Today 10/3

• Review Model Checking/Wumpus
• CNF
• WalkSat
• Break
• Start on FOL

Review

• Propositional logic provides
  • Propositions that have
  • Truth values and
  • Logical connectives that allow a
  • Compositional Semantics and
  • Inference
Models

- Models are formally structured worlds with respect to which truth can be evaluated.
- \( m \) is a model of a sentence \( \alpha \) if \( \alpha \) is true in \( m \)
- \( M(\alpha) \) is the set of all models of \( \alpha \)

Wumpus world model

Situation after detecting nothing in [1,1], moving right; breeze in [2,1]

Consider possible models for \( \neg A \) assuming only \( p \)

3 Boolean choices \( \Rightarrow \) 8 possible models
Effective propositional inference

- Two families of efficient algorithms for propositional inference based on model checking:
- Are used for checking satisfiability
- Complete backtracking search algorithms
  - DPLL algorithm (Davis, Putnam, Logemann, Loveland)
  - Incomplete local search algorithms
    - WalkSAT algorithm

Conversion to CNF

\[ B_{1} \Rightarrow (P_{1,2} \lor P_{2,1}) \]
- Eliminate \( \Rightarrow \), replacing \( a \Rightarrow b \) with \( (\neg b \lor a) \).
  - \( (B_{1} \Rightarrow (P_{1,2} \lor P_{2,1})) \cdot ((\neg (P_{1,2} \lor P_{2,1})) \Rightarrow B_{1}) \)
- Eliminate \( \Rightarrow \), replacing \( a \Rightarrow b \) with \( \neg a \lor b \).
  - \( \neg B_{1} \lor P_{1,2} \lor P_{2,1} \land (\neg (P_{1,2} \lor P_{2,1})) \Rightarrow B_{1} \)
- Move \( \neg \) inwards using de Morgan’s rules and double-negation:
  - \( \neg B_{1} \lor P_{1,2} \lor P_{2,1} \land (\neg (P_{1,2} \lor P_{2,1})) \Rightarrow B_{1} \)
- Apply distributivity law (\( \cdot \) over \( \lor \)) and flatten:
  - \( \neg B_{1} \lor P_{1,2} \lor P_{2,1} \land (\neg (P_{1,2} \lor P_{2,1})) \Rightarrow B_{1} \)
The DPLL algorithm

- Determine if an input propositional logic sentence (in CNF) is satisfiable by assigning values to variables.

1. Pure symbol heuristic
   - Pure symbol: always appears with the same "sign" in all clauses.
   - Example: In the three clauses (A ∨ ¬B), (¬B ∨ ¬C), (C ∨ A), A and B are pure, C is impure.
   - Assign a pure symbol so that their literals are true.

2. Unit clause heuristic
   - Unit clause: only one literal in the clause or only one literal which has not yet received a value. The only literal in a unit clause must be true.

The WalkSAT algorithm

- Incomplete, local search algorithm.
- Evaluation function: The min-conflict heuristic of minimizing the number of unsatisfied clauses.
- Steps are taken in the space of complete assignments, flipping the truth value of one variable at a time.
- Balance between greediness and randomness.
  - To avoid local minima
The WalkSAT algorithm

```
Function WalkSAT(clauses, n max-flips) returns a satisfying model or failure
Input: clauses, a set of clauses in propositional logic
      n, the probability of choosing to do a "random walk" move
      max-flips, number of flips allowed before giving up
Model = a random assignment of true/false to the symbols in clauses
For i = 1 to max-flips do
  If model satisfies clauses then return model
  Else randomly select clause from clauses that is false in model
  With probability p flip the value in model of a randomly selected symbol
  From clause
  Else flip whichever symbol in clause maximizes the number of satisfied clauses
  Return failure
```

Break

- Quiz 1: Average was 43

Pros and cons of propositional logic

- Propositional logic is declarative
- Propositional logic allows partial/disjunctive/negated information
- Propositional logic is compositional:
  - Meaning of \( B_{1,1} \land P_{1,2} \) is derived from meaning of \( B_{1,1} \) and of \( P_{1,2} \)
- Meaning in propositional logic is context-independent
- Propositional logic has very limited expressive power
  - E.g., cannot say "pits cause breezes in adjacent squares"
  - Except by writing one sentence for each square
**FOL**

- At a high level...
  - FOL allows you to represent objects, properties of objects, and relations among objects
  - Specific domains are modeled by developing knowledge-bases that capture the important parts of the domain (change, auto repair, medicine, time, set theory, etc)

**FOL**

- First order logic adds
  - Variables and quantifiers that allow
  - Statements about unknown objects and
  - Statements about classes of objects

**First-order logic**

- Whereas propositional logic assumes the world contains facts,
- first-order logic (like natural language) assumes the world contains
  - Objects: people, houses, numbers, colors, baseball games, wars, ...
  - Relations: red, round, prime, brother of, bigger than, part of, comes between, ...
  - Functions: father of, best friend, one more than, plus, ...
Syntax of FOL

- Constants: KingJohn, 2, ...
- Predicates: Brother, >, ...
- Functions: Sqrt, LeftLegOf, ...
- Variables: x, y, a, b, ...
- Connectives: ¬, ⇒, ∧, ∨, ⇔
- Equality: =
- Quantifiers: ∀, ∃

Atomic sentences

Atomic sentence = predicate(term₁,...,termₙ)
or term₁ = term₂

Term = function(term₁,...,termₙ)
or constant or variable

E.g.
- Brother(KingJohn, RichardTheLionheart)
- > (Length(LeftLegOf(Richard)), Length(LeftLegOf(KingJohn)))

Complex sentences

Complex sentences are made from atomic sentences using connectives
¬S₁, S₁ ∧ S₂, S₁ ∨ S₂, S₁ ⇒ S₂, S₁ ⇔ S₂

E.g.
Sibling(KingJohn, Richard) ⇒
Sibling(Richard, KingJohn)
Truth in first-order logic

- Sentences are true with respect to a model and an interpretation.
- Model contains objects (domain elements) and relations among them.
- Interpretation specifies referents for constant symbols → objects, predicate symbols → relations, function symbols → functional relations.
- An atomic sentence \( \text{predicate}(\text{term}_1, \ldots, \text{term}_n) \) is true iff the objects referred to by \( \text{term}_1, \ldots, \text{term}_n \) are in the relation referred to by \( \text{predicate} \).

Models for FOL: Example

Models as Sets

- Let’s populate a domain:
  - \((\text{R}, \text{J}, \text{RLL}, \text{JLL}, \text{C})\)
- Property Predicates
  - Person = \((\text{R}, \text{J})\)
  - Crown = \((\text{C})\)
  - King = \((\text{J})\)
- Relational Predicates
  - Brother = \(\{(\text{R}, \text{J}), (\text{J}, \text{R})\}\)
  - OnHead = \(\{(\text{C}, \text{J})\}\)
- Functional Predicates
  - LeftLeg = \(\{(\text{R}, \text{RLL}), (\text{J}, \text{JLL})\}\)
Quantifiers

- Allows us to express properties of collections of objects instead of enumerating objects by name
- Universal: "for all" ∀
- Existential: "there exists" ∃

Universal quantification

∀x At(x, CU) ⇒ Smart(x)

∀x P is true in a model m iff P is true with x being each possible object in the model

Roughly speaking, equivalent to the conjunction of instantiations of P

At(KingJohn, CU) ⇒ Smart(KingJohn)
At(Richard, CU) ⇒ Smart(Richard)
At(Ralphie, CU) ⇒ Smart(Ralphie)
...

Existential quantification

∃x At(x, CU) ∧ Smart(x)

∃x P is true in a model m iff P is true with x being some possible object in the model

Roughly speaking, equivalent to the disjunction of instantiations of P

At(KingJohn, CU) ∧ Smart(KingJohn)
At(Richard, CU) ∧ Smart(Richard)
At(Ralphie, CU) ∧ Smart(Ralphie)
...

Properties of quantifiers

- $\forall x \forall y$ is the same as $\forall y \forall x$
- $\exists x \forall y$ is not the same as $\forall x \exists y$
- $\exists x \forall y$ Loves($x,y$) is "There is a person who loves everyone in the world"
- $\forall x \exists y$ Loves($x,y$) is "Everyone in the world is loved by at least one person"

- Quantifier duality: each can be expressed using the other
  - $\forall x$ Likes(x, IceCream) $\Leftrightarrow \neg \exists x \neg$ Likes(x, IceCream)
  - $\exists x$ Likes(x, Broccoli) $\Leftrightarrow \neg \forall x \neg$ Likes(x, Broccoli)

Variables

- A big part of using FOL involves keeping track of all the variables while reasoning.
- Substitution lists are the means used to track the value, or binding, of variables as processing proceeds.

Examples

- $\text{Likes}[x]$
- $\text{Loves}[x] \rightarrow \text{Eats}[x]\{x\}$
- $x \in \text{Humans}$
- $\exists x$ Likes(x, Broccoli) $\rightarrow \text{Eats}[x]\{x\}$
Examples

\( P(x, y) \rightarrow Q(x, y) \)
\( P(x, y) \land Q(x, y) \rightarrow R(x, y) \)
\( P(x, y) \land Q(x, y) \rightarrow R(x, y) \)

Inference

• Inference in FOL involves showing that some sentence is true, given a current knowledge-base, by exploiting the semantics of FOL to create a new knowledge-base that contains the sentence in which we are interested.

Inference Methods

• Proof as Generic Search
• Proof by Modus Ponens
  - Forward Chaining
  - Backward Chaining
• Resolution
• Model Checking
Generic Search

• States are snapshots of the KB
• Operators are the rules of inference
• Goal test is finding the sentence you're seeking
  - I.e. Goal states are KBs that contain the sentence (or sentences) you're seeking

Example

• Harry is a hare
• Tom is a tortoise
• Hares outrun tortoises
• Harry outruns Tom?

Tom and Harry

• And introduction
• Universal elimination
• Modus ponens
What’s wrong?

- The branching factor caused by the number of operators is huge
- It’s a blind (undirected) search

So...

- So a reasonable method needs to control the branching factor and find a way to guide the search...
- Focus on the first one first

Forward Chaining

- When a new fact $p$ is added to the KB
  - For each rule such that $p$ unifies with part of the premise
    - If all the other premises are known
    - Then add consequent to the KB

This is a data-driven method.
Backward Chaining

• When a query \( q \) is asked
  - If a matching \( q' \) is found return substitution list
  - Else For each rule \( q' \) whose consequent matches \( q \), attempt to prove each antecedent by backward chaining

This is a goal-directed method. And it’s the basis for Prolog.

Is Tom faster than someone?

Notes

• Backward chaining is not abduction; we are not inferring antecedents from consequents.
• The fact that you can’t prove something by these methods doesn’t mean its false. It just means you can’t prove it.
Resolution

- Modus ponens is not complete. I.e. there are things we should be able to prove true that we can’t by using Modus ponens alone.
- Used appropriately, resolution is complete.

Resolution Example

Resolve 1 and 3
Resolve 2 and 5
Resolve 4 and 6
Convert to Normal Form