CHAPTER 9, SECTIONS I–A

Inference in First-Order Logic
Outline

- Forward and backward chaining
- Generalized Modus Ponens
- Unification
- Proofs
\( \Gamma \text{ must be a ground term (i.e., no variables)} \)

\[
\frac{(\forall x \phi(x)) \land \psi}{(\forall x \phi(x)) \land \psi} \quad \{\Gamma/x\} \\
\frac{x \in \Gamma}{x \in \Gamma}
\]

E 8 Universal Elimination (UE)

\[
\frac{(\phi \lor \psi) \land \phi \lor \psi}{\phi} \quad \frac{\phi \lor \psi \lor \phi}{\phi} \\
\frac{\phi \lor \psi \lor \phi}{\phi}
\]

E 8 And-Introduction (AI)

\[
\frac{\phi \land \psi \land \phi \land \psi}{\phi \land \psi} \\
\frac{\phi \land \psi \land \phi \land \psi}{\phi \land \psi}
\]

E 8 Modus Ponens (MP)

Proof process is a search, operators are inference rules.

Sound inference: find a such that \( KB \models a \).
Example Proof

\[ \text{Bob is a buffalo} \]

\[ \text{Buffaloes outrun pigs} \]

\[ \text{Pat is a pig} \]

\[ \text{Buffaloes outrun pigs} \]

\[ \forall x \forall y \ (\text{father}(x,y) \implies x \neq y) \]

1. \( \text{Buffalo}(Bob) \)
2. \( \text{Pig}(Pat) \)
3. \( \exists x \forall y \ (\text{Buffalo}(x) \land \text{Pig}(y)) \)
\[(\text{fast}(\text{Bob}, \text{Pat}) \iff \text{pig}(\text{Pat}) \land \text{buffalo}(\text{Bob})) \Rightarrow \{x/\text{Bob}, y/\text{Pat}\}\]

\[\text{UE 3', } x/\text{Bob}, y/\text{Pat}\]
MP 6 & 7

6. Faster (Bob, Pat)
a single, more powerful inference rule.

Idea: Find a substitution that makes the rule

Problem: Bunching factors huge, esp. for UE

AI, UE. MP is a common inference pattern

Goal test checks state to see if it contains query sentence

States are sets of sentences

Operators are inference rules

---

Search with primitive inference rules
A substitution $\sigma$ unifies atomic sentences $p$ and $q$ if $p\sigma = q\sigma$. 
Likes(John, Mother(John))
Likes(John, O)
Likes(John, Jane)
Knows(John, x) \iff Likes(John, x)

Then we conclude...

E.g., if we know 9 and like outer to conclusion.

\textbf{Idea:} Unify rule premises with known facts, apply unifier to conclusion.

\[
\begin{array}{c|c|c}
\{ \text{Likes(John, x)} \} & \{ \text{Knows(John, x)} \} & \{ \text{Likes(John, x)} \} \\
\{ \text{Knows(John, x)} \} & \{ \text{Likes(John, x)} \} & \{ \text{Knows(John, x)} \} \\
\{ \text{Likes(John, x)} \} & \{ \text{Knows(John, x)} \} & \{ \text{Likes(John, x)} \}
\end{array}
\]
All variables assumed universally quantified

(atomic sentence) (conjunction of atomic sentences) \( \iff \)

either a single atomic sentence or

GMP used with KB of definite clauses (exactly one positive literal):

\[
\begin{align*}
\text{Faster}(\text{Bob}, \text{Steve}) & = b_0 \\
\{x/\text{Bob}, y/\text{Pat}, z/\text{Steve} \} & = 0 \\
\text{Faster}(x', y) & \iff (\text{Faster}(x', y) \lor \text{Faster}(y', z)) \\
\text{Faster}(\text{Pat}, \text{Steve}) & = b \leftrightarrow \exists \overline{d} \lor \exists \overline{d} \\
\text{Faster}(\text{Bob}, \text{Pat}) & = \forall \overline{d} \\
E.G.: \text{Faster}(\text{Bob}, \text{Steve}) & = b_0
\end{align*}
\]

where \( b_0 = 0 \) for all \( \overline{d} \)

\[
(b \iff \overline{u}_d \lor \cdots \lor \overline{u}_d \lor \overline{1}_d) \iff \overline{u}_d \lor \cdots \lor \overline{1}_d
\]

**Generalized Modus Ponens (GMP)**
3. From 1 and 2, φ follows by simple MP

\[ \phi, u^d \lor \cdots \lor \phi, l^d \models u^d \lor \cdots \lor l^d \]

2. \[ (\phi b \iff \phi u^d \lor \cdots \lor \phi l^d) = \phi (b \iff u^d \lor \cdots \lor l^d) \equiv (b \iff u^d \lor \cdots \lor l^d) \]

I. Lemma: For any definite clause \( d \), we have \( d \models d \) by UE

provided that \( \phi i^d = \phi i^d \)

\[ \phi b \models (b \iff u^d \lor \cdots \lor l^d) \]

Need to show that
Forward chaining is data-driven and continues from percepts. e.g., inferring properties from categories.

When a new fact is added to the KB, if the other premises are known, then add the conclusion to the KB and continue chaining. For each rule such that unifies with a premise, then add the conclusion to the KB.
Number in [] = unification literal; \( \land \) indicates rule firing

Add facts 1, 2, 3, 4, 5, \( \ell \) in turn.

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

Two versions: find any solution, find all solutions.

More complications help to avoid infinite loops.

(Some added complications in keeping track of the unifiers.

Attempt to prove each premise of the rule by backward chaining.

$h$ matches $\overline{h}$ for each rule whose consequent matches $\overline{h}$ if a matching fact is known, return the unifier $\overline{h}$ when a query $h$ is asked.

Backward chaining
Backward chaining example