CS 416  
Artificial Intelligence  
Lecture 15  
First-Order Logic  
Chapter 9

Guest Speaker

• Topics in Optimal Control, Minimax Control, and Game Theory  
• March 28th, 2 p.m. OLS 005  
• Onesimo Hernandez-Lerma  
• Department of Mathematics  
• CINVESTAV-IPN, Mexico City  
• This is a nontechnical introduction, mainly thru examples, to some recent topics in control and game theory, including adaptive control, minimax control (a.k.a. "worst-case control" or "games against nature"), partially observable systems (a.k.a. controlled "hidden Markov models"), cooperative and noncooperative game equilibria, etc.

Final Exam

• Final Exam will be May 6th at 7:00 p.m.
• This conflicts with the fewest number of other exams

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

\[
\begin{align*}
  \text{King}(x) \land \text{Greedy}(x) & \Rightarrow \text{Evil}(x) \\
  \text{King}(\text{John}) & \\
  \text{Greedy}(y) & 
\end{align*}
\]

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

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

First iteration of forward chaining

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

Second iteration of forward chaining

- We can satisfy
  \[ \text{American}(x) \land \text{Weapon}(y) \land \text{Sells}(x, y, z) \land \text{Hostile}(z) \Rightarrow \text{Criminal}(x) \]
  - must satisfy unknown premises
  - We can satisfy this rule
    \[ \text{Missile}(x) \land \text{Owns}(\text{Nono}, x) \Rightarrow \text{Sells}(\text{West}, x, \text{Nono}) \]
    - by substituting \{x/M1\}
    - and adding \text{Sells}(\text{West}, \text{M1}, \text{Nono}) to KB

- We can satisfying the implication sentences first
  \[ \text{American}(x) \land \text{Weapon}(y) \land \text{Sells}(x, y, z) \land \text{Hostile}(z) \Rightarrow \text{Criminal}(x) \]
  - must satisfy unknown premises
  - We can satisfy this rule
    \[ \text{Missile}(x) \land \text{Owns}(\text{Nono}, x) \Rightarrow \text{Sells}(\text{West}, x, \text{Nono}) \]
    - with \{x/West, y/M1, z/Nono\}
    - and adding \text{Sells}(\text{West}, \text{M1}, \text{Nono}) to KB
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

Proving completeness

• Assume KB only has sentences with no function symbols
  – What’s the most number of iterations through algorithm?
  – Depends on the number of facts that can be added
    • Let k be the arity, the max number of arguments of any predicate and
    • Let p be the number of predicates
    • Let n be the number of constant symbols
    – At most \(pn^k\) distinct ground facts
    – Fixed point is reached after this many iterations
    – A proof by contradiction shows that the final KB is complete

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

• Conject 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
  • Optimal ordering is NP-hard, similar to matrixmult

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) ^ Weapon(y) ^ Sells(x, y, z) ^ Hostile (z) => Criminal(x)
  - ~American(x) V ~Weapon(y) V ~Sells(x, y, z) V ~Hostile(z) V Criminal(x)
- Every sentence of first-order logic can be converted into an inferentially equivalent CNF sentence (they are both unsatisfiable in same conditions)

**Example**

- Everyone who loves all animals is loved by someone

\[ \forall x \left( \forall y \right. \text{Animal}(y) \Rightarrow \text{Loves}(x, y) \) \] \[ = \exists y \text{Loves}(x, y) \]

**Example**

- Standard form: For sentences like \((\forall x \ F(x)) \lor (\exists y \ Q(y))\) 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 \left( \forall y \right. \text{Animal}(y) \land \neg \text{Loves}(x, y) \) \lor (\exists y \text{Loves}(x, y)) \]

Skolemization is the process of removing existential quantifiers by eliminating. In the simplest case, it is just like the existential instantiation rule of Section 9.8: translate \(\exists y F(x)\) into \(F(x, a)\), where \(a\) is a new constant. If we apply this rule to our sample sentence, however, we obtain

\[ \forall x \left( \forall y \right. \text{Animal}(y) \land \neg \text{Loves}(x, y) \) \lor (\exists y \text{Loves}(x, y)) \]

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 \(x\).
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

- 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

Resolution

\[ \text{SUBST}(\theta, f_1 \lor \cdots \lor f_k, m_1 \lor \cdots \lor m_n) \]

where \( \text{Unit}(f_i, \neg m_j) = \theta \). For example, we can resolve the two clauses

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

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 resultant clause

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