Inference in first-order logic
Outline

♦ Reducing first-order inference to propositional inference
♦ Unification
♦ Generalized Modus Ponens
♦ Forward and backward chaining
♦ Logic programming
♦ Resolution
## A brief history of first-order logic

<table>
<thead>
<tr>
<th>Year</th>
<th>Name</th>
<th>Contribution</th>
</tr>
</thead>
<tbody>
<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>∃ complete algorithm for FOL</td>
</tr>
<tr>
<td>1930</td>
<td>Herbrand</td>
<td>complete algorithm for FOL (reduce to propositional)</td>
</tr>
<tr>
<td>1931</td>
<td>Gödel</td>
<td>¬∃ 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>
Frege’s notation for FOL

In Frege’s notation, formulas looked like tree structures. He used

Example: $\forall x (A(x) \rightarrow B(x))$

Frege would have written

```
  a
  |
  B a
  |
  A a
```
Universal instantiation (UI)

Every instantiation of a universally quantified sentence is entailed by it

For every variable $v$ and ground term $g$, if $\theta$ is the substitution $\{v \leftarrow g\}$ then

$$
\forall v \alpha \\
\frac{}{\alpha \theta}
$$

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

King(John) $\land$ Greedy(John) $\Rightarrow$ Evil(John)

King(Richard) $\land$ Greedy(Richard) $\Rightarrow$ Evil(Richard)

King(father(John)) $\land$ Greedy(father(John)) $\Rightarrow$ Evil(father(John))

$:$
Existential instantiation (EI)

For any sentence $\alpha$, variable $v$, and constant symbol $k$ that doesn’t appear elsewhere in the knowledge base, if $\theta = \{v \leftarrow k\}$ then

$$
\exists v \quad \alpha

\Rightarrow

\alpha \theta
$$

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

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

where $C_1$ is a new constant symbol (i.e., doesn’t already appear somewhere)

In words:

If there is a crown on John’s head, then we can call the crown $C_1$

$C_1$ is called a Skolem constant
Existential instantiation, continued

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

Mathematicians use these techniques informally every day.
Example: proofs involving limits

Given \( \lim_{x \to 5} f(x) = 2 \), i.e.,
\[
\forall \epsilon > 0 \ \exists \delta > 0 \ \forall x \ |x - 5| < \delta, \ |f(x) - 2| < \epsilon.
\]

Let \( \epsilon \) be any number \( > 0 \).
Then \( \exists \delta > 0 \ \forall x \ |x - 5| < \delta, \ |f(x) - 2| < \epsilon. \)

Let \( \delta_1 > 0 \) be such that \( \forall x \ |x - 5| < \delta, \ |f(x) - 2| < \epsilon. \)

Let \( x \) be any number such that \( |x - 5| < \delta_1. \) Then \( |f(x) - 2| < \epsilon. \)

\[\ldots\]
Reduction to propositional inference

Suppose the KB contains just the following:

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

Instantiating the universal sentence in all possible ways, we have

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

The new KB is *propositionalized*: proposition symbols are

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

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 1: propositionalization can create lots of irrelevant sentences.
E.g., suppose we are given

\[ \forall x \; \text{King}(x) \land \text{Greedy}(x) \Rightarrow \text{Evil}(x) \]
\[
\text{King}(\text{John})
\]
\[
\forall y \; \text{Greedy}(y)
\]
\[
\text{Brother}(\text{Richard}, \text{John})
\]
\[
\text{Daughter}(\text{John}, \text{Joanna})
\]

To prove \( \text{Evil}(\text{John}) \), we first use propositionalization to get \( \text{Greedy}(\text{John}) \)

But propositionalization also produces \( \text{Greedy}(\text{Richard}) \) and \( \text{Greedy}(\text{Joanna}) \)

With \( p \) \( k \)-ary predicates and \( n \) constants, there are \( p \cdot n^k \) instantiations
Reduction, continued

Problem 2: with function symbols, propositionalization can create infinitely many sentences!

- \textit{Greedy}(John)
- \textit{Greedy}(father(John))
- \textit{Greedy}(father(father(John)))
  
  \ldots

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

Idea: For $n = 0$ to $\infty$ do
  
  create a propositional KB by instantiating with all terms of depth $\leq n$
  
  (e.g., up to $n$ nested occurrences of $\text{Father}$)

  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 \textit{semidecidable}
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 $\text{Greedy}(y)$

$\theta = \{ x \leftarrow \text{John}, y \leftarrow \text{John} \}$ works

A unifier for $\alpha$ and $\beta$ is a substitution $\theta$ such that $\alpha \theta = \beta \theta$ 

$\alpha$ and $\beta$ are unifiable if such a $\theta$ exists

<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></td>
</tr>
<tr>
<td>$\text{Knows}(\text{John}, x)$</td>
<td>$\text{Knows}(y, \text{Joanna})$</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>$\text{Knows}(x, \text{Joanna})$</td>
<td></td>
</tr>
<tr>
<td>$\text{Knows}(\text{John}, x)$</td>
<td>$\text{Knows}(x_{17}, \text{Joanna})$</td>
<td></td>
</tr>
<tr>
<td>$\text{Knows}(x, x)$</td>
<td>$\text{Knows}(z, \text{mother}(z))$</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 $Greedy(y)$

$\theta = \{x \leftarrow John, y \leftarrow John\}$ works

A unifier for $\alpha$ and $\beta$ is a substitution $\theta$ such that $\alpha \theta = \beta \theta$

$\alpha$ and $\beta$ are unifiable if such a $\theta$ exists

<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>${x \leftarrow Jane}$</td>
</tr>
<tr>
<td>$Knows(John, x)$</td>
<td>$Knows(y, Joanna)$</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>$Knows(x, Joanna)$</td>
<td></td>
</tr>
<tr>
<td>$Knows(John, x)$</td>
<td>$Knows(x_{17}, Joanna)$</td>
<td></td>
</tr>
<tr>
<td>$Knows(x, x)$</td>
<td>$Knows(z, mother(z))$</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 $\text{Greedy}(y)$.

$$\theta = \{x \leftarrow \text{John}, y \leftarrow \text{John}\}$$ works

A **unifier** for $\alpha$ and $\beta$ is a substitution $\theta$ such that $\alpha\theta = \beta\theta$.

$\alpha$ and $\beta$ are **unifiable** if such a $\theta$ exists.

<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>${x \leftarrow \text{Jane}}$</td>
</tr>
<tr>
<td>$\text{Knows}(\text{John}, x)$</td>
<td>$\text{Knows}(y, \text{Joanna})$</td>
<td>${x \leftarrow \text{Joanna}, y \leftarrow \text{John}}$</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>$\text{Knows}(x, \text{Joanna})$</td>
<td></td>
</tr>
<tr>
<td>$\text{Knows}(\text{John}, x)$</td>
<td>$\text{Knows}(x_{17}, \text{Joanna})$</td>
<td></td>
</tr>
<tr>
<td>$\text{Knows}(x, x)$</td>
<td>$\text{Knows}(z, \text{mother}(z))$</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 $\text{Greedy}(y)$

$\theta = \{x \leftarrow \text{John}, y \leftarrow \text{John}\}$ works

A unifier for $\alpha$ and $\beta$ is a substitution $\theta$ such that $\alpha \theta = \beta \theta$

$\alpha$ and $\beta$ are unifiable if such a $\theta$ exists

<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>${x \leftarrow \text{Jane}}$</td>
</tr>
<tr>
<td>Knows(John, x)</td>
<td>Knows(y, Joanna)</td>
<td>${x \leftarrow \text{Joanna}, y \leftarrow \text{John}}$</td>
</tr>
<tr>
<td>Knows(John, x)</td>
<td>Knows(y, mother(y))</td>
<td>${y \leftarrow \text{John}, x \leftarrow \text{mother(John)}}$</td>
</tr>
<tr>
<td>Knows(John, x)</td>
<td>Knows(x, Joanna)</td>
<td></td>
</tr>
<tr>
<td>Knows(John, x)</td>
<td>Knows(x_{17}, Joanna)</td>
<td></td>
</tr>
<tr>
<td>Knows(x, x)</td>
<td>Knows(z, mother(z))</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 $Greedy(y)$.

$\theta = \{x \leftarrow John, y \leftarrow John\}$ works

A unifier for $\alpha$ and $\beta$ is a substitution $\theta$ such that $\alpha\theta = \beta\theta$.

$\alpha$ and $\beta$ are unifiable if such a $\theta$ exists.

<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>${x \leftarrow Jane}$</td>
</tr>
<tr>
<td>Knows(John, x)</td>
<td>Knows(y, Joanna)</td>
<td>${x \leftarrow Joanna, y \leftarrow John}$</td>
</tr>
<tr>
<td>Knows(John, x)</td>
<td>Knows(y, mother(y))</td>
<td>${y \leftarrow John, x \leftarrow mother(John)}$</td>
</tr>
<tr>
<td>Knows(John, x)</td>
<td>Knows(x, Joanna)</td>
<td>fail</td>
</tr>
<tr>
<td>Knows(John, x)</td>
<td>Knows(x_{17}, Joanna)</td>
<td></td>
</tr>
<tr>
<td>Knows(x, x)</td>
<td>Knows(z, mother(z))</td>
<td></td>
</tr>
</tbody>
</table>

Knows(John, x) Knows(x_{17}, Joanna) fail

CMSC 421: Chapter 9  15
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 $\text{Greedy}(y)$

$\theta = \{x \leftarrow \text{John}, y \leftarrow \text{John}\}$ works

A unifier for $\alpha$ and $\beta$ is a substitution $\theta$ such that $\alpha\theta = \beta\theta$

$\alpha$ and $\beta$ are unifiable if such a $\theta$ exists

<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>${x \leftarrow \text{Jane}}$</td>
</tr>
<tr>
<td>$\text{Knows}(\text{John}, x)$</td>
<td>$\text{Knows}(y, \text{Joanna})$</td>
<td>${x \leftarrow \text{Joanna}, y \leftarrow \text{John}}$</td>
</tr>
<tr>
<td>$\text{Knows}(\text{John}, x)$</td>
<td>$\text{Knows}(y, \text{mother}(y))$</td>
<td>${y \leftarrow \text{John}, x \leftarrow \text{mother}(\text{John})}$</td>
</tr>
<tr>
<td>$\text{Knows}(\text{John}, x)$</td>
<td>$\text{Knows}(x, \text{Joanna})$</td>
<td>fail</td>
</tr>
<tr>
<td>$\text{Knows}(\text{John}, x)$</td>
<td>$\text{Knows}(x_{17}, \text{Joanna})$</td>
<td>${x_{17} \leftarrow \text{John}, x \leftarrow \text{Joanna}}$</td>
</tr>
<tr>
<td>$\text{Knows}(x, x)$</td>
<td>$\text{Knows}(z, \text{mother}(z))$</td>
<td></td>
</tr>
</tbody>
</table>

Standardizing apart eliminates overlap of variables, e.g., $\text{Knows}(x_{17}, \text{Joanna})$
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 $\text{Greedy}(y)$

$\theta = \{x \leftarrow \text{John}, y \leftarrow \text{John}\}$ works

A **unifier** for $\alpha$ and $\beta$ is a substitution $\theta$ such that $\alpha\theta = \beta\theta$

$\alpha$ and $\beta$ are **unifiable** if such a $\theta$ exists

<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>${x \leftarrow \text{Jane}}$</td>
</tr>
<tr>
<td>$\text{Knows}(\text{John}, x)$</td>
<td>$\text{Knows}(y, \text{Joanna})$</td>
<td>${x \leftarrow \text{Joanna}, y \leftarrow \text{John}}$</td>
</tr>
<tr>
<td>$\text{Knows}(\text{John}, x)$</td>
<td>$\text{Knows}(y, \text{mother}(y))$</td>
<td>${y \leftarrow \text{John}, x \leftarrow \text{mother}(\text{John})}$</td>
</tr>
<tr>
<td>$\text{Knows}(\text{John}, x)$</td>
<td>$\text{Knows}(x, \text{Joanna})$</td>
<td>$\text{fail}$</td>
</tr>
<tr>
<td>$\text{Knows}(\text{John}, x)$</td>
<td>$\text{Knows}(x_{17}, \text{Joanna})$</td>
<td>${x_{17} \leftarrow \text{John}, x \leftarrow \text{Joanna}}$</td>
</tr>
<tr>
<td>$\text{Knows}(x, x)$</td>
<td>$\text{Knows}(z, \text{mother}(z))$</td>
<td>$\text{fail}$</td>
</tr>
</tbody>
</table>

**Standardizing apart** eliminates overlap of variables, e.g., $\text{Knows}(x_{17}, \text{Joanna})$

Can’t unify a variable with a term that contains the variable
Unification (continued)

A most general unifier (mgu) for \( \alpha \) and \( \beta \) is a substitution \( \theta \) such that

1. \( \theta \) is a unifier for \( \alpha \) and \( \beta \);
2. for every unifier \( \theta' \) of \( \alpha \) and \( \beta \) and for every expression \( e \), 
   \( e\theta' \) is a substitution instance of \( e\theta \)

E.g., let \( \alpha = \text{Knows}(w, \text{father}(x)) \) and \( \beta = \text{Knows}(\text{mother}(y), y) \)

\( \theta_1 = \{w \leftarrow \text{mother}(\text{father}(x)), y \leftarrow \text{father}(x)\} \) is an mgu

\( \theta_2 = \{w \leftarrow \text{mother}(\text{father}(v)), y \leftarrow \text{father}(v), x \leftarrow v\} \) is an mgu

\( \theta_3 = \{w \leftarrow \text{mother}(\text{father}(\text{John})), y \leftarrow \text{father}(\text{John})\} \)

is a unifier but it is not an mgu

If \( \theta \) and \( \theta' \) are mgus for \( \alpha \) and \( \beta \), then they are identical except for renaming of variables
Algorithm to find an mgu

Compare the expressions element by element, building up a substitution along the way. Here’s the basic idea (the book gives additional details):

For each pair of corresponding elements:

- Apply the substitution we’ve built so far
- If the two elements are the same after substituting, keep going
- Else if one of them is a variable $x$ and the other is an expression $e$, and if $x$ doesn’t appear anywhere in $e$ (the ”occur check”) then incorporate $x = e$ into the substitution
- Else FAIL

$\text{Knows}(\text{John, } x) \uparrow \uparrow \downarrow$

$\text{Knows}(y, \text{mother}(y))$

$\theta = \{\} \quad \theta = \{y \leftarrow \text{John}\} \quad \theta = \{y \leftarrow \text{John}, x \leftarrow \text{mother(John)}\}$

Runs in quadratic time (would be linear time if it weren’t for the occur check)
Generalized Modus Ponens (GMP)

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

where \( \theta \) is a substitution such that \( p_i' \theta = p_i \theta \) for all \( i \), and all variables are assumed to be universally quantified.

Example:

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

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

with \( \theta = \{ x \leftarrow John, y \leftarrow John \} \), \( q \theta = \text{Evil}(x) \theta = \text{Evil}(John) \)

Equivalent formulation using \textit{definite clauses} (\textbf{exactly} one positive literal)

\[ p_1', p_2', \ldots, p_n', (\neg p_1 \lor \neg p_2 \lor \ldots \lor \neg p_n \lor q) \]

\[ q \theta \]
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 \( i \)

We know that for any definite clause \( p \), universal instantiation gives us

\[ p \models p\theta. \] Thus

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

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

If \( p_i'\theta = p_i\theta \) for all \( i \), then \( q\theta \) follows from 1 and 2 and ordinary Modus Ponens
Example knowledge base

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

Prove one of the following:
1. Russell & Norvig have a sense of humor
2. Col. West is a criminal
Example knowledge base, continued

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

\[\text{American}(x) \land \text{Weapon}(y) \land \text{Sells}(x, y, z) \land \text{Hostile}(z) \Rightarrow \text{Criminal}(x)\]
Example knowledge base, continued

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

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

\[
\exists x \: \text{Owns}(\text{Nono}, x) \land \text{Missile}(x)
\]

\[
\text{Owns}(\text{Nono}, M_1) \: \text{and} \: \text{Missile}(M_1)
\]
Example knowledge base, continued

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

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

\[ \exists x \ \text{Owns}(Nono, x) \land \text{Missile}(x) \]

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

\[ \forall x \ \text{Missile}(x) \land \text{Owns}(Nono, x) \Rightarrow \text{Sells}(West, x, Nono) \]
The law says it is a crime for an American to sell weapons to hostile nations. The country Nono, an enemy of America, has some missiles. All of its missiles were sold to it by Colonel West, who is American.

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

\[\exists x \text{ Owns}(Nono, x) \land \text{Missile}(x)\]

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

\begin{align*}
\forall x & \text{ Missile}(x) \land \text{Owns}(Nono, x) \Rightarrow \text{Sells}(West, x, Nono) \\
\text{Missile}(x) & \Rightarrow \text{Weapon}(x) \quad \text{Missiles are weapons}
\end{align*}
Example knowledge base, continued

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

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

\[
\exists x \text{ Owns}(\text{Nono}, x) \land \text{Missile}(x)
\]

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

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

\[
\text{Missile}(x) \Rightarrow \text{Weapon}(x) \quad \text{Missiles are weapons}
\]

\[
\text{Enemy}(x, \text{America}) \Rightarrow \text{Hostile}(x) \quad \text{An enemy of America is “hostile”}
\]
Example knowledge base, continued

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

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

\[
\exists x \ \text{Owns}(\text{Nono}, x) \land \text{Missile}(x)
\]

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

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

\[
\text{Missile}(x) \Rightarrow \text{Weapon}(x) \quad \text{Missiles are weapons}
\]

\[
\text{Enemy}(x, \text{America}) \Rightarrow \text{Hostile}(x) \quad \text{An enemy of America is “hostile”}
\]

\[
\text{American}(\text{West})
\]
The law says it is a crime for an American to sell weapons to hostile nations. The country Nono, an enemy of America, has some missiles. All of its missiles were sold to it by Colonel West, who is American.

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

\[
\exists x \: \text{Owns}(\text{Nono}, x) \land \text{Missile}(x)
\]

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

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

\[
\text{Missile}(x) \Rightarrow \text{Weapon}(x) \quad \text{Missiles are weapons}
\]

\[
\text{Enemy}(x, \text{America}) \Rightarrow \text{Hostile}(x) \quad \text{An enemy of America is “hostile”}
\]

\[
\text{American}(\text{West})
\]

\[
\text{Enemy}(\text{Nono}, \text{America})
\]
Forward chaining algorithm

function FOL-FC-Ask(KB, α) returns a substitution or false

repeat until new is empty

new ← { }

for each sentence r in KB do

(p₁ ∧ ... ∧ pₙ ⇒ q) ← Standardize-Apart(r)

for each θ such that (p₁ ∧ ... ∧ pₙ)θ = (p'₁ ∧ ... ∧ p'ₙ)θ

for some p'₁, ..., p'ₙ in KB

q' ← Subst(θ, q)

if q' is not a renaming of a sentence already in KB or new then do

add q' to new

φ ← Unify(q', α)

if φ is not fail then return φ

add new to KB

return false
Forward chaining proof

- American(West)
- Missile(M1)
- Owns(Nono,M1)
- Enemy(Nono,America)

\[
\begin{align*}
\text{American}(x) \land \text{Weapon}(y) \land \text{Sells}(x, y, z) \land \text{Hostile}(z) & \Rightarrow \text{Criminal}(x) \\
\forall x \quad \text{Missile}(x) \land \text{Owns}(Nono, x) & \Rightarrow \text{Sells}(\text{West}, x, \text{Nono}) \\
\text{Owns}(\text{Nono}, M_1) & \quad \text{Missile}(M_1) \\
\text{Missile}(x) & \Rightarrow \text{Weapon}(x) \\
\text{American}(\text{West}) & \quad \text{Enemy}(x, \text{America}) \Rightarrow \text{Hostile}(x) \\
\end{align*}
\]
Forward chaining proof

\[ \text{American}(x) \land \text{Weapon}(y) \land \text{Sells}(x, y, z) \land \text{Hostile}(z) \Rightarrow \text{Criminal}(x) \]
\[ \forall x \ (\text{Missile}(x) \land \text{Owns}(\text{Nono}, x) \Rightarrow \text{Sells}(\text{West}, x, \text{Nono}) \]
\[ \text{Owns}(\text{Nono}, M_1) \]
\[ \text{Missile}(x) \Rightarrow \text{Weapon}(x) \]
\[ \text{American}(\text{West}) \]
\[ \text{Enemy}(\text{Nono}, \text{America}) \]
\[ \text{Hostile}(\text{Nono}) \]
Forward chaining proof

\[
\begin{align*}
\text{American}(x) \land \text{Weapon}(y) \land \text{Sells}(x, y, z) \land \text{Hostile}(z) & \Rightarrow \text{Criminal}(x) \\
\forall x \quad \text{Missile}(x) \land \text{Owns}(\text{Nono}, x) & \Rightarrow \text{Sells}(\text{West}, x, \text{Nono}) \\
\text{Owns}(\text{Nono}, M_1) & \Rightarrow \text{Missile}(M_1) \\
\text{Missile}(x) & \Rightarrow \text{Weapon}(x) \\
\text{American}(\text{West}) & \Rightarrow \text{Enemy}(\text{West}, \text{America}) \\
\end{align*}
\]
Properties of forward chaining

Sound and complete for first-order definite clauses
(proof similar to propositional proof)

May not terminate in general if $\alpha$ is not entailed

This is unavoidable: entailment with definite clauses is semidecidable
(i.e., equivalent to the halting problem)

Can guarantee termination if restrictions are satisfied, e.g.,

\textit{Datalog} = first-order definite clauses + \textbf{no functions}
(e.g., the Colonel West example)

FC terminates for Datalog in poly iterations: at most $p \cdot n^k$ literals
Efficiency of forward chaining

Simple observation: no need to match a rule on iteration $k$ if a premise wasn’t added on iteration $k - 1$

⇒ match each rule whose premise contains a newly added literal

Matching itself can be expensive

◊ *Database indexing* allows $O(1)$ retrieval of known facts
  e.g., query $\text{Missile}(x)$ retrieves $\text{Missile}(M_1)$

◊ But matching conjunctive premises against known facts is NP-hard
  (see next page)

◊ Partial fix: store partial matches in data structures such as *rete networks*

Forward chaining is widely used in *deductive databases* and *expert systems*
Hard matching example

\[
\text{Diff}(wa, nt) \land \text{Diff}(wa, sa) \land \\
\text{Diff}(nt, q) \land \text{Diff}(nt, sa) \land \\
\text{Diff}(q, nsw) \land \text{Diff}(q, sa) \land \\
\text{Diff}(nsw, v) \land \text{Diff}(nsw, sa) \land \\
\text{Diff}(v, sa) \Rightarrow \text{Colorable}() \\
\text{Diff}(\text{Red}, \text{Blue}) \land \text{Diff}(\text{Red}, \text{Green}) \\
\text{Diff}(\text{Green}, \text{Red}) \land \text{Diff}(\text{Green}, \text{Blue}) \\
\text{Diff}(\text{Blue}, \text{Red}) \land \text{Diff}(\text{Blue}, \text{Green})
\]

Don’t need statements like \( nt = \text{Red} \lor nt = \text{Blue} \lor nt = \text{Green} \). Why?

\text{Colorable}() \text{ is inferred iff the CSP has a solution}

Need to try many combinations of variable values

More generally,

CSPs include 3SAT as a special case, hence matching is NP-hard
Backward chaining algorithm

function FOL-BC-Ask(\(KB, goals, \theta\)) returns a set of substitutions

inputs: \(KB\), a knowledge base
\(goals\), a list of conjuncts forming a query (\(\theta\) already applied)
\(\theta\), the current substitution, initially the empty substitution \(\{\}\)

local variables: \(answers\), a set of substitutions, initially empty

if \(goals\) is empty then return \(\{\theta\}\)
\(q' \leftarrow \text{Subst}(\theta, \text{First}(goals))\)
for each sentence \(r\) in \(KB\)
where \(\text{Standardize-Apart}(r) = (p_1 \land \ldots \land p_n \Rightarrow q)\)
and \(\theta' \leftarrow \text{Unify}(q, q')\) succeeds
\(\text{new_goals} \leftarrow [p_1, \ldots, p_n | \text{Rest}(goals)]\)
\(answers \leftarrow \text{FOL-BC-Ask}(KB, new_goals, \text{Compose}(\theta', \theta)) \cup answers\)
return \(answers\)
Backward chaining example

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

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

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

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

\[
\text{American}(\text{West})
\]

\[
\text{Enemy}(x, \text{America}) \Rightarrow \text{Hostile}(x)
\]

\[
\text{Enemy}(\text{Nono}, \text{America})
\]
Backward chaining example

\[ \text{American}(x) \land \text{Weapon}(y) \land \text{Sells}(x,y,z) \land \text{Hostile}(z) \Rightarrow \text{Criminal}(x) \]
\[ \forall x \ \text{Missile}(x) \land \text{Owns}(\text{Nono}, x) \Rightarrow \text{Sells}(\text{West}, x, \text{Nono}) \]
\[ \text{Owns}(\text{Nono}, M_1) \]
\[ \text{Missile}(M_1) \]
\[ \text{Missile}(x) \Rightarrow \text{Weapon}(x) \]
\[ \text{Enemy}(x, \text{America}) \Rightarrow \text{Hostile}(x) \]
\[ \text{American}(\text{West}) \]
\[ \text{Enemy}(\text{Nono}, \text{America}) \]
Backward chaining example

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

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

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

\[
\text{Missile}(M_1)
\]

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

\[
\text{Enemy}(x, \text{America}) \Rightarrow \text{Hostile}(x)
\]

\[
\text{American}(\text{West})
\]

\[
\text{Enemy}(\text{Nono, America})
\]
Backward chaining example

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

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

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

\[
\text{Missile}(x) \Rightarrow \text{Weapon}(x) \quad \text{Enemy}(x, \text{America}) \Rightarrow \text{Hostile}(x)
\]

\[
\text{American}(\text{West}) \quad \text{Enemy}(\text{Nono}, \text{America})
\]
Backward chaining example

\[ \text{American}(x) \land \text{Weapon}(y) \land \text{Sells}(x,y,z) \land \text{Hostile}(z) \Rightarrow \text{Criminal}(x) \]
\[ \forall x \quad \text{Missile}(x) \land \text{Owns}(\text{Nono}, x) \Rightarrow \text{Sells}(\text{West}, x, \text{Nono}) \]
\[ \text{Owns}(\text{Nono}, M_1) \]
\[ \text{Missile}(x) \Rightarrow \text{Weapon}(x) \]
\[ \text{American}(\text{West}) \]

\[ \text{American}(\text{West}) \land \text{Weapon}(y) \land \text{Sells}(x,y,z) \land \text{Hostile}(z) \Rightarrow \text{Criminal}(x) \]
\[ \forall x \quad \text{Missile}(x) \land \text{Owns}(\text{Nono}, x) \Rightarrow \text{Sells}(\text{West}, x, \text{Nono}) \]
\[ \text{Owns}(\text{Nono}, M_1) \]
\[ \text{Missile}(x) \Rightarrow \text{Weapon}(x) \]
\[ \text{American}(\text{West}) \]
Backward chaining example

American(x) ∧ Weapon(y) ∧ Sells(x, y, z) ∧ Hostile(z) ⇒ Criminal(x)
∀ x  Missile(x) ∧ Owns(Nono, x) ⇒ Sells(West, x, Nono)
Owns(Nono, M₁)
Missile(M₁)
Missile(x) ⇒ Weapon(x)  Enemy(x, America) ⇒ Hostile(x)
American(West)  Enemy(Nono, America)
Backward chaining example

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

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

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

\[ \text{Missile}(M_1) \]

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

\[ \text{Enemy}(x, \text{America}) \Rightarrow \text{Hostile}(x) \]

\[ \text{American}(\text{West}) \]

\[ \text{Enemy}(\text{Nono, America}) \]
Properties of backward chaining

♦ Depth-first recursive proof search: space is linear in size of proof

♦ Incomplete due to infinite loops
  
  Partial fix: check current goal against every goal on stack
  
  This prevents looping here:
  \[ P(x) \Rightarrow P(x) \]

  But it doesn’t prevent looping here:
  \[ Q(f(x)) \Rightarrow Q(x) \]

♦ Inefficient due to repeated subgoals (both success and failure)
  Fix using caching of previous results (extra space!)

♦ Widely used (without improvements!) for logic programming
Prolog systems

Basis: backward chaining with Horn clauses
+ extras (e.g., built-in “predicates” that do arithmetic, printing, etc.)

Program = set of clauses of the form head :- literal₁, ... literalₙ.

    criminal(X) :- american(X), weapon(Y), sells(X,Y,Z), hostile(Z).

Capitalized words (e.g., X) are variables, and
lower-case words (e.g., nono) are constants
  this is the opposite of what we’ve been doing

Depth-first, left-to-right backward chaining
Closed-world assumption (“negation as failure”)
  e.g., given alive(X) :- not dead(X).
  alive(joe) succeeds if dead(joe) fails

Compilation techniques ⇒ approaching a billion LIPS
  Efficient unification by open coding (generate unification code inline)
  Efficient retrieval of matching clauses by direct linking
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 $S$:
successor succeeds for each successor of $X$

Appending two lists to produce a third:

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

query: append(A,B,[1,2])
answers: A=[] B=[1,2]
A=[1,2] B=[]
Resolution in FOL

\[ \ell_1 \lor \cdots \lor \ell_i \lor \cdots \lor \ell_k, \quad m_1 \lor \cdots \lor m_j \lor \cdots \lor m_n \]
\[ (\ell_1 \lor \cdots \lor \ell_{i-1} \lor \ell_{i+1} \lor \cdots \lor \ell_k \lor m_1 \lor \cdots \lor m_{j-1} \lor m_{j+1} \lor \cdots \lor m_n)\theta \]

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

For example,

\[ \neg\text{Rich}(x) \lor \text{Unhappy}(x), \quad \text{Rich}(\text{Ken}) \]
\[ \text{Unhappy}(\text{Ken}) \]

with \( \theta = \{x \leftarrow \text{Ken}\} \)

To prove that \( KB \models \text{an instance of } \alpha \), convert \( KB \land \neg\alpha \) to CNF and do resolution repeatedly.

This is a complete proof procedure for FOL.

If there’s a substitution \( \theta \) such that \( KB \models \theta\alpha \), then it will return \( \theta \).

If there’s no such \( \theta \), then the procedure won’t necessarily terminate.
Conversion to CNF

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

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 \(\neg\) inwards:
\[\forall x, p \equiv \exists x \neg p, \ \neg \exists x, p \equiv \forall x \neg p:\]
\[\forall x \ [\exists y \neg (\neg Animal(y) \lor Loves(x, y))] \lor [\exists y \ Loves(y, x)]\]
\[\forall x \ [\exists y \neg \neg Animal(y) \land \neg Loves(x, y)] \lor [\exists y \ Loves(y, x)]\]
\[\forall x \ [\exists y \ Animal(y) \land \neg Loves(x, y)] \lor [\exists y \ Loves(y, x)]\]
Conversion to CNF, continued

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

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

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

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

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 $\lor$:

$$[\text{Animal}(F(x)) \lor \text{Loves}(G(x), x)] \land [\neg \text{Loves}(x, F(x)) \lor \text{Loves}(G(x), x)]$$
Resolution proof: definite clauses

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

\[ \neg \text{Criminal}(\text{West}) \]

\[ \neg \text{American}(\text{West}) \lor \neg \text{Weapon}(y) \lor \neg \text{Sells}(\text{West},y,z) \lor \neg \text{Hostile}(z) \]

\[ \neg \text{Weapon}(y) \lor \neg \text{Sells}(\text{West},y,z) \lor \neg \text{Hostile}(z) \]

\[ \neg \text{Missile}(x) \lor \text{Weapon}(x) \]

\[ \neg \text{Missile}(x) \lor \neg \text{Owns}(\text{Nono},x) \lor \text{Sells}(\text{West},x,\text{Nono}) \]

\[ \neg \text{Sells}(\text{West},\text{M1},z) \lor \neg \text{Hostile}(z) \]

\[ \neg \text{Sells}(\text{West},\text{M1},z) \lor \neg \text{Hostile}(z) \]

\[ \neg \text{Missile}(\text{M1}) \lor \neg \text{Owns}(\text{Nono},\text{M1}) \lor \neg \text{Hostile}(\text{Nono}) \]

\[ \neg \text{Owns}(\text{Nono},\text{M1}) \lor \neg \text{Hostile}(\text{Nono}) \]

\[ \neg \text{Enemy}(\text{Nono},\text{America}) \lor \text{Hostile}(\text{Nono}) \]

\[ \neg \text{Hostile}(\text{Nono}) \]

The figure omits all resolvents except for the ones in the proof
Homework

Problems 20.11, 20.17, 9.12 and 9.19

10 points each, 40 points total

Due in one week (i.e., April 6)