Reasoning with First-Order Logic

Chaining

Resolution

Compilation to SAT
FOL Reasoning: Motivation

- **What if we want to use modus ponens?**
  
  Propositional Logic:
  \[ a \land b, \quad a \land b \implies c \]
  \[\hline\]
  \[ c \]

- **In First-Order Logic?**
  
  \[ \forall x \text{ Monkey}(x) \implies \text{Curious}(x) \]
  \[ \text{Monkey}(George) \]
  \[ \hline \]
  \[ ??? \]

- **Must "unify" \( x \) with George:**
  
  Need to substitute \( \{x/George\} \) in \( \text{Monkey}(x) \implies \text{Curious}(x) \) to infer \( \text{Curious}(George) \)
What is Unification?

Not this kind of unification...
What is Unification?

• Match up expressions by finding variable values that make the expressions identical

Unify city(x) and city(seattle) using \{x/seattle\}

• Unify(a, b) returns most general unifier (MGU)
Most General Unifier

- **Unify** \((a, b)\) returns most general unifier (MGU)
- MGU places fewest restrictions on values of variables
- Examples:
  - \(\text{Unify}(\text{city}(x), \text{city}(\text{seattle}))\) returns \(\{x/\text{seattle}\}\)
  - \(\text{Unify}(\text{PokesInTheEyes}(\text{Moe}, x), \text{PokesInTheEyes}(y, z))\) returns \(\{y/\text{Moe}, z/x\}\)
  - \(\{y/\text{Moe}, x/\text{Moe}, z/\text{Moe}\}\) also possible but not MGU
Unification and Substitution

• **Unification** produces a mapping from variables to values (e.g., \{x/seattle,y/tacoma\})

• **Substitution**: \text{Subst}(\text{mapping}, \text{sentence}) returns new sentence with variables replaced by values

  \text{Subst}\{\{x/seattle,y/tacoma\}\},\ \text{connected}(x, y)),

  returns \text{connected}(seattle, Tacoma)
Unification Examples I

- **Unify(road(x, kent), road(seattle, y))**
  
  Returns \( \{x / \text{seattle}, \ y / \text{kent} \} \)
  
  When substituted in both expressions, the resulting expressions match:
  
  Each is \( \text{(road(seattle, kent))} \)

- **Unify(road(x, x), road(seattle, kent))**
  
  Not possible - Fails!
  
  \( x \) can’t be seattle and kent at the same time!
Unification Examples II

• Unify($f(g(x, \text{dog}), y))$, $f(g(\text{cat}, y), \text{dog})$
  
  \{x / \text{cat}, y / \text{dog}\}

• Unify($f(g(x))$, $f(x)$)

  Fails: no substitution makes them identical.  
  E.g.  \{x / g(x) \} yields $f(g(g(x)))$ and $f(g(x))$ which are not identical!

• Thus: A variable may not \textit{contain} itself in a substitution

  Directly or indirectly
Unification Examples III

• \text{Unify}(f(g(\text{cat}, y), y), f(x, \text{dog}))
  \{x / g(\text{cat}, \text{dog}), y / \text{dog}\}

• \text{Unify}(f(g(y)), f(x))
  \{x / g(y)\}

• Back to curious monkeys:

\[
\begin{align*}
\text{Monkey}(x) &\rightarrow \text{Curious}(x) \\
\text{Monkey}(\text{George}) &\rightarrow \text{Curious}(\text{George})
\end{align*}
\]

Unify and then use modus ponens =
\textit{generalized modus ponens} (GMP)
(“Lifted” version of modus ponens)
Inference I: Forward Chaining

• The algorithm:
  Start with the KB
  Add any fact you can generate with GMP (i.e., unify expressions and use modus ponens)
  Repeat until: goal reached or generation halts
Example

• 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.

• Is Col. West a criminal?

• KB of definite clauses (exactly 1 positive literal):

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

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

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

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

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

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

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

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

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

Initial facts in KB

\[
\begin{array}{c}
\text{American}(\text{West}) \\
\text{Missile}(\text{M}1) \\
\text{Owns}(\text{Nono}, \text{M}1) \\
\text{Enemy}(\text{Nono}, \text{America})
\end{array}
\]
Forward chaining example

1. $\text{Missile}(x) \Rightarrow \text{Weapon}(x)$
2. $\text{Missile}(x) \land \text{Owns}(\text{Nono}, x) \Rightarrow \text{Sells}(\text{West}, x, \text{Nono})$
3. $\text{Enemy}(x, \text{America}) \Rightarrow \text{Hostile}(x)$

Facts inferred after 1$^{st}$ iteration
Forward chaining example

American(x) ∧ Weapon(y) ∧ Sells(x, y, z) ∧ Hostile(z) ⇒ Criminal(x)

Facts inferred after 1\(^{st}\) iteration
Forward chaining example

Col. West is a criminal

\{x/\text{West}, y/M_1, z/\text{Nono}\}
Inference I: Forward Chaining

- **Sound? Complete? Decidable?**
  Yes; yes for definite KB; no (see p. 331 in text)

- **Speed concerns?** Inefficiencies due to:
  Unification via exhaustive pattern matching; premise rechecking on every iteration; irrelevant fact generation.
  (see Section 9.3.3 for strategies to increase speed)
Inference II: Backward Chaining

• The algorithm:
  Start with KB and goal.
  Find all rules whose results unify with goal:
    Add the premises of these rules to the goal list
    Remove the corresponding result from the goal list
  Stop when:
    Goal list is empty (SUCCEED) or
    Progress halts (FAIL)
Backward chaining example

Goal

Criminal(West)
Backward chaining example

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

\[ \text{Criminal}(\text{West}) \quad \{x/\text{West}\} \]
Backward chaining example

Depth-first traversal

New Subgoal

KB has:
Missile(y) ⇒ Weapon(y)
Missile(M_1)
Backward chaining example

KB has:
\[ \text{Missile}(y) \land \text{Owns}(\text{Nono}, y) \Rightarrow \text{Sells}(\text{West}, y, \text{Nono}) \]
\[ \text{Missile}(M_1) \]
\[ \text{Owns}(\text{Nono}, M_1) \]

New Subgoal
\[ \{z/?\} \]
Backward chaining example

KB has:
Enemy(z, America) ⇒ Hostile(z)
Enemy(Nono, America)
Backward chaining example

Criminal(West)

{ x/West, y/MI, z/Nono }

American(West)

{ }

Weapon(y)

{ }

Sells(West, MI, z)

{ z/Nono }

Hostile(Nono)

{ }
Properties of backward chaining

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

• Incomplete due to infinite loops (e.g. repeated states)
  ⇒ fix by checking current goal against goals on stack
  ⇒ Can’t fix infinite paths though (similar to DFS)

• Inefficient due to repeated computations
  ⇒ fix using caching of previous results (extra space)

• Widely used for logic programming
  E.g., Prolog (logic programming language) – see Section 9.4 in text
Inference III: Resolution

\{ (p \lor q), (\neg p \lor r \lor s) \} \models_R (q \lor r \lor s)

Recall Propositional Case:
• Literal in one clause
• Its negation in the other
• Result is disjunction of other literals
First-Order Resolution
[Robinson 1965]

\[ \{ (p(x) \lor q(A), \neg p(B) \lor r(x) \lor s(y)) \} \]

\[ \vdash R \]

\[ (q(A) \lor r(B) \lor s(y)) \]

- Literal in one clause
- Negation of something which unifies in other
- Result is disjunction of all other literals with substitution based on MGU

Substitute MGU \{x/B\} in all literals
Inference using First-Order Resolution

• As before, use “proof by contradiction”
  To show \( KB \models \alpha \), show \( KB \land \neg \alpha \) unsatisfiable

• Method
  Let \( S = KB \land \neg \text{goal} \)
  Convert \( S \) to clausal form
  • Standardize variables (replace \( x \) in all with \( y, z, x_1, \ldots \))
  • Move quantifiers to front, skolemize to remove \( \exists \)
  • Replace \( \Rightarrow \) with \( \lor \) and \( \neg \)
  • Use deMorgan’s laws to get CNF (and of ors)
  Resolve clauses in \( S \) until empty clause (unsatisfiable) or no new clauses added
Next Time

- Wrap up of FOL
- FOL Wumpus Agent
- To Do
  
  Project #2 due this Saturday NOON
  Read chapter 9