AUTOMATED REASONING

SLIDES 5:

COMPLETENESS of RESOLUTION
Basic idea of Completeness proof
Semantic Trees
Lifting a ground resolution refutation
Resolvents and a Semantic Tree
Refutations from a Semantic Tree

KB - AR - 09

Completeness of Resolution

We will show by construction:
If clauses S have no models then there is a resolution proof of [] from S.

As shown in Slides 4 the construction has two parts:

(i) find a ground resolution refutation for some finite subset of the ground instances of the given clauses, and then

(ii) transform this ground refutation to a general refutation (called *Lifting*).

Assume S has no models; then

(a) *find the appropriate ground instances*: construct a finite closed semantic tree for ground instances G of clauses in S;

(b) *find a ground refutation*: construct a ground resolution refutation from the closed semantic tree for G;

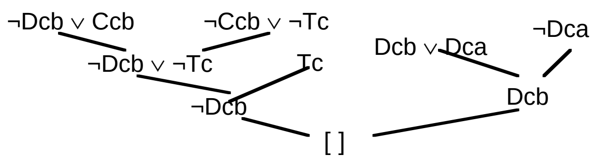
(c) *find a general refutation*: construct a resolution refutation for S from the ground refutation.

This works because of the relation between ground and general refutations:

Example of the relationship between a refutation of ground instances of clauses S and a resolution refutation of S (used for Step (c))

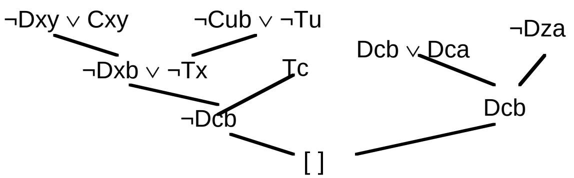
1. Dca ∨ Dcb 2. ¬Dxy ∨ Cxy 3. ¬Tu ∨ ¬Cub 4. Tc 5. ¬Dcz

Ground instances: {u == c, x == c, y == b, z == a}
Dca ∨ Dcb ¬Dcb ∨ Ccb ¬Tc ∨ ¬Ccb Tc ¬Dca



Ground proof

Resolution proof

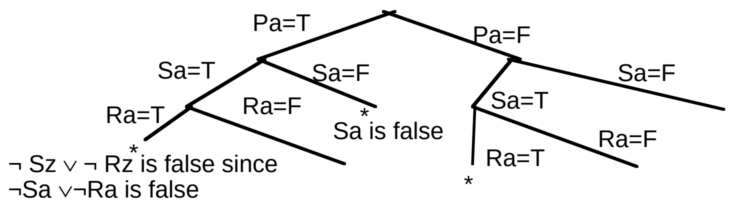


Each clause in the ground proof is an instance of a corrsponding clause in the resolution proof.

Semantic Tree

A *Semantic Tree* for S is an enumeration of all HIs over Sig(L), where S uses language L. Each branch represents a H-interpretation (HI) over Sig(L).

Example: Let Sig(L) = < {P,Q, R, S}, {f}, {a, b} > and
Given Px ∨ Ry ∨ ¬Qxy , ¬Sz ∨ ¬Rz, Pu ∨ Qf(v)v, Sa, Sb , ¬Pf(a) ∨ ¬Pf(b)



Each finite portion of a branch of a semantic tree gives a partial Herbrand Interpretation of S. A branch is terminated (marked *) if it cannot be a model for S.

eg the leftmost branch falsifies ¬Sa ∨ ¬Ra, instance of ¬Sz ∨ ¬Rz.
Which other branches falsify a given clause?
Any interpretation that uses the assignments in a terminated branch is impossible as a model of S.

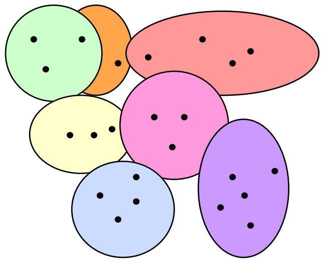
If every branch in a semantic tree for clauses in S is closed then S is unsatisfiable.

The "Umbrella" Property

5bii

- If S has no Hmodels then each H-interpretation must falsify a clause in S;
- To make a clause C false it is sufficient to make 1 ground instance of C false.
- Since clauses in S are finite, the falsifying part of the interpretation is found after consideration of a finite number of atoms.

BUT: Can we be sure there are a finite number of ground instances of S sufficient to be falsified by all the H-interpretations over the signature of S?



The "Umbrella" Property says we can!
Each dot is an H-interpretation
Each circle is a ground instance of S (one of the finite number) falsified by a number of H-interpretations

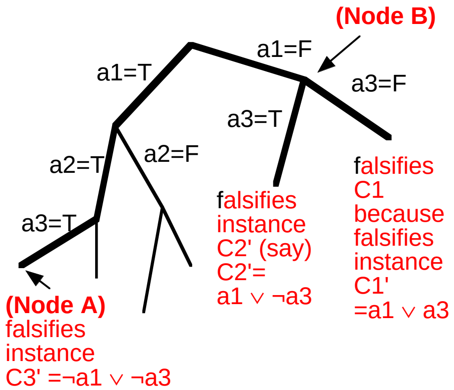
We can show: If S is unsatisfiable then there is a finite closed semantic tree for S (Called *compactness*.)

Assume S has no models:

- If the Semantic tree for S were infinite, then there would be an infinite branch. (Konig's Lemma)
- We claim such an infinite branch would yield a model because:
  - Assume for contradiction the branch did not give a model.
  - Then the branch could have been finite (by above observations)

Illustrating Compactness in a Semantic Tree

5biii



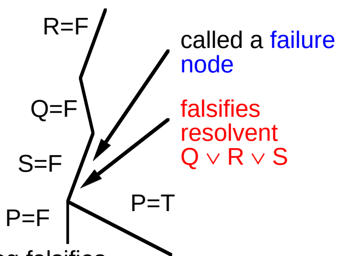
Any interpretation that assigns T to a1 and a3 will falsify instance C3'. Thus all branches of the tree through node A (called a closure node) can be terminated at A.

Node B is called a Failure Node. Notice that the two children are both closure nodes and that the ground instances C2' and C1' resolve with each other.

Let the *Herbrand Base* be {a1,a2,a3,a4,a5,...}
Let Givens include clauses C1, C2, C3, ...which have as instances
C3'= ¬a1 ∨ ¬a3, C2'=a1 ∨ ¬a3, C1' = a1 ∨ a3, ...
The semantic tree yields an unsatisfiable finite set of ground instances of S.

How to find a resolvent from a semantic tree.

5biv



eg falsifies
P ∨ Q ∨ R

Note1:
the false clause must include P

eg falsifies ¬P ∨ S makes P true and S false

Note2:
the false clause must include ¬P

The two children of a failure node will resolve by Nodes 1 and 2.

If atom A is tested, then the clause falsified by the A=F branch will contain A and the clause falsified by the A=T branch will contain ¬A.

The resolvent willl be false at the failure node. **WHY?**

Therefore, the resolvent cannot be a tautology. **Why?**

The resolvent can be added to the falsifying ground instances and will allow a smaller tree to be obtained, since the failure node will now become a closure node.

<u>How to obtain a ground refutation from a completed semantic tree.</u>

5bv

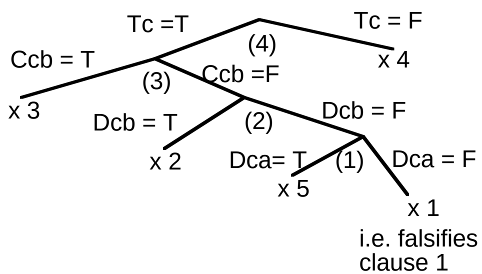
1. Dca ∨ Dcb   2. ¬ Dxy ∨ Cxy   3. ¬Tx ∨ ¬ Cxb   4. Tc   5. ¬ Dcz

Ground instances found from tree:
1g. Dca ∨ Dcb   2g. ¬ Dcb ∨ Ccb   3g. ¬Tc ∨ ¬ Ccb   4g. Tc   5g. ¬ Dca

<u>A semantic tree:</u>

Resolution steps - process nodes in order.

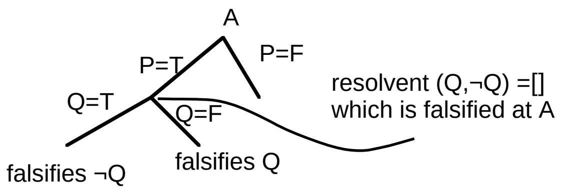
(1): (left) 5g; (right) 1g;
=> Dcb (6g and is false)
(2): (left) 2g; (right) 6g
=> Ccb (7g and is false)
(3): (left) 3g; (right) 7g
=> ¬ Tc (8g and is false)
(4): (left) 8g; (right) 4g
=> []



Check that at each failure node to be processed false clauses can be resolved and that if atom A is tested, then left clause contains ¬A and right clause contains A. (Assumes left branch of test makes A=T, right branch makes A = F.) Also check that the resolvent is false at the processed node.

Properties of the Semantic Tree method (1):

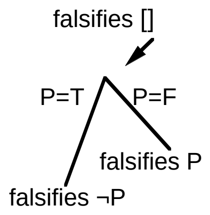
5ci



Sometimes a resolvent falsifies a node higher than the failure node at which it was formed - enables the tree to contract more quickly.

Suppose a fully closed semantic tree has been generated using clauses in T. Let T' be the used ground instances (i.e. the ground instances falsified at the leaves).

If the resolvent, eg QRS (ie Q∨R∨S) of Slide 5biv, is added by resolution to T' then T can be contracted since QRS is falsified at the failure node, which is higher than the two parent clauses. This gives rise to a new closure node which can be used to derive a resolvent and hence a smaller semantic tree. Eventually , since there were only a finite number of closure nodes at the start and each step removes at least one, a tree of the form on the right is derived, from which [] is deduced.



Properties of the Semantic Tree method (2):

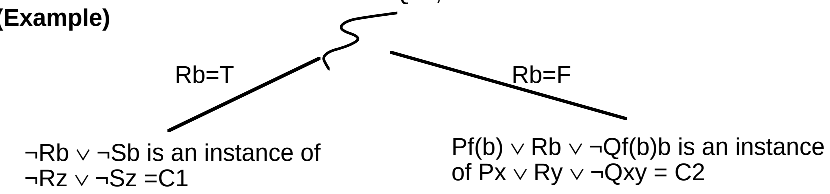
- Each tree gives rise to a ground resolution proof (refutation) using instances of the given clauses.
- Can derive a full resolution refutation from a semantic tree proof by LIFTING (see below and slide 5cv) because of the following invariant property:

• Each failed clause instance is either an instance of a given clause, or an instance of the resolvent of the involved clauses at the leaves (ie the closure nodes).

LIFTING LEMMA

(Example)

Resolve instances of C1 and C2 to give ¬Sb ∨ Pf(b) ∨ ¬Qf(b)b which is an instance of ¬Sz ∨ Px ∨ ¬Qxz, the resolvent of C1 and C2



5cii

Properties of the Semantic Tree method (3):

5ciii

Sometimes a factoring step is required:
It's indicated if the ground instance has fewer literals than the general clause.

Recall that at ground level, factoring is just merging of identical literals, whereas in general it requires a substitution to make 2 or more literals identical.

- In slide 5civ at (Z) the new ground clause {¬ Sb, Pf(b)} is obtained:
- by resolution of ¬Sb ∨ Pf(b) ∨ ¬Qf(b)b and Pf(b) ∨ Qf(b)b to give ¬Sb ∨ Pf(b) ∨ Pf(b)
- then by ***merging*** to give ¬Sb ∨ Pf(b)

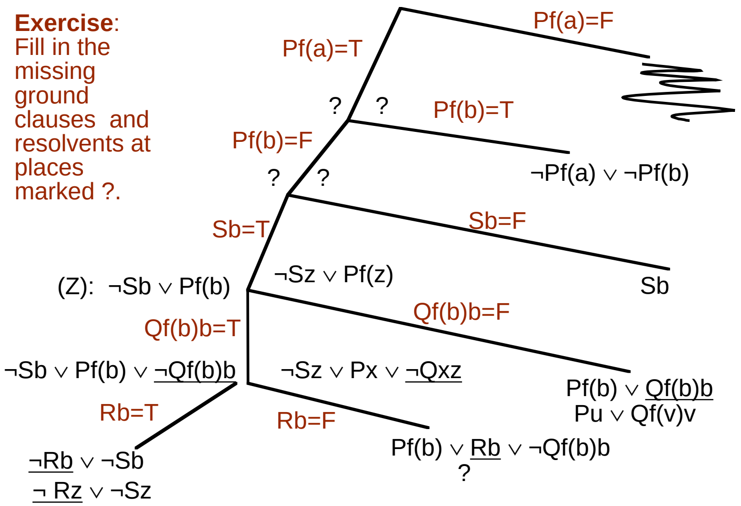
- The general resolvent clause is formed from ¬Sz ∨ Px ∨ ¬Qxz and Pu ∨ Qf(v)v, giving ¬Sz ∨ Pf(z) ∨ Pu which ***factors*** to ¬Sz ∨ Pf(z) with binding {u==f(z)}.

(The proof of the Lifting Lemma outlined on Slide 5cv also covers the case when the ground resolvent has a merge applied and the general resolvent factors. As an exercise you might like to see why this is so.)

Q. What is the problem with using the semantic tree method for showing unsatisfiability? What feature of resolution makes resolution better?

S = Px ∨ Ry ∨ ¬Qxy, ¬Sz ∨ ¬Rz, Pu ∨ Qf(v)v, Sa, Sb , ¬Pf(a) ∨ ¬Pf(b)

Exercise: Fill in the missing ground clauses and resolvents at places marked ?.



5civ

The Lifting Lemma (General Case) 5cv

Given C1 and C2 that resolve to give R and C1', C2', instances of C1 and C2, that resolve to give R', then R' is an instance of R (or of a factor of R).

$$\begin{tikzcd} C1,C2 \arrow[r, "\text{resolve and maybe factor}"] \arrow[d, "\text{take ground instances}"'] & R=(D\vee G)\theta \arrow[d, "\text{take ground instance}"] \\ C1'=C1\sigma=(E\vee D)\sigma,\ C2'=C2\lambda=(\neg F\vee G)\lambda,\ E\sigma=F\lambda \arrow[r, "\text{resolve}"] & R'=D\sigma\vee G\lambda=(D\vee G)\sigma\lambda \end{tikzcd}$$

$C1 = E\vee D$,
$C2 = \neg F\vee G$
D and G clauses
E, F unify,
θ=mgu(E,F)

σ and λ apply to different variables so are unaffected when combined (or composed) in $\sigma\lambda$. $E\sigma\lambda= F\sigma\lambda = E\sigma = F\lambda$ so $\sigma\lambda$ unifies E and F (so θ exists).
So $(D\vee G)\theta\rho =(D\vee G)\sigma\lambda$ (defn of mgu) for some substitution ρ
and R' is a ground instance of $(D\vee G)\theta$.

You can follow the diagram in two different ways:
C1,C2 have ground instances C1', C2' that resolve to R' (used by semantic tree)
C1, C2 resolve to R that has ground instance R' (used by resolution)

<u>The Lifting Lemma:</u> 5cvi

The *lifting lemma*, illustrated on Slide 5civ and proved in outline on 5cv, shows how the ground resolution obtained by the semantic tree method can be transformed into a full resolution proof. Each ground resolution step, working up the tree, yields a general resolution step. An illustration of such a step is shown on 5cii. The property, also illustrated in general, is that:

if C1 and C2 resolve (possibly after factoring) to give resolvent R, and ground instances C1' and C2', respectively, of C1 and C2, resolve to give resolvent R', then R' is a ground instance of R (or of a factor of R).

This property is then used to guarantee the fact, given on 5cii, that each clause labelling a failure node is an instance of a given clause or of a resolvent derived from the given clauses. Each step of the ground resolution proof (deriving R' from C1' and C2' by resolution and/or factoring) gives rise to a step between C1 and C2 deriving R and such that R' is a ground instance of R (or of a factor of R).

When carrying out the procedure by hand you can either: find the ground refutation and then obtain the general one by lifting, or add to the semantic tree the general resolvents and derive the refutation by resolving the clauses at each pair of leaf nodes.

A different ***completeness proof for ground resolution*** (ie not using semantic trees) uses induction on the number of different *atoms* occurring in the given set of clauses. (As identical literals in a clause are merged, each literal in a clause occurs only once.) You can assume also that there are no tautologies as such clauses can be removed without affecting satisfiability (or, as noted on Slide 5biv, no tautologies are needed), so all literals in a clause involve different atoms. The base case (for one atom A) is easy - all clauses must be unit clauses of the form A or ¬A. If S is unsatisfiable it must contain both kinds and a refutation can easily be found by resolving clauses such as A and ¬A.

The induction step (for k>0 atoms) assumes as Ind. Hyp. that a refutation can be obtained for an unsatisfiable set of ground clauses S with <k atoms. Now, there must be at least 1 atom (say B) that occurs both positively and negatively in different clauses (if not S can't be unsatisfiable - why?). Form two sets of unsatisfiable clauses, S' and S", as follows: First construct S1/S2 = {C | C in S and C does not contain B/¬B} and then delete any occurrences of ¬B/B from clauses in S1/S2 to give S'/S". (**Exercise**: show S' and S" must be unsatisfiable - this is the crucial step.) Hence by the Ind. Hyp. there is a resolution refutation of S' and of S", as all occurrences of B have been removed from S' and S", so they have <k atoms occurring. Now replace the removed literals ¬B/B into the refutations. That of S' will derive ¬B (or still be a refutation) and that of S" will derive B or still be a refutation. In case both ¬B and B are derived a refutation can be found by resolving them. In all cases a refutation is found for S.

e.g. for the ground clauses on slide 5bv, choose Dcb as B and form S1' =
{Ccb, ¬Ccb∨ ¬Tc, Tc, ¬Dca} and S1"= {Dca, ¬Ccb∨ ¬Tc, Tc, ¬Dca}.
For S1', repeat the step (say choosing Ccb) and eventually you will obtain the refutation
¬Ccb∨ ¬Tc + Tc ==> ¬Ccb, Ccb + ¬Ccb ==>[] .
S1" is obviously unsatisfiable, Dca+¬Dca ==>[] .
Putting back ¬Dcb into the first refutation will now yield ¬Dcb, and putting back Dcb into the second refutation gives Dcb. Then resolve these to give [] .
(e.g. Putting back Dcb into the second refutation gives Dca∨Dcb + ¬Dca ==> Dcb.)

Induction proofs are often used when resolution is restricted in some way - we will see some more examples later.
5cvii

Summary of Slides 5

5di

1. Completeness of resolution can be shown in several ways. Most proofs demonstrate completeness using two steps. First a refutation is found using ground instances of the given clauses. This ground refuation is then *lifted*, to use the original clauses.

2. There is a close relationship between the ground refutation and the lifted refutation.

3. A *Semantic Tree* formed from a set of unsatisfiable clauses will be finite. Each branch in the tree will falsify some ground instance of one of the given clauses.

4. A *failure node* A in a semantic tree is a node such that both descendants of A, formed by considering some atom D=T or D=F, are leaves and the falsifying ground instance of the leaf which considered D=T contains the literal ¬D and the falsifying instance for the other leaf contains the atom D. The resolvent of the two falsifying instances falsifies the branch ending at A.

5dii

5. A semantic tree can be used to obtain a ground refutation of ground instances of given clauses and also to find the corresponding refutation.

6. The refutation obtained from a semantic tree indicates where factoring is needed. The refutation never derives a tautology.

7. Semantic Trees could be used to show unsatisfiability of a set of clauses S. But it is not a very practical method in general, since if a "bad" order of atoms is selected the tree could be very large. For this purpose, there is no need to form resolvents, of course, it is enough to know that every branch in the tree falsifies some ground instance of S.

8. Inductive proofs can also be used to show the completeness of ground resolution.

Question for next week:

Suppose resolvents are restricted, in that only certain literals in a clause can be resolved upon.

Consider the restriction that forces literals in a clause to be selected in alphabetical order. (e.g. R(...) would be resolved before S(...).)

How can the Semantic Tree proof be modified to show completeness for this case?

5diii