**CSE 473**

Chapter 8

First-Order Logic

---

**Last Time:**

Efficient propositional inference

Two families of efficient algorithms for propositional inference based on *model checking*:

Complete backtracking search algorithms
- DPLL algorithm (Davis, Putnam, Logemann, Loveland)
- Similar to TT enumeration but with heuristics
- See lecture slides from last class

Incomplete local search algorithms
- *WalkSAT* algorithm for testing satisfiability
Why Satisfiability?

- Recall: $KB \models a$ iff $KB \land \neg a$ is unsatisfiable
- Thus, algorithms for satisfiability can be used for inference (entailment)

- More generally, showing a sentence is satisfiable (the SAT problem) is NP-complete
  i.e., Finding a fast algorithm for SAT automatically yields fast algorithms for hundreds of difficult (NP-complete) problems
Satisfiability Examples

E.g. 2-CNF sentences (2 literals per clause):

\((\neg A \lor \neg B) \land (A \lor B) \land (A \lor \neg B)\)

Satisfiable?
Yes (e.g., \(A = \text{true}, B = \text{false}\))

\((\neg A \lor \neg B) \land (A \lor B) \land (A \lor \neg B) \land (\neg A \lor B)\)

Satisfiable?
No

The WalkSAT algorithm

- Local search algorithm
  Incomplete: may not always find a satisfying assignment even if one exists

- Evaluation function?
  \(= \text{Number of unsatisfied clauses}\)
  WalkSAT tries to minimize this function

- Balance between greediness and randomness
The **WalkSAT algorithm**

```plaintext
function WalkSAT(clauses, p, max-flips) returns a satisfying model or failure
inputs: clauses, a set of clauses in propositional logic
        p, 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
    clause ← a randomly selected 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
```

Greed Randomness

---

**Hard Satisfiability Problems**

- Consider random 3-CNF sentences, e.g.,
  
  \[ (\neg D \lor \neg B \lor C) \land (B \lor \neg A \lor \neg C) \land (\neg C \lor \neg B \lor E) \land (E \lor \neg D \lor B) \land (B \lor E \lor \neg C) \]

  \[ m = \text{number of clauses} \]
  \[ n = \text{number of symbols} \]

  Hard instances of SAT seem to cluster near \( m/n = 4.3 \) (critical point)
Hard Satisfiability Problems

Median runtime for 100 satisfiable random 3-CNF sentences, $n = 50$
What about me?

Putting it all together: Logical Wumpus Agents

A wumpus-world agent using propositional logic:

\[-P_{1,1}\]
\[-W_{1,1}\]

For \(x = 1, 2, 3, 4\) and \(y = 1, 2, 3, 4\), add (with appropriate boundary conditions):

\[B_{x,y} \leftrightarrow (P_{x,y+1} \lor P_{x,y-1} \lor P_{x+1,y} \lor P_{x-1,y})\]
\[S_{x,y} \leftrightarrow (W_{x,y+1} \lor W_{x,y-1} \lor W_{x+1,y} \lor W_{x-1,y})\]

\[W_{1,1} \lor W_{1,2} \lor \ldots \lor W_{4,4}\]
\[\neg W_{1,1} \lor \neg W_{1,2}\]
\[\neg W_{1,1} \lor \neg W_{1,3}\]
\[\ldots\]

\(\Rightarrow 64\) distinct proposition symbols, 155 sentences!
function PL-WUMPUS-AGENT( percept ) returns an action
inputs: percept, a list. [stench, breeze, glitter]
static: KB, initially containing the “physics” of the wumpus world
z, y, orientation, the agent’s position (init. [1,1]) and orient. (init. right)
visited, an array indicating which squares have been visited, initially false
action, the agent’s most recent action, initially null
plan, an action sequence, initially empty

update z, y, orientation, visited based on action
if stench then TELL( KB, S_{x,y} ) else TELL( KB, \neg S_{x,y} )
if breeze then TELL( KB, B_{x,y} ) else TELL( KB, \neg B_{x,y} )
if glitter then action \leftarrow grab
else if plan is nonempty then action \leftarrow POP( plan )
else if for some fringe square \([i,j]\), ASK( KB, (\neg P_{i,j} \land \neg W_{i,j}) ) is true or
for some fringe square \([i,j]\), ASK( KB, (P_{i,j} \lor W_{i,j}) ) is false then do
plan \leftarrow A^*\text{-GRAPH-SEARCH}( ROUTE-PB( x, y, orientation, [i,j], visited ) )
action \leftarrow POP( plan )
else action \leftarrow a randomly chosen move
return action

Limitations of propositional logic

- KB contains "physics" sentences for every single square

- For every time step \( t \) and every location \([x,y]\), we need to add to the KB:
  \[ L_{x,y}^t \land FacingRight^t \land Forward^t \implies L_{x+1,y}^{t+1} \]

- Rapid proliferation of sentences
What we’d like is a way to talk about objects and groups of objects, and to define relationships between them.

Use: First-order logic
(aka “predicate logic”)

Propositional vs. First-Order

Propositional logic
Facts: p, q, ¬r, ¬P₁₁, ¬W₁₁ etc.
(p ∧ q) v (¬r v q ∧ p)

First-order logic
Objects: George, Monkey2, Raj, 473Student1, etc.
Relations:
Curious(George), Curious(473Student1), ...
Smarter(473Student1, Monkey2)
Smarter(Monkey2, Raj)
Stooges(Larry, Moe, Curly)
PokesInTheEyes(Moe, Curly)
PokesInTheEyes(473Student1, Raj)
FOL Definitions

- **Constants**: George, Monkey2, etc.
  Name a specific object.
- **Variables**: X, Y.
  Refer to an object without naming it.
- **Functions**: banana-of, grade-of, etc.
  Mapping from objects to objects.
- **Terms**: banana-of(George), grade-of(stdnt1)
  Refer to objects
- **Relations**: Curious, PokesInTheEyes, etc.
  State relationships between objects.
- **Atomic Sentences**: PokesInTheEyes(Moe, Curly)
  Can be true or false
  Correspond to propositional symbols P, Q

More Definitions

- **Logical connectives**: and, or, not, ⇒, ⇔
- **Quantifiers**:
  ∀ For all (Universal quantifier)
  ∃ There exists (Existential quantifier)
- **Examples**
  George is a monkey and he is curious
  \( \text{Monkey}(George) \land \text{Curious}(George) \)
  Monkeys are curious
  \( \forall m: \text{Monkey}(m) \Rightarrow \text{Curious}(m) \)
  There is a curious monkey
  \( \exists m: \text{Monkey}(m) \land \text{Curious}(m) \)
Quantifier / Connective Interaction

\[ M(x) = \text{“}x\text{ is a monkey”} \]
\[ C(x) = \text{“}x\text{ is curious”} \]

1. \( \forall x: M(x) \land C(x) \)
   “Everything is a curious monkey”

2. \( \forall x: M(x) \implies C(x) \)
   “All monkeys are curious”

3. \( \exists x: M(x) \land C(x) \)
   “There exists a curious monkey”

4. \( \exists x: M(x) \implies C(x) \)
   “There exists an object that is either a curious monkey, or not a monkey at all”

Nested Quantifiers: Order matters!

\[ \forall x \exists y \ P(x,y) \neq \exists y \forall x \ P(x,y) \]

- **Examples**
  Every monkey has a tail
  \[ \forall m \exists t \ \text{has}(m,t) \]
  Every monkey *shares* a tail!
  \[ \exists t \forall m \ \text{has}(m,t) \]

Try:
Everybody loves somebody *vs.* Someone is loved by everyone
Next Time

- FOL for the Wumpus
- Reasoning in FOL