Inference in first-order logic

Chapter 9, Sections 1–4

Outline

◇ Proofs

◇ Unification

◇ Generalized Modus Ponens

◇ Forward and backward chaining

Proofs

Sound inference: find α such that $KB \models \alpha$.
Proof process is a <u>search</u>, operators are inference rules.

E.g., Modus Ponens (MP)

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

E.g., And-Introduction (AI)

$$\frac{\alpha \quad \beta}{\alpha \wedge \beta} \qquad \frac{OK(Joe) \quad CSMajor(Joe)}{OK(Joe) \wedge CSMajor(Joe)}$$

E.g., Universal Elimination (UE)

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

τ 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	

AI 1 & 2

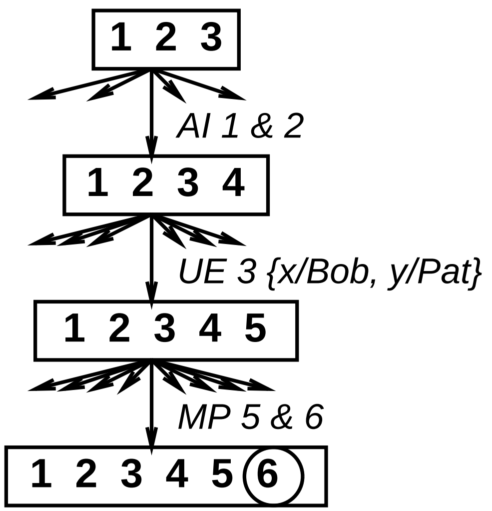
4. $Buffalo(Bob) \land Pig(Pat)$

UE 3, $\{x/Bob, y/Pat\}$	5. $Buffalo(Bob) \land Pig(Pat) \Rightarrow Faster(Bob, Pat)$

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

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
$\Rightarrow$ a single, more powerful inference rule

Unification

A substitution σ unifies atomic sentences p and q if $\underline{\underline{p\sigma = q\sigma}}$

p	q	σ
$Knows(John, x)$	$Knows(John, Jane)$	
$Knows(John, x)$	$Knows(y, OJ)$	
$Knows(John, x)$	$Knows(y, Mother(y))$	

.

		$\{x/Jane\}$
		$\{x/John, y/OJ\}$
		$\{y/John, x/Mother(John)\}$

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

E.g., if we know q and $Knows(John, x) \Rightarrow Likes(John, x)$

then we conclude $Likes(John, Jane)$

$Likes(John, OJ)$

$Likes(John, Mother(John))$

Generalized Modus Ponens (GMP)

$$\frac{p_1{}',\ \ p_2{}',\ \ldots,\ p_n{}',\ \ (p_1 \wedge p_2 \wedge \ldots \wedge p_n \Rightarrow q)}{q\sigma}$$

where $p_i{}'\sigma = p_i\sigma$ for all i

$$\begin{aligned}
\text{E.g. } p_1{}' &= \text{Faster(Bob,Pat)} \\
p_2{}' &= \text{Faster(Pat,Steve)} \\
p_1 \wedge p_2 \ \Rightarrow\ q \ &= \ Faster(x,y) \wedge Faster(y,z) \ \Rightarrow\ Faster(x,z) \\
\sigma &= \{x/Bob, y/Pat, z/Steve\} \\
q\sigma &= Faster(Bob, Steve)
\end{aligned}$$

GMP used with KB of <u>definite clauses</u> (*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

$$p_1', \ \ldots, \ p_n', \ \ (p_1 \wedge \ldots \wedge p_n \Rightarrow q) \models q\sigma$$

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. $(p_1 \wedge \ldots \wedge p_n \Rightarrow q) \models (p_1 \wedge \ldots \wedge p_n \Rightarrow q)\sigma = (p_1\sigma \wedge \ldots \wedge p_n\sigma \Rightarrow q\sigma)$
2. $p_1', \ \ldots, \ p_n' \models p_1' \wedge \ldots \wedge p_n' \models p_1'\sigma \wedge \ldots \wedge p_n'\sigma$
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; $\surd$ indicates rule firing

<u>1.</u> $Buffalo(x) \land Pig(y) \Rightarrow Faster(x, y)$
<u>2.</u> $Pig(y) \land Slug(z) \Rightarrow Faster(y, z)$
<u>3.</u> $Faster(x, y) \land Faster(y, z) \Rightarrow Faster(x, z)$
<u>4.</u> $Buffalo(Bob)$ [1a, $\times$]
<u>5.</u> $Pig(Pat)$ [1b, $\surd$] $\rightarrow$ <u>6.</u> $Faster(Bob, Pat)$ [3a, $\times$], [3b, $\times$]
[2a, $\times$]
<u>7.</u> $Slug(Steve)$ [2b, $\surd$]
$\rightarrow$ <u>8.</u> $Faster(Pat, Steve)$ [3a, $\times$], [3b, $\surd$]
$\rightarrow$ <u>9.</u> $Faster(Bob, Steve)$ [3a, $\times$], [3b, $\times$]

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 <u>any</u> solution, find <u>all</u> solutions

Backward chaining is the basis for <u>logic programming</u>, e.g., Prolog

Backward chaining example

$\underline{1.}\ Pig(y) \wedge Slug(z) \ \Rightarrow\ Faster(y, z)$
$\underline{2.}\ Slimy(z) \wedge Creeps(z) \ \Rightarrow\ Slug(z)$
$\underline{3.}\ Pig(Pat)$ $\underline{4.}\ Slimy(Steve)$ $\underline{5.}\ Creeps(Steve)$