Knowledge Representation II
(Inference in Propositional Logic)

CSE 473
Continued...

Last Time

• Inference Algorithms
  As search
  Systematic & stochastic  [[skipped this]]
• Themes
  Expressivity vs.
  Tractability

Special Syntactic Forms: CNF

• General Form:
  \(((\neg q \land r) \lor s) \land \neg(s \land t)\)

• Conjunction Normal Form (CNF)
  \((\neg q \lor r \lor s) \land (\neg s \lor \neg t)\)
  Set notation: \{ (\neg q, r, s), (\neg s, \neg t) \}
  empty clause () = false

Resolution

If the unicorn is mythical, then it is immortal, but
if it is not mythical, it is a mammal. If the
unicorn is either immortal or a mammal, then it
is horned.
Prove: the unicorn is horned.

\[
\begin{align*}
(A \lor H) \
(M \lor A) \
(\neg H) \
(\neg I \lor H) \
(M \lor \neg I) \
(\neg M)
\end{align*}
\]

\[
\begin{align*}
M = \text{mythical} \\
I = \text{immortal} \\
A = \text{mammal} \\
H = \text{horned}
\end{align*}
\]

Inference 4: DPLL
(Enumeration of Partial Models)
[Davis, Putnam, Loveland & Logemann 1962]
Version 1

\[
dpll_1(pa)\
  \text{if (pa makes F false) return false;}
  \text{if (pa makes F true) return true;}
  \text{choose P in F;}
  \text{if (dpll_1(pa U \{P=0\})) return true;}
  \text{return dpll_1(pa U \{P=1\});}
\]

Returns true if F is satisfiable, false otherwise

Improving DPLL

If literal \(L_i\) is true, then clause \((L_i \lor L_j \lor \ldots)\) is true
If clause \(C_i\) is true, then \(C_i \land C_j \land C_k \land \ldots\) has the same
value as \(C_i \land C_j \land \ldots\)
Therefore: Okay to delete clauses containing true literals!
If literal \(L_i\) is false, then clause \((L_i \lor L_j \lor L_k \lor \ldots)\) has
the same value as \((L_i \lor L_j \lor \ldots)\)
Therefore: Okay to delete clauses containing false literals!
If literal \(L_i\) is false, then clause \((L_i)\) is false
Therefore: the empty clause means false!
Further Improvements

• Unit Literals
  A literal that appears in a singleton clause
  \{\neg b \land c \land \neg a \land \neg e \land d \land b \land (e \land a \land \neg c)\}
  * Might as well set it true! And simplify
  \{\neg b\} \land (a \land \neg b \land e \land d \land b \land \neg (e \land a \land \neg c)\}

• Pure Literals
  A symbol that always appears with same sign
  \{(a \land \neg b \land c) \land (\neg c \land \neg a \land \neg b \land e \land d \land b \land (e \land a \land \neg c)\}
  * Might as well set it true! And simplify
  \{(a \land \neg b \land c) \land (\neg a \land \neg b \land e) \land (e \land a \land \neg c)\}

DPLL (for real!)

\[
dpll(F, \text{ literal})
\]
\[
\begin{aligned}
&\text{remove clauses containing literal} \\
&\text{shorten clauses containing } \neg \text{literals} \\
&\text{if (F contains no clauses) return true;} \\
&\text{if (F contains empty clause) return false;} \\
&\text{if (F contains a unit or pure L) return dpll(F, L);} \\
&\text{choose V in F;} \\
&\text{if (dpll(F, } \neg V)\) return true; \\
&\text{return } dpll_2(F, V) \}
\]

Heuristic Search in DPLL

• Heuristics are used in DPLL to select a (non-unit, non-pure) proposition for branching
• Idea: identify a most constrained variable
  Likely to create many unit clauses
• MOM's heuristic:
  Most occurrences in clauses of minimum length

Success of DPLL

• 1962 - DPLL invented
• 1992 - 300 propositions
• 1997 - 600 propositions (satz)
• Additional techniques:
  Learning conflict clauses at backtrack points
  Randomized restarts
  2002 (zChaff) 1,000,000 propositions - encodings of hardware verification problems

Outline

• Inference Algorithms
  As search
  Systematic & stochastic
• Themes
  Expressivity vs. Tractability
WalkSat

- Local search over space of complete truth assignments
  - With probability P: flip any variable in any unsatisfied clause
  - With probability (1-P): flip best variable in any unsat clause
    - Like fixed-temperature simulated annealing

- SAT encodings of N-Queens, scheduling
- Best algorithm for random K-SAT
  - Best DPLL: 700 variables
  - Walksat: 100,000 variables

Random 3-SAT

- Random 3-SAT
  - Sample uniformly from space of all possible 3-clauses
  - n variables, l clauses

- Which are the hard instances?
  - Around l/n = 4.3

Random 3-SAT

- Varying problem size, n
- Complexity peak appears to be largely invariant of algorithm
  - Backtracking algorithms like Davis-Putnam
  - Local search procedures like GSAT

- What's so special about 4.3?

Random 3-SAT

- Complexity peak coincides with solubility transition
  - l/n < 4.3 problems under-constrained and SAT
  - l/n > 4.3 problems over-constrained and UNSAT
  - l/n=4.3, problems on "knife-edge" between SAT and UNSAT

Real-World Phase Transition Phenomena

- Many NP-hard problem distributions show phase transitions
  - Job shop scheduling problems
  - TSP instances from TSPLib
  - Exam timetables @ Edinburgh
  - Boolean circuit synthesis
  - Latin squares (alias sports scheduling)

- Hot research topic: predicting hardness of a given instance, & using hardness to control search strategy (Horvitz, Kautz, Ruan 2001-3)

Restricted Expressiveness

- 2-SAT
- Horn Theories
Horn Theories

- Horn clause ≡ at most one positive literal
  \{ (\neg q \lor \neg r \lor s), (\neg s \lor \neg t) \}
  \{(q \land r) \Rightarrow s), (s \land t) \Rightarrow false\}\}

- Many problems naturally take the form of such if/then rules
  If (fever) AND (vomiting) then FLU

- Unit propagation is refutation complete for Horn theories
  Good implementation - linear time!

Summary: Algorithms

- Forward Chaining
- Resolution
- Model Enumeration
- Enumeration of Partial Models (DPLL)
- Walksat

Themes

- Expressiveness
  Expressive but awkward
  No notion of objects, properties, or relations
  Number of propositions is fixed

- Tractability
  NPC in general
  Completeness / speed tradeoff
  Horn clauses, binary clauses