EECS 391: Introduction to AI

Soumya Ray

Website: http://engr.case.edu/ray_soumya/eeCS391_sp15/
Email: sray@case.edu
Office: Olin 516
Office hours: F 11:30-2pm (email if stopping by between 12:30-2) or by appointment
Announcements

• PA2 out today
• Read Chapter 9 (First order logic inference)
Today

• Semantic Inference in Propositional Logic (chapter 7)

• First Order Logic (chapter 8)
Semantic Inference

• So far, we have considered algorithms for syntactic inference
  – These algorithms work by using inference rules to produce derivations

• Now we will look at algorithms that perform semantic inference
  – Inference via models and entailment (variation on inference by enumeration)
  – For propositional logic, this means considering \textit{satisfying assignments}
  – Also called inference via “model checking”
Definitions

• A formula $\alpha$ is **valid** if it is true in every model (tautology)

• A formula $\alpha$ is **satisfiable** if it is true in some model
Satisfiability and Inference

- Satisfiability and inference are linked by the proof-by-refutation theorem: $\alpha \vdash \beta$ iff $(\alpha \land \neg \beta)$ is unsatisfiable

- Satisfiability checking algorithms use search
  - Goal-Directed search (DPLL)
  - Local search (WalkSAT)
Satisfiability by Goal-Directed Search

• Given a CNF formula, I want to find out if it is satisfiable by checking possible models

\[(A \lor \neg B) \land (\neg B \lor \neg C) \land (C \lor A)\]

• How can we set up the problem?
  – States=value assignments to symbols
  – Initial state, goal state, operators, cost?
DPLL
(Davis, Putnam, Logemann, Loveland 1960)

• Input: formula in CNF
• Output: satisfying assignment if any
• Algorithm: Depth first search of space of all models
  – Uses heuristics to guide search (prune search space)
  – Complete (as in search)
$$\left( A \lor \neg B \right) \land \left( \neg B \lor \neg C \right) \land \left( C \lor A \right)$$
\[(A \lor \neg B) \land (\neg B \lor \neg C) \land (C \lor A)\]
Heuristic 1: Pure literals

• Suppose a CNF formula has a symbol that appears only negated or only unnegated in all clauses
  – This is a “pure” literal
  – If the formula is satisfiable, it is satisfiable with the pure literal set to true
  – Example:

\[ (A \lor \neg B) \land (\neg B \lor \neg C) \land (C \lor A) \]
Heuristic 2: Unit Propagation

• Suppose a clause has only a single literal (because everything else is assigned false)
• Then that literal has to be set to true
  – This can cascade
  – Example

\[
(A \lor \neg B) \land (\neg B \lor \neg C) \land (C \lor A)
\]

\[
B = true
\]
Heuristic 3: Early Termination

• Since the formula is in CNF, for any assignment to be a satisfying one, every clause has to be true
  – Any (partial) assignment that makes any clause false can be pruned

• Since a clause is a disjunction, once any literal is set to true, that clause does not have to be considered any more
DPLL

• Start with initial list of clauses and a current assignment (initially empty)
• If all clauses true return true
• If any clause false return false
• If :
  – There is an unassigned pure literal, set to true and call DPLL on result
  – Else if there is an unassigned unit clause, set to true and call DPLL on result
  – Else pick a random symbol, return DPLL with symbol set to true OR’ed with DPLL with symbol set to false
Satisfiability by Optimization

• How can we set up the problem?

\[(A \lor \neg B) \land (\neg B \lor \neg C) \land (C \lor A)\]
WalkSAT (Kautz and Selman, 1993)

• Start with initial complete assignment
• Repeat $MAX_{FLIPS}$ times:
  – If all clauses true, return current assignment
  – Else, pick random unsatisfied clause
  – With probability $p$, flip the assignment of a random variable in clause
  – Else flip the assignment of the variable in clause that maximizes satisfied clauses
Properties of WalkSAT

• Not Complete
  – If it returns no satisfying assignment, does not imply formula is unsatisfiable
  – But it turns out that depending on the formula, we can make a really good guess about this
Why is SAT hard?

• The great success of WalkSAT led to a huge amount of research on SAT
  – People knew theoretically that SAT was supposed to be a hard problem in the worst case
  – But here was a simple local search procedure which was able to solve extremely large SAT problems extremely fast
  – How to reconcile these facts? Could we isolate the “hard” SAT problems somehow? Are these an interesting subset of SAT problems?
Hard SAT Problems

• Suppose we generate a random CNF formula

• There are two key parameters
  – The number of clauses in the formula ($M$)
  – The number of propositional symbols ($N$)

• It turns out that the ratio $M/N$ determines the hardness of a SAT problem
Intuition

• If there are many symbols and few clauses, the problem is *underconstrained*: the probability of a random assignment being satisfying is close to 1

• If there are few symbols and many clauses, the problem is *overconstrained*: the probability of a random assignment being satisfying is close to 0

• For a critical value of $M/N$, the probability of a random assignment being satisfying is close to 0.5
  – These are the hardest SAT problems
If a problem domain has this sort of characteristic, it is said to have a “phase transition.” Typically in such cases, problems near the transition are hard to solve.
Runtime Characteristics

Median runtime for 100 *satisfiable* random 3CNF formulae, $N = 50$
Summary

• We learned about:
  – Syntax, semantics, models, entailment, derivations, soundness and completeness
  – Syntactic Inference Algorithms
    • Resolution
  – Semantic Inference Algorithms
    • Systematic: DPLL
    • Local Search: WalkSAT
  – Phase Transition Characteristics of SAT
Issues with Propositional Logic

• Want to say: “Somewhere in the grid there is one wumpus.”

\[(W_{1,1} \land \neg W_{1,2} \land \ldots \land \neg W_{m,n}) \lor\]

\[(\neg W_{1,1} \land W_{1,2} \land \ldots \land \neg W_{m,n}) \lor \ldots\]
First Order Logic

- Same thing in FOL:

$$\exists x, y \ Cell(x, y) \land Contains(x, y, Wumpus) \land$$

$$[\forall a, b \ (Cell(a, b) \land (a \neq x \lor b \neq y)) \Rightarrow \neg Contains(a, b, Wumpus))]$$

Quantifiers:
- “There exists”,
- “For all”

Also: Function symbols
Key Elements of FOL

• FOL introduces:
  – Objects
  – Relations between objects (predicates)
  – Functions that map objects to other objects
  – Variables that act as placeholders for objects
  – Quantifiers that allow us to talk about (infinite) collections of objects
Functions and Predicates

• A **function** takes a set of objects and returns an object, e.g. $Mother(A) = B$

• A **predicate** takes a set of objects and returns true/false, e.g. $isMotherOf(A,B)$?