Inference in Propositional Logic (and Intro to SAT)

CSE 473

Today

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

Reasoning Tasks

- Model finding (SAT)
  - KB = background knowledge
  - S = description of problem
  - Show (KB $\wedge$ S) is satisfiable
  - A kind of constraint satisfaction
- Deduction
  - S = question
  - Prove that KB $\models$ S
  - Two approaches:
    1. Rules to derive new formulas from old
    2. Show (KB $\wedge \neg$ S) is unsatisfiable

Inference 1: Forward Chaining

Forward (& Backward) Chaining

Based on rule of ***modus ponens***

If know $P_1, \ldots, P_n$ & know $(P_1 \wedge \ldots \wedge P_n) \Rightarrow Q$

Then can conclude Q

Pose as Search through Problem Space?

States?

Operators?

Is it sound? Complete?

Model finding (SAT), or deduction (proof)?

Special Syntactic Forms: CNF

- General Form:
  $((q \wedge \neg r) \rightarrow s)) \wedge \neg (s \wedge t)$

- Conjunctive Normal Form (CNF)
  $(\neg q \vee r \vee s) \wedge (\neg s \vee \neg t)$
  Set notation: $\{ (\neg q, r, s), (\neg s, \neg t) \}$
  empty clause () = *false*

© D. Weld D. Fox

Inference 2: Resolution

[Robinson 1965]

$\{ (p \vee \alpha), (\neg p \vee \beta \vee \gamma) \} \ |\!-_R \ (\alpha \vee \beta \vee \gamma)$

Correctness
If S1 $|\!-_R$ S2 then S1 $|=$ S2
Refutation Completeness:
If S is unsatisfiable then S $|\!-_R$ ()

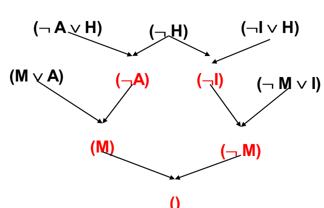
© D. Weld D. Fox

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 = mythical
I = immortal
A = mammal
H = horned



Model finding, or deduction?

© D. Weld D. Fox

Inference 3: Model Enumeration

Enumerate every possible world *w*.
For each *w*:
  Check whether S is true in *w*.
  If yes, we're done → S is satisfiable.
If no *w* satisfies S, S is unsatisfiable.

Model finding, or deduction?
View as Search?
Critique?

© D. Weld D. Fox

Inference 4: DPLL
(Enumeration of *Partial* Models)

[Davis, Putnam, Loveland & Logemann 1962]
Version 1

```
dpll(pa){
  if (pa makes F unsatisfiable) return
                                false;
  if (pa is a full assignment) return
                                true;
  choose P in F;
  if (dpll(pa U {P=0})) return true;
  return dpll(pa U {P=1});
}
```

Returns true if F is satisfiable, false otherwise

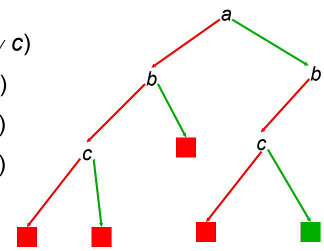
DPLL Version 1

$(a \vee b \vee c)$

$(a \vee \neg b)$

$(a \vee \neg c)$

$(\neg a \vee c)$



Improving DPLL

- We can intelligently rearrange our clauses at each step of the search to improve speed:
  - Remove clauses containing true literals.
  - Remove false literals from remaining clauses.

Improving DPLL

- Unit Literals

  A literal that appears in a singleton clause

  {{¬b c}{¬c}{a ¬b e}{d b}{e a ¬c}}

  Might as well set it true! And simplify

  {{¬b}   {a ¬b e}{d b}}

  {{d}}

- Pure Literals

  A symbol that always appears with same sign

  {{a ¬b c}{¬c d ¬e}{¬a ¬b e}{d b}{e a ¬c}}

  Might as well set it true!   And simplify

  {{a ¬b c}   {¬a ¬b e}   {e a ¬c}}

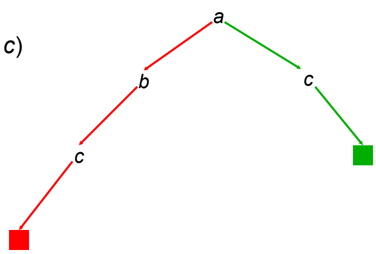
DPLL (for real)

([redacted] ∨ [redacted] ∨ c)

([redacted] ∨ $\neg b$)

([redacted] ∨ $\neg c$)

([redacted] ∨ 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
- Can we eliminate the exponential search time?

Success of DPLL

- 1962 – DPLL invented
- 1992 – 300 propositions
- 1997 – 600 propositions (satz)
- 2002 – 1,000,000 propositions (zChaff)

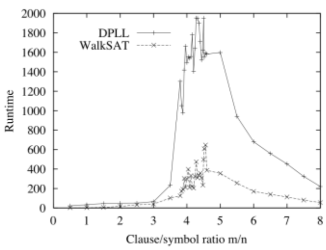
Chaff – fastest complete SAT solver

Created by 2 Princeton undergrads, for a summer project!

Inference 5: 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
- Faster than DPLL
- Completeness?

Random 3-SAT Performance



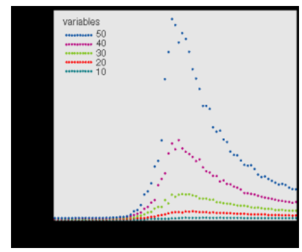
- Random 3-SAT
  - sample uniformly from space of all possible 3-clauses
  - *n* variables, *m* clauses

Which are the hard instances?

around *m/n* = 4.3

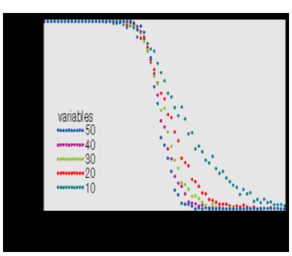
Random 3-SAT

- Varying problem size, *n*
- Complexity peak appears to be largely invariant of algorithm
  - -backtracking algorithms like DPLL
  - -local search procedures like WALKSAT



- *What's so special about 4.3?*

Random 3-SAT



- Complexity peak coincides with solubility transition

m/n < 4.3 problems under-constrained and SAT

m/n > 4.3 problems over-constrained and UNSAT

m/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 (aka 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

Analysis of Propositional Logic Inference / SAT

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