




Artificial Intelligence Programming

First-Order Logic

Chris Brooks

Department of Computer Science
University of San Francisco



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.

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.

Resolution

- Inference can be done using DeMorgan's, Modus Ponens, And-elimination, etc
- Sound, but not necessarily complete.
- Also, search can be inefficient: there might be many operators that can be applied in a particular state.
- Luckily, there is a complete rule for inference (when coupled with a complete search algorithm) that uses a single operator.
- This is called *resolution*.
 - $A \vee B$ and $\neg A \vee C$ allows us to conclude $B \vee C$.
 - A is either true or not true. If A is true, then C must be true.
 - if A is false, then B must be true.
 - This can be generalized to clauses of any length.

Conjunctive Normal Form

- Resolution works with disjunctions.
- This means that our knowledge base needs to be in this form.
- Conjunctive Normal Form is a conjunction of clauses that are disjunctions.
- $(A \vee B \vee C) \wedge (D \vee E \vee F) \wedge (G \vee H \vee I) \wedge \dots$
- Every propositional logic sentence can be converted to CNF.

Conjunctive Normal Form Recipe

1. Eliminate equivalence

- $A \Leftrightarrow B$ becomes $A \Rightarrow B \wedge B \Rightarrow A$

2. Eliminate implication

- $A \Rightarrow B$ becomes $\neg A \vee B$

3. Move \neg inwards using double negation and DeMorgan's

- $\neg(\neg A)$ becomes A

- $\neg(A \wedge B)$ becomes $(\neg A \vee \neg B)$

4. Distribute nested clauses

- $(A \vee (B \wedge C))$ becomes $(A \vee B) \wedge (A \vee C)$

Example

- $B_{1,1} \Leftrightarrow (P_{1,2} \vee P_{2,1})$
- Eliminating equivalence produces:
 - $(B_{1,1} \Rightarrow (P_{1,2} \vee P_{2,1})) \wedge ((P_{1,2} \vee P_{2,1}) \Rightarrow B_{1,1})$
- Removing implication gives us:
 - $(\neg B_{1,1} \vee (P_{1,2} \vee P_{2,1})) \wedge (\neg(P_{1,2} \vee P_{2,1}) \vee B_{1,1})$

Example

- We then use DeMorgan's rule to move negation inwards:
 - $(\neg B_{1,1} \vee P_{1,2} \vee P_{2,1}) \wedge ((\neg P_{1,2} \wedge \neg P_{2,1}) \vee B_{1,1})$
- Finally, we distribute OR over AND:
 - $(\neg B_{1,1} \vee P_{1,2} \vee P_{2,1}) \wedge (\neg P_{1,2} \vee B_{1,1}) \wedge (\neg P_{2,1}) \vee B_{1,1})$
- Now we have clauses that can be plugged into a resolution theorem prover. (can break ANDs into separate sentences)
- They're less readable by a human, but more computationally useful.

Proof By Refutation

- Once your KB is in CNF, you can do resolution by refutation.
 - In math, this is called proof by contradiction
- Basic idea: we want to show that sentence A is true.
- Insert $\neg A$ into the KB and try to derive a contradiction.

Example

- Prove that there is not a pit in (1,2). $\neg P_{1,2}$
- Relevant Facts:
 - $R_{2a} : (\neg B_{1,1} \vee P_{1,2} \vee P_{2,1})$
 - $R_{2b} : (\neg P_{1,2} \vee B_{1,1})$
 - $R_{2c} : (\neg P_{2,1} \vee B_{1,1})$
 - $R_4 : \neg B_{1,1}$
- Insert $R_n : P_{1,2}$ into the KB

Example

- Resolve R_n with R_{2b} to get: $R_6 : B_{1,1}$
- We already have a contradiction, since $R_4 : \neg B_{1,1}$
- Therefore, the sentence we inserted into the KB must be false.
- Most proofs take more than one step to get to a contradiction ...

Horn clauses

- Standard resolution theorem proving (and propositional inference in general) is exponentially hard.
- However, if we're willing to restrict ourselves a bit, the problem becomes (computationally) easy.
- A *Horn* clause is a disjunction with at most one positive literal.
 - $\neg A \vee \neg B \vee \neg C \vee D$
 - $\neg A \vee \neg B$
- These can be rewritten as implications with one consequent.
 - $A \wedge B \wedge C \Rightarrow D$
 - $A \wedge B \Rightarrow \text{False}$
- Horn clauses are the basis of logic programming (sometimes called rule-based programming)

How to use a KB: Forward Chaining

- Forward chaining involves starting with a KB and continually applying Modus Ponens to derive all possible facts.
- This is sometimes called data-driven reasoning
- Start with domain knowledge and see what that knowledge tells you.
- This is very useful for discovering new facts or rules
- Less helpful for proving a specific sentence true or false
 - Search is not directed towards a goal

How to Use A KB: Backward Chaining

- Backward chaining starts with the goal and “works backward” to the start.
- Example: If we want to show that A is entailed, find a sentence whose consequent is A .
- Then try to prove that sentence’s antecedents.
- This is sometimes called query-driven reasoning.
- More effective at proving a particular query, since search is focused on a goal.
- Less likely to discover new and unknown information.
- Means-ends analysis is a similar sort of reasoning.
- Prolog uses backward chaining.

Strengths of Propositional Logic

- Declarative - knowledge can be separated from inference.
- Can handle partial information
- Can compose more complex sentences out of simpler ones.
- Sound and complete inference mechanisms (efficient for Horn clauses)

Applications

- Propositional logic can work nicely in bounded domains
 - All objects of interest can be enumerated.
- Fast algorithms exist for solving SAT problems via model checking.
 - Search all models to find one that satisfies a sentence.
 - Can be used for some scheduling and planning problems
- Constraint satisfaction problems can be expressed in propositional logic.
 - Often, we'll use a predicate-ish notation as syntactic sugar.

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.

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*

Expressivity

- Functions: father-of(bart) = homer, fall-classes(student) = AI, etc.
- *First-order logic* gives us all of this.

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

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))

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:
 $\text{homer} = \text{father}(\text{Bart})$

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.
 - $\forall x \text{Simpson}(x) \Rightarrow \text{yellow}(x)$

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
 $\exists x \text{ female}(x) \Rightarrow \text{playsSaxophone}(x)?$

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.

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 $\forall x, y \text{ brotherOf}(x, y) \Rightarrow \text{siblingOf}(x, y)$

Negation

- We can negate quantifiers
 - $\neg\forall x \text{yellow}(x)$ says that it is not true that everyone is yellow.
 - $\exists x\neg\text{yellow}(x)$ has the same meaning - there is someone who is not yellow.
 - $\neg\exists x \text{daughterOf}(Bart, x)$ says that there does not exist anyone who is Bart's daughter.
 - $\forall x \neg\text{daughterOf}(Bart, x)$ 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 .

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)
 - $\forall x \text{ goesTo}(\text{moes}, x) \Rightarrow$
 $(\text{Likes}(x, \text{Homer}) \Leftrightarrow \neg \text{Likes}(x, \text{Barney}))$

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
 - $\forall x \text{ worksAt}(\text{PowerPlant}, x) \Rightarrow \text{scaredOf}(x, \text{burns})$

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.