Inference in First-Order Logic

- Proofs

- Unification
- Generalized modus ponens
- Forward and backward chaining

- Completeness

- Resolution

- Logic programming

Inference in First-Order Logic

- Proofs – extend propositional logic inference to deal with quantifiers
- Unification
- Generalized modus ponens
- Forward and backward chaining – inference rules and reasoning program
- Completeness – Gödel's theorem: for FOL, any sentence entailed by another set of sentences can be proved from that set
- Resolution – inference procedure that is complete for any set of sentences
- Logic programming

Remember: propositional logic

◇ **Modus Ponens** or **Implication-Elimination**: (From an implication and the premise of the implication, you can infer the conclusion.)

$$\frac{\alpha \Rightarrow \beta, \qquad \alpha}{\beta}$$

◇ **And-Elimination**: (From a conjunction, you can infer any of the conjuncts.)

$$\frac{\alpha_1 \land \alpha_2 \land \ldots \land \alpha_n}{\alpha_i}$$

◇ **And-Introduction**: (From a list of sentences, you can infer their conjunction.)

$$\frac{\alpha_1, \ \alpha_2, \quad \ldots, \quad \alpha_n}{\alpha_1 \land \alpha_2 \land \ldots \land \alpha_n}$$

◇ **Or-Introduction**: (From a sentence, you can infer its disjunction with anything else at all.)

$$\frac{\alpha_i}{\alpha_1 \lor \alpha_2 \lor \ldots \lor \alpha_n}$$

◇ **Double-Negation Elimination**: (From a doubly negated sentence, you can infer a positive sentence.)

$$\frac{\neg\neg\alpha}{\alpha}$$

◇ **Unit Resolution**: (From a disjunction, if one of the disjuncts is false, then you can infer the other one is true.)

$$\frac{\alpha \lor \beta, \qquad \neg\beta}{\alpha}$$

◇ **Resolution**: (This is the most difficult. Because β cannot be both true and false, one of the other disjuncts must be true in one of the premises. Or equivalently, implication is transitive.)

$$\frac{\alpha \lor \beta, \qquad \neg\beta \lor \gamma}{\alpha \lor \gamma} \quad \text{or equivalently} \quad \frac{\neg\alpha \Rightarrow \beta, \qquad \beta \Rightarrow \gamma}{\neg\alpha \Rightarrow \gamma}$$

Proofs

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

E.g., Modus Ponens (MP)

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

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)

Proofs

The three new inference rules for FOL (compared to propositional logic) are:

- **Universal Elimination (UE):**
  for any sentence α, variable x and ground term τ,

  $$\frac{\forall x \quad \alpha}{\alpha\{x/\tau\}}$$

- **Existential Elimination (EE):**
  for any sentence α, variable x and constant symbol k not in KB,

  $$\frac{\exists x \quad \alpha}{\alpha\{x/k\}}$$

- **Existential Introduction (EI):**
  for any sentence α, variable x not in α and ground term g in α,

  $$\frac{\alpha}{\exists x \quad \alpha\{g/x\}}$$

Proofs

The three new inference rules for FOL (compared to propositional logic) are:

- **Universal Elimination (UE):**
  for any sentence α, variable x and ground term τ,

  $$\frac{\forall x \quad \alpha}{\alpha\{x/\tau\}}$$

  e.g., from $\forall$x Likes(x, Candy) and {x/Joe} we can infer Likes(Joe, Candy)

- **Existential Elimination (EE):**
  for any sentence α, variable x and constant symbol k not in KB,

  $$\frac{\exists x \quad \alpha}{\alpha\{x/k\}}$$

  e.g., from $\exists$x Kill(x, Victim) we can infer Kill(Murderer, Victim), if Murderer new symbol

- **Existential Introduction (EI):**
  for any sentence α, variable x not in α and ground term g in α,

  $$\frac{\alpha}{\exists x \quad \alpha\{g/x\}}$$

  e.g., from Likes(Joe, Candy) we can infer $\exists$x Likes(x, Candy)

Example Proof

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

Example Proof

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

Example Proof

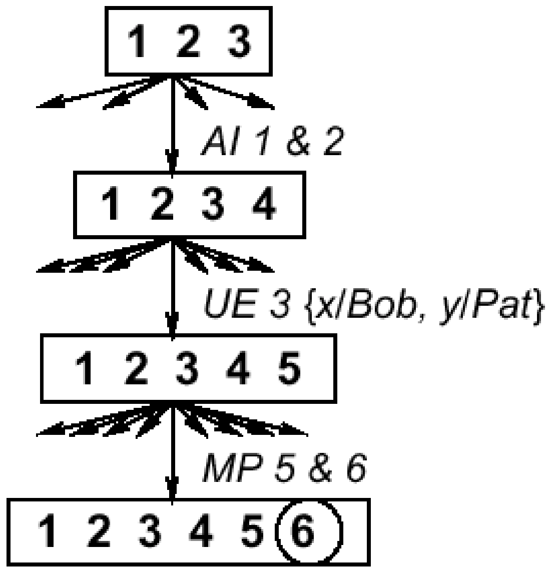
UE 3, $\{x/Bob, y/Pat\}$	5. $Buffalo(Bob) \wedge Pig(Pat) \Rightarrow Faster(Bob, Pat)$

Example Proof

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

Search with primitive example 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
$Knows(John, x)$	$Knows(John, Jane)$	
$Knows(John, x)$	$Knows(y, OJ)$	
$Knows(John, x)$	$Knows(y, Mother(y))$	

Unification

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

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 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

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

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

Properties of GMP

- Why is GMP and efficient inference rule?

  - It takes bigger steps, combining several small inferences into one

  - It takes sensible steps: uses eliminations that are guaranteed to help (rather than random UEs)

  - It uses a precompilation step which converts the KB to canonical form (Horn sentences)

  Remember: sentence in Horn from is a conjunction of Horn clauses (clauses with at most one positive literal), e.g.,
  $(A \lor \neg B) \land (B \lor \neg C \lor \neg D)$, that is $(B \Rightarrow A) \land ((C \land D) \Rightarrow B)$

Horn form

- We convert sentences to Horn form as they are entered into the KB
- Using Existential Elimination and And Elimination

- e.g., $\exists x$ Owns(Nono, x) $\wedge$ Missile(x) becomes

  Owns(Nono, M)
  Missile(M)

(with M a new symbol that was not already in the KB)

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

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, $\times$]
5. $Pig(Pat)$ [1b, $\surd$] $\rightarrow$ 6. $Faster(Bob, Pat)$ [3a, $\times$], [3b, $\times$]
   [2a, $\times$]
7. $Slug(Steve)$ [2b, $\surd$]
   $\rightarrow$ 8. $Faster(Pat, Steve)$ [3a, $\times$], [3b, $\surd$]
   $\rightarrow$ 9. $Faster(Bob, Steve)$ [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)

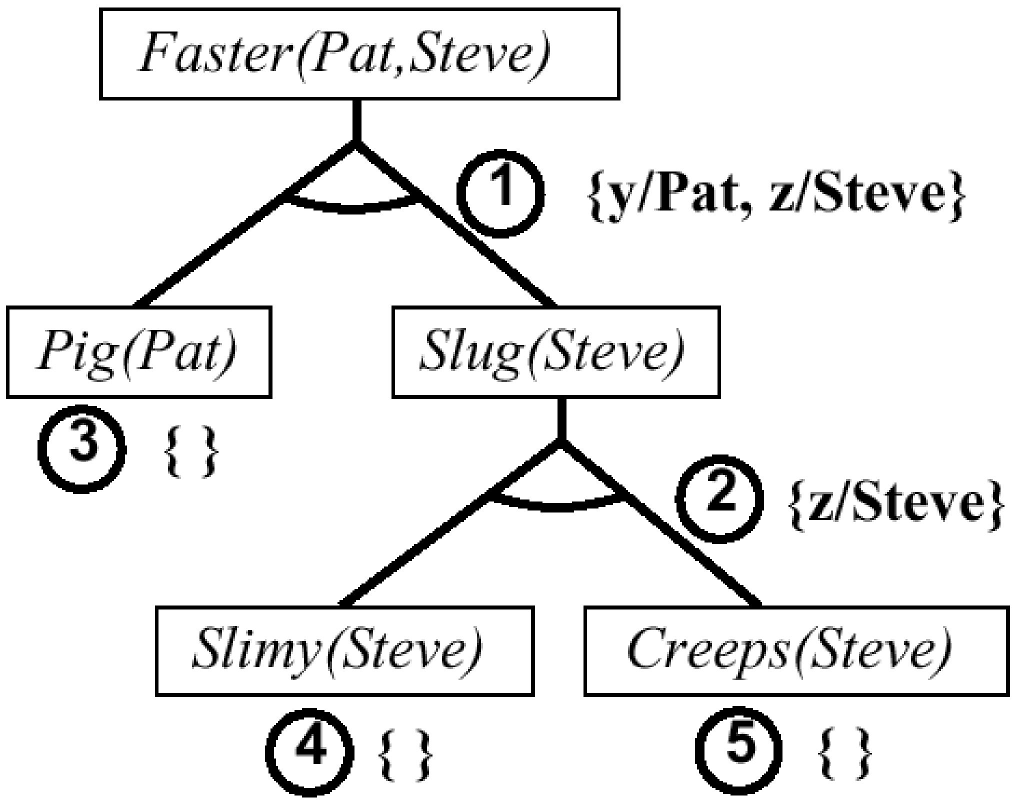
(More complications help to avoid infinite loops)

Two versions: find 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. $Creeps(Steve)$



Completeness in FOL

Procedure i is complete if and only if

$$KB \vdash_i \alpha \quad \text{whenever} \quad KB \models \alpha$$

Forward and backward chaining are complete for Horn KBs but incomplete for general first-order logic

E.g., from

$$\begin{aligned}
PhD(x) &\Rightarrow HighlyQualified(x) \\
\neg PhD(x) &\Rightarrow EarlyEarnings(x) \\
HighlyQualified(x) &\Rightarrow Rich(x) \\
EarlyEarnings(x) &\Rightarrow Rich(x)
\end{aligned}$$

should be able to infer $Rich(Me)$, but FC/BC won't do it

Does a complete algorithm exist?

Historical note

450B.C.	Stoics	propositional logic, inference (maybe)
322B.C.	Aristotle	"syllogisms" (inference rules), quantifiers
1565	Cardano	probability theory (propositional logic + uncertainty)
1847	Boole	propositional logic (again)
1879	Frege	first-order logic
1922	Wittgenstein	proof by truth tables
1930	Gödel	$\exists$ complete algorithm for FOL
1930	Herbrand	complete algorithm for FOL (reduce to propositional)
1931	Gödel	$\neg\exists$ complete algorithm for arithmetic
1960	Davis/Putnam	"practical" algorithm for propositional logic
1965	Robinson	"practical" algorithm for FOL—resolution

Resolution

Entailment in first-order logic is only semidecidable:
can find a proof of α if $KB \models \alpha$
cannot always prove that $KB \not\models \alpha$
Cf. Halting Problem: proof procedure may be about to terminate with success or failure, or may go on for ever

Resolution is a refutation procedure:
to prove $KB \models \alpha$, show that $KB \wedge \neg\alpha$ is unsatisfiable

Resolution uses KB, $\neg\alpha$ in CNF (conjunction of clauses)

Resolution inference rule combines two clauses to make a new one:

Inference continues until an empty clause is derived (contradiction)

Resolution inference rule

Basic propositional version:

$$\frac{\alpha \vee \beta, \quad \neg\beta \vee \gamma}{\alpha \vee \gamma} \quad \text{or equivalently} \quad \frac{\neg\alpha \Rightarrow \beta, \quad \beta \Rightarrow \gamma}{\neg\alpha \Rightarrow \gamma}$$

Full first-order version:

$$\frac{\begin{array}{c} p_1 \vee \ldots \; p_j \; \ldots \vee p_m, \\ q_1 \vee \ldots \; q_k \; \ldots \vee q_n \end{array}}{(p_1 \vee \ldots \; p_{j-1} \vee p_{j+1} \; \ldots p_m \vee q_1 \ldots \; q_{k-1} \vee q_{k+1} \; \ldots \vee q_n)\sigma}$$

where $p_j\sigma = \neg q_k\sigma$

For example,

$$\frac{\begin{array}{l} \neg Rich(x) \vee Unhappy(x) \\ Rich(Me) \end{array}}{Unhappy(Me)}$$

with $\sigma = \{x/Me\}$

Remember: normal forms

Other approaches to inference use syntactic operations on sentences, often expressed in standardized forms

Conjunctive Normal Form (CNF—universal)

$$\textit{conjunction} \text{ of } \underbrace{\textit{disjunctions} \text{ of } \textit{literals}}_{\textit{clauses}}$$

E.g., $(A \vee \neg B) \wedge (B \vee \neg C \vee \neg D)$

"product of sums of simple variables or negated simple variables"

Disjunctive Normal Form (DNF—universal)

$$\textit{disjunction} \text{ of } \underbrace{\textit{conjunctions} \text{ of } \textit{literals}}_{\textit{terms}}$$

E.g., $(A \wedge B) \vee (A \wedge \neg C) \vee (A \wedge \neg D) \vee (\neg B \wedge \neg C) \vee (\neg B \wedge \neg D)$

"sum of products of simple variables or negated simple variables"

Horn Form (restricted)

conjunction of *Horn clauses* (clauses with ≤ 1 positive literal)

E.g., $(A \vee \neg B) \wedge (B \vee \neg C \vee \neg D)$

Often written as set of implications:

$B \Rightarrow A$ and $(C \wedge D) \Rightarrow B$

Conjunctive normal form

Literal = (possibly negated) atomic sentence, e.g., $\neg Rich(Me)$

Clause = disjunction of literals, e.g., $\neg Rich(Me) \vee Unhappy(Me)$

The KB is a conjunction of clauses

Any FOL KB can be converted to CNF as follows:

1. Replace $P \Rightarrow Q$ by $\neg P \vee Q$
2. Move $\neg$ inwards, e.g., $\neg \forall x\, P$ becomes $\exists x\, \neg P$
3. Standardize variables apart, e.g., $\forall x\, P \vee \exists x\, Q$ becomes $\forall x\, P \vee \exists y\, Q$
4. Move quantifiers left in order, e.g., $\forall x\, P \vee \exists x\, Q$ becomes $\forall x \exists y\, P \vee Q$
5. Eliminate $\exists$ by Skolemization (next slide)
6. Drop universal quantifiers
7. Distribute $\wedge$ over $\vee$, e.g., $(P \wedge Q) \vee R$ becomes $(P \vee Q) \wedge (P \vee R)$

Skolemization

$\exists x\, Rich(x)$ becomes $Rich(G1)$ where $G1$ is a new "Skolem constant"

$\exists\, k\ \ \frac{d}{dy}(k^y) = k^y$ becomes $\frac{d}{dy}(e^y) = e^y$

More tricky when $\exists$ is inside $\forall$

E.g., "Everyone has a heart"

$$\forall\, x\ \ Person(x)\ \Rightarrow\ \exists\, y\ \ Heart(y) \wedge Has(x, y)$$

Incorrect:

$$\forall\, x\ \ Person(x)\ \Rightarrow\ Heart(H1) \wedge Has(x, H1)$$

Correct:

$$\forall\, x\ \ Person(x)\ \Rightarrow\ Heart(H(x)) \wedge Has(x, H(x))$$

where H is a new symbol ("Skolem function")

Skolem function arguments: all enclosing universally quantified variables

Resolution proof

To prove α:

- negate it
- convert to CNF
- add to CNF KB
- infer contradiction

E.g., to prove $Rich(me)$, add $\neg Rich(me)$ to the CNF KB

$$\neg PhD(x) \vee HighlyQualified(x)$$
$$PhD(x) \vee EarlyEarnings(x)$$
$$\neg HighlyQualified(x) \vee Rich(x)$$
$$\neg EarlyEarnings(x) \vee Rich(x)$$

Resolution proof