CS 2409S

Lecture 4

Notes by: Henry J. Pasko

January 17, 1973

Method I - proposed strategy for resolution

$\mathcal{S}$ = set of input clauses

n = number of atoms in $\mathcal{S}$

$\mathcal{T} \supseteq \mathcal{S}$,

$L_1,\ldots,L_m$ = stack

The Algorithm:

1) INITIALIZATION: $m \leftarrow 1$
$\mathcal{T} \leftarrow \mathcal{S}$

2) If $m > n$ then STOP (the stack describes a satisfying truth assignment for $\mathcal{S}$)

3) LITERAL SELECTION: Select a literal $L \in \mathcal{S}$ so that neither L nor $\bar{L}$ is on the stack $L_1,\ldots,L_{m-1}$. $L_m \leftarrow L$

4) FALSIFICATION CHECK: If $L_1,\ldots,L_m$ falsifies no member of $\mathcal{T}$, then $m \leftarrow m+1$. Go to 2.
Else let $C_1 \in \mathcal{T}$ be a clause falsified by $L_1,\ldots,L_m$ and set $L_m \leftarrow \bar{L}_m$. If the new $L_1,\ldots,L_m$ falsifies no member of $\mathcal{T}$, $m \leftarrow m+1$, go to 2. Else let $C_2 \in \mathcal{T}$ be a clause falsified by $L_1,\ldots,L_m$.

5) RESOLVE: $\mathcal{T} \leftarrow \mathcal{T} \cup \{R(C_1,C_2)\}$
If $R(C_1,C_2) = \square$ then STOP ($\mathcal{S}$ is inconsistent)

COMMENT: $R(C_1,C_2)$ is falsified by the stack, and therefore did not appear previously in $\mathcal{T}$. (Nor is it subsumed by anything in $\mathcal{T}$. For, suppose it was subsumed by $C \in \mathcal{T}$. But then C would also have been falsified by the stack.)

6) POP STACK: Find the largest $k < m$ so that $L_1,\ldots,L_k$ does not falsify $R(C_1,C_2)$.

$m \leftarrow k+1$ ($k=0 \equiv$ empty stack). go to 4.

This procedure always terminates either with a resolution proof of $\mathcal{S}$, or a satisfying assignment.

Example: $P \vee Q$, $P \vee \overline{Q}$, $\overline{P} \vee R$, $\overline{P} \vee \overline{R}$

For step 3 of the algorithm, pick the literal which occurs most often in $\mathcal{S}$. More will be said about this, later.

1) $m \leftarrow 1$
   $\mathcal{T} \leftarrow \mathcal{S}$

3) $L_1 \leftarrow P$

4) $m \leftarrow 2$

3) $L_2 \leftarrow Q$

4) $C_1 \leftarrow P \vee Q$

   $L_2 \leftarrow \overline{Q}$

   $C_2 \leftarrow P \vee \overline{Q}$

5) $\mathcal{T} \leftarrow \mathcal{T} \cup \{R(C_1,C_2)\} = \mathcal{T} \cup \{P\}$

6) $k=0$, $m \leftarrow 1$

4) $C_1 \leftarrow P$, $L_1 \leftarrow \overline{P}$, $m \leftarrow 2$

3) $L_2 \leftarrow R$

4) $C_1 \leftarrow \overline{P} \vee R$

   $L_2 \leftarrow \overline{R}$

   $C_2 \leftarrow \overline{P} \vee \overline{R}$

5) $\mathcal{T} \leftarrow \mathcal{T} \cup \{R(C_1,C_2)\} = \mathcal{T} \cup \{\overline{P}\}$

6) $k=0$, $m \leftarrow 1$

4) $C_1 \leftarrow P$

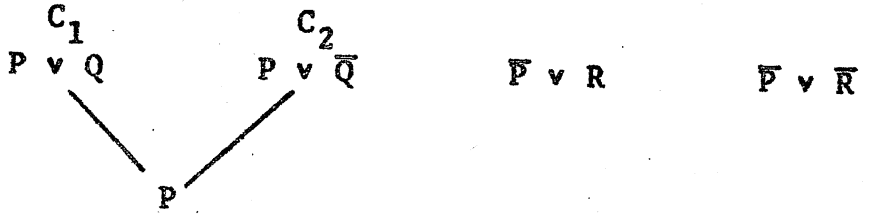
   $L_1 \leftarrow \overline{P}$

   $C_2 \leftarrow \overline{P}$

5) $\mathcal{T} \leftarrow \mathcal{T} \cup \{R(C_1, C_2)\} = \mathcal{T} \cup \{\square\}$

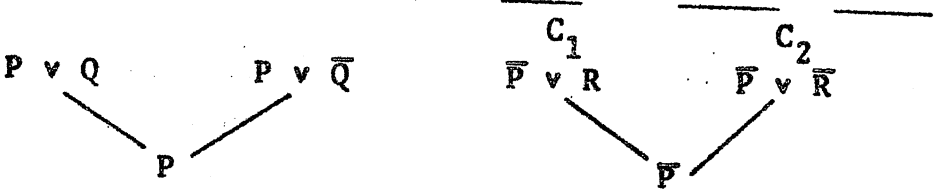
STOP - $\mathcal{S}$ is inconsistent.

Diagramatically, the algorithm looks like:



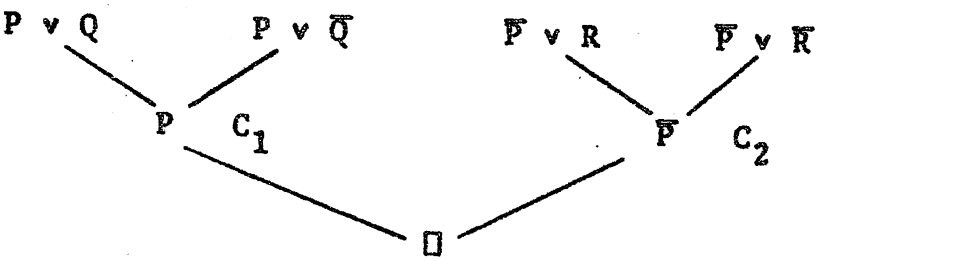
stack: P ~~P~~

Q̄



stack: P̄ ~~P̄~~

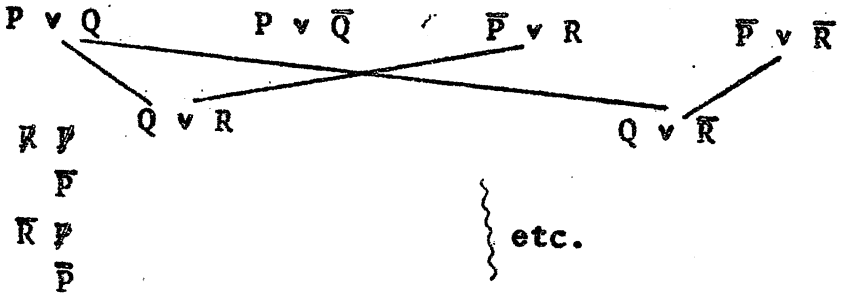
R̄



stack: ~~P~~

P̄

Now, let us try choosing the literals for step 3 of the algorithm differently:



We can immediately see that the method of selecting literals is indeed crucial. As shown above, it is possible to choose literals in such a manner that the clauses resulting from resolution are just as complex as those in the original set $\mathcal{S}$.

Definition: Literal Selection Strategy (LSS): A method for selecting L in step 3 of method I .

Exercise I: Show that if the D-P procedure without any subsumption is applied to $\mathcal{S}$, and atoms are eliminated in the order $P_1,\ldots,P_n$, then every clause generated by applying method I to $\mathcal{S}$ will be generated by D-P provided the following LSS is used:

Choose the highest index i so neither P_i nor $\bar{P}_i$ appears on the stack, and let $L = P_i$. (Thus, initially, the stack is loaded in the order $P_n,\ldots,P_1$) .

Conjecture (Research Problem)

For every inconsistent set $\mathcal{S}$ of clauses and every resolution proof P of $\mathcal{S}$, there is a way of selecting literals (step 3) such that method I yields a proof no longer than P .

Remark: If the subsumption rule is added to Method I, the resolution proof generated by Method I will be unaffected.

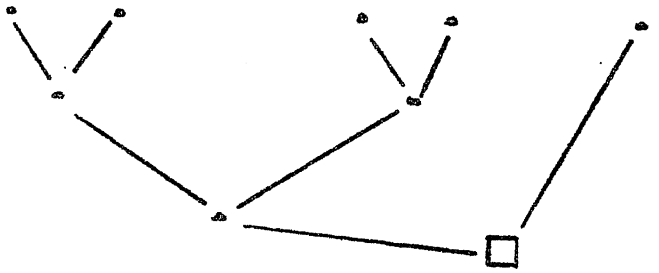
Method IA: This consists of Method I with the following LSS (Augment step 3).

Let $\mathcal{T}'$ consist of all clauses of $\mathcal{T}$ not verified by the stack, modified so that all literals falsified by the stack are deleted. Now, let $\mathcal{T}''$ consist of all the clauses of minimal length in $\mathcal{T}'$. For each atom P , count the total number of occurences of P or $\bar{P}$. Let Q be the atom with maximal count. Let $L=Q$ or $\bar{Q}$, which ever has more occurences.

Exercise II: a) Apply Method IA to the set $\{P \vee Q, R \vee S, \overline{P} \vee \overline{R}, \overline{P} \vee \overline{S}, \overline{Q} \vee \overline{R}, \overline{Q} \vee \overline{S}\}$

b) Prove there is no shorter resolution proof than the one obtained in (a). Hint: (Use the following lemma, and show that it is essential to use one of the original clauses twice.

Lemma: If $\mathcal{A}$ is a minimally inconsistent set of m clauses, then every resolution proof of $\mathcal{A}$ has length at least $m-1$. It will have length $m-1$ in the case the proof is a tree (no clause is used more than once in a resolvent).



Proof: This follows from the fact that a binary rooted tree with m leaves has $m-1$ internal nodes.

OPEN QUESTION: Is there a sound proof system (such as resolution, analytic tableaux, etc.), such that for some polynomial $P(n)$, every inconsistent formula of length n, has a proof in the system of length $\leq P(n)$.

POTENTIAL PROJECT: Pick a particular proof system in some text book (e.g., Mendelson, Shoenfield, Smullyan, etc.) and prove there is no polynomial bound on the length of minimal proofs in the system.

CS 2409S

Lecture 5

Notes by John Corbett

Jan. 22, 1973

Note: In Exercise 1 (lecture 4, page 4) assume that all variables are eliminated by the D-P procedure before the empty clause is generated.

Experimental Results Using Method IA

Observation: It has been observed that in most examples Method IA uses all resolvents it generates. That is, every resolvent produced is used later as a parent clause for a resolution. Examples can be found where this is not true, but the following is an open question:

> "Must this be the case if the original set of clauses is minimally inconsistent?"

Examples for Method IA

(1) Method IA generates the minimal proof for the sets of S_n of lecture 3 (page 3). (It produced a "tree" proof. Thus the length is $|S_n|-1$.)

(2) When tried on the symmetric case

$$S = \{P_{i_1} \vee P_{i_2} \vee P_{i_3}\ ,\ \overline{P}_{i_1} \vee \overline{P}_{i_2} \vee \overline{P}_{i_3} \quad 1 \le i_1 < i_2 < i_3 \le 5\}$$

(i.e. triples using $P_1, P_2, \ldots, P_5$ and their negations.) Method IA produced a minimal proof in 19 resolutions. (There are $\binom{5}{3} \cdot 2 = 20$ clauses.)

(3) (Most interesting) Graph Colouring Problems: Assert that an n-node complete graph (a graph in which each node is connected to every other node) can be coloured with n-1 colours.

Notation: P_{ij} - node i gets colour j for $1 \le i \le n$, $1 \le j \le n-1$.

Clauses in G_n:

for $1 \le i < j \le n, 1 \le k \le n-1$ $\overline{P}_{ik} \vee \overline{P}_{jk}$ (not allowed to have nodes i & j both coloured k at the same time)

for $1 \le i \le n$ $P_{i1} \vee P_{i2} \vee \ldots \vee P_{in-1}$

G_n contains $\binom{n}{2}(n-1) + n = \frac{n(n-1)^2}{2} + n$

clauses. A resolution proof for this problem (the shortest known) has been devised with length (*) $(n-1)(n+2)2^{n-3}$. (This is interesting since it is exponential in n.) Observe the results of Method IA in the following table for various values of n:

# of nodes	$\lvert G_n \rvert$	Length of Proof (Method IA)	Shortest known Proof (calculated from *)
3	9	10	10
4	22	42	36
5	45	124	112
6	81	342	320
7	133	896	864

This example is a good one to use if you are trying to get a high lower bound on minimal resolution proof lengths. (Karp first observed that this example seems to require long resolution proofs.)

An Extension of Resolution Calculus - generalize clauses into "phrases".

A phrase is a disjunction $(A_1 \vee \ldots \vee A_n)$ where each A_i is an &-clause. For example:

$$(P\&Q \vee R\&S \vee \bar{P}\&Q \vee P)$$

is a phrase. (We omit parentheses around the &-clauses.)

Extended Resolution:

$$ER(H_1, H_2) = (H_1 - S_1) \cup (H_2 - S_2) \quad \text{where}$$

S_1 = {set of all &-clauses of H_1 containing L (the literal resolved upon)}

S_2 = {set of all &-clauses of H_2 containing $\bar{L}$}.

H_1, H_2 - are phrases as described above.

Example:

$$ER[(P\&Q \vee P\&R \vee \bar{P}\&S \vee T), (\bar{P}\&S \vee \bar{P}\&U \vee S\&T \vee U)]$$

$$= (\bar{P}\&S \vee T \vee S\&T \vee U)$$

where P is the literal resolved upon and

$S_1 = \{P\&Q, P\&R\}$, $S_2 = \{\bar{P}\&S, \bar{P}\&U\}$

It should be noted that the above "extended resolution" rule is valid and complete. The rule can be used along with the subsumption rule without any loss. Although it is possible to operate ordinary resolution proofs on clauses such as was used by the example, simply by expanding and "multiplying"

through variables then carrying on an "ordinary" resolution procedure, it has been found that the above extended resolution procedure will never be any longer (and in fact is often shorter) than the resolution proof for an equivalent set of clauses.

POTENTIAL PROJECT: Prove the above assertions, give some examples, and extend the theory to predicate calculus.

Predicate Calculus

Notation:

Following is a list of basic symbols from which formulas in predicate calculus are built:

(a) An infinite list for:

(1) individual variables - use x,y,z to stand for these

(2) n-place function symbols for each n-use f,g,h for these

(3) n-place predicate symbols for each n-use P,Q,R for these

(b) Logical connectives: $\&$, $\vee$, $\neg$

(c) quantifiers: $\forall$, $\exists$

(d) parentheses and comma

Definition: *term*: (1) A variable or a constant is a *term*.

(2) If f is an n-place function symbol and $t_1,\ldots,t_n$ are terms, then $f(t_1,\ldots,t_n)$ is a *term*.

Definition: Atomic Formula: An expression of the form $P(t_1,\ldots,t_n)$ for P, an n-place predicate symbol and where $t_1,\ldots,t_n$ are terms is an atomic formula.

Definition: Formula: (1) Atomic formulas are formulas.

(2) If A,B are formulas and x is a variable, then $(A\&B)$, $(A\vee B)$, $\neg A$, $\exists xA$, $\forall x A$ are formulas.

Definition: Free & Bound Variables: An occurrence of the variable x in a formula A is bound iff the occurrence is in some "sub formula" of the form:

$$\exists x B \quad \text{or} \quad \forall x B$$

If the occurence is not bound, it is free.

e.g. $P(x_1) \,\&\, \exists x_2 P(x_2)$ $\quad x_1$ is free, x_2 is bound

Definition: Sentence: A is a sentence (or closed formula) if A has no free variables.

Definition: Interpretation: An interpretation, I, of a sentence A consists of the following:

(1) A domain (non-empty set) D,

(2) $I(f)$: $D^n \to D$ for each n-place function symbol, f, in A,

(3) $I(P)$: An n-place relation on D, for each n-place predicate symbol, P, in A.

$I(A)$ is the "truth value" (t or f) which I gives A.

Definition:

(1) A is valid iff $I(A) = t$ for all I .

(2) A is satisfiable iff $I(A) = t$ for some I .

(3) A is unsatisfiable iff $I(A) = f$ for all I .

(4) A is falsifiable iff $I(A) = f$ for some I .

Definition: Consistent: If S is a set of sentences, S is consistent iff there exists an I such that $I(A)$ is true for all $A \in S$.

Definition: Prenex Form: A is in prenex form iff A has the form $Q_1x_1 \ldots Q_nx_n\ B$ where B is quantifier free and each Q_i is a quantifier (i.e. $\forall$ or $\exists$).

Lemma: Every formula can be put in prenex form.

Functional Form (Skolem)

Given a formula (e.g. $\forall x\, \exists y\, B(x,y)$); if it is true under some interpretation I , then there is a function (known as a Skolem function) defined on the domain of I such that y (of the example above) can be represented as f(x) . The existential quantifier can subsequently be deleted to form the following:

$\forall x\, B(x,f(x))$.