Artificial Intelligence

Inference in First-Order Logic

Readings: Chapter 9 of Russell & Norvig.
## A brief history of reasoning

<table>
<thead>
<tr>
<th>Year</th>
<th>Person</th>
<th>Significance</th>
</tr>
</thead>
<tbody>
<tr>
<td>450 BC</td>
<td>Stoics</td>
<td>propositional logic, inference (maybe)</td>
</tr>
<tr>
<td>322 BC</td>
<td>Aristotle</td>
<td>“syllogisms” (inference rules), quantifiers</td>
</tr>
<tr>
<td>1565</td>
<td>Cardano</td>
<td>probability theory (propositional logic + uncertainty)</td>
</tr>
<tr>
<td>1847</td>
<td>Boole</td>
<td>propositional logic (again)</td>
</tr>
<tr>
<td>1879</td>
<td>Frege</td>
<td>first-order logic</td>
</tr>
<tr>
<td>1922</td>
<td>Wittgenstein</td>
<td>proof by truth tables</td>
</tr>
<tr>
<td>1930</td>
<td>Gödel</td>
<td>$\exists$ complete algorithm for FOL</td>
</tr>
<tr>
<td>1930</td>
<td>Herbrand</td>
<td>complete algorithm for FOL</td>
</tr>
<tr>
<td>1931</td>
<td>Gödel</td>
<td>$\neg\exists$ complete algorithm for arithmetic</td>
</tr>
<tr>
<td>1960</td>
<td>Davis/Putnam</td>
<td>“practical” algorithm for propositional logic</td>
</tr>
<tr>
<td>1965</td>
<td>Robinson</td>
<td>“practical” algorithm for FOL—resolution</td>
</tr>
</tbody>
</table>
Universal instantiation (UI)

Every instantiation of a universally quantified sentence is entailed by it:

$$\forall v \alpha \\
\frac{}{\text{SUBST}(\{v/g\}, \alpha)}$$

for any variable $v$ and ground term $g$

E.g., $\forall x \ King(x) \land Greedy(x) \implies Evil(x)$ yields

$$King(John) \land Greedy(John) \implies Evil(John)$$
$$King(Richard) \land Greedy(Richard) \implies Evil(Richard)$$
$$King(Father(John)) \land Greedy(Father(John)) \implies Evil(Father(John))$$

::
Existential instantiation (EI)

For any sentence \( \alpha \), variable \( v \), and constant symbol \( k \) that does not appear elsewhere in the knowledge base:

\[
\exists v \; \alpha \\
\text{Subst}(\{v/k\}, \alpha)
\]

E.g., \( \exists x \; \text{Crown}(x) \land \text{OnHead}(x, \text{John}) \) yields

\( \text{Crown}(C_1) \land \text{OnHead}(C_1, \text{John}) \)

provided \( C_1 \) is a new constant symbol, called a Skolem constant.
UI versus EI

From $\forall x \forall y \ (y + x = y)$ we obtain

$$y + a = y$$

From $\exists x \ \forall y \ (y + x = y)$ we obtain

$$y + e = y$$

provided $e$ is a new constant symbol.

- UI can be applied several times to add new sentences; the new KB is logically equivalent to the old.
- EI can be applied once to replace the existential sentence; the new KB is not equivalent to the old, but is satisfiable iff the old KB was satisfiable.
Reduction to Propositional Inference

Suppose the KB contains just the following:

\[ \forall x \ King(x) \land Greedy(x) \implies Evil(x) \]
\[ King(John) \]
\[ Greedy(John) \]
\[ Brother(Richard, John) \]

Instantiating the universal sentence in *all possible* ways, we have

\[ King(John) \land Greedy(John) \implies Evil(John) \]
\[ King(Richard) \land Greedy(Richard) \implies Evil(Richard) \]
\[ King(John) \]
\[ Greedy(John) \]
\[ Brother(Richard, John) \]

The new KB is propositionalized: proposition symbols are

\[ King(John), Greedy(John), Evil(John), King(Richard) \text{ etc.} \]
Reduction to Propositional Inference

Claim: a ground sentence is entailed by new KB iff entailed by original KB.

Claim: every FOL KB can be propositionalized so as to preserve entailment.

Idea: propositionalize KB and query, apply resolution, return result.

Problem: with function symbols, ground terms are infinitely many, e.g., $Father(Father(Father(John)))$.

Theorem: Herbrand (1930). If a sentence $\alpha$ is entailed by an FOL KB, it is entailed by a finite subset of the propositional KB.

Idea: For $n = 0$ to $\infty$, create a propositional KB by instantiating with depth-$n$ terms see if $\alpha$ is entailed by this KB.

Problem: works if $\alpha$ is entailed, loops if $\alpha$ is not entailed

Theorem (Turing (1936), Church (1936)) Entailment in FOL is semidecidable.
Problems with Propositionalization

- Propositionalization seems to generate lots of irrelevant sentences.
- E.g., from

\[ \forall x \ King(x) \land Greedy(x) \implies Evil(x) \]

\[ King(John) \]

\[ \forall y \ Greedy(y) \]

\[ Brother(Richard, John) \]

it seems obvious that \( Evil(John) \), but propositionalization produces lots of facts such as \( Greedy(Richard) \) that are irrelevant.

- In general, for one \( k \)-ary predicate and \( n \) constants, there are \( n^k \) instantiations!
Unification

We can get the inference immediately if we can find a substitution $\theta$ such that $King(x)$ and $Greedy(x)$ match $King(John)$ and $Greedy(y)$. That is, $\theta = \{x/John, y/John\}$ works.

$\textsc{Unify}(\alpha, \beta) = \theta$ if $\alpha\theta = \beta\theta$

<table>
<thead>
<tr>
<th>$p$</th>
<th>$q$</th>
<th>$\theta$</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>
<tr>
<td>$Knows(John, x)$</td>
<td>$Knows(x, OJ)$</td>
<td></td>
</tr>
</tbody>
</table>
Unification

- We can get the inference immediately if we can find a substitution $\theta$ such that $\text{King}(x)$ and $\text{Greedy}(x)$ match $\text{King}(\text{John})$ and $\text{Greedy}(y)$. That is, $\theta = \{x/\text{John}, y/\text{John}\}$ works.

- $\text{UNIFY}(\alpha, \beta) = \theta$ if $\alpha\theta = \beta\theta$

<table>
<thead>
<tr>
<th>$p$</th>
<th>$q$</th>
<th>$\theta$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$\text{Knows}(\text{John}, x)$</td>
<td>$\text{Knows}(\text{John}, \text{Jane})$</td>
<td>${x/\text{Jane}}$</td>
</tr>
<tr>
<td>$\text{Knows}(\text{John}, x)$</td>
<td>$\text{Knows}(y, \text{OJ})$</td>
<td></td>
</tr>
<tr>
<td>$\text{Knows}(\text{John}, x)$</td>
<td>$\text{Knows}(y, \text{Mother}(y))$</td>
<td></td>
</tr>
<tr>
<td>$\text{Knows}(\text{John}, x)$</td>
<td>$\text{Knows}(x, \text{OJ})$</td>
<td></td>
</tr>
</tbody>
</table>
Unification

We can get the inference immediately if we can find a substitution $\theta$ such that $\text{King}(x)$ and $\text{Greedy}(x)$ match $\text{King}(\text{John})$ and $\text{Greedy}(y)$. That is, $\theta = \{x/\text{John}, y/\text{John}\}$ works.

$\text{UNIFY}(\alpha, \beta) = \theta$ if $\alpha\theta = \beta\theta$

<table>
<thead>
<tr>
<th>$p$</th>
<th>$q$</th>
<th>$\theta$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$\text{Knows}(\text{John}, x)$</td>
<td>$\text{Knows}(\text{John}, \text{Jane})$</td>
<td>${x/\text{Jane}}$</td>
</tr>
<tr>
<td>$\text{Knows}(\text{John}, x)$</td>
<td>$\text{Knows}(y, \text{OJ})$</td>
<td>${x/\text{OJ}, y/\text{John}}$</td>
</tr>
<tr>
<td>$\text{Knows}(\text{John}, x)$</td>
<td>$\text{Knows}(y, \text{Mother}(y))$</td>
<td></td>
</tr>
<tr>
<td>$\text{Knows}(\text{John}, x)$</td>
<td>$\text{Knows}(x, \text{OJ})$</td>
<td></td>
</tr>
</tbody>
</table>
Unification

We can get the inference immediately if we can find a substitution \( \theta \) such that \( King(x) \) and \( Greedy(x) \) match \( King(John) \) and \( Greedy(y) \). That is, \( \theta = \{x/John, y/John\} \) works.

\( \text{UNIFY}(\alpha, \beta) = \theta \text{ if } \alpha\theta = \beta\theta \)

<table>
<thead>
<tr>
<th>( p )</th>
<th>( q )</th>
<th>( \theta )</th>
</tr>
</thead>
<tbody>
<tr>
<td>( Knows(John, x) )</td>
<td>( Knows(John, Jane) )</td>
<td>( {x/Jane} )</td>
</tr>
<tr>
<td>( Knows(John, x) )</td>
<td>( Knows(y, OJ) )</td>
<td>( {x/OJ, y/John} )</td>
</tr>
<tr>
<td>( Knows(John, x) )</td>
<td>( Knows(y, \text{Mother}(y)) )</td>
<td>( {y/John, x/\text{Mother}(y)} )</td>
</tr>
<tr>
<td>( Knows(John, x) )</td>
<td>( Knows(x, OJ) )</td>
<td></td>
</tr>
</tbody>
</table>
Unification

We can get the inference immediately if we can find a substitution $\theta$ such that $\text{King}(x)$ and $\text{Greedy}(x)$ match $\text{King}(\text{John})$ and $\text{Greedy}(y)$. That is, $\theta = \{x/\text{John}, y/\text{John}\}$ works.

$\text{Unify}(\alpha, \beta) = \theta$ if $\alpha\theta = \beta\theta$

<table>
<thead>
<tr>
<th>$p$</th>
<th>$q$</th>
<th>$\theta$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$\text{Knows}(\text{John}, x)$</td>
<td>$\text{Knows}(\text{John}, \text{Jane})$</td>
<td>${x/\text{Jane}}$</td>
</tr>
<tr>
<td>$\text{Knows}(\text{John}, x)$</td>
<td>$\text{Knows}(\text{y}, \text{OJ})$</td>
<td>${x/\text{OJ}, y/\text{John}}$</td>
</tr>
<tr>
<td>$\text{Knows}(\text{John}, x)$</td>
<td>$\text{Knows}(\text{y}, \text{Mother}(\text{y}))$</td>
<td>${y/\text{John}, x/\text{Mother}(\text{John})}$</td>
</tr>
<tr>
<td>$\text{Knows}(\text{John}, x)$</td>
<td>$\text{Knows}(\text{x}, \text{OJ})$</td>
<td>fail</td>
</tr>
</tbody>
</table>

Standardizing apart eliminates overlap of variables, e.g., $\text{Knows}(z_{17}, \text{OJ})$
Conversion to CNF

Everyone who loves all animals is loved by someone:

\[ \forall x \ [\forall y \ Animal(y) \implies Loves(x, y)] \implies [\exists y \ Loves(y, x)] \]

1. Eliminate biconditionals and implications

\[ \forall x \ [\neg \forall y \ \neg Animal(y) \lor Loves(x, y)] \lor [\exists y \ Loves(y, x)] \]

2. Move \text{\neg} inwards:

\[ \neg \forall x, p \equiv \exists x \ \neg p, \quad \neg \exists x, p \equiv \forall x \ \neg p: \]

\[ \forall x \ [\exists y \ \neg (\neg Animal(y) \lor Loves(x, y))] \lor [\exists y \ Loves(y, x)] \]
\[ \forall x \ [\exists y \ \neg Animal(y) \land \neg Loves(x, y)] \lor [\exists y \ Loves(y, x)] \]
\[ \forall x \ [\exists y \ Animal(y) \land \neg Loves(x, y)] \lor [\exists y \ Loves(y, x)] \]
Conversion to CNF contd.

3. Standardize variables: each quantifier should use a different one

\[ \forall x \ [\exists y \ Animal(y) \land \neg Loves(x, y)] \lor [\exists z \ Loves(z, x)] \]

4. Skolemize: a more general form of existential instantiation. Each existential variable is replaced by a Skolem function of the enclosing universally quantified variables:

\[ \forall x \ [Animal(F(x)) \land \neg Loves(x, F(x))] \lor Loves(G(x), x) \]
Conversion to CNF contd.

5. Drop universal quantifiers:

\[\text{Animal}(F(x)) \land \neg \text{Loves}(x, F(x)) \lor \text{Loves}(G(x), x)\]

6. Distribute \(\land\) over \(\lor\):

\[\text{Animal}(F(x)) \lor \text{Loves}(G(x), x) \land \neg \text{Loves}(x, F(x)) \lor \text{Loves}(G(x), x)\]
Generalized Modus Ponens (GMP)

\[ p_1', \ p_2', \ \ldots, \ p_n', \ (p_1 \land p_2 \land \ldots \land p_n \Rightarrow q) \]

\[ \frac{\ q \theta}{\ } \]

where \( p_i' \theta = p_i \theta \)

- \( p_1' \) is \( \text{King}(John) \)
- \( p_1 \) is \( \text{King}(x) \)
- \( p_2' \) is \( \text{Greedy}(y) \)
- \( p_2 \) is \( \text{Greedy}(x) \)
- \( \theta \) is \{\( x/\text{John}, \ y/\text{John} \)\}
- \( q \) is \( \text{Evil}(x) \)
- \( q \theta \) is \( \text{Evil}(\text{John}) \)

- GMP used with KB of definite clauses (those clauses having exactly one positive literal).
- All variables assumed universally quantified
Soundness of GMP

Need to show that

\[ p_1', \ldots, p_n', (p_1 \land \ldots \land p_n \Rightarrow q) \models q\theta \]

provided that \( p_i'\theta = p_i\theta \) for all \( i \)

Lemma: For any definite clause \( p \), we have \( p \models p\theta \) by UI

1. \( (p_1 \land \ldots \land p_n \Rightarrow q) \models (p_1 \land \ldots \land p_n \Rightarrow q)\theta \) and

\[ (p_1 \land \ldots \land p_n \Rightarrow q)\theta = (p_1\theta \land \ldots \land p_n\theta \Rightarrow q\theta) \]

2. \( p_1', \ldots, p_n' \models p_1' \land \ldots \land p_n' \models p_1'\theta \land \ldots \land p_n'\theta \)

3. From 1 and 2, \( q\theta \) follows by ordinary Modus Ponens
Resolution

Full first-order version:

\[
\begin{align*}
\ell_1 \lor \cdots \lor \ell_k, & \quad m_1 \lor \cdots \lor m_n \\
(\ell_1 \lor \cdots \lor \ell_{i-1} \lor \ell_{i+1} \lor \cdots \lor \ell_k \lor m_1 \lor \cdots \lor m_{j-1} \lor m_{j+1} \lor \cdots \lor m_n)
\end{align*}
\]

where \( \text{UNIFY}(\ell_i, -m_j) = \theta \).

For example,

\[
\begin{align*}
-\text{Rich}(x) \lor \text{Unhappy}(x) \\
\text{Rich}(\text{Ken})
\end{align*}
\]

\[
\begin{array}{c}
\hline
\text{Rich}(\text{Ken}) \\
\text{Unhappy}(\text{Ken}) \\
\hline
\end{array}
\]

with \( \theta = \{x/\text{Ken}\} \).
Soundness of Resolution

Need to show $l_1 \lor \cdots \lor l_k, \quad m_1 \lor \cdots \lor m_n \models c$ where $c$ is

$$(l_1 \lor \cdots \lor l_{i-1} \lor l_{i+1} \lor \cdots \lor l_k \lor m_1 \lor \cdots \lor m_{j-1} \lor m_{j+1} \lor \cdots \lor m_n)\theta$$

and $\text{UNIFY}(l_i, -m_j) = \theta$.

This is true because

1. $l_1 \lor \cdots \lor l_k \models (l_1 \lor \cdots \lor l_k)\theta$ by UI,
2. $m_1 \lor \cdots \lor m_n \models (m_1 \lor \cdots \lor m_n)\theta$ by UI, and
3. $(l_1 \lor \cdots \lor l_k)\theta$ and $(m_1 \lor \cdots \lor m_n)\theta$ imply $c$ by propositional resolution.
Using Resolution

1. Obtain $CNF(KB \land \neg \alpha)$
2. Apply resolution steps to the CNF
3. If the empty clause is generated, then $KB \models \alpha$.

Theorem: Resolution is a refutational completeness inference system for $KB \models \alpha$. 
Example Knowledge Base

The law says that it is a crime for an American to sell weapons to hostile nations. The country Nono, an enemy of America, has some missiles, and all of its missiles were sold to it by Colonel West, who is American.

Prove that Col. West is a criminal
... it is a crime for an American to sell weapons to hostile nations:
Example Knowledge Base contd.

... it is a crime for an American to sell weapons to hostile nations:

\[ \text{American}(x) \land \text{Weapon}(y) \land \text{Sells}(x, y, z) \land \text{Hostile}(z) \implies \text{Criminal}(x) \]

Nono ... has some missiles
... it is a crime for an American to sell weapons to hostile nations:

\[ \text{American}(x) \land \text{Weapon}(y) \land \text{Sells}(x, y, z) \land \text{Hostile}(z) \implies \text{Criminal}(x) \]

Nono ... has some missiles, i.e.,
\[ \exists x \ \text{Owns}(\text{Nono}, x) \land \text{Missile}(x) : \]

\[ \text{Owns}(\text{Nono}, M_1) \ \text{and} \ \text{Missile}(M_1) \]

... all of its missiles were sold to it by Colonel West
Example Knowledge Base contd.

... it is a crime for an American to sell weapons to hostile nations:

\[ \text{American}(x) \land \text{Weapon}(y) \land \text{Sells}(x, y, z) \land \text{Hostile}(z) \implies \text{Criminal}(x) \]

Nono ... has some missiles, i.e.,
\[ \exists x \ \text{Owns}(\text{Nono}, x) \land \text{Missile}(x) : \]

\[ \text{Owns}(\text{Nono}, M_1) \ \text{and} \ \text{Missile}(M_1) \]

... all of its missiles were sold to it by Colonel West

\[ \forall x \ \text{Missile}(x) \land \text{Owns}(\text{Nono}, x) \implies \text{Sells}(\text{West}, x, \text{Nono}) \]

Missiles are weapons:
Example Knowledge Base contd.

... it is a crime for an American to sell weapons to hostile nations:

\[
American(x) \land Weapon(y) \land Sells(x, y, z) \land Hostile(z) \implies Criminal(x)
\]

Nono ... has some missiles, i.e.,
\[
\exists x \text{ Owns}(Nono, x) \land \text{Missile}(x):
\]

\[
\text{Owns}(Nono, M_1) \text{ and } \text{Missile}(M_1)
\]

... all of its missiles were sold to it by Colonel West

\[
\forall x \text{ Missile}(x) \land \text{Owns}(Nono, x) \implies \text{Sells}(West, x, Nono)
\]

Missiles are weapons:

\[
\text{Missile}(x) \implies \text{Weapon}(x)
\]

An enemy of America counts as “hostile”:
... it is a crime for an American to sell weapons to hostile nations:

\[\text{American}(x) \land \text{Weapon}(y) \land \text{Sells}(x, y, z) \land \text{Hostile}(z) \implies \text{Criminal}(x)\]

Nono ... has some missiles, i.e.,
\[\exists x \ \text{Owns}(\text{Nono}, x) \land \text{Missile}(x):\]

\[\text{Owns}(\text{Nono}, M_1) \text{ and } \text{Missile}(M_1)\]
... all of its missiles were sold to it by Colonel West

\[\forall x \ \text{Missile}(x) \land \text{Owns}(\text{Nono}, x) \implies \text{Sells}(\text{West}, x, \text{Nono})\]

Missiles are weapons:

\[\text{Missile}(x) \implies \text{Weapon}(x)\]

An enemy of America counts as “hostile”:

\[\text{Enemy}(x, \text{America}) \implies \text{Hostile}(x)\]
Resolution Proof: Definite Clauses

neg American(x) ∨ neg Weapon(y) ∨ neg Sells(x,y,z) ∨ neg Hostile(z) ∨ Criminal(x) 

neg Criminal(West)

American(West)

neg American(West) ∨ neg Weapon(y) ∨ neg Sells(West,y,z) ∨ neg Hostile(z)

neg Weapon(y) ∨ neg Sells(West,y,z) ∨ neg Hostile(z)

neg Missile(x) ∨ Weapon(x)

neg Missile(y) ∨ neg Sells(West,y,z) ∨ neg Hostile(z)

Missile(M1)

neg Missile(M1) ∨ neg Owns(Nono,M1) ∨ neg Hostile(Nono)

neg Sells(West,M1,z) ∨ neg Hostile(z)

neg Missile(x) ∨ neg Owns(Nono,x) ∨ Sells(West,x,Nono)

neg Sells(West,M1,z) ∨ neg Hostile(z)

Missile(M1)

neg Missile(M1) ∨ neg Owns(Nono,M1) ∨ neg Hostile(Nono)

Missile(M1)

neg Enemy(x,America) ∨ Hostile(x)

neg Owns(Nono,M1) ∨ neg Hostile(Nono)

Enemy(Nono,America)

neg Hostile(Nono)

Enemy(Nono,America)
Logic programming

Sound bite: computation as inference on logical KBs

Logic programming
1. Identify problem
2. Assemble information
3. Tea break
4. Encode information in KB
5. Encode problem instance as facts
6. Ask queries
7. Find false facts

Ordinary programming
1. Identify problem
2. Assemble information
3. Figure out solution
4. Program solution
5. Encode problem instance as data
6. Apply program to data
7. Debug procedural errors

Should be easier to debug $Capital(New York, US) \text{ than } x := x + 2$!
Prolog Systems

- A resolution inference system on Horn clauses + bells & whistles
- Widely used in Europe, Japan (basis of 5th Generation project)
  Compilation techniques $\Rightarrow$ 60 million LIPS
- Program = set of definite clauses, which are of form:
  head :- literal$_1$, ... literal$_n$.

  weapon(X) :- missile(X).

- Efficient unification by open coding
- Efficient retrieval of matching clauses by direct linking
- Depth-first, left-to-right backward chaining
- Built-in predicates for arithmetic etc., e.g., $X \text{ is } Y \times Z + 3$
- Closed-world assumption (“negation as failure”) e.g., given
  alive(X) :- not dead(X). alive(joe) succeeds if
dead(joe) fails.
% it is a crime to sell weapons to hostile nations:
criminal(X):-american(X),weapon(Y),sells(X,Y,Z),hostile(Z).

% Nono ... has some missiles,
owns(nono,m1).
missile(m1).

% all of its missiles were sold to it by Colonel West
sells(west,X,nono) :- missile(X), owns(nono,X).

% Missiles are weapons
weapon(X) :- missile(X).

% An enemy of America counts as "hostile":
hostile(X) :- enemy(X,america).

% The country Nono, an enemy of America ...
enemy(nono,america).

% West, who is American ...
american(west).

?- criminal(Who).
Who = west.
Prolog Examples

Depth-first search from a start state \( X \):

\[
\text{dfs}(X) :- \text{goal}(X).
\]

\[
\text{dfs}(X) :- \text{successor}(X,S),\text{dfs}(S).
\]

No need to loop over \( S \): \textit{successor} succeeds for each

Appending two lists to produce a third:

\[
\text{append}([],Y,Y).
\]

\[
\text{append}([X|L],Y,[X|Z]) :- \text{append}(L,Y,Z).
\]

query: \( \text{append}(A,B,[1,2]) \) ?

answers: \( A=[] \quad B=[1,2] \)


\( A=[1,2] \quad B=[] \)