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

Completeness of Resolution

Theorem (Completeness) If a set of clauses S has no models then S ⇒ *[]
(i.e. there is a resolution refutation of [] from S.)

Proof Structure for Resolution Completeness

Assume
S $|=$ ⊥

(a): consider H-interpretations
Useful Theorem (*)

S $|=_H$ ⊥

(b): find unsatisfiable finite set of ground instances of S i.e. S_{FG}; existence is guaranteed by *compactness*

S_{FG} $|=_H$ ⊥

(Look for a ...)

Ground Refutation by resolution and factoring using S_{FG}

(c): "Lift" ground derivation to a first order derivation

Hence
S =>* [ ]

Assume that S $|=$ ⊥ and follow the arrows to show that S **=>*** [ ]

Completeness of Resolution

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

Most methods to show completeness rely on some very useful properties:

(a) A set of clauses S has no models iff S has no Hmodels
(Useful Theorem (*))
so it is sufficient to look at Herbrand Interpretations

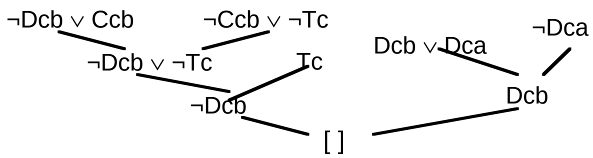
(b) If a set of clauses S is H-unsatisfiable (has no H-models) then there is a finite subset of ground instances of S also H-unsatisfiable
(*compactness*).
find the appropriate ground instances:
construct a finite closed semantic tree G for ground instances of S

(c) A resolution refutation for a set of clauses S has a similar structure to a ground resolution refutation using ground instances of S
(see slide 5aiii for an example).
find a ground refutation:
construct a ground resolution refutation from G
and lift it to give a resolution refutation from S

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

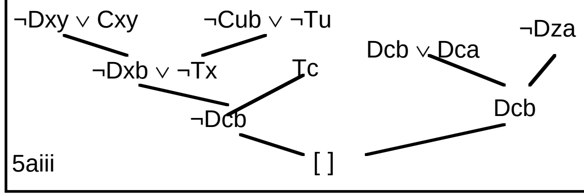
1. Dca ∨ Dcb 2. ¬Dxy ∨ Cxy 3. ¬Tu ∨ ¬Cub 4. Tc 5. ¬Dcz

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



Ground proof

Resolution proof



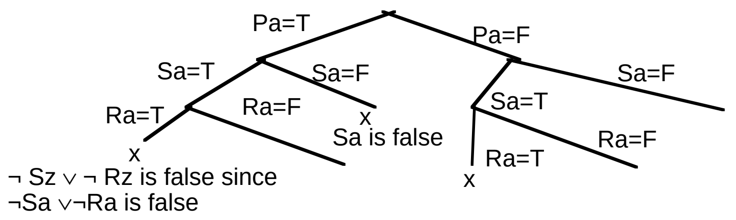
Each clause in the ground proof is an instance of a corresponding clause in the resolution proof.

Semantic Tree (used in steps b and c of completeness proof)

5bi

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



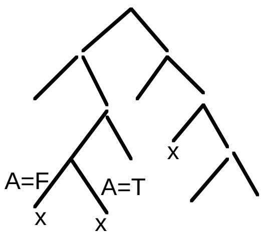
Each finite portion of a branch of a semantic tree gives a partial Herbrand Interpretation of S. A branch is terminated (marked x) if it cannot be a model for S.

eg the leftmost branch falsifies ¬Sa ∨ ¬Ra, instance of ¬Sz ∨ ¬Rz.
Which other branches falsify a given clause?

Observations about a Semantic Tree

5bii

General skeleton
of a semantic tree



1. 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. (Why?)

2. Any interpretation that uses the assignments in a terminated branch is impossible as a model of S.

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

Why?

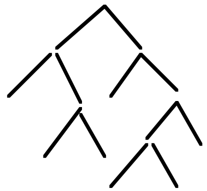
Every HI will be an extension of some branch of the tree and hence makes some clause in S false

4. We would like to know that ***if*** S is unsatisfiable we ***can find*** a finite closed tree

Could the tree have an infinte set of branches when S is unsatisfiable?
Let's see ....

Is a Semantic tree (for unsatisfiable S) always finite?

General skeleton
of a semantic tree



If S has no Hmodels then each H-interpretation must falsify a clause in S

For Step b of the completeness proof want to collect from a completed tree the set of ground clauses made false by the given general clauses

How do we know this set is finite?
Could the tree have an infinte set of branches?
Let's see what we know:

- 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?

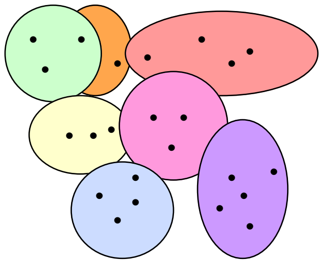
The "Umbrella" Property says we can!

In other words, the closed tree is finite

(We prove this next relying on König's Lemma)

5biii

Compactness for Clauses ("Umbrella" Property) 5biv



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*.)

We'll use **König's Lemma**:

A finitely branching tree can have an infinite number of nodes only if some branch is infinite (has an infinite length)

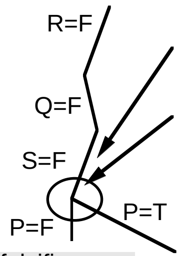
Assume S has no models:

- If the Semantic tree for S were infinite, then there would be an infinite branch. (Directly from König'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 earlier observations on ***5biii***)

How to find a resolvent from a semantic tree 5bv



called a failure node

falsifies resolvent Q ∨ R ∨ S

eg falsifies P ∨ Q ∨ R makes P, Q and R false

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

Note1: the false clause must include P

Note2: the false clause must include ¬P

The two children of a failure node will resolve (see ***Notes 1/2*** below the tree)

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.

How to obtain a ground refutation from a complete semantic tree (see ppt)

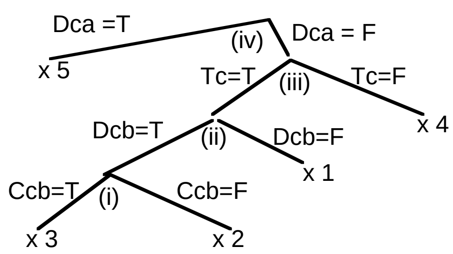
1. Dca ∨ Dcb 2. ¬ Dxy ∨ Cxy 3. ¬Tx ∨ ¬ Cxb 4. Tc 5. ¬ Dcz

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

Ground resolution steps - process nodes in order.

(i): (left) 3g; (right) 2g;
=> ¬Dcb ∨ ¬Tc (6g and is false)
(ii): (left) 6g; (right) 1g
=> Dca ∨ ¬Tc (7g and is false)
(iii): (left) 7g; (right) 4g
=> Dca (8g and is false)
(iv): (left) 5g; (right) 8g
=> []

A semantic tree:



Check that at each failure node the two false clauses below it 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.

5bvi

Properties of the Semantic Tree method (1)

- 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 because of the following invariant property:

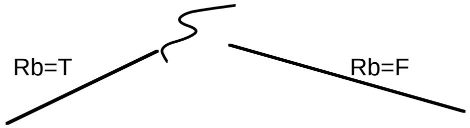
• Each failed clause instance is either an instance of a given clause, or an instance of the resolvent of the involved clauses at the leaves (ie the closure nodes).

See example below (and optional slides 5e for the general case)

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



¬Rb ∨ ¬Sb is an instance of ¬Rz ∨ ¬Sz =C1

Pf(b) ∨ Rb ∨ ¬Qf(b)b is an instance of Px ∨ Ry ∨ ¬Qxy = C2

5ci

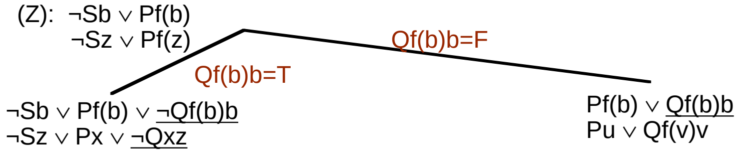
Properties of the Semantic Tree method (2)

5cii

Sometimes a factoring step is required to complete a refutation:
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.

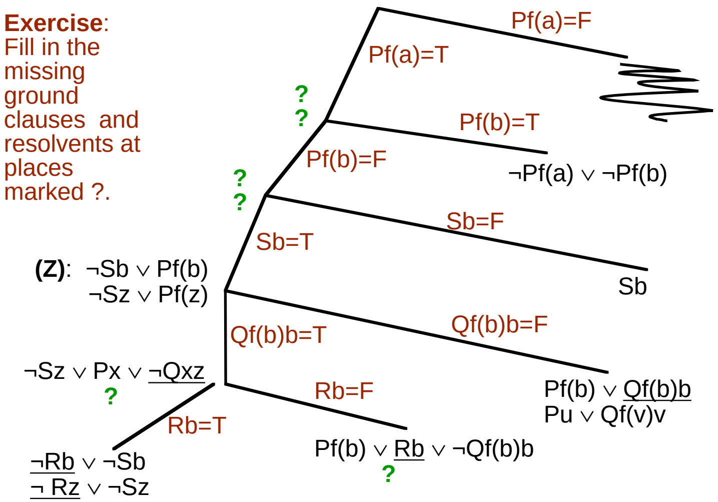
Example:
- In slide 5ciii at (Z) the new ground clause ¬ Sb ∨ Pf(b) is obtained:
- by resolution of ¬Sb ∨ Pf(b) ∨ ¬Qf(b)b and Pf(b) ∨ Qf(b)b to give ¬Sb ∨ Pf(b) ∨ Pf(b)
- then by ***merging*** to give ¬Sb ∨ Pf(b)
- The general resolvent clause is formed from ¬Sz ∨ Px ∨ ¬Qxz and Pu ∨ Qf(v)v, giving ¬Sz ∨ Pf(z) ∨ Pu which ***factors*** to ¬Sz ∨ Pf(z) with binding {u==f(z)}.



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

5ciii

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



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

The Lifting Lemma:

5civ

The *lifting lemma* is shown in action on Slides 5ci/5cii. (The general case and outline proof are part of the optional slides 5.) The lemma shows why 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. The property stated by the lemma is:

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

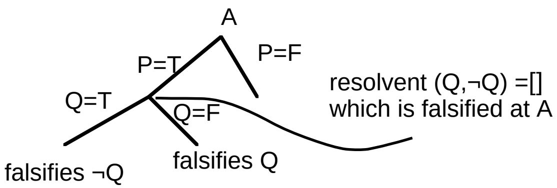
This property is then used to guarantee the fact, given on 5ci, that each clause labelling a failure node is an instance of a given clause or of a resolvent derived from the given clauses. Each step of the ground resolution proof (deriving R' from C1' and C2' by resolution and/or factoring) gives rise 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 leaves of the semantic tree the general resolvents and derive the refutation by resolving the clauses at each pair of leaf nodes and contracting the tree.

Properties of the Semantic Tree method (3)

5cv

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



When/how can this happen?

- Since the order of atoms chosen is arbitrary, any particular choice may not give the shortest or best tree. Sometimes (as above) this is detected.
- (Because there are usually several different refutations, different orders of atoms can give completely different trees)

Completing the Semantic Tree method

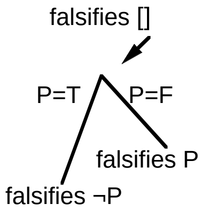
5cvi

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 at a failure node,
eg Q∨R∨S of Slide 5bv,
is added by resolution to T' then T can be contracted since the resolvent is falsified at the failure node.

The failure node is higher than the two parent clauses and gives rise to a new closure node.
This in turn 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 will be derived, from which [] is deduced.



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. (See optional slides!)

START of OPTIONAL MATERIAL (SLIDES 5)

Outline Proof of the Lifting Lemma
Alternative (inductive) completeness proof for ground resolution

---

The Lifting Lemma (General Case)

5ei

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).

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

C1 = E∨D,
C2 = ¬ F∨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 σλ. Eσλ= Fσλ = Eσ = Fλ so σλ unifies E and F (ie θ exists).
So (D∨G)θρ =(D∨G)σλ (defn of mgu) for some substitution ρ
and R' is a ground instance of (D ∨G)θ.

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)

This proof also covers the case when the ground resolvent has a merge applied and the general resolvent factors. **Exercise**: explain why this is so.

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 5bv, 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>1 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 one 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:

For S': First construct S1 = {C | C in S and C does not contain B} and then delete any occurrences of ¬B from clauses in S1 to give S'.

For S": First construct S2 = {C | C in S and C does not contain ¬B} and then delete any occurrences of B from clauses in S2 to give S".

(**Exercise**: **show S' and S'' must be unsatisfiable - actually, this is the crucial step.**
Hint: e.g. for S' assume S' is satisfiable and hence has a model, and then derive a contradiction.)

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.

5eii

5eiii

e.g. for the ground clauses on slide 5bvi, 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.