Forward Chaining

**Remember this from propositional logic?**

- Start with atomic sentences in KB
- Apply Modus Ponens
  - add new sentences to KB
  - discontinue when no new sentences
- Hopefully find the sentence you are looking for in the generated sentences
Lifting forward chaining

First-order definite clauses

• all sentences are defined this way to simplify processing
  – disjunction of literals with exactly one positive
  – clause is either atomic or an implication whose antecedent
    is a conjunction of positive literals and whose consequent
    is a single positive literal

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

- 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, and all of its missiles were sold to it by Colonel West, who is American.

- We will prove West is a criminal.
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
  - Owns (Nono, M1)
  - Missile (M1)

- 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

- We also need to know that missiles are weapons

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

- and we must know that 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

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

**Starting from the facts**

- find all rules with satisfied premises
- add their conclusions to known facts
- repeat until
  - query is answered
  - no new facts are added
First iteration of forward chaining

Look at the implication sentences first

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

- must satisfy unknown premises
- We can satisfy this rule

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

- by substituting \( \{x/M1\} \)
- and adding \( Sells(West, M1, Nono) \) to KB
First iteration of forward chaining

- We can satisfy
  \[ \text{Missile}(x) \Rightarrow \text{Weapon}(x) \]
  - with \( \{x/\text{M1}\} \)
  - and \( \text{Weapon (M1)} \) is added

- We can satisfy
  \[ \text{Enemy}(x, \text{America}) \Rightarrow \text{Hostile}(x) \]
  - with \( \{x/\text{Nono}\} \)
  - and \( \text{Hostile \{Nono\}} \) is added
Second iteration of forward chaining

- We can satisfy

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

- with \{x/West, y/M1, z/Nono\}

- and Criminal (West) is added

---

Figure 9.4  The proof tree generated by forward chaining on the crime example. The initial facts appear at the bottom level, facts inferred on the first iteration in the middle level, and facts inferred on the second iteration at the top level.
Analyze this algorithm

**Sound?**
- Does it only derive sentences that are entailed?
- Yes, because only Modus Ponens is used and it is sound

**Complete?**
- Does it answer every query whose answers are entailed by the KB?
- Yes if the clauses are definite clauses
Complexity of this algorithm

**Three sources of complexity**

- inner loop requires finding all unifiers such that premise of rule unifies with facts of database
  - this "pattern matching" is expensive
- must check every rule on every iteration to check if its premises are satisfied
- many facts are generated that are irrelevant to goal
Pattern matching

Conjunct ordering

• Missile (x) ^ Owns (Nono, x) => Sells (West, x, Nono)
  – Look at all items owned by Nono, call them X
  – for each element x in X, check if it is a missile

  – Look for all missiles, call them X
  – for each element x in X, check if it is owned by Nono
Incremental forward chaining

**Pointless (redundant) repetition**

- Some rules generate new information
  - this information may permit unification of existing rules
- Some rules generate preexisting information
  - we need not revisit the unification of the existing rules

*Every new fact inferred on iteration t must be derived from at least one new fact inferred on iteration t-1*
Irrelevant facts

Some facts are irrelevant and occupy computation of forward-chaining algorithm

• What if Nono example included lots of facts about food preferences?
  – Not related to conclusions drawn about sale of weapons
  – How can we eliminate them?
    ▪ Backward chaining is one way
Magic Set

Rewriting the rule set

• Sounds dangerous

• Add elements to premises that restrict candidates that will match
  – added elements are based on desired goal

• Let goal = Criminal (West)
  – Magic(x) ^ American(x) ^ Weapon(y) ^ Sells(x, y, z) ^ Hostile(z) => Criminal (x)
  – Add Magic (West) to Knowledge Base
Backward Chaining

Start with the premises of the goal

- Each premise must be supported by KB
- Start with first premise and look for support from KB
  - looking for clauses with a head that matches premise
  - the head’s premise must then be supported by KB

A recursive, depth-first, algorithm

- Suffers from repetition and incompleteness
Resolution

We saw earlier that resolution is a complete algorithm for refuting statements

- Must put first-order sentences into conjunctive normal form
  - conjunction of clauses, each is a disjunction of literals
    - literals can contain variables (which are assumed to be universally quantified)
First-order CNF

- For all x, American(x) \land Weapon(y) \land Sells(x, y, z) \land Hostile(z) \Rightarrow Criminal(x)

- \neg American(x) \lor \neg Weapon(y) \lor \neg Sells(x, y, z) \lor \neg Hostile(z) \lor Criminal(x)

*Every sentence of first-order logic can be converted into an inferentially equivalent CNF sentence*
Everyone who loves all animals is loved by someone

∀x [∀y Animal(y) ⇒ Loves(x, y)] ⇒ [∃y Loves(y, x)]

◊ Eliminate implications:
∀x [¬∀y ¬Animal(y) ∨ Loves(x, y)] ∨ [∃y Loves(y, x)].

◊ Move ¬ inwards: In addition to the usual rules for negated connectives, we need rules for negated quantifiers. Thus, we have

¬∀x p becomes ∃x ¬p
¬∃x p becomes ∀x ¬p.

Our sentence goes through the following transformations:

∀x [∃y ¬(¬Animal(y) ∨ Loves(x, y))] ∨ [∃y Loves(y, x)].
∀x [∃y ¬¬Animal(y) ∧ ¬Loves(x, y)] ∨ [∃y Loves(y, x)].
∀x [∃y Animal(y) ∧ ¬Loves(x, y)] ∨ [∃y Loves(y, x)].

Notice how a universal quantifier (∀y) in the premise of the implication has become an existential quantifier. The sentence now reads “Either there is some animal that x doesn’t love, or (if this is not the case) someone loves x.” Clearly, the meaning of the original sentence has been preserved.
Example

**Standardize variables:** For sentences like \((\forall x \ P(x)) \lor (\exists x \ Q(x))\) which use the same variable name twice, change the name of one of the variables. This avoids confusion later when we drop the quantifiers. Thus, we have

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

**Skolemize:** Skolemization is the process of removing existential quantifiers by elimination. In the simple case, it is just like the Existential Instantiation rule of Section 9.1: translate \(\exists x \ P(x)\) into \(P(A)\), where \(A\) is a new constant. If we apply this rule to our sample sentence, however, we obtain

\[
\forall x \ [Animal(A) \land \neg Loves(x, A)] \lor Loves(B, x)
\]

which has the wrong meaning entirely: it says that everyone either fails to love a particular animal \(A\) or is loved by some particular entity \(B\). In fact, our original sentence allows each person to fail to love a different animal or to be loved by a different person. Thus, we want the Skolem entities to depend on \(x\):

\[
\forall x \ [Animal(F(x)) \land \neg Loves(x, F(x))] \lor Loves(G(x), x).
\]
Example

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

**F and G are Skolem Functions**

- arguments of function are universally quantified variables in whose scope the existential quantifier appears
Example

Drop universal quantifiers: At this point, all remaining variables must be universally quantified. Moreover, the sentence is equivalent to one in which all the universal quantifiers have been moved to the left. We can therefore drop the universal quantifiers:

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

Distribute $\land$ over $\lor$:

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

This step may also require flattening out nested conjunctions and disjunctions.

- Two clauses
- $F(x)$ refers to the animal potentially unloved by $x$
- $G(x)$ refers to someone who might love $x$
Resolution inference rule

A lifted version of propositional resolution rule

- two clauses must be standardized apart
  - no variables are shared
- can be resolved if their literals are complementary
  - one is the negation of the other
  - if one unifies with the negation of the other
where UNIFY($\ell_i, \neg m_j$) = $\theta$. For example, we can resolve the two clauses

$[\text{Animal}(F(x)) \lor \text{Loves}(G(x), x)]$ and $[\neg \text{Loves}(u, v) \lor \neg \text{Kills}(u, v)]$

by eliminating the complementary literals $\text{Loves}(G(x), x)$ and $\neg \text{Loves}(u, v)$, with unifier $\theta = \{u/G(x), v/x\}$, to produce the **resolvent** clause

$[\text{Animal}(F(x)) \lor \neg \text{Kills}(G(x), x)]$. 