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:

\[ \text{at-home} \lor \text{at-work} \land \lnot \text{at-home} \]

\[ \text{at-work} \]

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

UNIFY

\[ \forall x \ \text{at-home}(x) \lor \text{at-work}(x) \land \lnot \text{at-home}(y) \]

\[ \text{at-work}(x) \]
Outline

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

• Logic programming
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)

\[ C1: \neg I(x) \lor H(x) \]
\[ C2: \neg H(D) \]

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

Proof also written as:
- C4: \( \neg I(D) \)
- C5: \( \square \)
- r[C1b, C2]
- r[C3, C4]
Resolution Examples (cont.)

Example 2:

Premises:
- Mother(Lulu, Fifi)
- Alive(Lulu)
- ∀x ∀y. Mother(x,y) ⇒ Parent(x,y)
- ∀x ∀y. (Parent(x,y) ∧ Alive(x)) ⇒ Older(x,y)

Prove:
- Older(Lulu, Fifi)

Denial:
- ¬Older(Lulu, Fifi)
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.
Example 3:

- Suppose the desired conclusion had been “Something is older than Fifi”
  \( \exists x. \text{Older}(x, \text{Fifi}) \)

- Denial:
  \( \neg \exists x. \text{Older}(x, \text{Fifi}) \)
  also written as: \( \forall x. \neg \text{Older}(x, \text{Fifi}) \)
  in clause form: \( \neg \text{Older}(x, \text{Fifi}) \)

- Last proof step would have been

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

- Conclusion:
  \( \exists x. \text{Older}(x, \text{Fifi}) \)
  clause form: \( \text{Older}(C, \text{Fifi}) \)
  denial: \( \neg \text{Older}(C, \text{Fifi}) \)

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”
    \[ \exists x. \text{Older}(x, \text{Fifi}) \]
  - Denial clause:
    \[ \neg \text{Older}(x, \text{Fifi}) \]
  - Substitution made in disproof:
    \[ \{x/\text{Lulu}\} \]
  - So Lulu is the “something” that’s older than Fifi.
    \[ \rightarrow \text{Answers question “what is older than Fifi?”} \]

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:
  \( \exists x. \text{Older}(\text{Lulu}, x) \)

- Denial:
  \( \neg \exists x. \text{Older}(\text{Lulu}, x) \)
  \( \forall x. \neg \text{Older}(\text{Lulu}, x) \)
  in clause form: \( \neg \text{Older}(\text{Lulu}, x) \)

- Successful proof gives
  \( \{x/\text{Fifi}\} \) [Verify!!]

Example 2:

“What is older than what?”

- In FOL form:
  \( \exists x \exists y. \text{Older}(x, y) \)

- Denial:
  \( \neg \exists x \exists y. \text{Older}(x, y) \)
  in clause form: \( \neg \text{Older}(x, y) \)

- Successful proof gives
  \( \{x/\text{Lulu}, y/\text{Fifi}\} \) [Verify!!]
Getting Multiple Answers

• Assume additional facts:
  Father(BowWow, Fifi)
  \neg Father(x, y) \lor 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 \lor \neg A_2 \lor \ldots \lor \neg A_n \lor B
  A_1 \land A_2 \land \ldots \land A_n \Rightarrow B

Achieved by finding all ways to refute a query
Getting Multiple Answers (cont.)

To find all ways of refuting \( \neg \text{Older}(x, \text{Fifi}) \):

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

  If \( \text{Older}(\text{Fang}, \text{Fifi}) \) was in KB, we would have

  \[
  \text{Answers} = \text{Answers} \cup \{x/\text{Fang}\}
  \]

- Find clauses of the form

  \[
  \neg A_1 \lor \neg A_2 \lor \cdots \lor \neg A_n \lor \text{Older}(x, \text{Fifi})
  \]

  and resolve

- If successful, with unifier \( \theta \), 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 \( \theta \) 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

\[
\begin{align*}
P(x,y) \lor P(u,v) & \quad \neg P(s,t) \lor \neg P(w,z) \\
\{u/x,v/y\} & \quad \{w/s,z/t\}
\end{align*}
\]
Equality

• Suppose we are given:
  \[
  \text{Older}(\text{Lulu}, \text{Fifi}) \\
  \neg \text{Older}(x, x)
  \]

  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

  \textbf{Paramodulation}: essentially, substitution of equals (but with unification)

• Proving \( \neg(\text{Lulu} = \text{Fifi}) \):

  \[
  \begin{align*}
  &\text{Older}(\text{Lulu}, \text{Fifi}) \quad \text{Lulu} = \text{Fifi} \\
  \quad \quad \text{Older}(\text{Fifi}, \text{Fifi}) \quad \neg \text{Older}(x, x) \\
  \quad \quad \quad \{x/\text{Fifi}\} \\
  \end{align*}
  \]
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.

\[ C2. \text{Alive}(Lulu) \]

\[ C5. \neg \text{Older}(Lulu, Fifi) \]
\[ C6. \text{Parent}(Lulu, Fifi) \]
\[ C7. \neg \text{Alive}(Lulu) \lor \text{Older}(Lulu, Fifi) \]
\[ C8. \text{Older}(Lulu, Fifi) \]
\[ C9. \square \]

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

\[ \text{at-home}(x) \lor \text{at-work}(x) \quad \text{at-home}(y) \]

\[ \text{at-work}(x) \]

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 ancestor of the other parent.

```
P ∨ Q  ¬P ∨ Q  P ∨ ¬Q  ¬P ∨ ¬Q
```

Linear resolution is complete.
Resolution Strategies (cont.)

- **Set-of-support resolution:**
  
  Given a set of clauses $\Gamma$, a set of support resolvent of $\Gamma$ is a resolvent whose parents are either clauses of $\Gamma$ 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:

\[
\text{Programming} = \text{Logic} + \text{Control}
\]

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:

\[
\forall x \forall l. \text{Member}(x, [x|l]) \\
\forall x \forall y \forall l. \text{Member}(x, l) \Rightarrow \text{Member}(x, [y|l])
\]

Logic programming can be thought of as a “declarative language”

\[
\text{Program} = \text{sequence declarations} \\
\text{Control} = \text{implicit} \\
\text{Program execution} = \text{proof} \\
\text{e.g., prove} \ \text{member}(3, [2, 1, 3])
\]