Knowledge Representation II
(Inference in Propositional Logic)

Propositional Logic
- Syntax
  - Atomic sentences: $P, Q, \ldots$
  - Connectives: $\land, \lor, \neg, \Rightarrow$
- Semantics
  - Truth Tables
- Inference
  - Modus Ponens
  - Resolution
  - DPLL
  - GSAT
- Complexity

Satisfiability, Validity, & Entailment
- $S$ is **satisfiable** if it is true in *some* world
- $S$ is **unsatisfiable** if it is false *all* worlds
- $S$ is **valid** if it is true in *all* worlds
- $S_1$ **entails** $S_2$ if wherever $S_1$ is true $S_2$ is also true

Semantics
- **Syntax**: a description of the *legal* arrangements of symbols (Def "sentences")
- **Semantics**: what the arrangement of symbols *means* in the world

Today
- **Inference Algorithms**
  - As search
    - Systematic & stochastic
- **Themes**
  - Expressivity vs. Tractability
Reasoning Tasks

- **Model Finding**
  - KB = background knowledge
  - S = description of problem
  - Show (KB ∨ S) is satisfiable
  - A kind of constraint satisfaction

- **Deduction**
  - S = question
  - Prove that KB ⊨ S
  - Two approaches:
    1. Rules to derive new formulas from old
    2. Show (KB ∨ ¬S) is unsatisfiable

Inference 1: Forward Chaining

**Forward (& Backward) Chaining**

Based on rule of *modus ponens*

If know P₁, .., Pₙ & know (P₁ ∧ .. ∧ Pₙ) ⊨ Q
Then can conclude Q

Pose as Search thru Problem Space?
- States?
- Operators?

Analysis

- **Sound?**
- **Complete?**

Can you prove

\[ \{ \} \models Q ∨ ¬Q \]

Inference 2: Resolution

[Robinson 1965]

\[ \{ (p ∨ α), (¬p ∨ β ∨ γ) \} \vdash (α ∨ β ∨ γ) \]

Correctness

- If S₁ \models S₂ then S₁ \models S₂

Refutation Completeness:

- If S is unsatisfiable then S \vdash ()

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.

\[ M = \text{mythical} \]
\[ I = \text{immortal} \]
\[ A = \text{mammal} \]
\[ H = \text{horned} \]
### Inference 3: Model Enumeration

for (m in truth assignments) {
  if (m makes \( \Phi \) true)
    then return "Sat!"
}
return "Unsat!"

View as Search?
Critique?

### DPLL Version 1

\[(a \lor b \lor c)\]
\[(a \lor \neg b)\]
\[(a \lor \neg c)\]
\[(\neg a \lor c)\]

### DPLL as Search

- Search Space?
- Algorithm?

### 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 \( C_i \land C_2 \land C_3 \land \ldots \) has the same value as \( C_i \land C_2 \land \ldots \) 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!

### DPLL version 2

\[
dpll_2(F, \text{ literal})\{
  \text{remove clauses containing literal}
  \text{shorten clauses containing} \neg \text{literal}
  \text{if (F contains no clauses) return true;}
  \text{if (F contains empty clause) return false;}
  \text{choose V in F;}
  \text{if (dpll}(F, \neg V))\text{return true;}
  \text{return dpll}_2(F, V);
\}
\]

Partial assignment corresponding to a node is the set of chosen literals on the path from the root to the node.

### Inference 4: DPLL

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

\[
dpll_1(pa)\{
  \text{if (pa makes F false) return false;}
  \text{if (pa makes F true) return true;}
  \text{choose F in F;}
  \text{if (dpll}_1(pa \cup \{F=0\})) \text{return true;}
  \text{return dpll}_1(pa \cup \{F=1\});
\}
\]

Returns true if F is satisfiable, false otherwise.
Further Improvements

Formula $(L \land C_1 \land C_2 \land \ldots)$ is only true when literal $L$ is true.

Therefore: Branch immediately on unit literals!

If literal $L$ does not appear negated in formula $F$, then setting $L$ true preserves satisfiability of $F$

Therefore: Branch immediately on pure literals!

May view this as adding constraint propagation techniques into play

DPLL (for real!

Davis - Putnam - Loveland - Logemann

dpll(F, literal){
    remove clauses containing literal
    shorten clauses containing $\neg$-literal
    if (F contains no clauses) return true;
    if (F contains empty clause)
        return false;
    if (F contains a unit or pure L)
        return dpll(F, L);
    choose V in F;
    if (dpll(F, $\neg$V)) return true;
    return dpll_2(F, V);
}

DPLL (for real!

Davis - Putnam - Loveland - Logemann

dpll(F, literal){
    remove clauses containing literal
    shorten clauses containing $\neg$-literal
    if (F contains no clauses) return true;
    if (F contains a unit or pure L)
        return dpll(F, L);
    choose P in F;
    if (dpll(F, $\neg$P)) return true;
    return dpll_2(F, P);
}

Where could we use an heuristic to further improve performance?
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

DPLL

Davis Putnam Loveland Logmann

- Model finding: Search (what kind?) over space of partial truth assignments

DPLL \( \text{wff} \):
- If \( P \) a pure literal \( P \), return DPLL\( \text{wff} \) extended by \( P \)
- For each unit clause \( \neg Y \), simplify \( \text{wff} \)
- Remove clauses containing \( Y \)
- Shorten clauses contain \( \neg Y \)
- If no clause left, return true (satisfiable)
- If \( \text{empty clause, return false} \)
- Choose a variable \( X \) & choose a value \( 0/1 \)
- If DPLL\( \text{wff} \) extended by \( X \) return true
- else return DPLL\( \text{wff}, \neg X \)

Horn Theories

- Recall the special case of Horn clauses:
  \( \{ (\neg q \lor \neg r \lor s), (\neg s \lor \neg t) \} \)
  \( \{ ((q \land r) \lor s), (s \land t) \lor \text{false} \} \)
- 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!

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, $3^i$ clauses
- Which are the hard instances? around $i/n = 4.3$

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
  
  $i/n < 4.3$ problems under-constrained and SAT
  
  $i/n > 4.3$ problems over-constrained and UNSAT
  
  $i/n = 4.3$, 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)

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