

*Artificial Intelligence Programming*  
*First-Order Logic*

Chris Brooks

Department of Computer Science  
University of San Francisco

## 12-2: Representation

---

- Propositional Logic has several nice features
  - Lets us easily express disjunction and negation
    - “There is a pit in (1,1) or in (2,2)”
    - “There is not a pit in (1,1)”
    - This is hard to do in C/Java/Python - variables can only take on a single value.
    - There’s no obvious way to *assign*  $x$  the value “3 or 4” or “some value less than 10”.
  - Separates declarative knowledge from inference procedures
  - Compositional
    - The meaning of a sentence is a function of the meaning of its parts.

## 12-3: Review of Propositional Logic

---

- Sentences are composed of atomic terms conjoined by operators
  - $P_{1,1} \wedge B_{1,2}$
  - $\neg B_{2,2} \vee \neg P_{1,2}$
- Terms are either true or false.
- A model is an assignment of values to terms.
  - The set of possible worlds that make a sentence true
- A model satisfies a sentence if the sentence is true given the assignments in the model.

## 12-4: Inference in Propositional Logic

---

- Inference can be done using DeMorgan's, Modus Ponens, And-elimination, etc
- Or, we can convert sentences to Conjunctive Normal Form
  - $(a \vee b \vee \neg c) \wedge (d \vee e) \wedge (f \vee \neg g)$
- We can then use resolution to derive new sentences.
- Resolution is sound and complete.

## 12-5: Weaknesses of Propositional Logic

---

- There are several problems with propositional logic:
  - Lack of conciseness. (How to say “There is a pit in every square”?)
  - Lack of ability to quantify (How to say “There is a square that has a pit”?)
  - Lack of ability to deal with large domains.
    - What if there were an infinite number of rooms?
    - What if we wanted to write sentences about the integers?
- The problem stems from the fact that propositional logic deals with facts that are either true or false.
- No way to indicate that terms in different sentences refer to the same object.
- No way to talk about relations between objects.

## 12-6: Expressivity

---

- We would like the sorts of structures that are useful in programming languages. In particular, we would like to have:
  - Objects: Wumpi, pits, gold, vacuum cleaners, etc.
  - Variables: how do we talk about objects without knowing their names?
  - Relations: These can include:
    - Unary relations (or properties): smelly(wumpus), shiny(gold), sleepy(student), etc.
    - Binary relations: brother-of(bart, lisa), holding(agent, gold), after(Tuesday, Monday)
    - $n$ -ary relations: simpsons(homer, marge, bart, lisa, maggie)
    - These are sometimes called *predicates*
  - Functions: father-of(bart) = homer, fall-classes(student) = AI, etc.
- *First-order logic* gives us all of this.

## 12-7: Models in first-order logic

---

- Recall that a model is the set of “possible worlds” for a collection of sentences.
- In propositional logic, this meant truth assignments to facts.
- In FOL, models have objects in them.
- The *domain* of a model is the set of objects in that world.
- For example, the Simpsons model might have the domain
  - {Marge, Homer, Lisa, Bart, Maggie}
- We can then specify relations and functions between these objects
  - married-to(marge, homer), baby(maggie), father(bart) = homer

## 12-8: Terms and sentences

---

- A *term* is an expression that refers to a single object.
  - Bart, Lisa, Homer
  - We can also use functions as terms - Saxophone(Lisa) refers to the object that is Lisa's saxophone
- An *atomic sentence* consists of a predicate applied to terms
  - Brother-of(Lisa, Bart), Married(Homer, Marge), Married(Mother(Lisa), Father(Bart))
  - Plays(Lisa, Saxophone(Lisa))

## 12-9: Terms and sentences

---

- A Complex sentence uses logical connectives  $\neg, \vee, \wedge, \Rightarrow, \Leftrightarrow$  to join atomic sentences.
  - $\neg \text{BrotherOf}(\text{Homer}, \text{Bart}),$
  - $\text{MotherOf}(\text{Lisa}, \text{Marge}) \Rightarrow \text{MotherOf}(\text{Bart}, \text{Marge})$
  - $\text{Oldest}(\text{Bart}) \vee \text{Oldest}(\text{Lisa})$
- We can also use equality to relate objects:  
 $\text{homer} = \text{father}(\text{Bart})$

## 12-10: Quantifiers and variables

---

- Often, it's not enough to make a statement about particular objects. Instead, we want to make a statement about some or all objects.
  - “All of the Simpsons are yellow.”
  - “At least one of the Simpsons is a baby.”
  - Quantifiers allow us to do this.
  - $\forall$  is the symbol for universal quantification
    - It means that a sentence holds for every object in the domain.
    - $\forall x \text{Simpson}(x) \Rightarrow \text{yellow}(x)$

## 12-11: 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.
  - $\exists x \text{female}(x) \wedge \text{playsSaxophone}(x)$
  - What would happen if I said  $\exists x \text{female}(x) \Rightarrow \text{playsSaxophone}(x)$ ?

## 12-12: Quantifiers

---

- In general,  $\Rightarrow$  makes sense with  $\forall$  ( $\wedge$  is usually too strong).
- $\wedge$  makes sense with  $\exists$  ( $\Rightarrow$  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.

## 12-13: Nesting quantifiers

---

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

## 12-14: Negation

---

- We can negate quantifiers
  - $\neg\forall xyellow(x)$  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  $\wedge$  and  $\vee$ .

## 12-15: More examples

---

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

## 12-16: More examples

---

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

## 12-17: 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.

## 12-18: Inference

---

- Recall that we have two basic approaches to performing inference in propositional logic:
  - Encode sentences in any way we see fit, and apply Modus Ponens, AND-elimination, OR-introduction, and DeMorgan's Rule.
  - Encode sentences in CNF, and apply resolution.
- We can use the exact same approaches in first-order logic.
  - There are a couple of extra details to deal with