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?

Can't get
¬satisfaction

3

Why Satisfiability?

- Recall: $KB \models \alpha$ iff $KB \wedge \neg\alpha$ 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

4

Satisfiability Examples

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

$(\neg A \vee \neg B) \wedge (A \vee B) \wedge (A \vee \neg B)$
Satisfiable?
Yes (e.g., A = true, B = false)

$(\neg A \vee \neg B) \wedge (A \vee B) \wedge (A \vee \neg B) \wedge (\neg A \vee B)$
Satisfiable?
No

5

The WalkSAT algorithm

- Local search algorithm
  - Incomplete: may not always find a satisfying assignment even if one exists
- Evaluation function?
  = Number of unsatisfied clauses
  - WalkSAT tries to minimize this function
- Balance between greediness and randomness

6

The WalkSAT algorithm

```
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**

7

Hard Satisfiability Problems

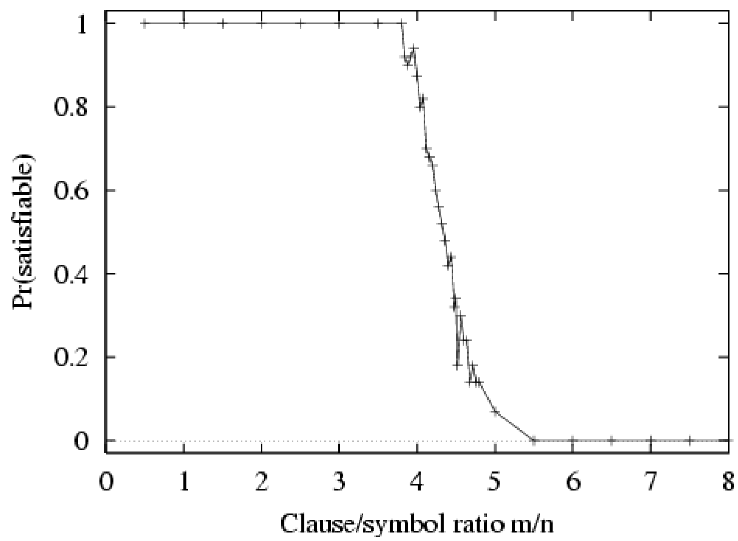
- Consider random 3-CNF sentences. e.g., $(\neg D \vee \neg B \vee C) \wedge (B \vee \neg A \vee \neg C) \wedge (\neg C \vee \neg B \vee E) \wedge (E \vee \neg D \vee B) \wedge (B \vee E \vee \neg C)$

  m = number of clauses
  n = number of symbols

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

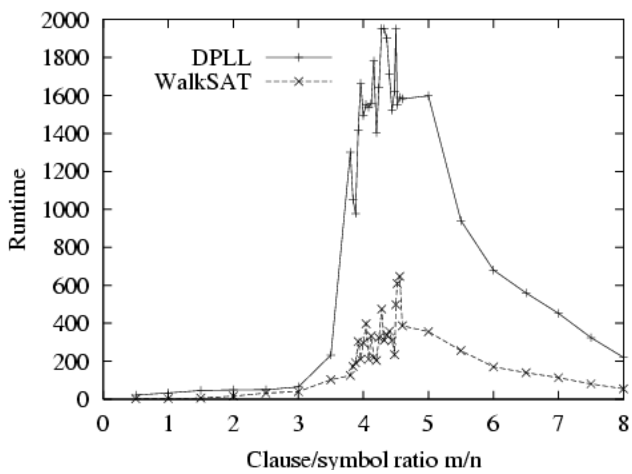
8

Hard Satisfiability Problems



9

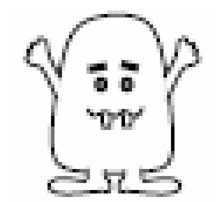
Hard Satisfiability Problems



Median runtime for 100 satisfiable random 3-CNF sentences, $n = 50$

10

What about me?



Putting it all together: Logical Wumpus Agents

A wumpus-world agent using propositional logic:

$\neg P_{1,1}$
$\neg W_{1,1}$
For x = 1, 2, 3, 4 and y = 1, 2, 3, 4, add (with appropriate boundary conditions):
$B_{x,y} \Leftrightarrow (P_{x,y+1} \lor P_{x,y-1} \lor P_{x+1,y} \lor P_{x-1,y})$
$S_{x,y} \Leftrightarrow (W_{x,y+1} \lor W_{x,y-1} \lor W_{x+1,y} \lor W_{x-1,y})$
$W_{1,1} \lor W_{1,2} \lor \ldots \lor W_{4,4}$
$\neg W_{1,1} \lor \neg W_{1,2}$
$\neg W_{1,1} \lor \neg W_{1,3}$
…

$\Rightarrow$ 64 distinct proposition symbols, 155 sentences!

12

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
    x, *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 *x*,*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* ← *grab*
  else if *plan* is nonempty **then** *action* ← POP(*plan*)
  else **if** for some fringe square [*i*,*j*], ASK($KB, (\neg P_{i,j} \wedge \neg W_{i,j})$) is *true* **or**
    for some fringe square [*i*,*j*], ASK($KB, (P_{i,j} \vee W_{i,j})$) is *false* **then do**
    plan ← A*-GRAPH-SEARCH(ROUTE-PB([*x*,*y*], *orientation*, [*i*,*j*], *visited*))
    action ← POP(*plan*)
  else *action* ← a randomly chosen move
  return *action*

13

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:

$$L^{t}_{x,y} \wedge FacingRight^{t} \wedge Forward^{t} \Rightarrow L^{t+1}_{x+1,y}$$

- Rapid proliferation of sentences

14

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, $\neg P_{1,1}$, $\neg W_{1,1}$ 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)

16

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

17

More Definitions

- **Logical connectives**: and, or, not, $\Rightarrow$, $\Leftrightarrow$
- **Quantifiers**:
  - $\forall$ For all (Universal quantifier)
  - $\exists$ There exists (Existential quantifier)
- **Examples**
  - George is a monkey and he is curious
    Monkey(George) $\wedge$ **Curious(George)**
  - Monkeys are curious
    $\forall$**m: Monkey(m)** $\Rightarrow$ **Curious(m)**
  - There is a curious monkey
    $\exists$**m: Monkey(m)** $\wedge$ **Curious(m)**

18

Quantifier / Connective Interaction

M(x) == "x is a monkey"
C(x) == "x is curious"

1. $\forall x$: $M(x) \wedge C(x)$
   "Everything is a curious monkey"
2. $\forall x$: $M(x) \Rightarrow C(x)$
   "All monkeys are curious"
3. $\exists x$: $M(x) \wedge C(x)$
   "There exists a curious monkey"
4. $\exists x$: $M(x) \Rightarrow C(x)$
   "There exists an object that is *either* a curious monkey, *or* not a monkey at all"

19

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

$\forall m \exists t\ has(m,t)$

?

Every monkey ***shares*** a tail!

$\exists t \forall m\ has(m,t)$

Try:

Everybody loves somebody *vs.* Someone is loved by everyone

20

Next Time

- FOL for the Wumpus
- Reasoning in FOL

21