like demodulation, paramodulation or theory resolution apply to cycles.

The first idea in order to shut down this source of reduncancy is to use cycles not directly for resolving. Instead, they are taken as *literal demodulators* (Wos 1967): Cycles, expressing the equivalence of literals, can be transformed into rules, in the same way as equations can be directed yielding rules. Consider for example a clause set containing among others the two clauses $\neg P \vee Q$ and $\neg Q \vee P$. These two clauses represent the formula $P \equiv Q$, which can be directed yielding for instance the rule $P \rightarrow Q$. Application of this rule means substituting Q for each occurrence of P and $\neg Q$ for each occurrence of $\neg P$. (Note that this effect can be achieved also with an ordering restriction (Loveland 1978), which precludes resolution with the Q and $\neg Q$ literal in the two clauses.)

However, the method of literal demodulators applies only for cycle clauses that can be directed, such as ground clauses or clauses with different predicate symbols. Thus, a lot of relevant clause sets does not allow this approach. For instance, the (cyclic) clause $\neg Pxy \vee Pyx$, expressing the symmetry of P, cannot be directed. The following example shows how literal demodulation accounts for an improvement of resolution's efficiency.

Another approach, which overcomes the problem with directing equivalences, is based on the idea of "compiling" cycles into a theory serving as the basis for *theory resolution.* Theory resolution, first proposed by Stickel (1985), is a generalization of ordinary resolution. The literals resolved upon need not be syntactically complementary, it is sufficient for them to be complementary under some theory. If S is any set of clauses, then the theory of S is the set $\mathcal{T}_S=\{C \mid C \text{ is a clause with } S \models C\}$. A clause set S' is *complementary* under some theory $\mathcal{T}$, if $\mathcal{T} \cup S'$ is unsatisfiable. In particular, each clause $C = L_1...L_n$ defines a theory $\mathcal{T}_C$ in which, for instance, the set of literals $\{\neg L_1,..,\neg L_n\}$ is complementary. Taking clauses (or clause sets) as a particular theory to perform theory resolution is essentially the same idea as Ohlbach's (1988) *link resolution.* For an arbitrary clause set $\mathcal{S}$, the set $\mathcal{T}_\mathcal{S}$ is not in general computable. Yet, in order to perform theory resolution on a clause set $\mathcal{S}$, it is sufficient to compute the resolution closure $\mathfrak{R}(\mathcal{S})$ of $\mathcal{S}$.

4.1 Lemma:

Let $\mathcal{S}$ be a set of clauses and $D \in \mathcal{T}_\mathcal{S}$ be a clause, which is not a tautology. Then there is some clause $C \in \mathfrak{R}(\mathcal{S})$, such that C subsumes D. ■

In order to find the substitutions σ, which make a set $\{L_1,..,L_n\}$ of literals complementary under the theory $\mathcal{T}_\mathcal{S}$, or, equivalently, which map the clause $D:=\{\neg L_1,..,\neg L_n\}$ onto an element of $\mathcal{T}_\mathcal{S}$, it is sufficient, according to the previous lemma, to find some σ and a clause $C \in \mathfrak{R}(\mathcal{S})$, such that C subsumes $D\sigma$. Thus, having a finite resolution closure $\mathfrak{R}(\mathcal{S})$, the set $\mathcal{S}$ guarantees the number of $\mathcal{T}_\mathcal{S}$-resolvents of two clauses to be finite. In particular, this is the case for cycles that produce only copies as resolvents, since these cycles are function free. The next lemma shows that the theory of a complete, function free cycle $\mathcal{D}$ with cycle substitution σ can be completely described by the pairwise equivalence of all I-literals of $\mathcal{D}\sigma$.

4.2 Lemma:

Let $\mathcal{D}$ be a complete, function free cycle with cycle substitution σ. Then $\mathfrak{R}(\mathcal{D}) = \{\neg L\sigma K\sigma \mid L$ and K are I-literals occurring in $\mathcal{D}\}$. ■

4.3 Examples:

a) Let $\mathcal{D} = \{\neg Pxy\ Qxy,\ \neg Quv\ Puv\}$. $\mathcal{D}$ is a complete and function free cycle with cycle substitution $\sigma=\{x\rightarrow u, y\rightarrow v\}$. The theory of $\mathcal{D}$ consists of the formula Puv ≡ Quv.
b) Let $\mathcal{D}=\{\neg Pxy\ Pyx\}$. $\mathcal{D}$ is function free, but not complete. By adding a copy of the clause $\neg Pxy\ Pyx$, a complete cycle $\mathcal{D}'$ is obtained.
Then $\mathfrak{R}(\mathcal{D}) = \mathfrak{R}(\mathcal{D}') = \{\neg Pxy\ Pyx,\ \neg Pxy\ Pxy\}$ is finite and thus the number of $\mathcal{T}_\mathcal{D}$-resolvents of any two clauses is finite, too.
c) Let $\mathcal{D}=\{\neg Pf(f(x,y),\ z)\ Pf(x,f(y,z))\}$.
Then $\mathfrak{R}(\mathcal{D})= \mathcal{D} \cup \{\neg Pf(f(f(x,y),z),u)\ Pf(x,f(y,f(z,u))),\ ...\}$ is infinite and also the number of $\mathcal{T}_\mathcal{D}$-resolvents of two clauses is infinite in general.

Taking into account the tight connection between $\mathcal{E}$-unification (see Bürckert, Herold & Schmidt-Schauß 1987) and theory resolution for equational theories, the previous examples'

behaviour is not surprising, as the number of most general unifiers under the theory of commutativity is finite, whereas it is infinite under associativity (see Herold & Siekmann 1985).

5 Conclusion

In this chapter we want to show by means of two examples, how the information about cycles in clause graphs can be used to reduce the search space in resolution theorem proving. In the first example literal demodulation is used to reduce the generation of redundant clauses.

5.1 Example:

Let n be any even natural number and let S be the clause set $\{C_1, C_2\}$ with

$C_1 = Px \vee Pfx$,

$C_2 = \neg Px \vee \neg Pf^n x$.

Except from the rather trivial cases n=2 and n=4, this clause set is not easy to refute. For instance, the Markgraph Karl theorem prover (Karl 1984) failed for numbers n>10. However, the problem has a straightforward solution based on literal demodulation: First, the following four resolvents are deduced:

$C_3 = \quad Pfx \vee \neg Pf^n x$

$C_4 = \neg Px \quad \vee \; Pf^{n-1} x$

$C_5 = \neg Px \quad \vee \; Pf^{n+1} x$

$C_6 = \quad Px \quad \vee \neg Pf^{n+1} x$

C_5 and C_6 form a (complete) cycle, expressing the equivalence $Px \equiv Pf^{n+1}x$, which can be directed resulting in the rule $R_1 = Pf^{n+1}x \rightarrow Px$. In the same way C_3 and the instance $\neg Pfx \vee Pf^n x$ of C_4 can be transformed into a rule $R_2 = Pf^n x \rightarrow Pfx$. Using R_2, clause C_2 can be rewritten into $C_2' = \neg Px \vee \neg Pfx$. Interreducing R_1 with R_2 yields the rule $R_3 = Pf^2 x \rightarrow Px$, and finally we obtain $Pfx \rightarrow Px$ by repeatedly applying R_3 to R_1 (note that n is even). This rule reduces C_2' to $\neg Px$ and C_1 to Px, which concludes the refutation. The length of this refutation has order O(n/2).

The next example illustrates the application of theory resolution to problems containing cyclic structures.

5.2 Example

Translating SAM's lemma into a clause set, avoiding the use of the equality predicate (see Wos 1988), results in a clause set S that consists of the following 11 units

$\{min(0\ x\ 0),\ max\ (x\ 0\ x),\ max(a\ b\ d_1),\ min(d_1\ c_1\ 0),\ min(a\ b\ d_2),\ min(d_2\ c_2\ 0),$
$min(c_2\ b\ e),\ min(c_2\ a\ f),\ max(c_1\ e\ g),\ max(c_1\ f\ h),\ \neg min(g\ h\ c_1)\}$

and 6 non unit clauses:

(C_1) $\neg min(x\ y\ z)\ min(y\ x\ z)$

(C_2) $\neg max(x\ y\ z)\ max(y\ x\ z)$

(C_3) $\neg min(x\ y\ u)\ \neg min(y\ z\ v)\ \neg min(x\ v\ w)\ min(u\ z\ w)$

(C_4) $\neg min(x\ y\ u)\ \neg min(y\ z\ v)\ \neg min(u\ z\ w)\ min(x\ v\ w)$

(C_5) $\neg max(x\ y\ z)\ min(x\ z\ x)$

(C_6) $\neg min(x\ z\ x)\ \neg max(x\ y\ x_1)\ \neg min(y\ z\ y_1)\ \neg max(x\ y_1\ z_1)\ min(z\ x_1\ z_1)$

In the following we describe a refutation of this clause set using positive hyperresolution together with theory resolution, as in chapter 4. The cyclic clauses C_1 and C_2, describing the symmetry of the *min* and *max* predicates, can be used for theory resolution, as in example 4.3.b). The same holds for the clause C_5, which is neither self-resolving nor a member of a cycle. The clauses C_3 and C_4, describing the associativity of the *min* predicate, form a semicycle, which produces copies in the following way: Let D_1, D_2, D_3 be clauses that resolve with C_3 to the unit clause D_4. Then (D_1, D_2, D_4) resolves with C_4 to a copy of D_3. The resolution closure of the semicycle $\mathcal{D}=\{C_3, C_4\}$ is not finite, this cycle thus cannot be used directly for theory resolution. But C_3, C_4 can be used to produce instances of $\mathcal{T}_\mathcal{D}$ in the way, as shown in figure 9:

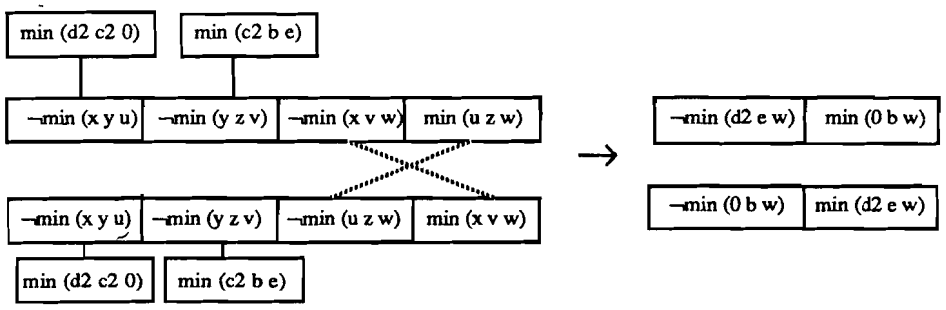


fig. 9

Only the relevant links are shown in this figure. The dotted links belong to the cycle. The result of resolving $\{C_3,C_4\}$ with the two units is a complete cycle, which is added to the theory box. A resolution step of this particular kind will be abbreviated by the following diagram (figure 10), where a cycle is represented by an equivalence within a bold faced box.

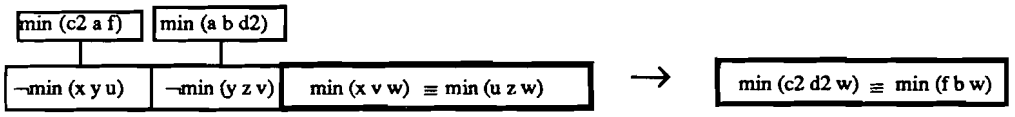
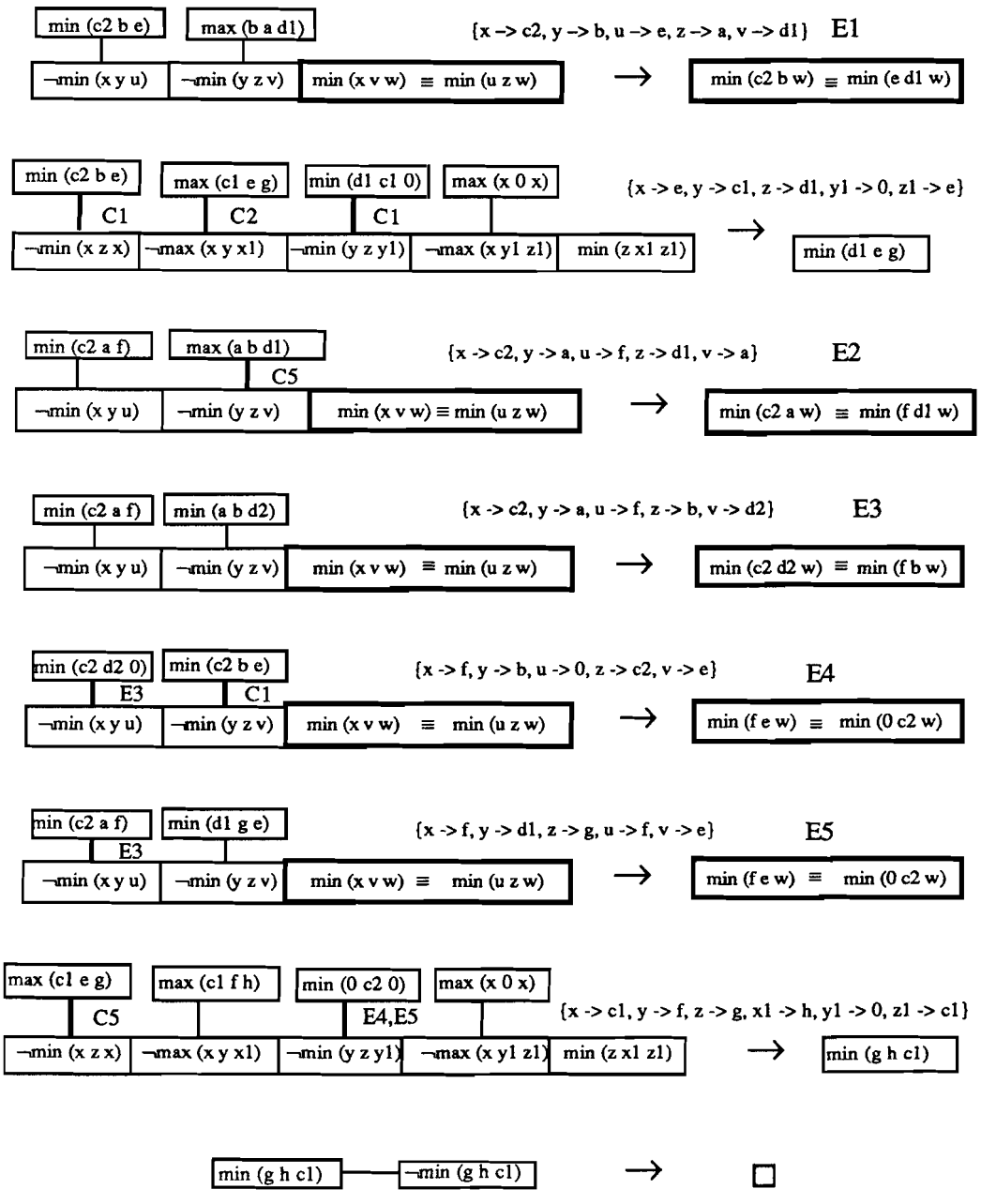


fig. 10

Diagram 10 shows that such a resolution step in fact is a way to obtain a rewrite rule from a *conditional* rewrite rule.

Proceeding this way, only the clause C_6 is needed to produce "ordinary" resolvents. Of course, there is also the possibility to produce copies of already retained cycles. Taking into account these reduncancies, a total 660 copies or instances of already present clauses are generated in the proof.

The proof of SAM's lemma, as illustrated below, consists of 8 steps. The bold faced links denote theory links, the used theories are denoted by their clauses. For instance, a link numbered with C_1 denotes a theory link under the theory of C_1, that is the commutativity of *min*.



Acknowledgement

I would like to thank my colleague Norbert Eisinger for thoroughly reading this manuscript. His comprehensive knowledge of the field, particularly about many examples and counter-examples that arise in theorem proving, resulted in improvements of the paper.

References

Bibel, W. (1981). On Matrices with connections. *Journal of the ACM* **28/4**, 633 - 645.

Bürckert, H.-J., Herold, A. & Schmidt-Schauß, M. (1987). On Equational Theories, Unification and Decidability. in: Lescanne, P. (ed): *Proc. of 2nd Conference on Rewriting Techniques and Applications*, Bordeaux, France. Springer LNCS 256, 204 - 215.

Chang, C.L. & Lee, R.C. (1973). Symbolic Logic and Mechanical Theorem Proving. Academic Press. New York.

Eisinger, N. (1981). Subsumption and Connection Graphs. *Proceedings of the 7th IJCAI*, Vancouver, 480 - 486.

Eisinger, N. (1988). Completeness, Confluence, and Related Properties of Clause Graph Resolution. PhD thesis and SEKI-Report SR-88-07, Universität Kaiserslautern.

Guard, J. et al (1969). Semi-Automated Mathematics. *Journal of the ACM*. **16**, 49 - 62.

Herold, A. (1983). Some Basic Notions of First Order Unification Theory. Internal Report, Universität Kaiserslautern.

Herold, A. & Siekmann, J. (1985). Unification in Abelian Semigroups. Memo-SEKI-85-III. Universität Kaiserslautern.

Loveland, D.W. (1978). Automated Theorem Proving: A Logical Basis. North-Holland.

Markgraph, K. (1984). The Markgraph Karl Refutation Procedure. SEKI Memo MK-84-01, Universität Kaiserslautern.

Ohlbach, H.J. (1988). Using Automated Reasoning Techniques for Deductive Databases. SEKI-Report SR-88-06, Universität Kaiserslautern.

Overbeek, R. (1975). An Implementation of Hyper-Resolution. *Computational Mathematics with Applications* **1**, 201 - 214.

Shostak, R.E. (1976). Refutation Graphs. *Artificial Intelligence*. **7/1**, 51 - 64.

Shostak, R.E. (1979). A Graph-Theoretic View of Resolution Theorem-Proving. Report SRI International, Menlo Park.

Stickel, M. E. (1985). Automated Deduction by Theory Resolution. *Journal of Automated Reasoning*. **1/4**, 333 - 356.

Stickel, M. E. (1986). Schubert's steamroller problem: Formulations and Solutions. *Journal of Automated Reasoning*. **2/1**, 89 - 102.

Vieille, L. (1987). Recursive Query Processing: The Power of Logic. ECRC Munich, Technical Report TR-KB-17.

Wos, L. et.al. (1967). The Concept of Demodulation in Theorem Proving. *Journal of the ACM*, **14**, 698 - 709.

Wos, L. (1988). Automated Reasoning: 33 Basic Research Problems. Prentice Hall, Englewood Cliffs.