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.)
\[
\frac{\alpha \Rightarrow \beta, \quad \alpha}{\beta}
\]

◊ **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.)
\[
\frac{\alpha_1, \quad \alpha_2, \quad \ldots, \quad \alpha_n}{\alpha_1 \land \alpha_2 \land \ldots \land \alpha_n}
\]

◊ **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.)
\[
\frac{\alpha \lor \beta, \quad \neg \beta}{\alpha}
\]

◊ **Resolution**: (This is the most difficult. Because $\beta$ cannot be both true and false, one of the other disjuncts must be true in one of the premises. Or equivalently, implication is transitive.)
\[
\frac{\alpha \lor \beta, \quad \neg \beta \lor \gamma}{\alpha \lor \gamma} \quad \text{or equivalently} \quad \frac{\neg \alpha \Rightarrow \beta, \quad \beta \Rightarrow \gamma}{\neg \alpha \Rightarrow \gamma}
\]
Proofs

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

E.g., Modus Ponens (MP)

\[
\begin{align*}
\alpha, \quad &\alpha \Rightarrow \beta \\
\underline{At(Joe, UCB)} &\quad At(Joe, UCB) \Rightarrow OK(Joe) \\
\beta &\quad OK(Joe)
\end{align*}
\]

E.g., And-Introduction (AI)

\[
\begin{align*}
\alpha &\quad \beta \\
\underline{OK(Joe) \quad CSMajor(Joe)} &\quad OK(Joe) \land CSMajor(Joe)
\end{align*}
\]

E.g., Universal Elimination (UE)

\[
\begin{align*}
\forall x \quad &\alpha \\
\underline{\forall x \quad At(x, UCB) \Rightarrow OK(x)} &\quad At(Pat, UCB) \Rightarrow OK(Pat)
\end{align*}
\]

$\tau$ must be a ground term (i.e., no variables)
The three new inference rules for FOL (compared to propositional logic) are:

- **Universal Elimination (UE):**
  for any sentence $\alpha$, variable $x$ and ground term $\tau$,
  \[
  \forall x \quad \alpha \\
  \underline{\alpha \{x/\tau\}}
  \]

- **Existential Elimination (EE):**
  for any sentence $\alpha$, variable $x$ and constant symbol $k$ not in KB,
  \[
  \exists x \quad \alpha \\
  \underline{\alpha \{x/k\}}
  \]

- **Existential Introduction (EI):**
  for any sentence $\alpha$, variable $x$ not in $\alpha$ and ground term $g$ in $\alpha$,
  \[
  \alpha \\
  \underline{\exists x \quad \alpha \{g/x\}}
  \]
Proofs

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

• **Universal Elimination (UE):**
  for any sentence $\alpha$, variable $x$ and ground term $\tau$,
  \[
  \forall x \quad \alpha \quad \Rightarrow \quad \alpha\{x/\tau\}
  \]
  e.g., from $\forall x \text{ Likes}(x, \text{ Candy})$ and $\{x/\text{ Joe}\}$
  we can infer $\text{ Likes}(\text{ Joe, Candy})$

• **Existential Elimination (EE):**
  for any sentence $\alpha$, variable $x$ and constant symbol $k$ not in KB,
  \[
  \exists x \quad \alpha \quad \Rightarrow \quad \alpha\{x/k\}
  \]
  e.g., from $\exists x \text{ Kill}(x, \text{ Victim})$ we can infer
  $\text{ Kill}($Murderer, Victim$)$, if Murderer new symbol

• **Existential Introduction (EI):**
  for any sentence $\alpha$, variable $x$ not in $\alpha$ and ground term $g$ in $\alpha$,
  \[
  \alpha \quad \Rightarrow \quad \exists x \quad \alpha\{g/x\}
  \]
  e.g., from $\text{ Likes}(\text{ Joe, Candy})$ we can infer
  $\exists x \text{ Likes}(x, \text{ Candy})$
Example Proof

Bob is a buffalo
Pat is a pig
Buffaloes outrun pigs

Bob outruns Pat

1. \( \text{Buffalo}(Bob) \)
2. \( \text{Pig}(Pat) \)
3. \( \forall x, y \ \text{Buffalo}(x) \land \text{Pig}(y) \Rightarrow \text{Faster}(x, y) \)
Example Proof

1. \textit{Buffalo}(Bob) \land \textit{Pig}(Pat)

AI 1 & 2

4. \textit{Buffalo}(Bob) \land \textit{Pig}(Pat)
Example Proof

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

1 2 3
   \[\text{AI 1 & 2}\]

1 2 3 4
   \[\text{UE 3 \{x/\text{Bob}, y/\text{Pat}\}}\]

1 2 3 4 5
   \[\text{MP 5 & 6}\]

1 2 3 4 5 6

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 $\sigma$ 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>
Unification

\[
\begin{align*}
\{x/Jane\} \\
\{x/John, y/OJ\} \\
\{y/John, x/Mother(John)\}
\end{align*}
\]

**Idea:** Unify rule premises with known facts, apply unifier to conclusion
E.g., if we know \(q\) and then we conclude

\[
\begin{align*}
\text{Knows}(John, x) & \Rightarrow \text{Likes}(John, x) \\
\text{Likes}(John, Jane) \\
\text{Likes}(John, OJ) \\
\text{Likes}(John, \text{Mother}(John))
\end{align*}
\]
Generalized Modus Ponens (GMP)

\[
p_1', \ p_2', \ \ldots, \ p_n', \ (p_1 \land p_2 \land \ldots \land p_n \implies q) \implies q^\sigma
\]

where \( p_i^\sigma = p_i^\sigma \) for all \( i \)

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

GMP used with KB of **definite clauses** (**exactly** one positive literal):  
either a single atomic sentence or  
(conjunction of atomic sentences) \implies (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 \( p_i'\sigma = p_i\sigma \) 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
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 form is a conjunction of Horn clauses (clauses with at most one positive literal), e.g.,

\[(\neg A \lor B) \land (B \lor \neg C \lor \neg D)\], 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., $\exists x \text{ Owns(Nono, x)} \land \text{Missile(x)}$ 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; √ indicates rule firing

1. Buffalo(x) ∧ Pig(y) ⇒ Faster(x, y)
2. Pig(y) ∧ Slug(z) ⇒ Faster(y, z)
3. Faster(x, y) ∧ Faster(y, z) ⇒ Faster(x, z)
4. Buffalo(Bob) [1a, ×]
5. Pig(Pat) [1b, √] → 6. Faster(Bob, Pat) [3a, ×], [3b, ×]
   [2a, ×]
7. Slug(Steve) [2b, √]
   → 8. Faster(Pat, Steve) [3a, ×], [3b, √]
   → 9. Faster(Bob, Steve) [3a, ×], [3b, ×]
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. \( Creeps(Steve) \)

Diagram:

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

\[
\begin{align*}
\text{PhD}(x) & \Rightarrow \text{HighlyQualified}(x) \\
\neg\text{PhD}(x) & \Rightarrow \text{EarlyEarnings}(x) \\
\text{HighlyQualified}(x) & \Rightarrow \text{Rich}(x) \\
\text{EarlyEarnings}(x) & \Rightarrow \text{Rich}(x)
\end{align*}
\]

should be able to infer \( \text{Rich}(Me) \), but FC/BC won’t do it.

Does a complete algorithm exist?
Historical note

450 B.C. Stoics   propositional logic, inference (maybe)
322 B.C. Aristotle  “syllogisms” (inference rules), quantifiers
1575 Cardano   probability theory (propositional logic + uncertainty)
1847 Boole   propositional logic (again)
1879 Frege   first-order logic
1922 Wittgenstein   proof by truth tables
1930 Gödel   ∃ complete algorithm for FOL
1930 Herbrand   complete algorithm for FOL (reduce to propositional)
1931 Gödel   →∃ 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 $\alpha$ 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 $KB \land \neg \alpha$ is unsatisfiable

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

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

\[
\begin{array}{c}
C_1 \\
\bigtriangledown \\
C_2 \\
\hline
C
\end{array}
\]

Inference continues until an empty clause is derived (contradiction)
Resolution inference rule

Basic propositional version:

\[
\frac{\alpha \lor \beta, \neg \beta \lor \gamma}{\alpha \lor \gamma}
\]

or equivalently

\[
\frac{-\alpha \Rightarrow \beta, \beta \Rightarrow \gamma}{-\alpha \Rightarrow \gamma}
\]

Full first-order version:

\[
\frac{p_1 \lor \ldots \lor p_j \lor \ldots \lor p_m, \quad \neg p_j \lor \ldots \lor \neg q_k \lor \ldots \lor \neg q_n}{\neg (p_1 \lor \ldots \lor p_{j-1} \lor p_{j+1} \lor \ldots \lor p_m \lor q_1 \lor \ldots \lor q_{k-1} \lor q_{k+1} \lor \ldots \lor q_n)\sigma}
\]

where \( p_j \sigma = \neg q_k \sigma \)

For example,

\[
\neg Rich(x) \lor Unhappy(x) \quad Rich(Me) \quad \frac{}{Unhappy(Me)}\]

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)**

\[
\text{conjunction of disjunctions of literals} \quad \text{clauses}
\]

E.g., \((A \lor \neg B) \land (B \lor \neg C \lor \neg D)\)

“product of sums of simple variables or negated simple variables”

**Disjunctive Normal Form (DNF—universal)**

\[
\text{disjunction of conjunctions of literals} \quad \text{terms}
\]

E.g., \((A \land B) \lor (A \land \neg C) \lor (A \land \neg D) \lor (\neg B \land \neg C) \lor (\neg B \land \neg D)\)

“sum of products of simple variables or negated simple variables”

**Horn Form (restricted)**

\[
\text{conjunction of Horn clauses} \quad \text{(clauses with } \leq 1 \text{ positive literal)}
\]

E.g., \((A \lor \neg B) \land (B \lor \neg C \lor \neg D)\)

Often written as set of implications:
\[B \Rightarrow A \text{ and } (C \land D) \Rightarrow B\]
Conjunctive normal form

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

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

The KB is a conjunction of clauses

Any FOL KB can be converted to CNF as follows:
1. Replace \( P \Rightarrow Q \) by \( \neg P \lor Q \)
2. Move \( \neg \) inwards, e.g., \( \neg \forall x P \) becomes \( \exists x \neg P \)
3. Standardize variables apart, e.g., \( \forall x P \lor \exists x Q \) becomes \( \forall x P \lor \exists y Q \)
4. Move quantifiers left in order, e.g., \( \forall x P \lor \exists x Q \) becomes \( \forall x \exists y P \lor Q \)
5. Eliminate \( \exists \) by Skolemization (next slide)
6. Drop universal quantifiers
7. Distribute \( \land \) over \( \lor \), e.g., \( (P \land Q) \lor R \) becomes \( (P \lor Q) \land (P \lor R) \)
Skolemization

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

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

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

E.g., "Everyone has a heart"

$\forall x \quad \text{Person}(x) \Rightarrow \exists y \quad \text{Heart}(y) \land \text{Has}(x, y)$

**Incorrect:**

$\forall x \quad \text{Person}(x) \Rightarrow \text{Heart}(H1) \land \text{Has}(x, H1)$

**Correct:**

$\forall x \quad \text{Person}(x) \Rightarrow \text{Heart}(H(x)) \land \text{Has}(x, H(x))$

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

Skolem function arguments: all enclosing universally quantified variables

CS 561, Session 16-18
Resolution proof

To prove \( \alpha \):
- negate it
- convert to CNF
- add to CNF KB
- infer contradiction

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

\[
\neg \text{PhD}(x) \lor \text{HighlyQualified}(x) \\
\text{PhD}(x) \lor \text{EarlyEarnings}(x) \\
\neg \text{HighlyQualified}(x) \lor \text{Rich}(x) \\
\neg \text{EarlyEarnings}(x) \lor \text{Rich}(x)
\]
Resolution proof

\[ \neg PhD(x) \lor HQ(x) \]
\[ \neg HQ(x) \lor Rich(x) \]
\[ \{ \} \]
\[ \neg PhD(x) \lor Rich(x) \]
\[ PhD(x) \lor ES(x) \]
\[ \{ \} \]
\[ Rich(x) \lor ES(x) \]
\[ \neg ES(x) \lor Rich(x) \]
\[ \{ \} \]
\[ Rich(x) \]
\[ \neg Rich(Me) \]
\[ \{x/Me\} \]