Artificial Intelligence Programming

Inference

Chris Brooks

Department of Computer Science
University of San Francisco

Logic-based agents

- The big picture: our agent needs to make decisions about what to do in a complex environment.
- In order to do this, it needs a model of the world.
- Logic is a language for representing this model.
- Our agent will maintain and update a *knowledge base* (KB) which stores this model.
- Our agent will use *inference* to derive new facts that are added to the KB.
- These new facts will help our agent make decisions.

Logic Summary

- Recall that propositional logic concerns:
  - facts about the world that are true or false.
  - These facts can be used to construct sentences using logical connectives ($\wedge, \vee, \neg, \rightarrow, \leftrightarrow$)
  - Example: $P_{1,1} \vee P_{3,3}$.
- We can convert these sentences to CNF and use resolution as an inference mechanism.
  - Resolution is sound and complete.

Logic Summary

- A big weakness of propositional logic is its closed-world assumption
  - No way to talk about objects that cannot be named.
- Also, no way to specify classes or relations
- In practice, this leads to an exponential number of sentences in the agent's KB.

Logic Summary

- First order logic addresses these issues by providing variables and quantification over these variables.
  - FOL works with objects and relations between them, which form atomic sentences.
  - Atomic sentences can be joined with logical connectives. objects.
  - Example: $Siblings(Bart, Lisa), Valikes(x, Marge) \rightarrow likes(x, homer)$

Quantifiers and variables

- $\exists$ is the symbol for existential quantification
  - It means that the sentence is true for at least one element in the domain.
  - $\exists x\ female(x) \wedge playsSaxophone(x)$
  - What would happen if I said $\exists x\ female(x) \rightarrow playsSaxophone(x)$?

Quantifiers

- In general, → makes sense with ∀ (∧ is usually too strong).
- ∧ makes sense with ∃ (→ is generally too weak.)
- Some examples:
  - One of the Simpsons works at a nuclear plant.
  - All of the Simpsons are cartoon characters.
  - There is a Simpson with blue hair and a green dress.
  - There is a Simpson who doesn't have hair.

Nesting quantifiers

- Often, we'll want to express more complex quantifications. For example, "every person has a mother"
  - $\forall x \exists y\ mother(x, y)$
  - Notice the scope - for each x, a different y is (potentially) chosen.
- What if we said $\exists y \forall x\ mother(x, y)$?
- this is not a problem when nesting quantifiers of the same type.
- $\forall x \forall y\ brotherOf(x, y) \rightarrow siblingOf(x, y)$ and $\forall y \forall x\ brotherOf(x, y) \rightarrow siblingOf(x, y)$ are equivalent.
- We often write that as $\forall x, y\ brotherOf(x, y) \rightarrow siblingOf(x, y)$

Negation

- We can negate quantifiers
  - $\neg \forall x\ yellow(x)$ says that it is not true that everyone is yellow.
  - $\exists x \neg yellow(x)$ has the same meaning - there is someone who is not yellow.
  - $\neg \exists x\ daughterOf(Bart, x)$ says that there does not exist anyone who is Bart's daughter.
  - $\forall x \neg daughterOf(Bart, x)$ says that for all individuals they are not Bart's daughter.
- In fact, we can use DeMorgan's rules with quantifiers just like with ∧ and ∨.

More examples

- A husband is a male spouse
  - $\forall x, y\ husband(x, y) \Leftrightarrow spouse(x, y) \wedge male(x)$
- Two siblings have a parent in common
  - $\forall x, y\ sibling(x, y) \Leftrightarrow \neg(x = y) \wedge \exists p\ Parent(x, p) \wedge Parent(y, p)$
- Everyone who goes to Moe's likes either Homer or Barney (but not both)
  - $\forall x\ goesTo(moes, x) \rightarrow (Likes(x, Homer) \Leftrightarrow \neg Likes(x, Barney))$

More examples

- Everyone knows someone who is angry at Homer.
  - $\forall x\ \exists y\ knows(x, y) \wedge angryAt(y, homer)$
- Everyone who works at the power plant is scared of Mr. Burns
  - $\forall x\ worksAt(PowerPlant, x) \rightarrow scaredOf(x, burns)$

Audience Participation

- Everyone likes Lisa.
- Someone who works at the power plant doesn't like Homer. (both ways)
- Bart, Lisa, and Maggie are Marge's only children.
- People who go to Moe's are depressed.
- There is someone in Springfield who is taller than everyone else.
- When a person is fired from the power plant, they go to Moe's
- Everyone loves Krusty except Sideshow Bob
- Only Bart skateboards to school
- Someone with large feet robbed the Quickie-mart.

Representing useful knowledge in FOL

- We can use FOL to represent class/subclass information, causality, existence, and disjoint sets.
- Example: Let's suppose we are interested in building an agent that can help recommend music.
- We want it to be able to reason about musical artists, songs, albums, and genres.
- It would be tedious to enter every bit of information about every artist; instead, we'll enter some rules and let our agent derive entailed knowledge.

Music example

- $\forall x genre(x, Punk) \rightarrow genre(x, Rock)$ - subclass: all Punk songs are Rock songs.
- $member(JohnLennon, Beatles) \wedge member(PaulMcCartney, Beatles) \wedge member(GeorgeHarrison, Beatles) \wedge member(RingoStarr, Beatles) \wedge \forall x\, member(x, Beatles) \rightarrow x \in \{John, Paul, George, Ringo\}$ - exclusive membership: John, Paul, George, and Ringo are the Beatles.
- $performedBy(Beatles, WhiteAlbum)$ The WhiteAlbum is a Beatles album
- $\forall x, y, z\, member(x, y) \wedge performedBy(y, z) \rightarrow playedOn(x, z)$ If someone is a member of a group, and that group performed an album, then that person played on that album.

Music example

- $genre(HolidaysInTheSun, Punk)$ - "Holidays in the Sun" is a Punk song.
- $\forall x\, genre(x, Rock) \rightarrow likes(Bob, x)$ Bob likes all rock songs.
  - We should be able to infer that Bob will like "Holidays in the Sun"
- $\forall x, y, z\, likes(x, y) \wedge member(z, y) \wedge performedBy(z, w) \rightarrow likes(x, w)$ - If someone likes albums by a band Y, and Z is a member of band Y, then that person will like albums by person Z.

Inference in Propositional logic

- We talked about two basic mechanisms for performing inference in propositional logic:
  - Forward chaining: Begin with the facts in your KB. Apply inference rules until no more conclusions can be drawn.
  - Backward chaining: Begin with a fact (or its negation) that you wish to prove. Find facts that will justify this fact. Work backwards until you find facts in your KB that support (contradict) the fact you wish to prove.

Inference in FOL

- Can we do the same sorts of inference with First-order logic that we do with propositional logic?
- Yes, with some extra details.
- Need to keep track of variable bindings (substitution)

Inference in FOL

- As with propositional logic, we'll need to convert our knowledge base into a canonical form.
- In the simplest approach, we can convert our FOL sentences to propositional sentences. We do this by removing quantification.
  - we leave in predicates for readability, but remove all variables.