Knowledge Representation II
(Inference in Propositional Logic)

CSE 473
473 Topics

Perception  NLP  Robotics  Multi-agent

Logic  Supervised Learning  Reinforcement Learning

Inference  Knowledge Representation  Planning

Probability

Search  Problem Spaces  Agency
Propositional Logic

• **Syntax**
  Atomic sentences: P, Q, ...
  Connectives: ∧, ∨, ¬, ⊃

• **Semantics**
  Truth Tables

• **Inference**
  Modus Ponens
  Resolution
  DPLL
  GSAT

• **Complexity**
Semantics

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

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

• **Model finding**
  \begin{itemize}
  \item $KB = \text{background knowledge}$
  \item $S = \text{description of problem}$
  \item Show $(KB \land S)$ is satisfiable
  \item A kind of constraint satisfaction
  \end{itemize}

• **Deduction**
  \begin{itemize}
  \item $S = \text{question}$
  \item Prove that $KB \models S$
  \item Two approaches:
    \begin{enumerate}
    \item Rules to derive new formulas from old
    \item Show $(KB \land \neg S)$ is unsatisfiable
    \end{enumerate}
  \end{itemize}
Inference 1: Forward Chaining

Forward (& Backward) Chaining
Based on rule of *modus ponens*

If know $P_1, \ldots, P_n$ & know $(P_1 \land \ldots \land P_n) \implies Q$
Then can conclude $Q$

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

- Sound?
- Complete?

Can you prove
\[
\emptyset \models Q \lor \neg Q
\]
Special Syntactic Forms: CNF

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

• 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
Inference 2: Resolution
[Robinson 1965]

\[
\{ (p \lor \alpha), (\neg p \lor \beta \lor \gamma) \} \vdash_R (\alpha \lor \beta \lor \gamma)
\]

Correctness
If \( S1 \vdash_R S2 \) then \( S1 \models S2 \)

Refutation Completeness:
If \( S \) is unsatisfiable then \( S \vdash_R (\)
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.

\[ (\neg A \lor H) \rightarrow (\neg H) \rightarrow (\neg I \lor H) \]

\[ (M \lor A) \rightarrow (\neg A) \rightarrow (\neg I) \rightarrow (\neg M \lor I) \]

\[ (M) \rightarrow (\neg M) \rightarrow () \]

\( M = \text{mythical} \)

\( I = \text{immortal} \)

\( A = \text{mammal} \)

\( H = \text{horned} \)
Inference 3: Model Enumeration

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

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

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

Returns true if F is satisfiable, false otherwise
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 $L_1$ is true, then clause $(L_1 \lor L_2 \lor ...)$ is true
If clause $C_1$ is true, then $C_1 \land C_2 \land C_3 \land ...$ has the same value as $C_2 \land C_3 \land ...$
Therefore: Okay to delete clauses containing true literals!
If literal $L_1$ is false, then clause $(L_1 \lor L_2 \lor L_3 \lor ...)$ has the same value as $(L_2 \lor L_3 \lor ...)$
Therefore: Okay to delete shorten containing false literals!
If literal $L_1$ is false, then clause $(L_1)$ is false
Therefore: the empty clause means false!
DPLL version 2
Davis - Putnam - Loveland - Logemann

dplll_2(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;
choose V in F;
if (dplll(F, \( \neg \)V)) return true;
return dplll_2(F, V);
}

Partial assignment corresponding to a node is the
set of chosen literals on the path from the root
to the node
DPLL Version 2

((\neg a \lor c) 
\land (\neg \neg b \lor \neg c) 
\land (\neg \neg \neg b \lor \neg \neg c))
Structure in Clauses

• **Unit Literals**

  A literal that appears in a singleton clause
  \[\{\neg b \ c\}\{\neg c\}\{a \neg b \ e\}\{d \ b\}\{e \ a \neg c\}\]

  *Might as well set it true!* And simplify
  \[\{\neg b\}\{a \neg b \ e\}\{d \ b\}\{d\}\]

• **Pure Literals**

  A symbol that always appears with same sign
  \[\{a \neg b \ c\}\{\neg c \ d \neg e\}\{\neg a \neg b \ e\}\{d \ b\}\{e \ a \neg c\}\]

  *Might as well set it true!* And simplify
  \[\{a \neg b \ c\}\{\neg a \neg b \ e\}\{e \ a \neg c\}\]
Further Improvements

Formula \((L) \land C_2 \land C_3 \land \ldots\) 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 ¬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, ¬V)) return true;
    return dpll_2(F, V);
}
DPLL (for real)

Note: bug in animation?!?
DPLL (for real!)
Davis - Putnam - Loveland - Logemann

dplll(F, literal)\
remove clauses containing literal\
shorten clauses containing ¬literal\
if (F contains no clauses) return true;
if (F contains a unit or pure L)\
    return dplll(F, L);
choose P in F;
if (dplll(F, ¬P)) return true;
return dplll_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( wff ):
  If ∃ a pure literal P, return DPLL(wff extended by P)
  For each unit clause (Y), simplify wff:
    Remove clauses containing Y
    Shorten clauses contain ¬Y
  If no clause left, return true (satisfiable)
  If ∃ empty clause, return false
  Choose a variable X & choose a value (0/1)
  If DPLL(wff extended by X ) return true
  else return DPLL(wff, ¬X)

© Daniel S. Weld
DPLL

• Developed 1962 – still the best complete algorithm for propositional reasoning

• State of the art solvers use:
  Smart variable choice heuristics
  “Clause learning” – at backtrack points, determine minimum set of choices that caused inconsistency, add new clause
  • Limited resolution (Agarwal, Kautz, Beame 2002)
  Randomized tie breaking & restarts

• Chaff – fastest complete SAT solver
  Created by 2 Princeton undergrads, for a summer project!
  Superscaler processor verification
  AI planning - Blackbox
Horn Theories

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

- 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, $\ell$ clauses

- Which are the hard instances? around $\ell/n = 4.3$
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
  
- $1/n < 4.3$ problems under-constrained and SAT
  
- $1/n > 4.3$ problems over-constrained and UNSAT
  
- $1/n = 4.3$, problems on “knife-edge” between SAT and UNSAT

© Daniel S. Weld
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