CSE 473 Propositional Logic
SAT Algorithms
Dan Weld
(With some slides from Mausam, Stuart Russell, Dieter Fox, Henry Kautz, Min-Yen Kan...)

Irrationally held truths may be more harmful than reasoned errors.
- Thomas Huxley (1825-1895)

Overview
- Introduction & Agents
- Search, Heuristics & CSPs
- Adversarial Search
- Logical Knowledge Representation
- Planning & MDPs
- Reinforcement Learning
- Uncertainty & Bayesian Networks
- Machine Learning
- NLP & Special Topics

Propositional Logic

• Syntax
  – Atomic sentences: P, Q, ...
  – Connectives: ∧, ∨, ¬, ⇒

• Semantics
  – Truth Tables

• Inference
  – Modus Ponens
  – Resolution
  – DPLL
  – GSAT

• Complexity

Semantics

• Syntax: which arrangements of symbols are legal
  – (Def "sentences")
• Semantics: what the symbols mean in the world
  – (Mapping between symbols and worlds)

Truth tables for connectives

<table>
<thead>
<tr>
<th>P</th>
<th>Q</th>
<th>¬P</th>
<th>P ∧ Q</th>
<th>P ∨ Q</th>
<th>P ⇒ Q</th>
<th>P ⇔ Q</th>
</tr>
</thead>
<tbody>
<tr>
<td>false</td>
<td>false</td>
<td>true</td>
<td>false</td>
<td>false</td>
<td>true</td>
<td>false</td>
</tr>
<tr>
<td>false</td>
<td>true</td>
<td>false</td>
<td>false</td>
<td>true</td>
<td>false</td>
<td>false</td>
</tr>
<tr>
<td>true</td>
<td>false</td>
<td>false</td>
<td>false</td>
<td>true</td>
<td>false</td>
<td>false</td>
</tr>
<tr>
<td>true</td>
<td>true</td>
<td>false</td>
<td>false</td>
<td>true</td>
<td>false</td>
<td>false</td>
</tr>
</tbody>
</table>

Models

• Logicians often think in terms of models, which are formally structured worlds with respect to which truth can be evaluated
  – In propositional case, each model = truth assignment
  – Set of models can be enumerated in a truth table

• We say m is a model of a sentence α if α is true in m
  – M[α] is the set of all models of α
  – Then KB ⊨ α iff M[KB] ⊆ M[α]
  
  E.g. KB = \{P \lor Q, \neg (P \land R)\}
  α = \{P, R\}

• How to check?
  – One way is to enumerate all elements in the truth table – slow
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
- **S1 entails S2** if wherever S1 is true S2 is also true

Types of Reasoning (Inference)

- **Deduction** (showing entailment, \( \models \))
  
  - \( S \) = question
  
  Prove that \( KB \models S \)
  
  - Two approaches:
    - Rules to derive new formulas from old (inference)
    - Show \( KB \land \neg S \) is unsatisfiable

- **Model Finding** (showing satisfiability)
  
  - \( S \) = description of problem
  
  Show \( S \) is satisfiable
  
  A kind of constraint satisfaction

---

Propositional Logic:

- **Inference Algorithms**

  1. Backward & Forward Chaining
  2. Resolution (Proof by Contradiction)
  3. Exhaustive Enumeration
  4. DPLL (Davis, Putnam Loveland & Logemann)
  5. GSAT

Wumpus World

- **Performance measure**
  
  - Gold: +1000, death: -1000
  
  - -1 per step, -10 for using the arrow

- **Environment**
  
  - Squares adjacent to wumpus are smelly
  
  - Squares adjacent to pit are breezy
  
  - Glitter iff gold is in the same square
  
  - Shooting kills wumpus if you are facing it
  
  - Shooting uses up the only arrow
  
  - Grabbing picks up gold if in same square
  
  - Releasing drops the gold in same square

- **Sensors**: Stench, Breeze, Glitter, Bump, Scream
- **Actuators**: Left turn, Right turn, Forward, Grab, Release, Shoot

Exploring a wumpus world
Exploring a wumpus world

Wumpus world sentences: KB

Let $P_{i,j}$ be true if there is a pit in $[i, j]$.
Let $B_{i,j}$ be true if there is a breeze in $[i, j]$.

KB:

- $\neg P_{1,1}$
- $\neg B_{1,1}$

“Pits cause breezes in adjacent squares”

$B_{1,1} \Leftrightarrow (P_{1,2} \lor P_{2,1})$
$B_{2,1} \Leftrightarrow (P_{1,1} \lor P_{2,2} \lor P_{3,1})$

Full Encoding of Wumpus World

In propositional logic:

- $\neg P_{1,1}$
- $\neg W_{1,1}$
- $B_{1,1} \Leftrightarrow (P_{1,2} \lor P_{2,1} \lor P_{3,1} \lor P_{4,1})$
- $S_{1} \Leftrightarrow (W_{1,2} \lor W_{2,1} \lor W_{2,3} \lor W_{4,1})$
- $W_{1,1} \lor W_{1,2} \lor \ldots \lor W_{4,4}$
- $\neg W_{1,1} \lor \neg W_{1,2}$
- $\neg W_{1,2} \lor \neg W_{1,3}$
- $\ldots$

$\Rightarrow$ 64 distinct proposition symbols, 155 sentences

Model Finding

- Find assignments to variables that makes a formula true

$\alpha = \neg P_{1,2}$ (ie "[1,2] is safe")

- Note: this is a kind of CSP, right?
State Estimation

- Maintaining a KB which records what you know about the (partially observed) world state
  - Prop logic
  - First order logic
  - Probabilistic encodings

Propositional Logic: Inference Algorithms

1. Backward & Forward Chaining
2. Resolution (Proof by Contradiction)
3. Exhaustive Enumeration
4. DPLL (Davis, Putnam, Loveland & Logemann)
5. GSAT

Inference 3: Model Enumeration

```python
for (m in truth assignments){
  if (m makes φ true) then return "Sat!"
}
return "Unsat!"
```

Inference 4: DPLL (Enumeration of Partial Models)

```
dpll_1(pa){
  if (pa makes F false) return false;
  if (pa makes F true) return true;
  choose F in F;
  if (dpll_1(pa ∪ {F=0})) return true;
  return dpll_1(pa ∪ {F=1});
}
```

Returns true if F is satisfiable, false otherwise
DPLL Version 1

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

\(\neg \neg a \lor a\)

DPLL Version 1

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

DPLL Version 1

\[(F \lor b \lor c)\]
\[(F \lor \neg b)\]
\[(F \lor \neg c)\]
\[(T \lor c)\]

\(T \lor \neg c\)

DPLL Version 1

\[(F \lor F \lor c)\]
\[(F \lor T)\]
\[(F \lor \neg c)\]
\[(T \lor c)\]

\(T \lor \neg c\)

DPLL Version 1

\[(F \lor F \lor F)\]
\[(F \lor T)\]
\[(F \lor T)\]
\[(T \lor F)\]

\(T \lor F\)

DPLL Version 1

\[F\]
\[T\]
\[T\]
\[T\]

\(T\)
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 \( C_i \land C_j \land C_j \land \ldots \) has the same value as \( C_i \land C_j \land \ldots \)
Therefore: Okay to delete clauses containing true literals!

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 \( C_i \land C_j \land C_j \land \ldots \) 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 \( (L_i \lor L_j \lor \ldots) \) has the same value as \( (L_i \lor L_j \lor \ldots) \)
Therefore: Okay to delete clauses containing false literals!

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 \( C_i \land C_j \land C_j \land \ldots \) 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 \( (L_i \lor L_j \lor \ldots) \) has the same value as \( (L_i \lor L_j \lor \ldots) \)
Therefore: Okay to delete clauses containing false literals!

Representing Formulae

• \( \text{CNF} = \text{Conjunctive Normal Form} \)
  – Conjunction (\( \land \)) of Disjunctions (\( \lor \))
• Represent as set of sets
  – \( ((A, B), (\neg A, C), (\neg C)) \)
  – \( ((\neg A), (A)) \)
  – \( (() \)  
  – \( ((A)) \)
  – \( () \)
DPLL version 2

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

(a \lor b \lor c)
(a \lor \neg b)
(a \lor \neg c)
(\neg a \lor c)
**Benefit**

- Can backtrack before getting to leaf

**Structure in Clauses**

- **Unit Literals**
  A literal that appears in a singleton clause
  \[\{\neg b\ c\}\{a \rightarrow b\ e\}\{d\ b\}\{e\ a \rightarrow c\}\]\n  *Might as well set it true! And simplify*
  \[\{\neg b\}\{a \rightarrow b\ e\}\{d\ b\}\{e\}\{d\}\{e\ a \rightarrow c\}\]\n
- **Pure Literals**
  - A symbol that always appears with same sign
  - \[\{(a \rightarrow b\ c)\ (\neg c\ d\ \neg e)\ (\neg a \rightarrow b\ e)\ (d\ b)\ (e\ a\ \neg c)\]\n  *Might as well set it true! And simplify*
  \[\{(a \rightarrow b\ c)\ (\neg a\ \neg b\ e)\ (e\ a\ \neg c)\}\]

**In Other Words**

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

Therefore: Branch immediately on unit literals!

May view this as adding constraint propagation techniques into play

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

Therefore: Branch immediately on pure literals!

May view this as adding constraint propagation techniques into play
DPLL (previous version)
Davis – Putnam – Loveland – Logemann

dpll(F, literal)
remove clauses containing literal
if (F contains no clauses) return true;
shorten clauses containing \neg literal
if (F contains empty clause) return false;

choose V in F;
if (dpll(F, \neg V)) return true;
return dpll(F, V);

DPLL (for real)

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

Compare with DPLL Version 1

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

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

Where could we use a heuristic to further improve performance?
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

Other Ideas?

- How else could we solve SAT problems?

WalkSat (Take 1)

- **Local** search (Hill Climbing + Random Walk) over space of complete truth assignments
  - With prob p: flip any variable in any unsatisfied clause
  - With prob (1-p): flip best variable in any unsat clause
  - best = one which minimizes #unsatisfied clauses

Refining Greedy Random Walk

- Each flip
  - makes some false clauses become true
  - breaks some true clauses, that become false
- Suppose \( s_1 \rightarrow s_2 \) by flipping \( x \). Then:
  \[ \#\text{unsat}(s_2) = \#\text{unsat}(s_1) - \text{make}(s_1, x) + \text{break}(s_1, x) \]
- Idea 1: if a choice breaks nothing, it’s likely good!
- Idea 2: near the solution, only the break count matters
  - the make count is usually 1

Walksat (Take 2)

\[
\begin{align*}
\text{state} &= \text{random truth assignment}; \\
\text{while} \! & \! \text{!GoalTest(state) do} \\
& \quad \text{clause} := \text{random member } \{ C \mid C \text{ is false in state } \}; \\
& \quad \text{for each } x \text{ in clause do compute } \text{break}[x]; \\
& \quad \text{if exists } x \text{ with } \text{break}(x) = 0 \text{ then } \text{var} := x; \\
& \quad \text{else with probability } p \text{ do} \\
& \quad \quad \text{var} := \text{random member } \{ x \mid x \text{ is in clause } \}; \\
& \quad \text{else with probability } 1 - p \text{ do} \\
& \quad \quad \text{var} := \text{arg min } \{ \text{break}[x] \mid x \text{ is in clause } \}; \\
& \quad \text{end if} \\
& \quad \text{state[var]} := 1 - \text{state[var]}; \\
\end{align*}
\]

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?

Special Syntactic Forms

- General Form:
  $((q \land \neg r) \rightarrow s) \land \neg (s \land t)$
- Conjunction Normal Form (CNF)
  $(-q \lor r \lor s) \land (-s \lor -t)$
  Set notation: $\{ (-q, r, s), (-s, -t) \}$
  empty clause $() = \text{false}$
- Binary clauses: 1 or 2 literals per clause
  $(-q \lor r)$  $(-s \lor -t)$
- Horn clauses: 0 or 1 positive literal per clause
  $(-q \lor -r \lor s)$  $(-s \lor -t)$
  $(q \land r) \rightarrow s$  $(s \land t) \rightarrow \text{false}$

Prop. Logic Themes

- Expressiveness
  Expressive but awkward
  No notion of objects, properties, or relations
  Number of propositions is fixed
- Tractability
  NP in general
  Completeness / speed tradeoff
  Horn clauses, binary clauses