INFEERENCE IN
FIRST-ORDER LOGIC

CHAPTER 9, SECTIONS 1–4

Outline

♦ Proofs
♦ Unification
♦ Generalized Modus Ponens
♦ Forward and backward chaining
**Proofs**

Sound inference: find $\alpha$ such that $KB \models \alpha$.

Proof process is a search, operators are inference rules.

E.g., Modus Ponens (MP)

$$
\frac{\alpha, \quad \alpha \Rightarrow \beta \quad \text{At}(Joe, UCB) \quad \text{At}(Joe, UCB)}{\beta \quad \text{OK}(Joe)}
$$

E.g., And-Introduction (AI)

$$
\frac{\alpha \quad \beta \quad \text{OK}(Joe) \quad \text{CSMajor}(Joe)}{\alpha \land \beta \quad \text{OK}(Joe) \land \text{CSMajor}(Joe)}
$$

E.g., Universal Elimination (UE)

$$
\frac{\forall x \quad \alpha \quad \forall x \quad \text{At}(x, UCB) \Rightarrow \text{OK}(x)}{\alpha \{x/\tau\} \quad \text{At}(Pat, UCB) \Rightarrow \text{OK}(Pat)}
$$

$\tau$ must be a ground term (i.e., no variables)

---

**Example proof**

| Bob is a buffalo | 1. Buffalo(Bob) |
| Pat is a pig | 2. Pig(Pat) |
| Buffaloes outrun pigs | 3. $\forall x, y$ Buffalo($x$) $\land$ Pig($y$) $\Rightarrow$ Faster($x, y$) |
| Bob outruns Pat |

---
Al 1 & 2

4. $\text{Buffalo}(\text{Bob}) \land \text{Pig}(\text{Pat})$

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

5. $\text{Buffalo}(\text{Bob}) \land \text{Pig}(\text{Pat}) \Rightarrow \text{Faster}(\text{Bob}, \text{Pat})$
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

AI, UE, MP is a common inference pattern

Problem: branching factor huge, esp. for UE

Idea: find a substitution that makes the rule premise match some known facts
⇒ a single, more powerful inference rule
A substitution $\sigma$ unifies atomic sentences $p$ and $q$ if $p\sigma = q\sigma$

<table>
<thead>
<tr>
<th>$p$</th>
<th>$q$</th>
<th>$\sigma$</th>
</tr>
</thead>
<tbody>
<tr>
<td>Knows(John,x)</td>
<td>Knows(John,Jane)</td>
<td></td>
</tr>
<tr>
<td>Knows(John,x)</td>
<td>Knows(y,OJ)</td>
<td></td>
</tr>
<tr>
<td>Knows(John,x)</td>
<td>Knows(y,Mother(y))</td>
<td></td>
</tr>
</tbody>
</table>

*Idea*: Unify rule premises with known facts, apply unifier to conclusion

E.g., if we know $q$ and then we conclude

$\text{Knows}(\text{John},x) \Rightarrow \text{Likes}(\text{John},x)$

$\text{Likes}(\text{John},\text{Jane})$

$\text{Likes}(\text{John},\text{OJ})$

$\text{Likes}(\text{John},\text{Mother}(\text{John}))$
Generalized Modus Ponens (GMP)

\[
p_1', \ p_2', \ldots, \ p_n', \ (p_1 \land p_2 \land \ldots \land p_n \Rightarrow q) \quad \frac{\Box}{q \sigma} \quad \text{where } p_i' \sigma = p_i \sigma \text{ for all } i
\]

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

GMP used with KB of definite clauses (exactly one positive literal):
- either a single atomic sentence or
- (conjunction of atomic sentences) \(\Rightarrow\) (atomic sentence)

All variables assumed universally quantified.

Soundness of GMP

Need to show that

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

provided that \( p_i' \sigma = p_i \sigma \text{ for all } i \)

Lemma: For any definite clause \( p \), we have \( p \models p \sigma \) by UE

1. \( (p_1 \land \ldots \land p_n \Rightarrow q) \models (p_1 \land \ldots \land p_n \Rightarrow q) \sigma = (p_1 \sigma \land \ldots \land p_n \sigma \Rightarrow q \sigma) \)

2. \( p_1', \ldots, p_n' \models p_1' \land \ldots \land p_n' \models p_1' \sigma \land \ldots \land p_n' \sigma \)

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. \( Buffalo(x) \land Pig(y) \Rightarrow Faster(x, y) \)
2. \( Pig(y) \land Slug(z) \Rightarrow Faster(y, z) \)
3. \( Faster(x, y) \land Faster(y, z) \Rightarrow Faster(x, z) \)
4. \( Buffalo(Bob) [1a, \checkmark] \)
5. \( Pig(Pat) [1b, \checkmark] \rightarrow 6. Faster(Bob, Pat) [3a, \checkmark], [3b, \times] \)
   \[2a, \checkmark]\n7. \( Slug(Steve) [2b, \checkmark] \)
   \rightarrow 8. \( Faster(Pat, Steve) [3a, \checkmark], [3b, \checkmark] \)
   \rightarrow 9. \( Faster(Bob, Steve) [3a, \checkmark], [3b, \checkmark] \)
**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. $Pig(y) \land Slug(z) \Rightarrow Faster(y, z)$
2. $Slimy(z) \land Creeps(z) \Rightarrow Slug(z)$
3. $Pig(Pat)$
4. $Slimy(Steve)$
5. $Creeps(Steve)$

1. $Faster(Pat, Steve)$
   - $\{y/\text{Pat}, z/\text{Steve}\}$
   - 3. $Pig(Pat)$
   - 4. $Slug(Steve)$
   - 2. $\{z/\text{Steve}\}$
   - 4. $\{\}$
   - 5. $\{\}$

---

1. $Faster(Pat, Steve)$
   - $\{y/\text{Pat}, z/\text{Steve}\}$
   - 3. $Pig(Pat)$
   - 4. $Slug(Steve)$
   - 2. $\{z/\text{Steve}\}$
   - 4. $\{\}$
   - 5. $\{\}$