Overview of Inference in First-Order Logic

Chapter 9
Outline

- Reducing first-order inference to propositional inference
- Lifting inference in propositional logic to first-order logic.
  - Unification
  - Resolution
Two Approaches for Inference in FOL

Propositionalisation:

- Treat a first-order sentences as a template.
- Instantiating all variables with all possible constants gives a set of ground propositional clauses.
- Apply efficient propositional solver, e.g. SAT.
Two Approaches for Inference in FOL

Propositionalisation:

• Treat a first-order sentences as a template.
• Instantiating all variables with all possible constants gives a set of ground propositional clauses.
• Apply efficient propositional solver, e.g. SAT.

Lifted Inference:

• Generalize propositional methods to 1\textsuperscript{st}-order methods.
• Issue: dealing with variables and quantifiers
• Rule of inference: resolution
• Unification: instantiate variables where necessary.
Propositionalisation

- **Easy case**: A finite world in which all individuals have names
  - E.g. the wumpus world
  - But also many planning, scheduling, etc. problems
Propositionalisation

- **Easy case:** A finite world in which all individuals have names
  - E.g. the wumpus world
  - But also many planning, scheduling, etc. problems

- **Idea:**
  - Replace a universally-quantified sentence with all of its instances
  - Replace an existentially-quantified sentence with a disjunction of its instances
Propositionalisation

- **Easy case:** A finite world in which all individuals have names
  - E.g. the wumpus world
  - But also many planning, scheduling, etc. problems
- **Idea:**
  - Replace a universally-quantified sentence with all of its instances
  - Replace an existentially-quantified sentence with a disjunction of its instances
- A formula (KB, etc.) with no variables is called *ground*
- **Inference procedure:**
  - Ground the KB and the query, and
  - Run an inference procedure for propositional logic.
Universals

- E.g., $\forall x \ King(x) \land Greedy(x) \Rightarrow Evil(x)$

  yields

  $King(John) \land Greedy(John) \Rightarrow Evil(John)$
  $King(Richard) \land Greedy(Richard) \Rightarrow Evil(Richard)$
  $King(car_{54}) \land Greedy(car_{54}) \Rightarrow Evil(car_{54})$
  ...

Existentials

• E.g., $\exists x \text{ Likes}(John, x)$

  yields

  $$\text{Likes}(John, John) \lor \text{Likes}(John, Richard) \lor \cdots \lor \text{Likes}(John, \text{car}_{54}) \lor \cdots$$
Existentials

- E.g., \( \exists x \ Likes(John, x) \)
  
  yields

  \[
  Likes(John, John) \lor Likes(John, Richard) \lor \cdots \lor Likes(John, \text{car}_{54}) \lor \cdots
  \]

  \[Q:\text{ What does “Everyone likes someone” look like?}\]
Reduction to propositional inference

- Suppose the KB contains just the following:
  \[ \forall x \text{King}(x) \land \text{Greedy}(x) \Rightarrow \text{Evil}(x) \]
  \[ \text{King}(\text{John}), \text{Greedy}(\text{John}), \text{Brother}(\text{Richard}, \text{John}) \]

- Instantiating the universal sentence in all possible ways, we get
  \[ \text{King}(\text{John}) \land \text{Greedy}(\text{John}) \Rightarrow \text{Evil}(\text{John}) \]
  \[ \text{King}(\text{Richard}) \land \text{Greedy}(\text{Richard}) \Rightarrow \text{Evil}(\text{Richard}) \]
  \[ \text{King}(\text{John}), \text{Greedy}(\text{John}), \text{Brother}(\text{Richard}, \text{John}) \]

- The new KB is propositionalized.

- Proposition symbols are
  \[ \text{King}(\text{John}), \]
  \[ \text{Greedy}(\text{John}), \]
  \[ \text{Brother}(\text{John}, \text{Richard}), \]
  \[ \text{Brother}(\text{John}, \text{John}), \text{etc.} \]
Problems with propositionalization

- Usually generates lots of irrelevant sentences.
- E.g., consider:
  \[
  \forall x \ King(x) \land Greedy(x) \Rightarrow Evil(x), \\
  \forall y \ Greedy(y), \\
  King(John), \quad Brother(Richard, John)
  \]
- For query \( Evil(John) \), propositionalization produces lots of facts (like \( Greedy(Richard) \)) that are irrelevant
- \( k \)-ary predicate and \( n \) constants \( \Rightarrow n^k \) instances
Problems with propositionalization

- Usually generates lots of irrelevant sentences.
- E.g., consider:
  \[ \forall x \ King(x) \land Greedy(x) \Rightarrow Evil(x), \]
  \[ \forall y \ Greedy(y), \]
  \[ King(John), \quad Brother(Richard, John) \]
  - For query \( Evil(John) \), propositionalization produces lots of facts (like \( Greedy(Richard) \)) that are irrelevant

- \( k \)-ary predicate and \( n \) constants \( \Rightarrow n^k \) instances
- However, many recent AI applications use propositionalization for FO KBs over a finite domain.
  - Has led to work in \textit{intelligent grounding}.
- Can make propositionalization work for \textit{arbitrary} FO theories
  - See text for more
General FOL: Dealing with Variables

Consider the KB:
\[
\{ \forall x (Grad(x) \Rightarrow Student(x)), \\
\forall y (Student(y) \Rightarrow Happy(y)), \\
Grad(ZeNian), \\
UGrad(Andrei) \}
\]

- Intuitively \(Happy(ZeNian)\) is inferrable.
- For such a deduction \(Andrei\) is irrelevant.

Idea: Try to limit instantiation of variables to \textit{useful} instances.
Unification

• If two formulas can be made the same by substitutions of variables, they are said to be *unified*

• Unification is the process of making 2 formulas (terms, etc) the same by finding an appropriate substitution for variables.

\[
\forall x (\text{Grad}(x) \Rightarrow \text{Student}(x)), \text{Grad}(\text{ZeNian})
\]

• To obtain \(\text{Student}(\text{ZeNian})\) we have the following steps:

• Figure out how to make \(\text{Grad}(x)\) and \(\text{Grad}(\text{ZeNian})\) the same.

• This is easy: Bind \(x\) to \(\text{ZeNian}\).

• Substituting, we get the rule instance:

\[
\text{Grad}(\text{ZeNian}) \Rightarrow \text{Student}(\text{ZeNian}).
\]

• Can now derive \(\text{Student}(\text{ZeNian})\).
Unification

• If two formulas can be made the same by substitutions of variables, they are said to be \textit{unified}.

• Unification is the process of making 2 formulas (terms, etc) the same by finding an appropriate substitution for variables.

• Consider:

\[ \forall x (\text{Grad}(x) \Rightarrow \text{Student}(x)), \quad \text{Grad}(\text{ZeNian}) \]
Unification

• If two formulas can be made the same by substitutions of variables, they are said to be unified.

• Unification is the process of making 2 formulas (terms, etc) the same by finding an appropriate substitution for variables.

• Consider:
  \[ \forall x (\text{Grad}(x) \Rightarrow \text{Student}(x)), \quad \text{Grad}(\text{ZeNian}) \]

• To obtain \text{Student}(\text{ZeNian}) we have the following steps:
Unification

- If two formulas can be made the same by substitutions of variables, they are said to be *unified*.
- Unification is the process of making 2 formulas (terms, etc) the same by finding an appropriate substitution for variables.
- Consider:
  $$\forall x (\text{Grad}(x) \Rightarrow \text{Student}(x)), \quad \text{Grad}(\text{ZeNian})$$
- To obtain \textit{Student(ZeNian)} we have the following steps:
  - Figure out how to make \textit{Grad}(x) and \textit{Grad(ZeNian)} the same.
    - This is easy: Bind \(x\) to \(\text{ZeNian}\).
Unification

- If two formulas can be made the same by substitutions of variables, they are said to be *unified*.
- Unification is the process of making 2 formulas (terms, etc) the same by finding an appropriate substitution for variables.

Consider:
\[ \forall x (Grad(x) \Rightarrow Student(x)), \quad \text{Grad(ZeNian)} \]

To obtain *Student(ZeNian)* we have the following steps:
- Figure out how to make \(\text{Grad}(x)\) and \(\text{Grad(ZeNian)}\) the same.
  - This is easy: Bind \(x\) to \(\text{ZeNian}\).
- Substituting, we get the rule instance:
  \[ \text{Grad(ZeNian)} \Rightarrow \text{Student(ZeNian)}. \]
Unification

- If two formulas can be made the same by substitutions of variables, they are said to be unified.
- Unification is the process of making two formulas (terms, etc) the same by finding an appropriate substitution for variables.
- Consider:
  \[ \forall x (Grad(x) \Rightarrow Student(x)), \quad Grad(ZeNian) \]
- To obtain \(Student(ZeNian)\) we have the following steps:
  - Figure out how to make \(Grad(x)\) and \(Grad(ZeNian)\) the same.
    - This is easy: Bind \(x\) to \(ZeNian\).
  - Substituting, we get the rule instance:
    \[Grad(ZeNian) \Rightarrow Student(ZeNian).\]
  - Can now derive \(Student(ZeNian)\).
Unification Examples

Look for substitution $\theta$ such that $\alpha \theta = \beta \theta$

<table>
<thead>
<tr>
<th>$\alpha$</th>
<th>$\beta$</th>
<th>$\theta$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$\text{Knows}(\text{John}, x)$</td>
<td>$\text{Knows}(\text{John}, \text{Jane})$</td>
<td></td>
</tr>
<tr>
<td>$\text{Knows}(\text{John}, x)$</td>
<td>$\text{Knows}(y, \text{OJ})$</td>
<td></td>
</tr>
<tr>
<td>$\text{Knows}(\text{John}, x)$</td>
<td>$\text{Knows}(y, \text{Mother}(y))$</td>
<td></td>
</tr>
<tr>
<td>$\text{Knows}(\text{John}, x)$</td>
<td>$\text{Knows}(x, \text{OJ})$</td>
<td></td>
</tr>
</tbody>
</table>
### Unification Examples

Look for substitution $\theta$ such that $\alpha\theta = \beta\theta$

<table>
<thead>
<tr>
<th>$\alpha$</th>
<th>$\beta$</th>
<th>$\theta$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$\text{Knows}(\text{John}, x)$</td>
<td>$\text{Knows}(\text{John}, \text{Jane})$</td>
<td>${x/\text{Jane}}$</td>
</tr>
<tr>
<td>$\text{Knows}(\text{John}, x)$</td>
<td>$\text{Knows}(y, \text{OJ})$</td>
<td></td>
</tr>
<tr>
<td>$\text{Knows}(\text{John}, x)$</td>
<td>$\text{Knows}(y, \text{Mother}(y))$</td>
<td></td>
</tr>
<tr>
<td>$\text{Knows}(\text{John}, x)$</td>
<td>$\text{Knows}(x, \text{OJ})$</td>
<td></td>
</tr>
</tbody>
</table>
Unification Examples

Look for substitution $\theta$ such that $\alpha\theta = \beta\theta$

<table>
<thead>
<tr>
<th>$\alpha$</th>
<th>$\beta$</th>
<th>$\theta$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$\text{Knows}(\text{John}, x)$</td>
<td>$\text{Knows}(\text{John}, \text{Jane})$</td>
<td>${x/\text{Jane}}$</td>
</tr>
<tr>
<td>$\text{Knows}(\text{John}, x)$</td>
<td>$\text{Knows}(y, \text{OJ})$</td>
<td>${x/\text{OJ}, y/\text{John}}$</td>
</tr>
<tr>
<td>$\text{Knows}(\text{John}, x)$</td>
<td>$\text{Knows}(y, \text{Mother}(y))$</td>
<td></td>
</tr>
<tr>
<td>$\text{Knows}(\text{John}, x)$</td>
<td>$\text{Knows}(x, \text{OJ})$</td>
<td></td>
</tr>
</tbody>
</table>
Unification Examples

Look for substitution $\theta$ such that $\alpha \theta = \beta \theta$

<table>
<thead>
<tr>
<th>$\alpha$</th>
<th>$\beta$</th>
<th>$\theta$</th>
</tr>
</thead>
<tbody>
<tr>
<td>\text{Knows}(John, x)</td>
<td>\text{Knows}(John, Jane)</td>
<td>{x / Jane}</td>
</tr>
<tr>
<td>\text{Knows}(John, x)</td>
<td>\text{Knows}(y, OJ)</td>
<td>{x / OJ, y / John}</td>
</tr>
<tr>
<td>\text{Knows}(John, x)</td>
<td>\text{Knows}(y, \text{Mother}(y))</td>
<td>{y / John, x / \text{Mother}(John)}</td>
</tr>
<tr>
<td>\text{Knows}(John, x)</td>
<td>\text{Knows}(x, OJ)</td>
<td></td>
</tr>
</tbody>
</table>
Unification Examples

Look for substitution $\theta$ such that $\alpha\theta = \beta\theta$

<table>
<thead>
<tr>
<th>$\alpha$</th>
<th>$\beta$</th>
<th>$\theta$</th>
</tr>
</thead>
<tbody>
<tr>
<td>Knows(John, x)</td>
<td>Knows(John, Jane)</td>
<td>${x/\text{Jane}}$</td>
</tr>
<tr>
<td>Knows(John, x)</td>
<td>Knows(y, OJ)</td>
<td>${x/OJ, y/\text{John}}$</td>
</tr>
<tr>
<td>Knows(John, x)</td>
<td>Knows(y, Mother(y))</td>
<td>${y/\text{John}, x/\text{Mother(John)}}$</td>
</tr>
<tr>
<td>Knows(John, x)</td>
<td>Knows(x, OJ)</td>
<td>fail</td>
</tr>
</tbody>
</table>

**Problem:** Can’t substitute both John and OJ for $x$ at the same time.

**Solution:** Standardize variables apart:
- Replace $\text{Knows}(x, \text{OJ})$ with $\text{Knows}(y, \text{OJ})$
Reasoning and Unification

• Unification lets us work with both universally quantified variables and arbitrary terms.

• We can use unification in rules such as:

  \[ \text{Parent}(x, y) \land \text{Parent}(y, z) \Rightarrow \text{GrandParent}(x, z) \]

  where the variables are taken as being universally quantified.

• Then forward chaining and backward chaining with unification can be defined for such rules.

ForKeyward chaining, following one line of development, one ends up with the programming language Prolog.
Resolution: Brief summary

- Resolution can be used in the first-order case (where it forms the basis for much of theorem proving).
- Full first-order version:
  \[
  \frac{\ell_1 \lor C_1, \quad \ell_2 \lor C_2}{(C_1 \lor C_2)\theta} \quad \text{where } \ell_1\theta = \neg\ell_2\theta.
  \]

- For example,
  \[
  \neg\text{Rich}(x) \lor \text{Unhappy}(x)
  \frac{\text{Rich}(Ken)}{\text{Unhappy}(Ken)} \quad \text{with } \theta = \{x/\text{Ken}\}
  \]

- For details see the text or CMPT 411.
Inference in FOL

For $KB$ and query $\alpha$: 

- Convert $KB \land \neg \alpha$ to CNF.
  - This is trickier than in propositional logic, since we have to deal with variables and quantifiers.
- Apply resolution steps to $CNF(KB \land \neg \alpha)$
  - No longer guaranteed to terminate if satisfiable
  - FOL is *undecidable*

้ว Complete for FOL
Summary

• Propositionalization
  • Grounding approach: reduce all sentences to PL and apply propositional inference techniques.

• FOL/Lifted inference techniques
  • Propositional techniques + Unification.
  • Generalized Modus Ponens
  • Resolution-based inference.

• Many other aspects of FOL inference not discussed in class