

Propositional Satisfiability

Brian C. Williams
16.410-13
November 9th, 2005

10/6/2005

copyright Brian Williams

1

Reading Assignment: Propositional Satisfiability

- AIMA Ch. 6 – Propositional Logic

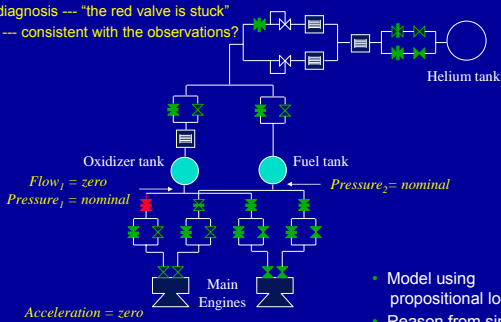
10/6/2005

copyright Brian Williams

2

How Do We Reason About Complex Systems at a Commonsense Level?

Is the diagnosis --- "the red valve is stuck" closed --- consistent with the observations?



10/6/2005

copyright Brian Williams

- Model using propositional logic.
- Reason from single model to operate, diagnose and repair.

Reduction to Clausal Form: Engine Example



(mode(E1) = ok implies
(thrust(E1) = on iff (flow(V1) = on and flow(V2) = on))) and
(mode(E1) = ok or mode(E1) = unknown) and
not (mode(E1) = ok and mode(E1) = unknown)



not (mode(E1) = ok) or not (thrust(E1) = on) or flow(V1) = on;
not (mode(E1) = ok) or not (thrust(E1) = on) or flow(V2) = on;
not (mode(E1) = ok) or not (flow(V1) = on) or not (flow(V2) = on)
or thrust(E1) = on;
mode(E1) = ok or mode(E1) = unknown;
not (mode(E1) = ok) or not (mode(E1) = unknown);

10/6/2005

copyright Brian Williams

4

Propositional Satisfiability

Find a truth assignment that satisfies logical sentence T:

- Reduce sentence T to clausal form.
- Perform search similar to MAC = (BT+CP)
[Davis, Logmann & Loveland, 1962]

Propositional satisfiability testing:

1990: 100 variables / 200 clauses (constraints)
1998: 10,000 - 100,000 vars / 10^6 clauses

Novel applications:

e.g. diagnosis, planning, software / circuit testing,
machine learning, and protein folding

10/6/2005

copyright Brian Williams

5

Outline

- Propositional Satisfiability
 - Backtrack Search
 - Unit Propagation
 - DPLL: Unit Propagation + Backtrack Search
- Appendices

10/6/2005

copyright Brian Williams

6

Propositional Satisfiability

Input: A **Propositional Satisfiability Problem** is a pair $\langle P, \Phi \rangle$, where:

- P is a finite set of **propositions**.
- Φ is a **propositional sentence** on P
 - May be reduced to a set of **clauses**.

Output: **True** iff there exists a **model** of Φ .

Is an instance of a CSP:

- Variables: Propositions
- Domain: {True, False}
- Constraints: Clauses that must be true

10/6/2005

copyright Brian Williams

7

Propositional Satisfiability

- An interpretation (truth assignment to all propositions) such that **all clauses are satisfied**:
- A clause is **satisfied** if and only if **at least one literal is true**.
- A clause is **violated** if and only if **all literals are false**.

C1: Not A or B

C2: Not C or A

C3: Not B or C

10/6/2005

copyright Brian Williams

8

Satisfiability Testing Procedures

Reduce to CNF (Clausal Form) then:

1. **Apply systematic, complete procedure**
 - Depth-first backtrack search [Davis, Logmann, & Loveland 1962]
 - unit propagation, shortest clause heuristic
2. **Apply stochastic, incomplete procedure**
 - **MinConflict** [Minton et al. 90],
GSAT [Selman et al. 1993]— see Appendix
3. **Apply exhaustive clausal resolution**
 - [Davis, Putnam, 1960]

10/6/2005

copyright Brian Williams

9

Outline

- Propositional Satisfiability
 - Backtrack Search
 - Unit Propagation
 - DPLL: Unit Propagation + Backtrack Search
- Appendices

10/6/2005

copyright Brian Williams

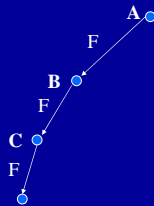
10

Propositional Satisfiability using Backtrack Search

- Assign true or false to an unassigned proposition.
- Backtrack as soon as a clause is violated.

Example:

- C1: Not A or B s
- C2: Not C or ~~A~~ s
- C3: Not B or C s



10/6/2005

copyright Brian Williams

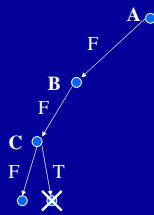
11

Propositional Satisfiability using Backtrack Search

- Assign true or false to an unassigned proposition.
- Backtrack as soon as a clause is violated.

Example:

- C1: Not A or B s
- C2: ~~Not C~~ or ~~A~~ u
- C3: Not B or C s



10/6/2005

copyright Brian Williams

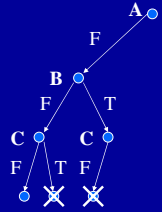
12

Propositional Satisfiability using Backtrack Search

- Assign true or false to an unassigned proposition.
- Backtrack as soon as a clause is violated.

Example:

- C1: Not A or B **s**
- C2: Not C or A **s**
- C3: Not B or C **v**



10/6/2005

copyright Brian Williams

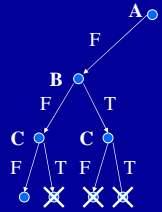
13

Propositional Satisfiability using Backtrack Search

- Assign true or false to an unassigned proposition.
- Backtrack as soon as a clause is violated.

Example:

- C1: Not A or B **s**
- C2: Not C or A **v**
- C3: Not B or C **s**



10/6/2005

copyright Brian Williams

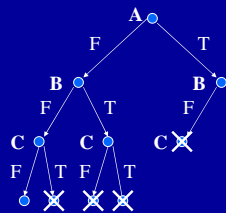
14

Propositional Satisfiability using Backtrack Search

- Assign true or false to an unassigned proposition.
- Backtrack as soon as a clause is violated.

Example:

- C1: Not A or B **v**
- C2: Not C or A **s**
- C3: Not B or C **s**



10/6/2005

copyright Brian Williams

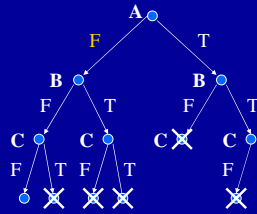
15

Propositional Satisfiability using Backtrack Search

- Assign true or false to an unassigned proposition.
- Backtrack as soon as a clause is violated.

Example:

- C1: Not A or B **s**
- C2: Not C or A **s**
- C3: Not B or C **v**



10/6/2005

copyright Brian Williams

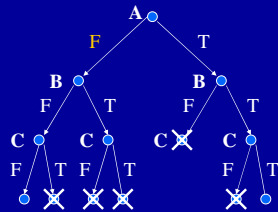
16

Propositional Satisfiability using Backtrack Search

- Assign true or false to an unassigned proposition.
- Backtrack as soon as a clause is violated.

Example:

- C1: Not A or B **s**
- C2: Not C or A **s**
- C3: Not B or C **s**



10/6/2005

copyright Brian Williams

17

Clausal Backtrack Search: Recursive Definition

BT(Phi, A)

Input: A *cnf* theory Phi,
An assignment A to propositions in Phi

Output: A decision of whether Phi is satisfiable.

- If a clause is violated, Return false;
- Else If all propositions are assigned, Return true;
- Else Q = some unassigned proposition in Phi;
- Return (BT(Phi, A[Q = True]) or
- BT(Phi, A[Q = False]))

10/6/2005

copyright Brian Williams

18

Outline

- Propositional Satisfiability
 - Backtrack Search
 - Unit Propagation
 - DPLL: Unit Propagation + Backtrack Search
- Appendices

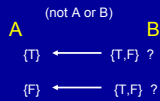
10/6/2005

copyright Brian Williams

19

Unit Propagation

Idea: Arc consistency (AC-3) on binary clauses



Unit propagation rule:

If all literals are false save L, then assign true to L:

- $\frac{(\text{not } A) \quad (\text{not } B) \quad (A \text{ or } B \text{ or } C)}{C}$

10/6/2005

copyright Brian Williams

20

Unit Propagation Examples

- C1: Not A or B Satisfied
 - C2: Not C or A Satisfied
 - C3: Not B or C Satisfied
 - C4: A Satisfied
- $C4 \rightarrow A \xrightarrow{\text{True}} C1 \rightarrow B \xrightarrow{\text{True}} C3 \rightarrow C$

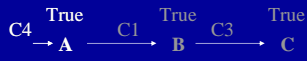
10/6/2005

copyright Brian Williams

21

Unit Propagation Examples

- C1: Not A or B Satisfied
- C2: Not C or A Satisfied
- C3: Not B or C Satisfied
- C4: A



- C4': Not B Satisfied

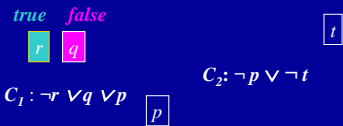


10/6/2005

copyright Brian Williams

22

Unit Propagation



```

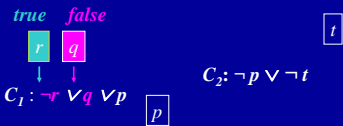
procedure propagate(C) // C is a clause
  if all literals in C are false except L, and L is unassigned
  then assign true to L and
       record C as a support for L and
       for each clause C' mentioning "not L",
           propagate(C')
end propagate
  
```

10/6/2005

copyright Brian Williams

23

Unit Propagation



```

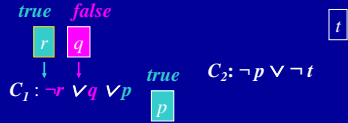
procedure propagate(C) // C is a clause
  => if all literals in C are false except L, and L is unassigned
  then assign true to L and
       record C as a support for L and
       for each clause C' mentioning "not L",
           propagate(C')
end propagate
  
```

10/6/2005

copyright Brian Williams

24

Unit Propagation

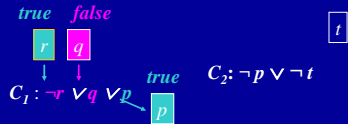


```

procedure propagate(C) // C is a clause
  if all literals in C are false except L, and L is unassigned
  then assign true to L and
  record C as a support for L and
  for each clause C' mentioning "not L",
    propagate(C')
end propagate
  
```

10/6/2005 copyright Brian Williams 25

Unit Propagation

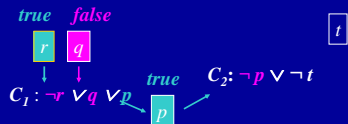


```

procedure propagate(C) // C is a clause
  if all literals in C are false except L, and L is unassigned
  then assign true to L and
  record C as a support for L and
  for each clause C' mentioning "not L",
    propagate(C')
end propagate
  
```

10/6/2005 copyright Brian Williams 26

Unit Propagation

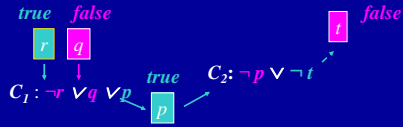


```

procedure propagate(C) // C is a clause
  if all literals in C are false except L, and L is unassigned
  then assign true to L and
  record C as a support for L and
  for each clause C' mentioning "not L",
    propagate(C')
end propagate
  
```

10/6/2005 copyright Brian Williams 27

Unit Propagation



```

procedure propagate(C) // C is a clause
  if all literals in C are false except L, and L is unassigned
  then assign true to L and
  record C as a support for L and
  for each clause C' mentioning "not L",
    propagate(C')
end propagate
  
```

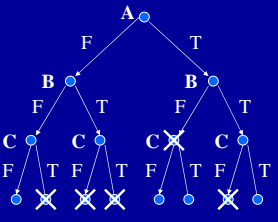
10/6/2005 copyright Brian Williams 28

Outline

- Propositional Satisfiability
 - Backtrack Search
 - Unit Propagation
 - DPLL: Unit Propagation + Backtrack Search
- Appendices

How Do We Combine Unit Propagation and Back Track Search?

- Backtrack Search
- Assign true or false to an unassigned proposition.
 - Backtrack as soon as a clause is violated.
 - Theory is satisfiable if assignment is complete.



- Example:
- C1: Not A or B
 - C2: Not C or A
 - C3: Not B or C

- Similar to MAC and Forward Checking:
 - Perform limited form of inference
 - apply inference rule after assigning each variable.

Propositional Satisfiability by D(PLL)

[Davis, Logmann, Loveland, 1962]

Initially:

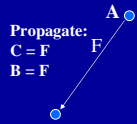
- **Unit propagate.**

Repeat:

1. Assign true or false to an unassigned proposition.
2. **Unit propagate.**
3. Backtrack as soon as a clause is violated.
4. Satisfiable if assignment is complete.

Example:

- C1: Not A or B **S**
- C2: Not C or A **S**
- C3: Not B or C **S**



10/6/2005

copyright Brian Williams

31

Propositional Satisfiability by DPLL

[Davis, Logmann, Loveland, 1962]

Initially:

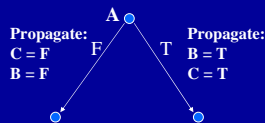
- **Unit propagate.**

Repeat:

1. Assign true or false to an unassigned proposition.
2. **Unit propagate.**
3. Backtrack as soon as a clause is violated.
4. Satisfiable if assignment is complete.

Example:

- C1: Not A or B **S**
- C2: Not C or A **S**
- C3: Not B or C **S**



10/6/2005

copyright Brian Williams

32

D(P)LL Procedure

[Davis, Logmann, Loveland, 1961]

DPLL(Phi,A)

Input: A *cnf* theory Phi,

An assignment A to propositions in Phi

Output: A decision of whether Phi is satisfiable.

1. A' = propagate(Phi);
2. If a clause is violated given A' return(false);
3. Else if all propositions in A' are assigned, return(true);
4. Else Q = some unassigned proposition in Phi;
6. Return (DPLL(Phi, A'[Q = True]) or
7. DPLL(Phi, A'[Q = False])

10/6/2005

copyright Brian Williams

33

Satisfiability Testing Procedures

- Reduce to CNF (Clausal Form) then:
- Apply systematic, complete procedure
 - Depth-first backtrack search [Davis, Logmann, & Loveland 1961]
 - unit propagation, shortest clause heuristic
 - State-of-the-art implementations:
 - *ntab* [Crawford & Auton, 1997]
 - *itms* [Nayak & Williams, 1997]
 - many others! See *SATLIB 1998 / Hoos & Stutzle*
- Apply stochastic, incomplete procedures
 - *GSAT* [Selman et. al 1993]
 - *Walksat* [Selman & Kautz 1993]
 - greedy local search + noise to escape local minima

10/6/2005

copyright Brian Williams

34

Required Appendices

You are responsible for reading and knowing this material:

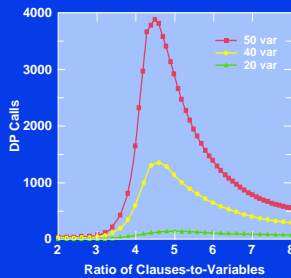
1. Characteristics of DPLL
2. Local Search using GSAT

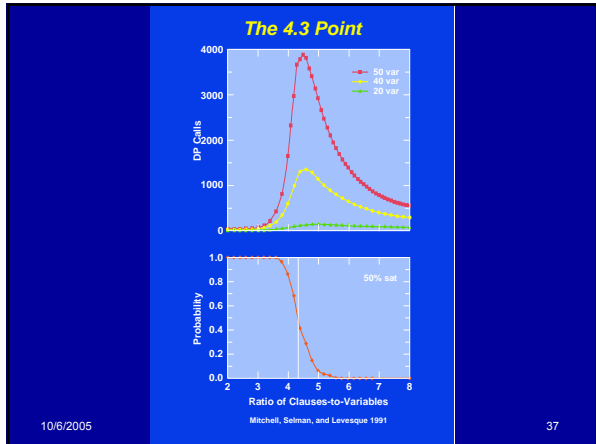
10/6/2005

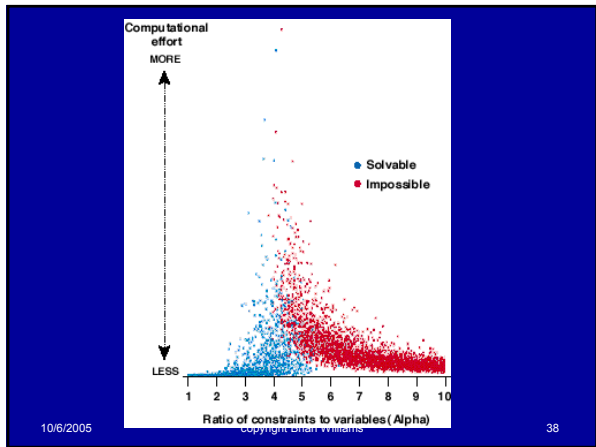
copyright Brian Williams

35

Hardness of 3SAT





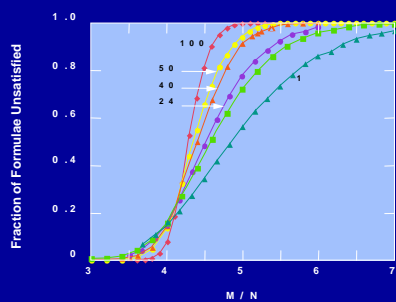


Intuition

- At low ratios:
 - few clauses (constraints)
 - many assignments
 - easily found
- At high ratios:
 - many clauses
 - inconsistencies easily detected

10/6/2005 39

Phase Transitions for Different Numbers of Variables



10/6/2005

copyright Brian Williams

40

Required Appendices

You are responsible for reading and knowing this material:

1. Characteristics of DPLL
2. Local Search using GSAT

10/6/2005

copyright Brian Williams

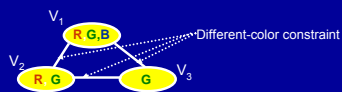
42

Incremental Repair (min-conflict heuristic)

Spike Hubble Telescope Scheduler [Minton et al.]

1. Initialize a candidate solution using "greedy" heuristic – get solution "near" correct one.
2. Select a variable in conflict and assign it a value that minimizes the number of conflicts (break ties randomly).

Graph Coloring
Initial Domains



10/6/2005

copyright Brian Williams

43

GSAT

- C1: Not A or B
- C2: Not C or Not A
- C3: or B or Not C

1. Init: Pick random assignment
2. Check effect of flipping each assignment, counting violated clauses.
3. Pick assignment with fewest violations,
4. End if consistent, Else goto 2

C1, C2, C3 violated	True A	False B	True C
	False	True	False
C3 violated	C2 violated	C1 violated	

10/6/2005 copyright Brian Williams 44

GSAT

- C1: Not A or B
- C2: Not C or Not A
- C3: or B or Not C

1. Init: Pick random assignment
2. Check effect of flipping each assignment, counting violated clauses.
3. Pick assignment with fewest violations,
4. End if consistent, Else goto 2

C1 violated	True A	False B	False C
	False	True	True
Satisfied	Satisfied	C1,C2,C3 violated	

10/6/2005 copyright Brian Williams 45

GSAT

- C1: Not A or B
- C2: Not C or Not A
- C3: or B or Not C

1. Init: Pick random assignment
2. Check effect of flipping each assignment, counting violated clauses.
3. Pick assignment with fewest violations,
4. End if consistent, Else goto 2

Satisfied	True A	True B	False C
-----------	-----------	-----------	------------

Problem: Pure hill climbers get stuck in local minima.
Solution: Add random moves to get out of minima (*WalkSAT*)

10/6/2005 copyright Brian Williams 46
