Inference in first-order logic

Russell and Norvig Chapter 9
Outline

- Reducing first-order inference to propositional inference
- Generalized Modus Ponens
- Forward chaining
- Backward chaining
- Resolution
First order inference can be done by converting the knowledge base to PL and using propositional inference.

- How to convert universal quantifiers?
  - Replace variable by ground term.
- How to convert existential quantifiers?
  - Skolemization.
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\((x) \land \) Greedy\((x) \Rightarrow \) Evil\((x) \) yields:

- King\((\text{John}) \land \) Greedy\((\text{John}) \Rightarrow \) Evil\((\text{John}) \)
- King\((\text{Father} (\text{John})) \land \) Greedy\((\text{Father} (\text{John})) \Rightarrow \) 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., $\exists x \ Crown(x) \land OnHead(x, John)$ yields:

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

provided $C_1$ is a new constant symbol, called a Skolem constant
EI versus UI

- 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.
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:

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

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

The new KB is propositionalized
Reduction (cont)

- **CLAIM**: A ground sentence is entailed by the new KB iff entailed by the 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, there are infinitely many ground terms,
  - e.g., \(\text{Father}(\text{Father}(\text{Father}(\text{John})))\)
  - In our natural numbers example: \(\text{NatNum}(S(0)), \text{NatNum}(S(S(0)))\)…

*The question of entailment for FOL is semidecidable: no algorithm exists that says no to every nonentailed sentence.*
Reduction (cont)

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

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

- **PROBLEM**: works if $\alpha$ is entailed, does not halt if $\alpha$ is not entailed.

- **THEOREM**: Turing (1936), Church (1936) Entailment for FOL is semi decidable
  - algorithms exist that say yes to every entailed sentence, but no algorithm exists that also says no to every non-entailed sentence.

- With $p$ $k$-ary predicates and $n$ constants, there are $p \cdot n^k$ instantiations!
Is there another way?

- Instead of translating the knowledge base to PL, we can make the inference rules work in FOL.
- For example, given
  \[ \forall x \ King(x) \land Greedy(x) \implies Evil(x) \]
  \[ King(John) \]
  \[ \forall y \ Greedy(y) \]
  It is intuitively clear that we can substitute \{x/John,y/John\} and obtain that Evil(John)
Unification

- We can make the inference if we can find a substitution such that $\text{King}(x)$ and $\text{Greedy}(x)$ match $\text{King}(\text{John})$ and $\text{Greedy}(y)$
  
  \[
  \{x/\text{John}, y/\text{John}\} \text{ works}
  \]

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

<table>
<thead>
<tr>
<th>$\alpha$</th>
<th>$\beta$</th>
<th>Subst</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>$\text{Knows}(y,\text{OJ})$</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

Unify(α, β) = θ if Subst(θ, α) = Subst(θ, β)

<table>
<thead>
<tr>
<th>α</th>
<th>β</th>
<th>Subst</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></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**

- Unify(α, β) = θ if Subst(θ, α) = Subst(θ, β)

<table>
<thead>
<tr>
<th>α</th>
<th>β</th>
<th>Subst</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,Mother(y))</td>
<td></td>
</tr>
<tr>
<td>Knows(John,x)</td>
<td>Knows(x,OJ)</td>
<td></td>
</tr>
</tbody>
</table>
Unification

- **Unify(α, β) = θ if Subst(θ, α) = Subst(θ, β)**

<table>
<thead>
<tr>
<th>α</th>
<th>β</th>
<th>Subst</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,Mother(y))</td>
<td>{y/John,x/Mother(John)}</td>
</tr>
<tr>
<td>Knows(John,x)</td>
<td>Knows(x,OJ)</td>
<td></td>
</tr>
</tbody>
</table>
Unification

- Unify($\alpha$, $\beta$) = $\theta$ if Subst($\theta$, $\alpha$) = Subst($\theta$, $\beta$)

<table>
<thead>
<tr>
<th>$\alpha$</th>
<th>$\beta$</th>
<th>Subst</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,Mother(y))</td>
<td>${y$/John,$x$/Mother(John)$}$</td>
</tr>
<tr>
<td>Knows(John,x)</td>
<td>Knows(x,OJ)</td>
<td>${\text{fail}}$</td>
</tr>
</tbody>
</table>
Unification

- Unifiers of $\text{Knows}(John,x)$ and $\text{Knows}(y,z)$ are 
  \{y/John, x/z \} or \{y/John, x/John, z/John\}

- The first unifier is more general than the second.

- There is a single most general unifier (MGU) that is unique up to renaming of variables.
  
  \[ \text{MGU} = \{ y/John, x/z \} \]
The unification algorithm

function Unify\( (x, y, \theta) \) returns a substitution to make \( x \) and \( y \) identical

inputs: \( x \), a variable, constant, list, or compound
\( y \), a variable, constant, list, or compound
\( \theta \), the substitution built up so far

if \( \theta = \text{failure} \) then return failure
else if \( x = y \) then return \( \theta \)
else if \( \text{VARIABLE?}(x) \) then return Unify-Var\( (x, y, \theta) \)
else if \( \text{VARIABLE?}(y) \) then return Unify-Var\( (y, x, \theta) \)
else if \( \text{COMPOUND?}(x) \) and \( \text{COMPOUND?}(y) \) then
    return Unify\( (\text{ARGS}[x], \text{ARGS}[y], \text{Unify}(\text{OP}[x], \text{OP}[y], \theta)) \)
else if \( \text{LIST?}(x) \) and \( \text{LIST?}(y) \) then
    return Unify\( (\text{REST}[x], \text{REST}[y], \text{Unify}(\text{FIRST}[x], \text{FIRST}[y], \theta)) \)
else return failure
The unification algorithm

```
function UNIFY-VAR(var, x, θ) returns a substitution
    inputs: var, a variable
            x, any expression
            θ, the substitution built up so far
    if {var/val} ∈ θ then return UNIFY(val, x, θ)
    else if {x/val} ∈ θ then return UNIFY(var, val, θ)
    else if OCCUR-CHECK?(var, x) then return failure
    else return add {var/x} to θ
```
Generalized Modus Ponens (GMP)

Suppose that $\text{Subst}(\theta, p_i') = \text{Subst}(\theta, p_i)$ for all $i$ then:

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

\[
\text{Subst}(\theta, q)
\]

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

- All variables assumed universally quantified.
Example

- 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

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

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

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

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

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

Nono ... has some missiles
Example

... it is a crime for an American to sell weapons to hostile nations:
\[
\text{American}(x) \land \text{Weapon}(y) \land \text{Sells}(x,y,z) \land \text{Hostile}(z) \Rightarrow \text{Criminal}(x)
\]

Nono … has some missiles, i.e., \(\exists x \text{ Owns}(\text{Nono},x) \land \text{Missile}(x)\):
\[
\text{Owns}(\text{Nono},M_1) \land \text{Missile}(M_1)
\]
Example

... it is a crime for an American to sell weapons to hostile nations:
\[ \text{American}(x) \land \text{Weapon}(y) \land \text{Sells}(x,y,z) \land \text{Hostile}(z) \Rightarrow \text{Criminal}(x) \]

Nono ... has some missiles, i.e., \( \exists x \) \( \text{Owns}(\text{Nono},x) \land \text{Missile}(x) \):
\[ \text{Owns}(\text{Nono},M_1) \land \text{Missile}(M_1) \]

... all of its missiles were sold to it by Colonel West
... it is a crime for an American to sell weapons to hostile nations:
\[\text{American}(x) \land \text{Weapon}(y) \land \text{Sells}(x,y,z) \land \text{Hostile}(z) \Rightarrow \text{Criminal}(x)\]

Nono … has some missiles, i.e., \(\exists x \text{ Owns}(\text{Nono},x) \land \text{Missile}(x)\):
\[\text{Owns}(\text{Nono},M_1) \land \text{Missile}(M_1)\]

… all of its missiles were sold to it by Colonel West
\[\text{Missile}(x) \land \text{Owns}(\text{Nono},x) \Rightarrow \text{Sells}(\text{West},x,\text{Nono})\]
Example

... it is a crime for an American to sell weapons to hostile nations:
\[American(x) \land Weapon(y) \land Sells(x,y,z) \land Hostile(z) \Rightarrow Criminal(x)\]

Nono … has some missiles, i.e., \(\exists x\) Owns(Nono,x) \& Missile(x):
\[Owns(Nono,M_1) \land Missile(M_1)\]

… all of its missiles were sold to it by Colonel West
\[Missile(x) \land Owns(Nono,x) \Rightarrow Sells(West,x,Nono)\]

Missiles are weapons:
Example

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

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

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

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

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

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

Missiles are weapons:

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

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

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

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

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

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

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

Missiles are weapons:

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

An enemy of America counts as "hostile":
Example

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

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

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

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

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

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

Missiles are weapons:

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

An enemy of America counts as "hostile“:

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

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

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

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

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

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

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

Missiles are weapons:

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

An enemy of America counts as "hostile“:

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

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

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

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

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

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

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

Missiles are weapons:

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

An enemy of America counts as "hostile“:

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

West, who is American ...

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

... it is a crime for an American to sell weapons to hostile nations:
\[\text{American}(x) \land \text{Weapon}(y) \land \text{Sells}(x,y,z) \land \text{Hostile}(z) \Rightarrow \text{Criminal}(x)\]

Nono … has some missiles, i.e., \(\exists x \ \text{Owns}(\text{Nono},x) \land \text{Missile}(x)\):
\[\text{Owns}(\text{Nono},M_1) \land \text{Missile}(M_1)\]

… all of its missiles were sold to it by Colonel West
\[\text{Missile}(x) \land \text{Owns}(\text{Nono},x) \Rightarrow \text{Sells}(\text{West},x,\text{Nono})\]

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

An enemy of America counts as "hostile“:
\[\text{Enemy}(x,\text{America}) \Rightarrow \text{Hostile}(x)\]

West, who is American …
\[\text{American}(\text{West})\]

The country Nono, an enemy of America …
Example

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

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

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

\[ \text{Ow}(Nono,M_1) \land \text{Missile}(M_1) \]

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

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

Missiles are weapons:

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

An enemy of America counts as "hostile“:

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

West, who is American ...

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

The country Nono, an enemy of America ...

\[ \text{Enemy}(Nono,America) \]
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
            (p_1 ∧ ... ∧ p_n ⇒ q) ← STANDARDIZE-APART(r)
            for each θ such that (p_1 ∧ ... ∧ p_n)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 example

✧ American(\(x\)) \land Weapon(\(y\)) \land 
Sells(\(x,y,z\)) \land Hostile(\(z\)) \Rightarrow 
Criminal(\(x\)) \\
✧ Owns(Nono,\(M_1\)) \land Missile(\(M_1\)) \\
✧ Missile(\(x\)) \land Owns(Nono,\(x\)) \Rightarrow 
Sells(West,\(x,Nono\)) \\
✧ Missile(\(x\)) \Rightarrow Weapon(\(x\)) \\
✧ Enemy(\(x,America\)) \Rightarrow Hostile(\(x\)) \\
✧ American(West) \\
✧ Enemy(Nono,America)
Forward chaining example

- American(x) \land Weapon(y) \land 
  Sells(x,y,z) \land Hostile(z) \Rightarrow 
  Criminal(x)
- Owns(Nono,M_1) \land Missile(M_1)
- Missile(x) \land Owns(Nono,x) \Rightarrow 
  Sells(West,x,Nono)
- Missile(x) \Rightarrow Weapon(x)
- Enemy(x,America) \Rightarrow Hostile(x)
- American(West)
- Enemy(Nono,America)
Forward chaining example

- \( \text{American}(x) \land \text{Weapon}(y) \land \text{Sells}(x,y,z) \land \text{Hostile}(z) \Rightarrow \text{Criminal}(x) \)
- \( \text{Owns}(\text{Nono}, M_1) \land \text{Missile}(M_1) \)
- \( \text{Missile}(x) \land \text{Owns}(\text{Nono}, x) \Rightarrow \text{Sells}(\text{West}, x, \text{Nono}) \)
- \( \text{Missile}(x) \Rightarrow \text{Weapon}(x) \)
- \( \text{Enemy}(x, \text{America}) \Rightarrow \text{Hostile}(x) \)
- \( \text{American}(\text{West}) \)
- \( \text{Enemy}(\text{Nono}, \text{America}) \)

Diagram:

```
American(West) -> Weapon(M1) -> Sells(West,M1,Nono) -> Hostile(Nono) -> Criminal(West)
```

\[
\begin{align*}
\text{American}(\text{West}) & \\
\text{Missile}(M1) & \\
\text{Owns}(\text{Nono}, M1) & \\
\text{Enemy}(\text{Nono}, \text{America}) &
\end{align*}
\]
Forward chaining for FOL

- Sound and complete for first-order definite clauses.
- **Datalog** = first-order definite clauses with *no functions* (e.g. crime KB)
  - FC terminates for Datalog in finite number of iterations

- May not terminate in general definite clauses with functions. E.g. in the natural number example can generate an infinite number of entailed facts:

  \[
  \text{NatNum}(0) \\
  \forall n \text{ NatNum}(n) \implies \text{NatNum}(S(n))
  \]
Backward chaining

- As in propositional logic, can work backwards from the goal
  - Requires a generator that tries multiple bindings
  - Depth-first recursive proof search
- Widely used for logic programming: problem solving by inference.
  - Example: Prolog
Backward chaining example

Criminal(West)
Backward chaining example

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

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

```
Criminal(West)

American(West)
{ }

Weapon(y)

Sells(x,y,z)

Hostile(z)

Missile(y)
{ y/ML }
```

{x/West, y/ML}
Backward chaining example

\[ \text{Missile}(x) \land \text{Owns}(\text{Nono}, x) \Rightarrow \text{Sells}(\text{West}, x, \text{Nono}) \]
Backward chaining example
Backward chaining algorithm

function FOL-BC-ASK(KB, query) returns a generator of substitutions
return FOL-BC-OR(KB, query, {})

generator FOL-BC-OR(KB, goal, θ) yields a substitution
for each rule (lhs ⇒ rhs) in FETCH-RULES-FOR-GOAL(KB, goal) do
  (lhs, rhs) ← STANDARDIZE-VARIABLES((lhs, rhs))
  for each θ’ in FOL-BC-AND(KB, lhs, UNIFY(rhs, goal, θ)) do
    yield θ’

generator FOL-BC-AND(KB, goals, θ) yields a substitution
if θ = failure then return
else if length(goals) = 0 then yield θ
else do
  first, rest ← FIRST(goals), REST(goals)
  for each θ’ in FOL-BC-OR(KB, SUBST(θ, first), θ) do
    for each θ” in FOL-BC-AND(KB, SUBST(θ’, rest), θ’) do
      yield θ”

There may be multiple relevant substitutions, so the functions are generators
Properties of backward chaining

Depth-first recursive proof search
- space is linear in size of proof.
- Incomplete due to infinite loops
  - Fixable by checking current goal against every goal on stack
- Inefficient due to repeated subgoals
  - Fixable by caching previous results (extra space!!)
- Widely used for logic programming: problem solving by inference.
  - Example: Prolog
Logic programming: Prolog

- BASIS: backward chaining with Horn clauses + bells & whistles
- Program = set of clauses of the form
  \[\text{head} :- \text{literal}_1, \ldots \text{literal}_n.\]

  \[
  \text{criminal}(X) :- \text{american}(X), \text{weapon}(Y), \text{sells}(X,Y,Z), \text{hostile}(Z).
  \]
Prolog is a ‘declarative’ language

- Clauses are statements about what is true about a problem, instead of instructions how to accomplish the solution.
- The Prolog system uses the clauses to work out how to accomplish the solution by searching through the space of possible solutions.
Example

- Prolog program consists of facts and rules.
  
  animal(lion).
  animal(sparrow).
  hasfeathers(sparrow).
  bird(X) :- animal(X), hasfeathers(X).

- “Run” by asking questions or queries. Or (using logic terminology) by setting a goal for Prolog to try to prove:
  
  ?- bird(sparrow).
  yes

- Or to find a value of a variable that makes it true:
  
  ?- bird(What).
  What = sparrow
Example

- 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=\[]
  \]
Example

path(X, Z) :- link(X, Z).

path(X, Z) :- path(X, Y), link(Y, Z).
Infinite loops

Proof that a path exists from a to c:

What goes wrong when the clauses are in the wrong order
Permutations

- Permutation(X, Y) is true whenever Y is a permutation of X.

?-permutation([a,b,c], P).
P = [a,b,c];
P = [a,c,b];
P = [b,a,c];

Two cases:
- The only permutation of the empty list is the empty list.
- If the first list is not empty, the it has the form [X|L], and a permutation of such a list can be constructed by first permuting L and then inserting X at any position into the permuted list.

permutation([], []).  
permutation([X|L], P) :-
    permutation(L, L1), insert(X, L1, P).
Resolution

- Full first-order version:
  \[
  l_1 \lor \cdots \lor l_k, \quad m_1 \lor \cdots \lor m_n \\
  \]

  \[
  \text{Subst}(\theta, l_1 \lor \cdots \lor l_{i-1} \lor l_{i+1} \lor \cdots \lor l_k \lor m_1 \lor \cdots \lor m_{j-1} \lor m_{j+1} \lor \cdots \lor m_n)
  \]

  where \( \theta = Unify(l_i, \neg m_j) \)

- The two clauses are assumed to be standardized apart so that they share no variables.

- For example,
  \[
  \neg Rich(x) \lor Unhappy(x), \quad Rich(Ken) \\
  \]

  \[
  Unhappy(Ken)
  \]

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

- Apply resolution steps to \( \text{CNF}(KB \land \neg \alpha) \); complete for FOL
Conversion to CNF

- Need to convert KB to CNF, eliminating existential quantifiers
- Consider the sentence “There is someone who is loved by everyone”:
  \[ \exists y \forall x \text{Loves}(x, y) \]
  Let’s name that someone using a constant that does not appear elsewhere in the KB (Skolem constant):
  \[ \forall x \text{Loves}(x, \text{Sk1}) \]
- Let’s try now the sentence “Everyone is loved by someone”
  \[ \forall y \exists x \text{Loves}(x, y) \]
  Consider the following Skolemization:
  \[ \forall y \text{Loves}(\text{SK2}, y) \]
Conversion to CNF

- Need to convert KB to CNF, eliminating existential quantifiers
- Consider the sentence “There is someone who is loved by everyone”:
  \[ \exists y \forall x \text{ Loves}(x, y) \]
  Let’s name that someone using a constant that does not appear elsewhere in the KB (Skolem constant):
  \[ \forall x \text{ Loves}(x, \text{Sk}_1) \]
- Let’s try now the sentence “Everyone is loved by someone”
  \[ \forall y \exists x \text{ Loves}(x, y) \]
  Consider the following Skolemization:
  \[ \forall y \text{ Loves}(\text{SK}_2, y) \]
- This doesn’t work, but Skolemization using a function does:
  \[ \forall y \text{ Loves}(\text{SK}_2(y), y) \]
Conversion to CNF

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

- Eliminate implications (and biconditionals)
  \[ \forall x ([\neg (\forall y \neg \text{Animal}(y) \lor \text{Loves}(x,y))] \lor [\exists y \text{Loves}(y,x)]) \]

- Move \( \neg \) inwards:
  \[ \neg \forall x p \equiv \exists x \neg p, \ \neg \exists x p \equiv \forall x \neg p \]
  \[ \forall x [\exists y \neg (\neg \text{Animal}(y) \lor \text{Loves}(x,y))] \lor [\exists y \text{Loves}(y,x)] \]
  \[ \forall x [\exists y \neg \neg \text{Animal}(y) \land \neg \text{Loves}(x,y)] \lor [\exists y \text{Loves}(y,x)] \]
  \[ \forall x [\exists y \text{Animal}(y) \land \neg \text{Loves}(x,y)] \lor [\exists y \text{Loves}(y,x)] \]
Conversion to CNF

- Standardize variables: each quantifier should use a different one:
  \[ \forall x \ [ \exists y \ Animal(y) \land \neg Loves(x,y)] \lor [\exists z \ Loves(z,x)] \]

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

- Drop universal quantifiers:
  \[ [Animal(F(x)) \land \neg Loves(x,F(x))] \lor Loves(G(x),x) \]

- Distribute \( \lor \) over \( \land \):
  \[ [Animal(F(x)) \lor Loves(G(x),x)] \land [\neg Loves(x,F(x)) \lor Loves(G(x),x)] \]
Resolution proof

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

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

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

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

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

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

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

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

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

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

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

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

\[ \text{Enemy}(\text{Nono},\text{America}) \]
Connection with CSP

- Colorable() is inferred iff the CSP has a solution
- Hardness of inference problem related to the difficulty of the corresponding CSP.

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

\[
\text{Diff}(\text{Red},\text{Blue}) \quad \text{Diff}(\text{Red},\text{Green}) \\
\text{Diff}(\text{Green},\text{Red}) \quad \text{Diff}(\text{Green},\text{Blue}) \\
\text{Diff}(\text{Blue},\text{Red}) \quad \text{Diff}(\text{Blue},\text{Green})
\]
Theorem provers

- Theorem prover – a system that does full first order logic inference (using resolution)

- Uses:
  - Prove mathematical theorems (known successes!)
  - Hardware/software verification
    - Verify that circuit/software produces correct output for all possible inputs (the RSA algorithm was verified this way)