

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 search, operators are inference rules.

E.g., Modus Ponens (MP)

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

E.g., And-Introduction (AI)

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

E.g., Universal Elimination (UE)

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

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. $\forall x, y \text{ } Buffalo(x) \wedge Pig(y) \Rightarrow Faster(x, y)$
Bob outruns Pat	

Al 1 & 2	4. $Buffalo(Bob) \wedge Pig(Pat)$

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



MP 6 & 7

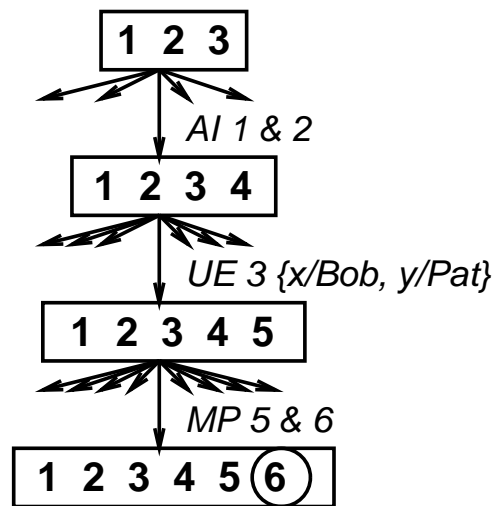
6. *Faster(Bob, Pat)*

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

⇒ a single, more powerful inference rule

Unification

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

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)

$$\frac{p_1', p_2', \dots, p_n', (p_1 \wedge p_2 \wedge \dots \wedge p_n \Rightarrow q)}{q\sigma} \quad \text{where } p_i'\sigma = p_i\sigma \text{ for all } i$$

E.g. $p_1' = \text{Faster}(\text{Bob}, \text{Pat})$

$p_2' = \text{Faster}(\text{Pat}, \text{Steve})$

$p_1 \wedge p_2 \Rightarrow q = \text{Faster}(x, y) \wedge \text{Faster}(y, z) \Rightarrow \text{Faster}(x, z)$

$\sigma = \{x/\text{Bob}, y/\text{Pat}, z/\text{Steve}\}$

$q\sigma = \text{Faster}(\text{Bob}, \text{Steve})$

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

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

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. $(p_1 \wedge \dots \wedge p_n \Rightarrow q) \models (p_1 \wedge \dots \wedge p_n \Rightarrow q)\sigma = (p_1\sigma \wedge \dots \wedge p_n\sigma \Rightarrow q\sigma)$
2. $p_1', \dots, p_n' \models p_1' \wedge \dots \wedge p_n' \models p_1'\sigma \wedge \dots \wedge p_n'\sigma$
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 \square = unification literal; \checkmark indicates rule firing

1. $Buffalo(x) \wedge Pig(y) \Rightarrow Faster(x, y)$

2. $Pig(y) \wedge Slug(z) \Rightarrow Faster(y, z)$

3. $Faster(x, y) \wedge Faster(y, z) \Rightarrow Faster(x, z)$

4. $Buffalo(Bob)$ $\square_{1a, \times}$

5. $Pig(Pat)$ $\square_{1b, \checkmark}$ \rightarrow 6. $Faster(Bob, Pat)$ $\square_{3a, \times}$, $\square_{3b, \times}$
 $\square_{2a, \times}$

7. $Slug(Steve)$ $\square_{2b, \checkmark}$

\rightarrow 8. $Faster(Pat, Steve)$ $\square_{3a, \times}$, $\square_{3b, \checkmark}$

\rightarrow 9. $Faster(Bob, Steve)$ $\square_{3a, \times}$, $\square_{3b, \times}$

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 any solution, find all solutions

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

Backward chaining example

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

