Proof Methods in FOL

Major Families:

- GMP
- Reduction
- Resolution
- Forward chaining
- Backward chaining

Some Other inference tools:

Entailment/ Unification/

Existential instantiation (EI)

- For any sentence α, variable ***v***, and constant symbol ***k*** that does not appear elsewhere in the knowledge base:

$$\frac{\exists v\, \alpha}{\text{Subst}(\{v/k\}, \alpha)}$$

- E.g., $\exists x\, Crown(x) \wedge OnHead(x, John)$ yields:

$$Crown(C_1) \wedge OnHead(C_1, John)$$

provided C_1 is a new constant symbol, called a Skolem constant

Universal instantiation (UI)

- Every instantiation of a universally quantified sentence is entailed by it:

$$\frac{\forall \, v \, \alpha}{\text{Subst}(\{v/g\}, \alpha)}$$

for any variable *v* and ground term *g*

- E.g., $\forall x \; King(x) \wedge Greedy(x) \Rightarrow Evil(x)$ yields:

$King(John) \wedge Greedy(John) \Rightarrow Evil(John)$
$King(Richard) \wedge Greedy(Richard) \Rightarrow Evil(Richard)$
$King(Father(John)) \wedge Greedy(Father(John)) \Rightarrow Evil(Father(John))$

Resolution in PC

- **Propositional version.**

$\{a \vee b, \neg b \vee g\} \mid\!- \; a \vee g$ **OR** $\{\neg a \Rightarrow b, b \Rightarrow g\} \mid\!- \; \neg a \Rightarrow g$

- Reasoning by cases **OR** transitivity of implication
- **First-order form**
- For two literals p_k and q_l in two clauses

$p_1 \vee p_2 \vee \ldots \vee p_n$

$q_1 \vee q_2 \vee \ldots \vee q_n$

such that θ =UNIFY(p_k, $\neg q_l$), derives

SUBST(θ, $p_1 \vee p_2 \vee \ldots p_{k-1} \vee p_{k+1} \vee \ldots \vee p_n \vee q_1 \vee q_2 \vee \ldots q_{l-1} \vee q_{l+1} \vee \ldots \vee q_n$)

- For resolution to apply, all sentences must be in **conjunctive normal form,**

Conjunction Normal Form (CNF) in PC

We like to prove: $KB \models \alpha$

equivalent to : $KB \wedge \neg\alpha$ *unsatifiable*

We first rewrite $KB \wedge \neg\alpha$ into conjunctive normal form (CNF).

A "conjunction of disjunctions"

$(A \vee \neg B) \wedge (B \vee \neg C \vee \neg D)$

literals

Clause   Clause

In theory
- Any KB can be converted into CNF.
- In fact, any KB can be converted into CNF-3, i.e. using clauses with at most 3 literals.

Example: Conversion to CNF in PC

$B_{1,1} \Leftrightarrow (P_{1,2} \vee P_{2,1})$

1. Eliminate $\Leftrightarrow$, replacing $\alpha \Leftrightarrow \beta$ with $(\alpha \Rightarrow \beta)\wedge(\beta \Rightarrow \alpha)$.

   $(B_{1,1} \Rightarrow (P_{1,2} \vee P_{2,1})) \wedge ((P_{1,2} \vee P_{2,1}) \Rightarrow B_{1,1})$

2. Eliminate $\Rightarrow$, replacing $\alpha \Rightarrow \beta$ with $\neg\alpha \vee \beta$.

   $(\neg B_{1,1} \vee P_{1,2} \vee P_{2,1}) \wedge (\neg(P_{1,2} \vee P_{2,1}) \vee B_{1,1})$

3. Move $\neg$ inwards using de Morgan's rules and double-negation: $\neg(\alpha \vee \beta) = \neg\alpha \wedge \neg\beta$

   $(\neg B_{1,1} \vee P_{1,2} \vee P_{2,1}) \wedge ((\neg P_{1,2} \wedge \neg P_{2,1}) \vee B_{1,1})$

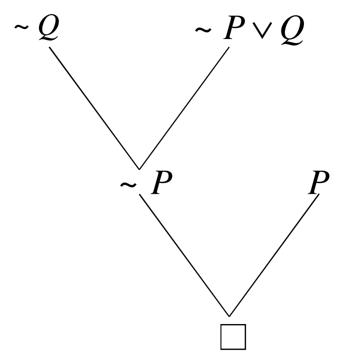
4. Apply distributive law ($\wedge$ over $\vee$) and flatten:

   $(\neg B_{1,1} \vee P_{1,2} \vee P_{2,1}) \wedge (\neg P_{1,2} \vee B_{1,1}) \wedge (\neg P_{2,1} \vee B_{1,1})$

Resolution in PC

1. P
2. $P \rightarrow Q$ converted to $\sim P \vee Q$
3. $\sim Q$

Draw the resolution tree (actually an inverted tree). Every node is a clausal form and branches are intermediate inference steps.



Resolution Algorithm in PC

- The resolution algorithm tries to prove: $KB \models \alpha$ *equivalent to* $KB \wedge \neg\alpha$ *unsatisfiable*

- Generate all new sentences from KB and the query.
- One of two things can happen:

1. We find $P \wedge \neg P$ which is unsatisfiable. i.e. we can entail the query.

2. We find no contradiction: there is a model that satisfies the sentence $KB \wedge \neg\alpha$ (non-trivial) and hence we cannot entail the query.

Resolution as Search

Given a database in clausal normal form KB

Find a sequence of resolution steps from KB to the empty clauses

<u>States:</u> current cnf KB + new clauses
<u>Operators:</u> resolution
<u>Initial state:</u> KB + negated goal
<u>Goal State:</u> a database containing the empty clause

Resolution Algorithm in FOPC

1) Convert sentences in the KB to CNF (clausal form)
2) Take the negation of the proposed query, convert it to CNF, and add it to the KB.
3) Repeatedly apply the resolution rule to derive new clauses.
4) If the empty clause (False) is eventually derived, stop and conclude that the proposed theorem is true.

<u>Procedure</u>:

✓Eliminate implications and biconditionals
✓Move ¬ inward
✓Standardize variables
✓Move quantifiers left
✓Skolemize: replace each existentially quantified variable with a Skolem constant or Skolem function
✓Distribute ∧ over ∨ to convert to conjunctions of clauses
✓Convert clauses to implications if desired for readability
(¬ a ∨ ¬ b ∨ c ∨ d) To a ∨ b => c ∨ d

Conversion to CNF

- Everyone who loves all animals is loved by someone:

$\forall x([\forall y\ \mathit{Animal}(y) \Rightarrow \mathit{Loves}(x,y)] \Rightarrow [\exists y\ \mathit{Loves}(y,x)])$

1. Eliminate biconditionals and implications

$\forall x([\neg\forall y\ (\neg \mathit{Animal}(y) \vee \mathit{Loves}(x,y))] \vee [\exists y\ \mathit{Loves}(y,x)])$

2. Move ¬ inwards:"$\neg\forall x\ p \equiv \exists x\ \neg p,\ \neg \exists x\ p \equiv \forall x\ \neg p$"

$\forall x\ ([\exists y\ (\neg(\neg \mathit{Animal}(y) \vee \mathit{Loves}(x,y)))] \vee [\exists y\ \mathit{Loves}(y,x)]\ )$

$\forall x\ ([\exists y\ (\neg\neg \mathit{Animal}(y) \wedge \neg \mathit{Loves}(x,y))] \vee [\exists y\ \mathit{Loves}(y,x)]\ )$

$\forall x([\exists y\ (\mathit{Animal}(y) \wedge \neg \mathit{Loves}(x,y))] \vee [\exists y\ \mathit{Loves}(y,x)]\ )$

Conversion to CNF contd.

3. Standardize variables: each quantifier should use a different one

$\forall x([\exists y\ \mathit{Animal}(y) \wedge \neg \mathit{Loves}(x,y)] \vee [\exists z\ \mathit{Loves}(z,x)])$

4. Skolemize: a more general form of existential instantiation.
Each existential variable is replaced by a Skolem function of the enclosing universally quantified variables:

$\forall x([\mathit{Animal}(F(x)) \wedge \neg \mathit{Loves}(x,F(x))] \vee \mathit{Loves}(G(x),x))$

5. Drop universal quantifiers:

$[\mathit{Animal}(F(x)) \wedge \neg \mathit{Loves}(x,F(x))] \vee \mathit{Loves}(G(x),x)$

6. Distribute ∨ over ∧ :

$[\mathit{Animal}(F(x)) \vee \mathit{Loves}(G(x),x)] \wedge [\neg \mathit{Loves}(x,F(x)) \vee \mathit{Loves}(G(x),x)]$

Recall: Resolution in PC

- Resolution inference rule (for CNF):

$$\frac{l_i \vee \ldots \vee l_k, \qquad m_1 \vee \ldots \vee m_n}{l_i \vee \ldots \vee l_{s-1} \vee l_{s+1} \vee \ldots \vee l_k \vee m_1 \vee \ldots \vee m_{j-1} \vee m_{j+1} \vee \ldots \vee m_n}$$

where l_s and m_j are complementary literals.

E.g., $\dfrac{P_{1,3} \vee P_{2,2}, \qquad \neg P_{2,2}}{P_{1,3}}$

Resolution is sound and complete for propositional logic

Resolution in FOL

- Full first-order version:

$$\frac{l_1 \vee \cdots \vee l_k, \qquad m_1 \vee \cdots \vee m_n}{(l_1 \vee \cdots \vee l_{i-1} \vee l_{i+1} \vee \cdots \vee l_k \vee m_1 \vee \cdots \vee m_{j-1} \vee m_{j+1} \vee \cdots \vee m_n)\theta}$$

where `Unify`(l_i, $\neg m_j$) = θ.

The two clauses are assumed to be standardized apart so that they share no variables.

- For example,

$$\frac{\neg Rich(x) \vee Unhappy(x) \qquad Rich(Ken)}{Unhappy(Ken)}$$

with θ = {x/Ken}

Empty Clause means False

- Resolution theorem proving ends
  - When the resolved clause has no literals (empty)
- This can only be because:
  - Two **unit clauses** were resolved
    - One was the negation of the other (after substitution)
  - Example: q(X) and ¬q(X) or: p(X) and ¬p(bob)
- Hence if we see the empty clause
  - This was because there was an inconsistency
  - Hence the proof by refutation

Resolution as Search

- **Initial State**: Knowledge base (KB) of axioms and negated theorem in CNF
- **Operators**: Resolution rule picks 2 clauses and adds new clause
- **Goal Test**: Does KB contain the empty clause?
- Search space of KB states

Socrates' Example

- KB: ***Socrates is a man and all men are mortal Therefore Socrates is mortal***
- Initial state

  1) is_man(socrates)

  2) ¬is_man(X) ∨ is_mortal(X)

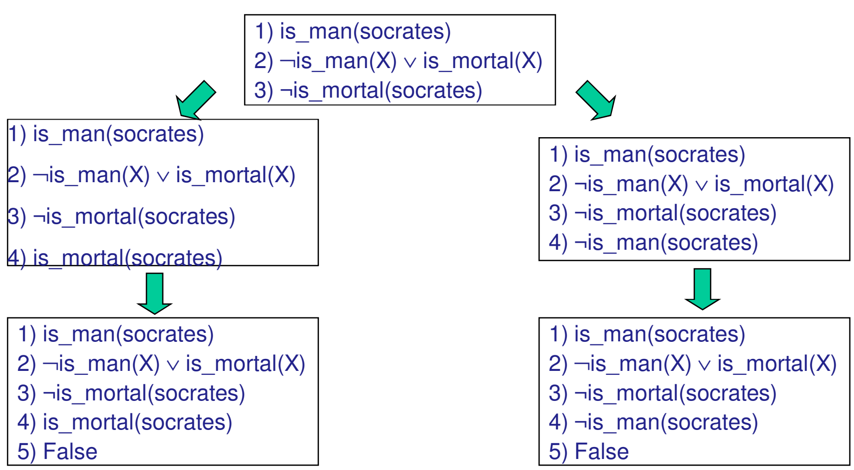
  3) ¬is_mortal(socrates)   (negation of theorem)
- Resolving (1) & (2) gives new state

  4) is_mortal(socrates)

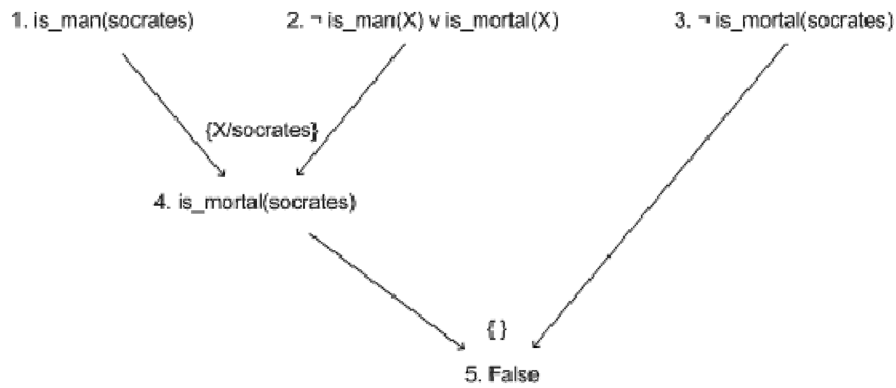
  Resolving (3) & (4) gives new state

  empty

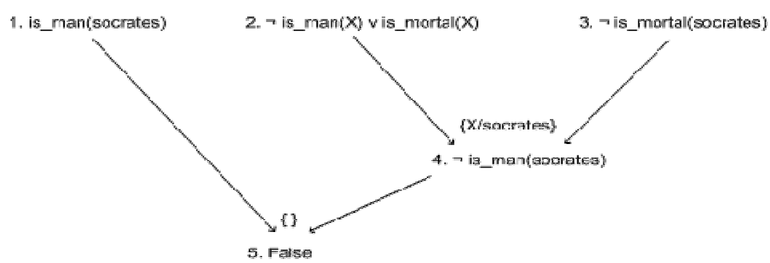
Aristotle's Example: Search Space



Resolution Proof Tree (Proof 1)



Resolution Proof Tree (Proof 2)

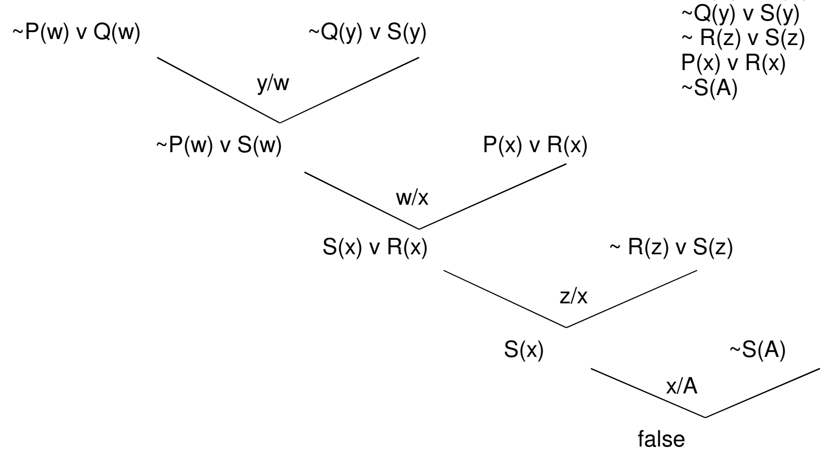


Read as:

> You said that all men were mortal. That means that for all things X, either X is not a man, or X is mortal. If we assume that Socrates is not mortal, then, given your previous statement, this means Socrates is not a man. But you said that Socrates ***is*** a man, which means that our assumption was false, so Socrates must be mortal.

Building Refutation resolution proof tree

KB
~P(w) v Q(w)
~Q(y) v S(y)
~ R(z) v S(z)
P(x) v R(x)
~S(A)



Conversion to CNF

- Everyone who loves all animals is loved by someone:

$\forall x([\forall y\ \textit{Animal}(y) \Rightarrow \textit{Loves}(x,y)] \Rightarrow [\exists y\ \textit{Loves}(y,x)])$

1. Eliminate biconditionals and implications

$\forall x([\neg\forall y\ (\neg \textit{Animal}(y) \vee \textit{Loves}(x,y))] \vee [\exists y\ \textit{Loves}(y,x)])$

2. Move ¬ inwards:"$\neg\forall x\ p \equiv \exists x\ \neg p$, $\neg \exists x\ p \equiv \forall x\ \neg p$"

$\forall x\ ([\exists y\ (\neg(\neg \textit{Animal}(y) \vee \textit{Loves}(x,y)))] \vee [\exists y\ \textit{Loves}(y,x)]\ )$

$\forall x\ ([\exists y\ (\neg\neg \textit{Animal}(y) \wedge \neg \textit{Loves}(x,y))] \vee [\exists y\ \textit{Loves}(y,x)]\ )$

$\forall x([\exists y\ (\textit{Animal}(y) \wedge \neg \textit{Loves}(x,y))] \vee [\exists y\ \textit{Loves}(y,x)]\ )$

Conversion to CNF contd.

3. Standardize variables: each quantifier should use a different one

   $\forall x([\exists y\ \mathit{Animal}(y) \wedge \neg \mathit{Loves}(x,y)] \vee [\exists z\ \mathit{Loves}(z,x)])$

4. Skolemize: a more general form of existential instantiation.
   Each existential variable is replaced by a Skolem function of the enclosing universally quantified variables:

   $\forall x([\mathit{Animal}(F(x)) \wedge \neg \mathit{Loves}(x,F(x))] \vee \mathit{Loves}(G(x),x))$

5. Drop universal quantifiers:

   $[\mathit{Animal}(F(x)) \wedge \neg \mathit{Loves}(x,F(x))] \vee \mathit{Loves}(G(x),x)$

6. Distribute $\vee$ over $\wedge$:

   $[\mathit{Animal}(F(x)) \vee \mathit{Loves}(G(x),x)] \wedge [\neg \mathit{Loves}(x,F(x)) \vee \mathit{Loves}(G(x),x)]$

Example: KB

Jack owns a dog.
Every dog owner is an animal lover.
No animal lover kills an animal.
Either Jack or Curiosity killed the cat, who is named Tuna.
Did Curiosity kill the cat?

Example: KB

Jack owns a dog.
Every dog owner is an animal lover.
No animal lover kills an animal.
Either Jack or Curiosity killed the cat, who is named Tuna.
Did Curiosity kill the cat?

A. $\exists x\ Dog(x) \land Owns(Jack, x)$
B. $\forall x\ (\exists y\ Dog(y) \land Owns(x, y)) \Rightarrow AnimalLover(x)$
C. $\forall x\ AnimalLover(x) \Rightarrow \forall y\ Animal(y) \Rightarrow \neg Kills(x, y)$
D. $Kills(Jack, Tuna) \lor Kills(Curiosity, Tuna)$
E. $Cat(Tuna)$
F. $\forall x\ Cat(x) \Rightarrow Animal(x)$

Example: (CNF)

A1. $Dog(D)$
A2. $Owns(Jack, D)$
B. $Dog(y) \land Owns(x, y) \Rightarrow AnimalLover(x)$
C. $AnimalLover(x) \land Animal(y) \land Kills(x, y) \Rightarrow False$
D. $Kills(Jack, Tuna) \lor Kills(Curiosity, Tuna)$
E. $Cat(Tuna)$
F. $Cat(x) \Rightarrow Animal(x)$

Example: Proof Tree