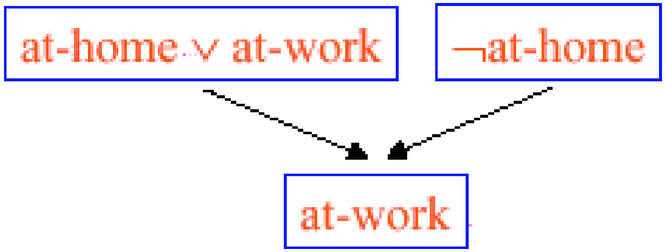
Resolution in propositional and first-order logic

- **Next time:**
  - More on resolution & theorem proving systems (Chapter 10 of R&N)
  - Read chapter 6 in Luger/Stubblefield about Prolog

Resolution in First-Order Logic

In propositional logic:

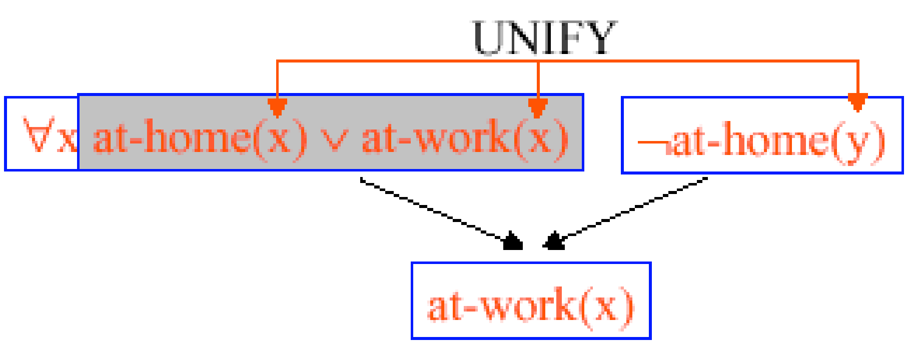


In first-order logic:

To generalize resolution proofs to FOL we must account for

- Predicates
- Unbound variables
- Existential & universal quantifiers

Idea: First convert sentences to clause form
Then unify variables



Outline

- Resolution in first-order logic
  - Proving logic sentences using resolution
  - Answering questions using resolution
  - Extensions to basic resolution
  - Resolution strategies

- Logic programming

Resolution in First-Order Logic

Basic steps for proving a conclusion S given premises

$\text{Premise}_1, \ldots, \text{Premise}_n$

(all expressed in FOL):

1. Convert all sentences to CNF
2. Negate conclusion S & convert result to CNF
3. Add negated conclusion S to the premise clauses
4. Repeat until contradiction or no progress is made:
   a. Select 2 clauses (call them parent clauses)
   b. Resolve them together, performing all required unifications
   c. If resolvent is the empty clause, a contradiction has been found (i.e., S follows from the premises)
   d. If not, add resolvent to the premises

If we succeed in Step 4, we have proved the conclusion

Resolution Examples

Example 1:

- If something is intelligent, it has common sense
- Deep Blue does not have common sense
- Prove that Deep Blue is not intelligent

1. $\forall x. I(x) \Rightarrow H(x)$
2. $\neg H(D)$

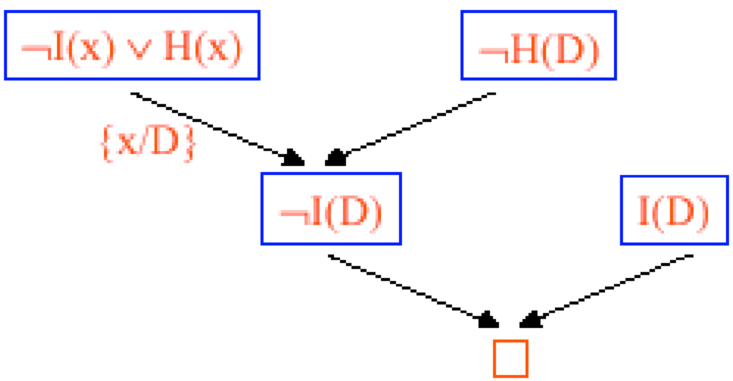
Conclusion: $\neg I(D)$
Denial: C3: $I(D)$

→

C1: $\neg I(x) \vee H(x)$
C2: $\neg H(D)$

CNF

A resolution proof of $\neg I(D)$:



Proof also written as:

2nd literal, 1st clause

C4: $\neg I(D)$ r[C1b, C2]

C5: □ r[C3, C4]

Resolution Examples (cont.)

Example 2:

Premises:

Mother(Lulu, Fifi)

Alive(Lulu)

$\forall x\ \forall y.\mathrm{Mother}(x,y) \Rightarrow \mathrm{Parent}(x,y)$

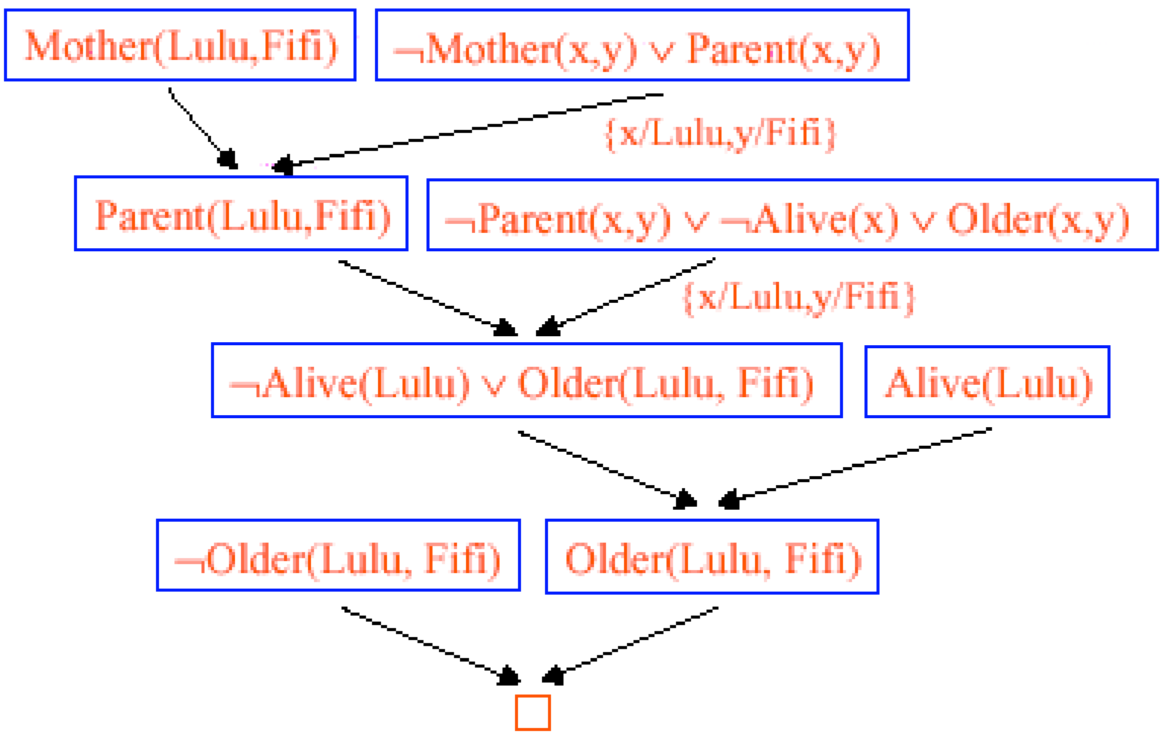
$\forall x\ \forall y.(\mathrm{Parent}(x,y) \wedge \mathrm{Alive}(x)) \Rightarrow \mathrm{Older}(x,y)$

Prove:

Older(Lulu, Fifi)

Denial:

¬Older(Lulu, Fifi)



Resolution Examples (cont.)

Could also have written the proof as:

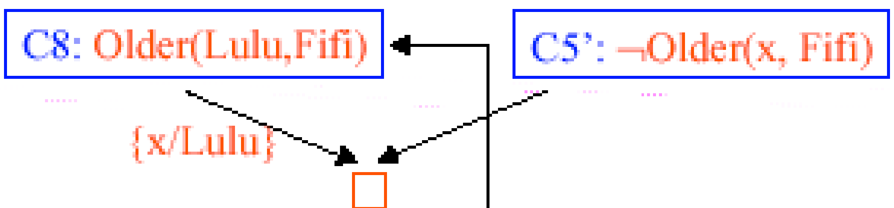
C1.	Mother(Lulu,Fifi)	given
C2.	Alive(Lulu)	given
C3.	¬Mother(x,y) ∨ Parent(x,y)	given
C4.	¬Parent(x,y) ∨ ¬Alive(x) ∨ Older(x,y)	given
C5.	¬Older(Lulu, Fifi)	denial of concl.
C6.	Parent(Lulu,Fifi)	r[C1,C3a]
C7.	¬Alive(Lulu) ∨ Older(Lulu, Fifi)	r[C6,C4a]
C8.	Older(Lulu, Fifi)	r[C8,C5]
C9.	□	

Proof consists of 4 resolution steps: longer than the proof with GMP, because we can only resolve two clauses at once using this form of resolution

Resolution Examples (cont.)

Example 3:

- Suppose the desired conclusion had been "Something is older than Fifi"
  ∃x.Older(x, Fifi)
- Denial:
  ¬∃x.Older(x, Fifi)
  also written as: ∀x.¬Older(x, Fifi)
  in clause form: ¬Older(x, Fifi)
- Last proof step would have been



Don't make mistake of first forming clause from conclusion & then denying it:

- Conclusion:
  ∃x.Older(x, Fifi)
  clause form: Older(C, Fifi)
  denial: ¬Older(C, Fifi)

Cannot unify Lulu,C!!

Outline

- Resolution in first-order logic
  - Proving logic sentences using resolution
  - Answering questions using resolution
  - Extensions to basic resolution
  - Resolution strategies

- Logic programming

Resolution for Question-Answering

- So far, resolution was used to just prove logic sentences
- Resolution's unification mechanism allows us to answer questions as well:
  - Consider again the proof of "Something is older than Fifi"
    $\exists x.Older(x, Fifi)$
  - Denial clause:
    $\neg Older(x, Fifi)$
  - Substitution made in disproof:
    $\{x/Lulu\}$
  - So Lulu is the "something" that's older than Fifi.
    →Answers question "what is older than Fifi?"

In general, to answer
"what x has such-and-such properties?"

- Prove "there exists an x with such-and-such properties"
- Extract substitution for x

Question-Answering

Example 1:

"Who is Lulu older than?"

- Prove that
  "there is an x such that Lulu is older than x"
- In FOL form:
  $\exists x.Older(Lulu, x)$
- Denial:
  $\neg\exists x.Older(Lulu, x)$
  $\forall x.\neg Older(Lulu, x)$
  in clause form: $\neg Older(Lulu, x)$
- Successful proof gives
  $\{x/Fifi\}$ [Verify!!]

Example 2:

"What is older than what?"

- In FOL form:
  $\exists x\ \exists y.Older(x, y)$
- Denial:
  $\neg\exists x\exists y.Older(x, y)$
  in clause form: $\neg Older(x, y)$
- Successful proof gives
  $\{x/Lulu, y/Fifi\}$ [Verify!!]

Getting Multiple Answers

- Assume additional facts:

  Father(BowWow, Fifi)

  $\neg$Father(x, y) $\vee$ Parent(x,y)

  Alive(BowWow)

- We can then answer $\exists$x.Older(x, Fifi) using

  {x/Lulu} or {x/BowWow}

  (i.e., 2 distinct proofs exist)

Q: Is it possible to find all answers to a given question using the resolution rule?

Ans: Yes, if the premises in the knowledge base are all Horn clauses

$$\neg A_1 \vee \neg A_2 \vee \ldots \vee \neg A_n \vee B$$

$$A_1 \wedge A_2 \wedge \ldots \wedge A_n \Rightarrow B$$

Achieved by finding all ways to refute a query

Getting Multiple Answers (cont.)

To find all ways of refuting $\neg$Older(x, Fifi):

- Find unit clauses this resolves with (if any), adding substitutions for successful refutations to **`Answers`**

  If Older(Fang, Fifi) was in KB, we would have
  `Answers = Answers ∪ {x/Fang}`

- Find clauses of the form

  $$\neg A_1 \vee \neg A_2 \vee \ldots \vee \neg A_n \vee \text{Older}(x, \text{Fifi})$$

  and resolve

- If successful, with unifier θ, recursively find all refutations of the corresponding antecedent instances ($\neg A_1, \neg A_2, \ldots, \neg A_n$)

- "Compose" the substitutions for these refutations with θ and add to **`Answers`**

Details in (R&N, p. 275)

Outline

- Resolution in first-order logic
    - Proving logic sentences using resolution
    - Answering questions using resolution
    - Extensions to basic resolution
    - Resolution strategies

- Logic programming

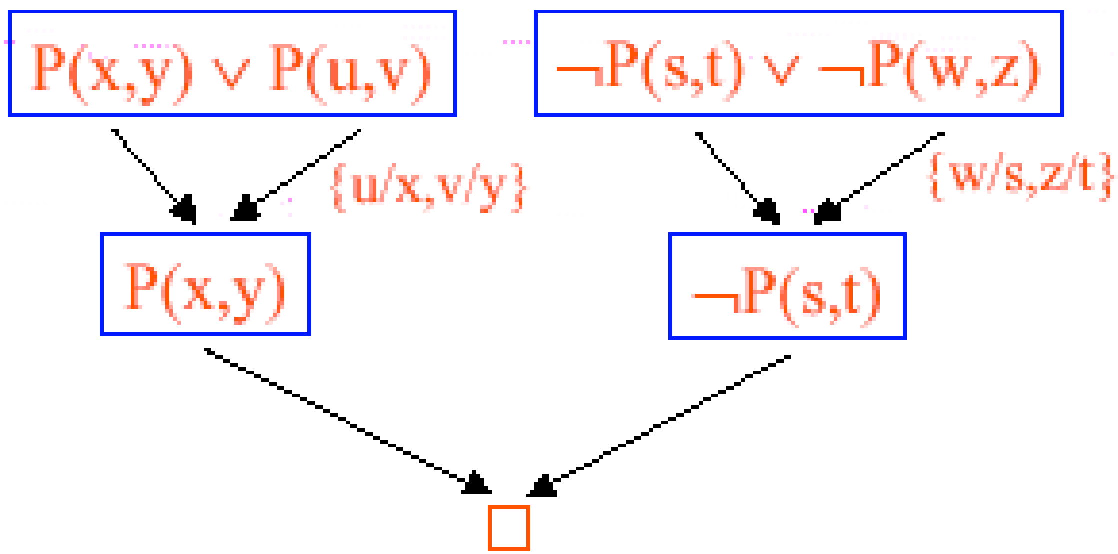
Factoring

- Resolution is "not quite" refutation-complete

  e.g. $P(x,y) \lor P(u,v)$ and $\neg P(s,t) \lor \neg P(w,z)$ are clearly contradictory, yet we can't derive □

- Factoring:

  Allows us to unify 2 literals of the same clause



Equality

- Suppose we are given:

  Older(Lulu, Fifi)

  ¬Older(x,x)

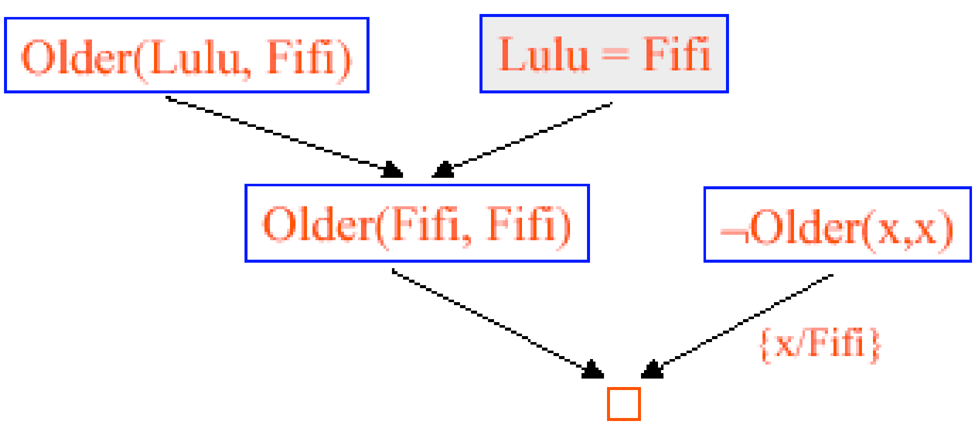
  resolution cannot be applied here

- Now, what if we know that Lulu & Fifi refer to the same entity?

  Need an additional rule & axioms to treat equality

  Paramodulation: essentially, substitution of equals (but with unification)

- Proving ¬(Lulu = Fifi):



Outline

- Resolution in first-order logic
  - Proving logic sentences using resolution
  - Answering questions using resolution
  - Extensions to basic resolution
  - Resolution strategies
- Logic programming

Resolution Strategies

In a general KB, there may be many resolutions that can be applied at a given step

	given
C2. Alive(Lulu)	given
	given
	given
C5. ¬Older(Lulu, Fifi)	denial of concl.
C6. Parent(Lulu,Fifi)	r[C1,C3a]
C7. ¬Alive(Lulu) ∨ Older(Lulu, Fifi)	r[C6,C4a]
C8. Older(Lulu, Fifi)	r[C8,C5]
C9. □	

We can use specific resolution strategies to ensure that we do not perform “useless” resolutions

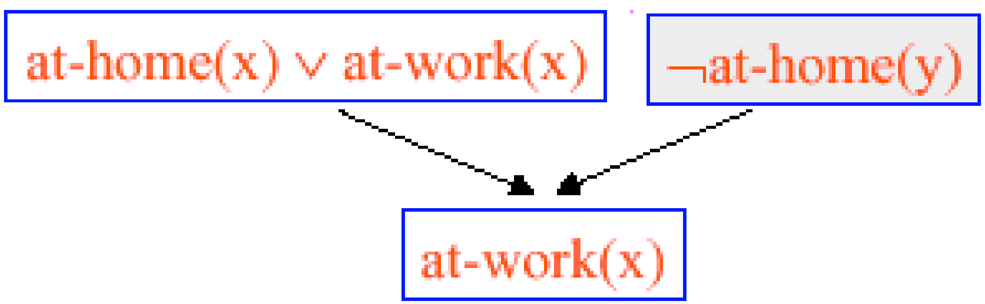
Resolution Strategies

- Backward chaining strategy:

  Reason backwards from a goal
  (used for finding multiple answers to a query)

- Unit resolution:

  One of the parent clauses is always chosen to contain a single literal



Idea: Length of resolvent always decreases by 1

→ gets closer to empty clause
(i.e., unit resolution is a Greedy method)

Caveat: Unit resolution is not complete!

Resolution Strategies (cont.)

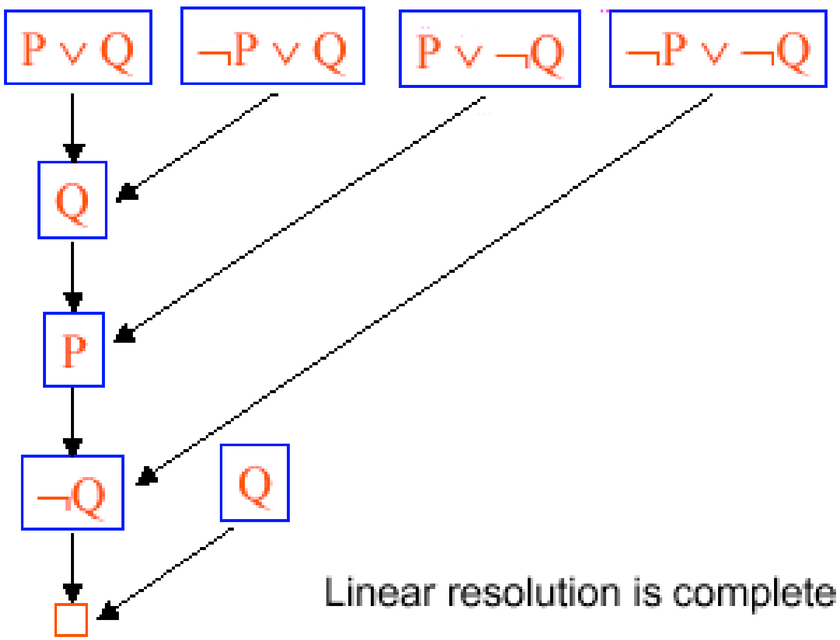
- Input resolution:

  One of the parent clauses is contained in the original KB

  Input resolution is equivalent to unit resolution (and hence also incomplete)

- Linear resolution:

  Each parent is a linear resolvent, i.e., is either in the initial KB or is an ansestor of the other parent



Linear resolution is complete

Resolution Strategies (cont.)

- Set-of-support resolution:

  Given a set of clauses Γ, a set of support resolvent of Γ is a resolvent whose parents are either clauses of Γ or descendants of such clauses

  Set-of-support resolution: always use a denial clause or a descendant of a denial clause as one parent

  Idea: "Focus" the proof on using the denial clause(s) to derive a contradiction rather than grinding arbitrary KB facts together

Outline

- Resolution in first-order logic
  - Proving logic sentences using resolution
  - Answering questions using resolution
  - Extensions to basic resolution
  - Resolution strategies

- Logic programming

Logic Programming

Robert Kowalski's equation:

Programming = Logic + Control

In logic programming, algorithms are created by augmenting logical sentences with information to control the inference process (Russell & Norvig)

An FOL definition of the list member function:

$\forall x \forall l . Member(x, [x|l])$
$\forall x \forall y \forall l . Member(x, l) \Rightarrow Member(x, [y|l])$

Logic programming can be thought of as a "declarative language"

Program = sequence declarations

Control = implicit

Program execution = proof

e.g., prove member(3, [2,1,3])