A brief history of reasoning

- **450 b.c.** Stoics: propositional logic, inference (maybe)
- **322 b.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
- **1931** Gödel: \( \neg \exists \) complete algorithm for arithmetic
- **1960** Davis/Putnam: “practical” algorithm for propositional logic
- **1965** Robinson: “practical” algorithm for FOL—resolution

Universal instantiation (UI)

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

\[
\forall v \alpha \\
\text{Subst}(\{v/g\}, \alpha)
\]

for any variable \( v \) and ground term \( g \)

- **E.g.,** \( \forall x \ King(x) \land \text{Greedy}(x) \implies \text{Evil}(x) \) yields

\[
\text{King}(\text{John}) \land \text{Greedy}(\text{John}) \implies \text{Evil}(\text{John})
\]

\[
\text{King}(\text{Richard}) \land \text{Greedy}(\text{Richard}) \implies \text{Evil}(\text{Richard})
\]

\[
\text{King}(\text{Father}(\text{John})) \land \text{Greedy}(\text{Father}(\text{John})) \implies \text{Evil}(\text{Father}(\text{John}))
\]

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.
Reduction to Propositional Inference

Suppose the KB contains just the following:

\[ \forall x \ (\text{King}(x) \land \text{Greedy}(x)) \Rightarrow \text{Evil}(x) \]

\[ \text{King}(\text{John}) \]

\[ \text{Greedy}(\text{John}) \]

\[ \text{Brother}(\text{Richard}, \text{John}) \]

Instantiating the universal sentence in all possible ways, we have

\[ \text{King}(\text{John}) \land \text{Greedy}(\text{John}) \Rightarrow \text{Evil}(\text{John}) \]

\[ \text{King}(\text{Richard}) \land \text{Greedy}(\text{Richard}) \Rightarrow \text{Evil}(\text{Richard}) \]

\[ \text{King}(\text{John}) \]

\[ \text{Greedy}(\text{John}) \]

\[ \text{Brother}(\text{Richard}, \text{John}) \]

The new KB is propositionalized: proposition symbols are

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

Problems with Propositionalization

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

\[ \forall x \ (\text{King}(x) \land \text{Greedy}(x)) \Rightarrow \text{Evil}(x) \]

\[ \text{King}(\text{John}) \]

\[ \forall y \ \text{Greedy}(y) \]

\[ \text{Brother}(\text{Richard}, \text{John}) \]

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

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

Reduction to Propositional Inference

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

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., \( \text{Father}(\text{Father}(\text{Father}(\text{John}))) \).

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
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>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>${x/OJ, y/\text{John}}$</td>
</tr>
<tr>
<td>Knows(John, x)</td>
<td>Knows(y, Mother(y))</td>
<td>${y/\text{John}, x/Mother(y)}$</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>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>${x/OJ, y/\text{John}}$</td>
</tr>
<tr>
<td>Knows(John, x)</td>
<td>Knows(y, Mother(y))</td>
<td>${y/\text{John}, x/Mother(y)}$</td>
</tr>
<tr>
<td>Knows(John, x)</td>
<td>Knows(x, OJ)</td>
<td>${}$</td>
</tr>
</tbody>
</table>
Conversion to CNF

- Everyone who loves all animals is loved by someone:
  \[ \forall x \left[ \forall y \ Animal(y) \implies Loves(x, y) \right] \implies \exists y \ Loves(y, x) \]

  1. Eliminate biconditionals and implications
  \[ \forall x \left[ \neg \forall y \neg Animal(y) \lor Loves(x, y) \right] \lor \exists y \ Loves(y, x) \]

  2. Move \( \neg \) inwards: \[ \forall x, p \equiv \exists x, p, \quad \neg \exists x, p \equiv \forall x, \neg p: \]
  \[ \forall x \left[ \exists y \neg \left( \neg Animal(y) \lor Loves(x, y) \right) \right] \lor \exists y \ Loves(y, x) \]

  3. Standardize variables: each quantifier should use a different one
  \[ \forall x \left[ Animal(F(x)) \land \neg Loves(x, F(x)) \right] \lor Loves(G(x), x) \]

Conversion to CNF contd.

- Drop universal quantifiers:
  \[ \exists y \ Animal(y) \land \neg Loves(x, y) \lor \exists z \ Loves(z, x) \]

- Distribute \( \land \) over \( \lor \):
  \[ \forall x \left[ Animal(F(x)) \lor Loves(G(x), x) \right] \land \left[ \neg Loves(x, F(x)) \lor Loves(G(x), x) \right] \]

Conversion to CNF contd.

- Skolemize: a more general form of existential instantiation. Each existential variable is replaced by a Skolem function of the enclosing universally quantified variables:
  \[ \forall x \left[ Animal(F(x)) \land \neg Loves(x, F(x)) \right] \lor Loves(G(x), x) \]
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

Soundness of Resolution

Need to show \( \ell_1 \lor \cdots \lor \ell_k, m_1 \lor \cdots \lor m_n \models c \) where \( c \) is
\( (\ell_1 \lor \cdots \lor \ell_{i-1} \lor \ell_{i+1} \lor \cdots \lor \ell_k \lor m_1 \lor \cdots \lor m_{j-1} \lor m_{j+1} \lor \cdots \lor m_n)\theta \)
and \( \text{UNIFY}(\ell_i, -m_j) = \theta \).

This is true because
1. \( \ell_1 \lor \cdots \lor \ell_k \models (\ell_1 \lor \cdots \lor \ell_k)\theta \) by UI,
2. \( m_1 \lor \cdots \lor m_n \models (m_1 \lor \cdots \lor m_n)\theta \) by UI, and
3. \( (\ell_1 \lor \cdots \lor \ell_k)\theta \) and \( (m_1 \lor \cdots \lor m_n)\theta \) imply \( c \) by propositional resolution.

Generalized Modus Ponens (GMP)

\[
\frac{p_1', p_2', \ldots, p_n', (p_1 \land p_2 \land \ldots \land p_n \Rightarrow q)}{q\theta}
\]

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

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

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

Resolution

Full first-order version:
\[
\frac{\ell_1 \lor \cdots \lor \ell_k, m_1 \lor \cdots \lor m_n}{\ell_1 \lor \cdots \lor \ell_{i-1} \lor \ell_{i+1} \lor \cdots \lor \ell_k \lor m_1 \lor \cdots \lor m_{j-1} \lor m_{j+1} \lor \cdots \lor m_n}
\]

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

For example,
\[
\begin{align*}
\neg \text{Rich}(x) \lor \text{Unhappy}(x) \\
\text{Rich}(Ken) \\
\text{Unhappy}(Ken)
\end{align*}
\]

with \( \theta = \{x/\text{Ken}\} \).
**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

**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

**Using Resolution**

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

Theorem: Resolution is a refutationally complete inference system for $KB \models \alpha$.

**Example Knowledge Base contd.**

... 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, 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”:

\[ \text{Enemy}(x, \text{America}) \implies \text{Hostile}(x) \]
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. Apply program to data
6. Debug procedural errors

Should be easier to debug Capital(New York, US) than $x := x + 2$!

Resolution Proof: Definite Clauses

Resolution Proof in Prolog

% 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 ...
al enemy(nono,america).
% West, who is American ...
al american(west).
?- criminal(Who).
Who = west.

Prolog Systems

A resolution inference system on Horn clauses + bells & whistles

- Widely used in Europe, Japan (basis of 5th Generation project)
- Compilation techniques ⇒ 60 million LIPS
- Program = set of definite clauses, which are of form:
  head :- literal₁, ..., literalₙ.
  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., $X$ is $Y*Z+3$
- Closed-world assumption (“negation as failure”) e.g., given alive(X) :- not dead(X). alive(joe) succeeds if dead(joe) fails.
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 \( S \): \text{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:

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