Inference in first-order logic

Chapter 9, Sections 1–4

Proofs

Sound inference: find α such that KB ⊨ α.
Proof process is a search, operators are inference rules.

E.g., Modus Ponens (MP)

\[ α, α \Rightarrow β \quad \text{At}(\text{Joe}, \text{UCB}) \quad \text{At}(\text{Joe}, \text{UCB}) \Rightarrow \text{OK}(\text{Joe}) \]

\[ β \quad \text{OK}(\text{Joe}) \]

E.g., And-Introduction (AI)

\[ α, β \quad \text{OK}(\text{Joe}) \quad \text{CSMajor}(\text{Joe}) \]

\[ α \land β \quad \text{OK}(\text{Joe}) \land \text{CSMajor}(\text{Joe}) \]

E.g., Universal Elimination (UE)

\[ ∀x α \quad ∀x \text{At}(x, \text{UCB}) \Rightarrow \text{OK}(x) \]

\[ α[x/τ] \quad \text{At}(\text{Pat}, \text{UCB}) \Rightarrow \text{OK}(\text{Pat}) \]

τ must be a ground term (i.e., no variables)

Example proof

Bob is a buffalo

Pat is a pig

Buffaloes outrun pigs

Bob outruns Pat

1. Bufal(o)\{Bob\}
2. Pig\{Pat\}
3. ∀x, y Bufal(o)(x) ∧ Pig(y) ⇒ Faster(x, y)

<table>
<thead>
<tr>
<th>Al 1 &amp; 2</th>
<th>4. Bufal(o){Bob} \land Pig{Pat}</th>
</tr>
</thead>
</table>

UE 3, \{x/\text{Bob}, y/\text{Pat}\}

5. Bufal(o)\{Bob\} \land Pig\{Pat\} ⇒ Faster(\text{Bob}, \text{Pat})
Search with primitive inference rules

Operators are inference rules
States are sets of sentences
Goal test checks state to see if it contains query sentence

MP 6 & 7 6. Faster(Bob, Pat)

---

Unification

A substitution σ unifies atomic sentences p and q if pσ = qσ

<table>
<thead>
<tr>
<th>p</th>
<th>Kown(John, x)</th>
<th>Kown(John, Jane)</th>
<th>σ</th>
</tr>
</thead>
<tbody>
<tr>
<td>q</td>
<td>Kown(John, x)</td>
<td>Kown(y, OJ)</td>
<td></td>
</tr>
<tr>
<td>Kown(x, x)</td>
<td>Kown(y, Mother(y))</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Idea: Unify rule premises with known facts, apply unifier to conclusion
E.g., if we know q and Kown(John, x) ⇒ Likes(John, x)
then we conclude Likes(John, Jane)
Likes(John, OJ)
Likes(John, Mother(John))

---

Generalized Modus Ponens (GMP)

\[ p_1', p_2', \ldots, p_n', (p_1 \land p_2 \land \ldots \land p_n \Rightarrow q) \Rightarrow q \sigma \]

where \( p_i' \sigma = p_i \sigma \) for all \( i \)

E.g., \( p_1' = \text{Faster}(Bob, Pat) \)
\( p_2' = \text{Faster}(Pat, Steve) \)
\( p_1 \land p_2 \Rightarrow q = \text{Faster}(x, y) \land \text{Faster}(y, z) \Rightarrow \text{Faster}(x, z) \)
\( \sigma = \{x/John, y/Pat, z/Steve\} \)
\( q \sigma = \text{Faster}(Bob, Steve) \)

GMP used with KB of definite clauses (exactly one positive literal):
either a single atomic sentence or
(conjunction of atomic sentences) ⇒ (atomic sentence)
All variables assumed universally quantified

---

Soundness of GMP

Need to show that
\[ p_1', \ldots, p_n', (p_1 \land \ldots \land p_n \Rightarrow q) \Rightarrow q \sigma \]
provided that \( p_i' \sigma = p_i \sigma \) for all \( i \)

Lemma: For any definite clause \( p \), we have \( p \models p \sigma \) by UE
1. \( (p_1 \land \ldots \land p_n \Rightarrow q) \models (p_1 \sigma \land \ldots \land p_n \sigma \Rightarrow q) \sigma \)
2. \( p_1', \ldots, p_n' \models p_1' \land \ldots \land p_n' \Rightarrow p_1' \sigma \land \ldots \land p_n' \sigma \)
3. From 1 and 2, \( q \sigma \) follows by simple MP
**Forward chaining**

When a new fact \( p \) is added to the KB
for each rule such that \( p \) unifies with a premise
if the other premises are known
then add the conclusion to the KB and continue chaining

Forward chaining is data-driven
- e.g., inferring properties and categories from percepts

---

**Forward chaining example**

Add facts 1, 2, 3, 4, 5, 7 in turn.
Number in \([]\) = unification literal; \( \checkmark \) indicates rule firing

1. \( \text{Buffalo}(x) \land \text{Pig}(y) \Rightarrow \text{Faster}(x, y) \)
2. \( \text{Pig}(y) \land \text{Slug}(z) \Rightarrow \text{Faster}(y, z) \)
3. \( \text{Faster}(x, y) \land \text{Faster}(y, z) \Rightarrow \text{Faster}(x, z) \)
4. \( \text{Buffalo}(Bob) \land [3a, x] \)
5. \( \text{Pig}(Pat) \land [1b, \checkmark] \Rightarrow \checkmark \text{Faster}(Bob, Pat) \land [3a, x], [3b, x] \)

---

**Backward chaining**

When a query \( q \) is asked
- if a matching fact \( q' \) is known, return the unifier
  for each rule whose consequent \( q' \) matches \( q \)
  attempt to prove each premise of the rule by backward chaining

(Some added complications in keeping track of the unifiers)
(More complications help to avoid infinite loops)

Two versions: find any solution, find all solutions

Backward chaining is the basis for logic programming, e.g., Prolog

---

**Backward chaining example**

1. \( \text{Pig}(y) \land \text{Slug}(z) \Rightarrow \text{Faster}(y, z) \)
2. \( \text{Slimy}(z) \land \text{Creeps}(z) \Rightarrow \text{Slug}(z) \)
3. \( \text{Pig}(Pat) \land [\{\}, y/Pat, z/\text{Steve}] \)
4. \( \text{Slimy}(\text{Steve}) \)
5. \( \text{Creeps}(\text{Steve}) \)

\( \text{Faster}(\text{Pat}, \text{Steve}) \)

\( \text{Pig}(\text{Pat}) \)

\( \text{Slug}(\text{Steve}) \)

\( \{\} \)

\( \{\} \)

---