EECS 391: Introduction to AI

Soumya Ray
Website: http://engr.case.edu/ray_soumya/eecs391_sp16/
Email: sray@case.edu
Office: Olin 516
Office hours: M 12:30-2pm
Announcements

• HW3 out
• Read Chapter 10.1-2, 10.4.2, 10.4.4 (Automated Planning)
Today

• First order Logic and inference (chapters 9.1,2,5)
• Automated Planning (ch 10)
Recap

• What is a literal in FOL?
• What is a clause in FOL?
• How do we perform inference in FOL?
• What is unification?
• What is the “most general unifier”? 
Note on ArcherAgent, PA2

• It does not belong to any package
• Place it in the root of your class hierarchy
• The config files should reference just “ArcherAgent” (NOT edu.cwru. Etc)
  – Change GameConfig2fv1a_Obstacles.xml
• Sorry for the confusion!
Lifted resolution inference rule

\[ l_1 \lor l_2 \lor m \ldots \lor l_k, \quad r_1 \lor r_2 \lor \neg m \ldots \lor r_k \]

\[ l_1 \lor l_2 \ldots \lor l_k \lor r_1 \lor r_2 \ldots \lor r_k \]

Clause: Universally quantified disjunction

\[ l_1 \lor l_2 \lor m_i \ldots \lor l_k, \quad r_1 \lor r_2 \lor \neg n_j \ldots \lor r_k, \quad m_i \theta = n_j \theta \]

\[ l_1 \lor l_2 \ldots \lor l_k \lor r_1 \lor r_2 \ldots \lor r_k \{\theta\} \]

Atomic Formula or negation: \( P(t_1, \ldots, t_n) \) (universally quantified)

Unification: Substitution \( \theta \) makes \( m_i \) syntactically identical to \( n_j \)
Standardizing Apart

• Suppose we have two FOL formulae $P(x, Const1)$ and $P(Const2, x)$
  
  – Are these unifiable?
  
  – Yes! Variables are placeholders. Assuming these $x$’s come from different universal quantifiers, we can replace the second $x$ with $y$.
  
  – This is done during standardizing apart
  
  – **Always do this** before unification unless it is stated that this has already been done
Most General Unifier (MGU)

• In general, multiple unifiers exist for two formulae

• E.g. \( p(x,y) \) and \( p(m,A) \) can be unified with 
  \( \{x/A, m/A, y/A\} \), \( \{x/B, m/B, y/A\} \),..., \( \{x/m, y/A\} \)

• A substitution \( \theta_1 \) is more general than \( \theta_2 \) if there is a nontrivial substitution \( \sigma \) so that 
  \( \theta_1 \sigma = \theta_2 \)
Most General Unifier (MGU)

• Consider a substitution that unifies two formulae and is more general than any other unifying substitution
  – This is called the most general unifier
  – E.g. for $p(x,y)$ and $p(m,A)$, $\{x/m, y/A\}$ is the MGU
    • Any other unifier like $\{x/A, m/A, y/A\}$ would require more substitutions

• For any two unifiable formulae, there is a unique MGU (upto renaming)

• So if we can find this MGU, we can use Resolution for FOL inference
Unification Algorithm

- **Input:** Two atomic formulae $s_1$ and $s_2$, standardized apart
- **Start with a list containing** $(s_1, s_2)$
- **While list is nonempty and one of the following cases applies:**
  - **Case 1:** $(x, x)$ is on the list; remove it
  - **Case 2:** $(x, T)$ or $(T, x)$ where $x$ is a variable and $T$ is a term and $x$ occurs in somewhere else in the list
    - If $x$ occurs in $T$, FAIL ("occurs check")
    - Else apply substitution $\{x/T\}$ to all other elements of list
  - **Case 3:** $(T_1, T_2)$ on the list, where neither is a variable
    - $T_1, T_2$ constants? If the same, remove, else FAIL
    - $T_1$ is $g(E_1, \ldots, E_n)$ and $T_2$ is $g(F_1, \ldots, F_n)$? Erase $(T_1, T_2)$ and replace with $(E_1, F_1), \ldots, (E_n, F_n)$
      - Else FAIL
- **At end, return** $\{x_i/T_i\}$, the elements left on the list
Example

Unify\( (p(f(x), y, x), p(a, a, f(C)))\)

\(\{p(f(x), y, x), p(a, a, f(C))\}\)

\(\{(a, f(x)), (y, a), (x, f(C))\}\)

\(\{(a, f(f(C))), (y, a), (x, f(C))\}\)

\(\{(a, f(f(C))), (y, f(f(C))), (x, f(C))\}\)

\(p(f(f(C)), f(f(C)), f(C))\)
Resolution Refutation Algorithm

• Convert \((KB \land \neg \alpha)\) to universally-quantified CNF

• Starting with this KB, generate all possible consequences using resolution as operator

• Continue until:
  • No new clauses are generated. Then KB does NOT entail \(\alpha\)
  • Two clauses resolve to yield the empty clause. Then KB entails \(\alpha\)
Completeness of FOL inference

- There are theorems (Skolem, Herbrand, Godel 1929-1930) that says that if a FOL formula is entailed by a KB, it must have a finite proof.

- So if a formula is entailed, it can be proved eventually---completeness!

- But what if it is not entailed?
FOL Entailment is Semidecidable

• If a formula is *not* entailed by a FOL KB, in general there is no way to tell by an algorithm
  – i.e., no algorithm exists that will always finitely determine non-entailment

• This creates problems for real logic systems
  – Prolog has “negation as failure”
Conversion to CNF

1. Standardize Apart:
   \( \forall x P(x) \land \exists x Q(x) \rightarrow \forall x P(x) \land \exists y Q(y) \)

2. Eliminate implications:
   \( \alpha \Rightarrow \beta \equiv \neg \alpha \lor \beta \)

3. Move negation inwards: de Morgan’s Laws,
   \( \neg \forall x p \equiv \exists x \neg p, \quad \neg \exists x p \equiv \forall x \neg p \)

4. Skolemize (get rid of \( \exists \))

5. “Drop” universal quantifiers

6. Distribute \( \land \) over \( \lor \)
Skolemization

• Dealing with $\exists$:
  – If our KB has $\exists x P(x)$, we can substitute $x$ with some object from our model’s domain
  – We don’t specifically know the name of this object, so we will invent one
    • This is called a “Skolem constant”
  – So $\exists x P(x)$ becomes $P(\text{NewSymbol})$, where \text{NewSymbol} is a constant symbol not used by anything else in our KB
    • The semantics for \text{NewSymbol} is “that/some object for which $P$ is true”
Skolemization part 2

- If our KB has $\forall x \exists y P(x,y)$, the new object for $y$ might need to depend on $x$

- Example:
  
  $\forall x \exists y \text{Friend}(x,y) \models \forall x \text{Friend}(x,\text{Billy})$?

- In this case we use a “Skolem function”

  $\forall x \exists y \text{Friend}(x,y)$ becomes

  $\forall x \text{Friend}(x, \text{FriendOf}(x))$
Skolemization part 2

• In general, the Skolem function needs as many arguments as there are quantifiers in front of the $\exists$ being quantified
Example

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

1. ∀x(∀y Animal(y) ⇒ Loves(x, y)) ⇒ (∃p Loves(p, x))

2. ∀x(¬∀y ¬Animal(y) ∨ Loves(x, y)) ∨ (∃p Loves(p, x))

3. ∀x(∃y ¬(¬Animal(y) ∨ Loves(x, y))) ∨ (∃p Loves(p, x))

4. ∀x(Animal(LovedBy(x)) ∧ ¬Loves(x, LovedBy(x))) ∨ (Loves(LoverOf(x), x))

5. (Animal(LB(x)) ∧ ¬Loves(x, LB(x))) ∨ (Loves(LO(x), x))

6. (Animal(LB(x)) ∨ Loves(LO(x), x)), (¬Loves(x, LB(x)) ∨ Loves(LO(x), x))
Example

\( \forall x \ Feathers(x) \land Flies(x) \Rightarrow Bird(x) \)

\( \forall x \ Bird(x) \Rightarrow Animal(x) \)

\( \forall x \forall y \ Animal(x) \land CanTalk(x) \land Human(y) \Rightarrow Likes(x, y) \)

Feathers(Parrot), Flies(Parrot), CanTalk(Parrot), Human(Socrates)

? Likes(Parrot, Socrates)

\[ \neg Feathers(x) \lor \neg Flies(x) \lor Bird(x) \]

\[ \neg Bird(x) \lor Animal(x) \]

\[ \neg Animal(x) \lor \neg CanTalk(x) \lor \neg Human(y) \lor Likes(x, y) \]

Feathers(Parrot), Flies(Parrot), CanTalk(Parrot), Human(Socrates)

\[ \neg Likes(Parrot, Socrates) \]
Example

\[\neg \text{Animal}(x) \lor \neg \text{CanTalk}(x) \lor \neg \text{Human}(y) \lor \text{Likes}(x,y), \neg \text{Likes(Parrot, Socrates)}\]

\[\neg \text{Animal(Parrot)} \lor \neg \text{CanTalk(Parrot)} \lor \neg \text{Human(Socrates)}\]

\[\text{Animal(Parrot)} \lor \neg \text{CanTalk(Parrot)} \lor \neg \text{Human(Socrates)}, \text{Human(Socrates)}\]

\[\neg \text{Animal(Parrot)} \lor \neg \text{CanTalk(Parrot)}\]

\[\text{Animal(Parrot)} \lor \neg \text{CanTalk(Parrot)}, \text{CanTalk(Parrot)}\]

\[\neg \text{Animal(Parrot)}\]

\[\neg \text{Bird}(x) \lor \text{Animal}(x), \neg \text{Animal(Parrot)}\]

\[\neg \text{Bird(Parrot)}\]

\[\neg \text{Feathers}(x) \lor \neg \text{Flies}(x) \lor \text{Bird}(x), \neg \text{Bird(Parrot)}\]

\[\neg \text{Feathers(Parrot)} \lor \neg \text{Flies(Parrot)}\]

\[\neg \text{Feathers(Parrot)} \lor \neg \text{Flies(Parrot)}, \text{Feathers(Parrot)}, \text{Flies(Parrot)}\]
Variants of Resolution

• In order to make the resolution procedure more efficient, people have explored several variants

• These are heuristics that (in some cases) sacrifice completeness for efficiency

• Unit Resolution

• Linear Resolution

• Book has others
Summary

• We learned about:
  – FOL syntax and semantics
    • Quantifiers
  – Syntactic Inference through Lifting
  – Lifted Resolution
  – Unification
  – Conversion to CNF
  – Complexity of FOL Inference

• Next: Automated Planning
Automated Planning (Ch 10)

• Consider a situation where an agent has to carry out a sequence of actions to achieve a goal

• Suppose the agent starts off with detailed, structured knowledge of the world
  – Could we take advantage of this?
  – E.g. a chess playing agent should start knowing rules
The Planning Problem

• Given:
  – An initial state of the world, described as a set of logical facts
  – A set of goal states, described as a set of logical facts
  – A set of actions, also described in logic

• Find a sequence of actions that will move the world from the initial state to the final state
  – This sequence is called a plan
  – Often also try to optimize some criteria
“Classical” Planning

• We’ll study planning algorithms designed to work when the world is:
  – Deterministic
  – Static
  – Fully observable
  – Actions are instantaneous

• These restrictions can be relaxed (more or less)
Task: Starting with initial configuration of blocks, produce a desired goal configuration by moving block around.
Situation Calculus (Chapter 10.3)

• It is natural to think of using full FOL to encode states of the world and actions
  – Then use general FOL inference as planner

• People developed a general method for encoding states and actions based on FOL
  – Called the “Situation Calculus”
  – Situations=predicates + time indices
Issues with Situation Calculus

• SC is appealing because no special algorithms are needed for planning
  – Given an SC knowledge base, query “Is there a sequence of actions leading to a situation where the goal holds?”
  – Apply resolution
• But this is very slow, even for small planning problems
• So specialized fragments of FOL have been developed to represent planning problems instead
Representing a Planning Problem

• For classical planning, one fragment of FOL that is used is called STRIPS (“Stanford Research Institute Problem Solver”)

• States, actions and goals will be represented in this language
  – Then we’ll see planning algorithms (which are inference algorithms in disguise) that find plans in this language
Representing States in STRIPS

• States in STRIPS are conjunctions of unnegated, ground, function-free literals
  – All conditions that hold in that state
  – Block(A), Block(B), On(A,B), On(B, Table), GripperEmpty
  – The “Closed World Assumption” is used
Representing Goals in STRIPS

• Goals are conjunctions of unnegated, ground, function-free literals

• Goals may not fully determine a state of the world
  – In this case, the goal is any state where these literals hold

• Example: \( On(A,E) \land On(B,D) \)
Representing Actions in STRIPS

• Want to represent an action of picking up a block from the table

\textit{Pickup\_from\_Table(x)}

\textbf{Preconditions:} \textit{Block(x), GripperEmpty, Clear(x), On(x,Table)}

\textbf{Add List:} \textit{Holding(x)}

\textbf{Delete List:} \textit{GripperEmpty, On(x,Table)}

“Applicability”: action can be used at a state iff its preconditions are satisfied