Propositional Satisfiability: Unit Propagation and DPLL

Empirical analysis slides draw upon material from:
Prof. Bart Selman
Cornell University

Brian C. Williams
16.410/413
November 9th, 2015

Assignments

- Assignment:
  - Problem Set #7: Due Wednesday.
- Reading:
  - Today: *[AIMA] Ch. 7, 8.*

Monday:

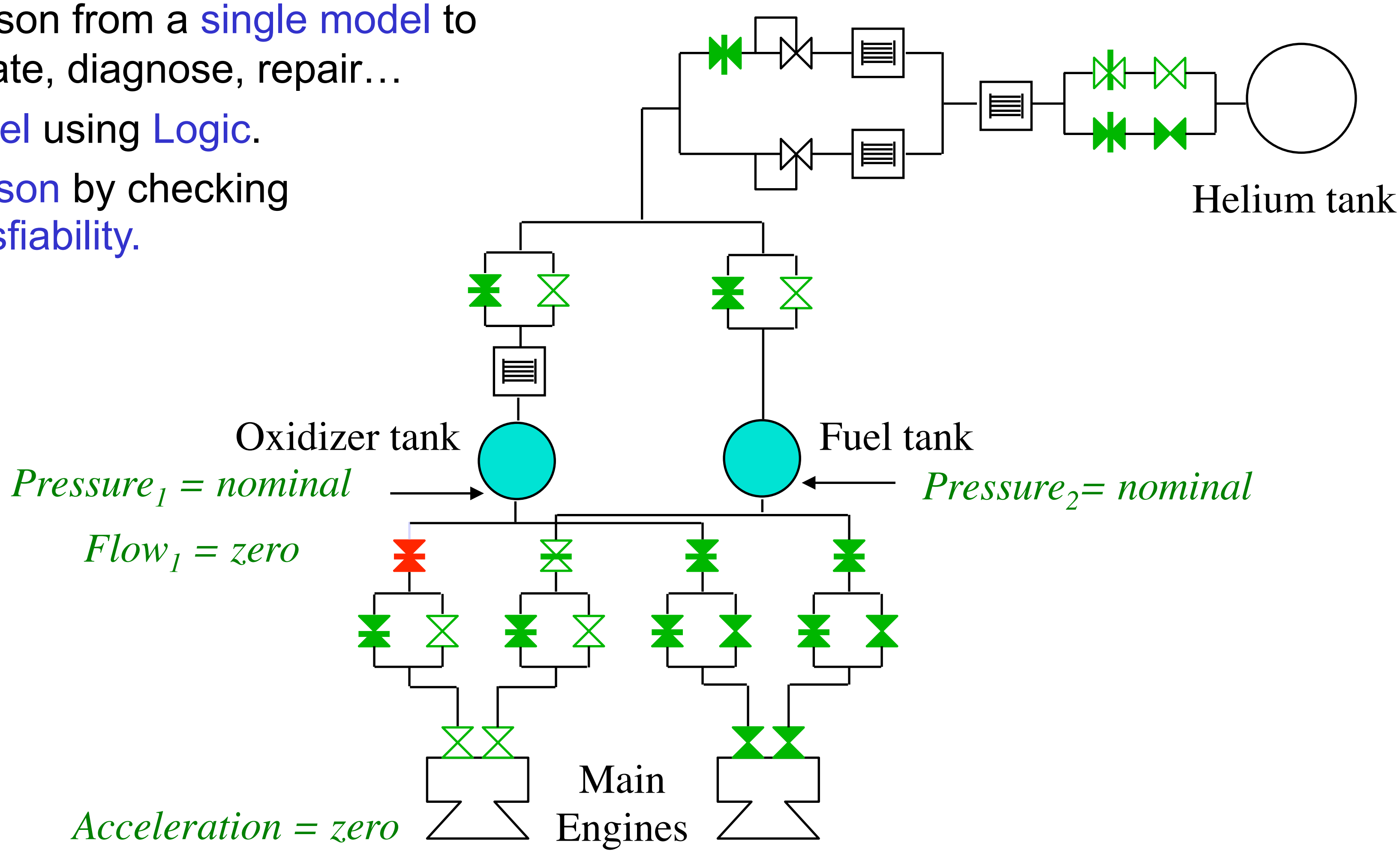
- J. de Kleer and B. C. Williams, "Diagnosing Multiple Faults," *Artificial Intelligence*, 32:100-117, 1987.
- Wednesday: B. C. Williams, and R. Ragno, "Conflict-directed A* and its Role in Model-based Embedded Systems," Special Issue on Theory and Applications of Satisfiability Testing, *Journal of Discrete Applied Math, January 2003*.

Outline

- Review
- Propositional Satisfiability with Inference
- Empirical, Average Case Analysis
- Model-based Diagnosis (separate slide packet).

Model-based Reasoning

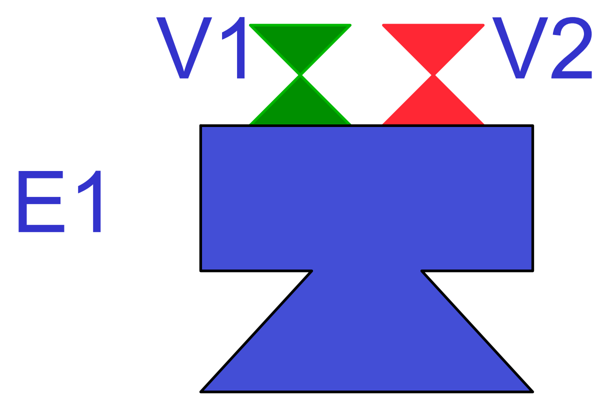
- Reason from a single model to operate, diagnose, repair…
- Model using Logic.
- Reason by checking Satisfiability.



Engine Model in Propositional Logic

"An Engine E1 can either be okay, or broken in some unknown way.

When E1 is okay, it will thrust when there is a flow through V1 and V2."



(mode(E1) = ok or mode(E1) = unknown) and
not (mode(E1) = ok and mode(E1) = unknown) and

(mode(E1) = ok implies
(thrust(E1) = on if and only if flow(V1) = on and flow(V2) = on))

Propositional Satisfiability

Given: a logical sentence S

Find: a truth assignment (true / false) that satisfies S:

1. Reduce S to *clausal form*.

2. Perform search + inference
   – similar to MAC = Backtrack + Constraint Propagation [Davis, Logmann & Loveland, 1962].

Propositional Satisfiability as Backtrack Search

Procedure: BT(Φ, A)
Input: A *cnf theory* Φ,
An assignment A to some propositions in Φ.
Output: true if Φ is satisfiable; false otherwise.

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

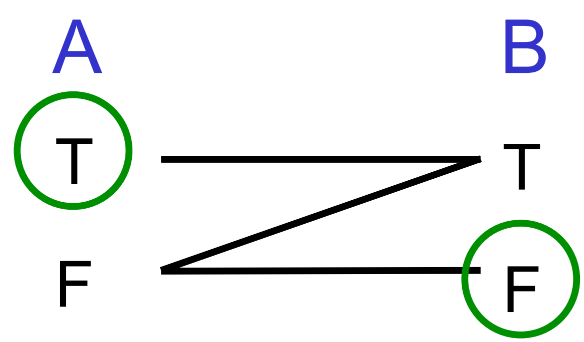
Outline

- Review
- Propositional Satisfiability with Inference
  - Unit Propagation
  - DPLL: Unit Propagation + Backtrack Search
- Empirical, Average Case Analysis
- Model-based Diagnosis

Unit Clause Resolution

Idea: Apply arc consistency (AC-3) to binary clauses.

Clause: (not A or B)



Unit clause resolution (aka unit propagation rule):

If all literals are false save L, then L must be true:

- $$\frac{(\text{not } A) \quad (\text{not } B) \quad (A \text{ or } B \text{ or } C)}{C}$$
- Unit propagation = repeated application of unit clause resolution rule.

Unit Propagation Examples

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

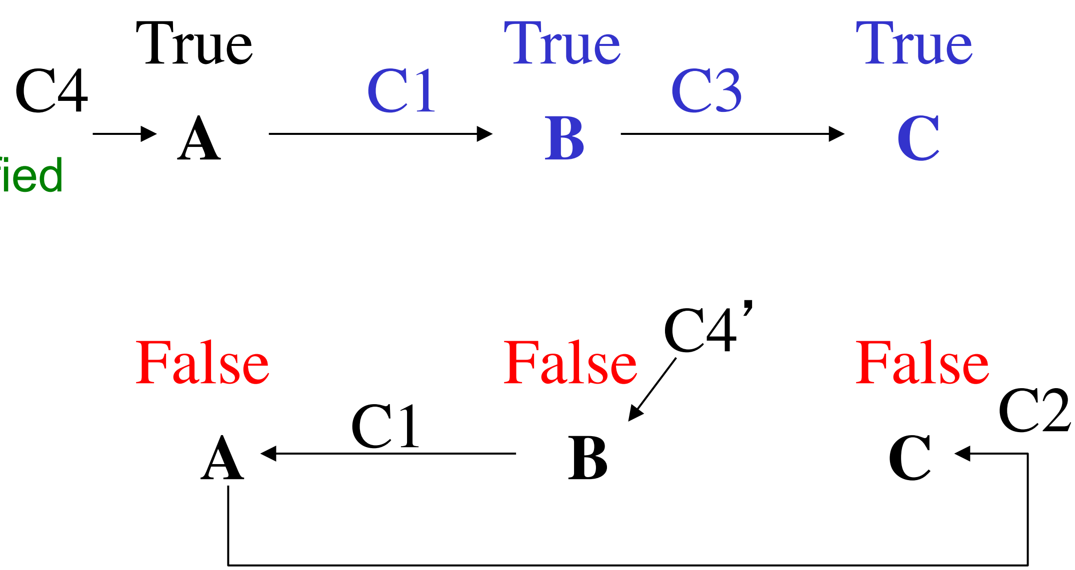
Support

$$\xrightarrow{\text{C4}} \overset{\text{True}}{\mathbf{A}} \xrightarrow{\text{C1}} \overset{\text{True}}{\mathbf{B}} \xrightarrow{\text{C3}} \overset{\text{True}}{\mathbf{C}}$$

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



Unit Propagation

true *false*

r q t

$C_2: \neg p \vee \neg t$

$C_1 : \neg r \vee q \vee p$ p

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*

Unit Propagation

true *false*

r q

$C_1 : \neg r \vee q \vee p$

p

t

$C_2 : \neg p \vee \neg t$

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*

Unit Propagation

true *false*

$\boxed{r}$ $\boxed{q}$ $\boxed{t}$

$C_1 : \neg r \lor q \lor p$ *true* $\boxed{p}$ $C_2 : \neg p \lor \neg t$

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

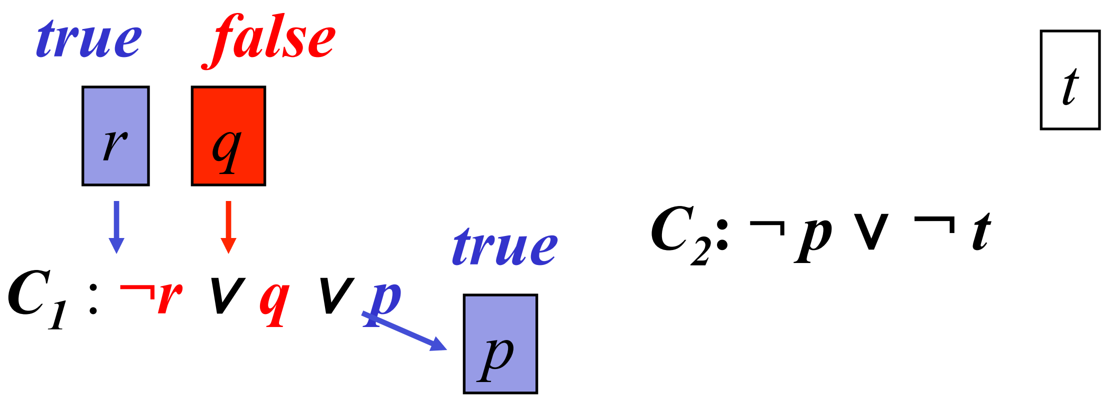
record **C** as a support for **L** and

for each clause ***C'*** mentioning "not **L**",

propagate(***C'***)

end *propagate*

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

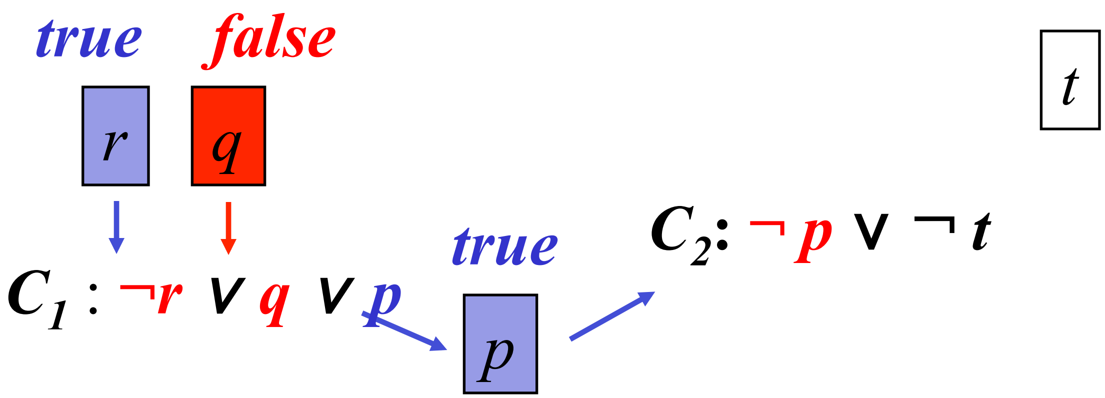
⇨ record **C** as a support for **L** and

for each clause ***C'*** mentioning "not **L**",

propagate(***C'***)

end *propagate*

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

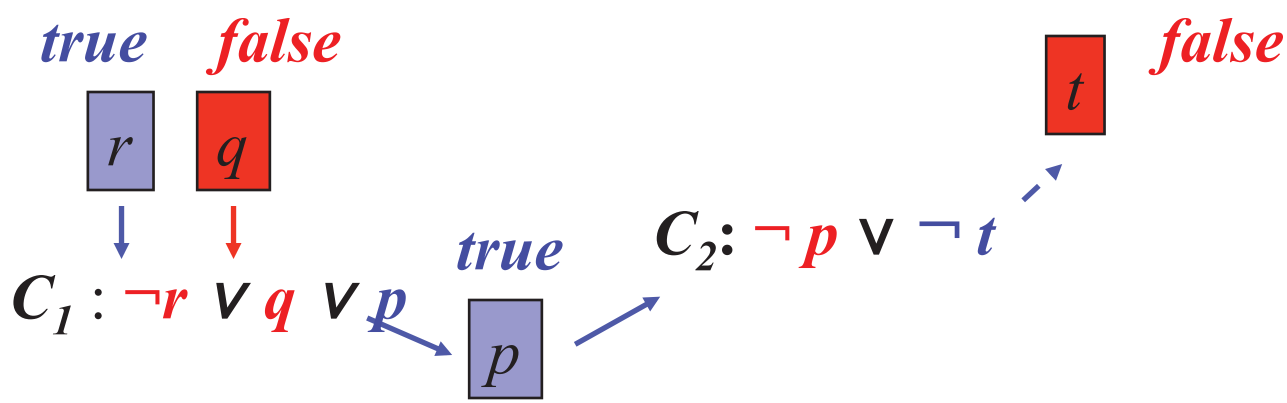
record **C** as a support for **L** and

⇨ **for each** clause ***C'*** mentioning "not **L**",

propagate(***C'***)

end *propagate*

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*

Outline

- Review
- Propositional Satisfiability with Inference
  - Unit Propagation
  - DPLL: Unit Propagation + Backtrack Search
- Empirical, Average Case Analysis
- Model-based Diagnosis

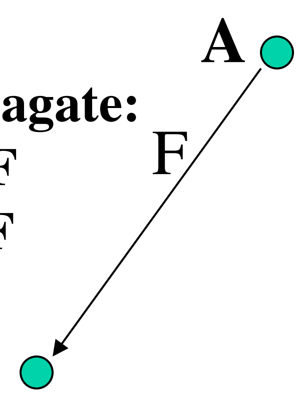
Propositional Satisfiability using DPLL
[Davis, Logmann, Loveland, 1962]

Initially:
- Unit propagate.

Repeat:
1. Assign true or false to unassigned proposition.
2. Unit propagate.
3. Backtrack when clause violated.
4. Satisfiable if assignment complete.

Example:
- C1: Not A or B **satisfied**
- C2: Not C or ~~A~~ **satisfied**
- C3: Not B or ~~C~~ **satisfied**

Propagate:
C = F
B = F

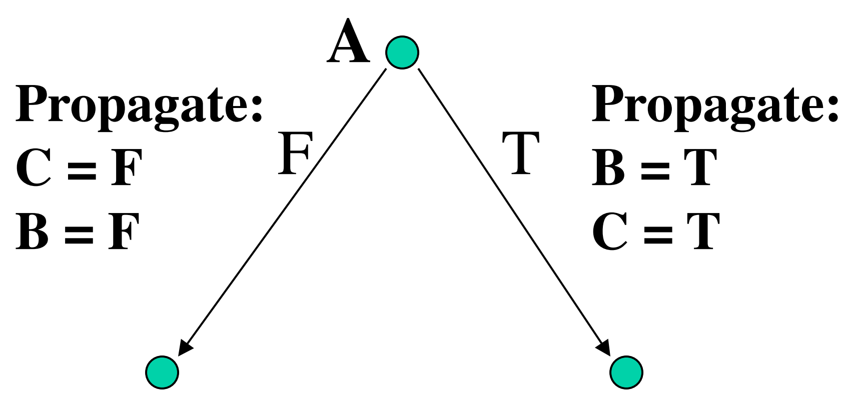


Propositional Satisfiability using DPLL
[Davis, Logmann, Loveland, 1962]

Initially:
- Unit propagate.

Repeat:
1. Assign true or false to unassigned proposition.
2. Unit propagate.
3. Backtrack when clause violated.
4. Satisfiable if assignment complete.

Example:
- C1: ~~Not~~ A or B **satisfied**
- C2: Not C or A **satisfied**
- C3: ~~Not~~ B or C **satisfied**



How Do We Fold Unit Propagation into Backtracking?

Procedure: BT(Φ, A)
Input: A *cnf* theory Φ,
An assignment A to some propositions in Φ.
Output: A decision of whether Φ is satisfiable.

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

Hint: Like MAC and Forward Checking:

- limited inference
- apply inference after assigning each variable.

D(P)LL Procedure
[Davis, Logmann, Loveland, 1961]

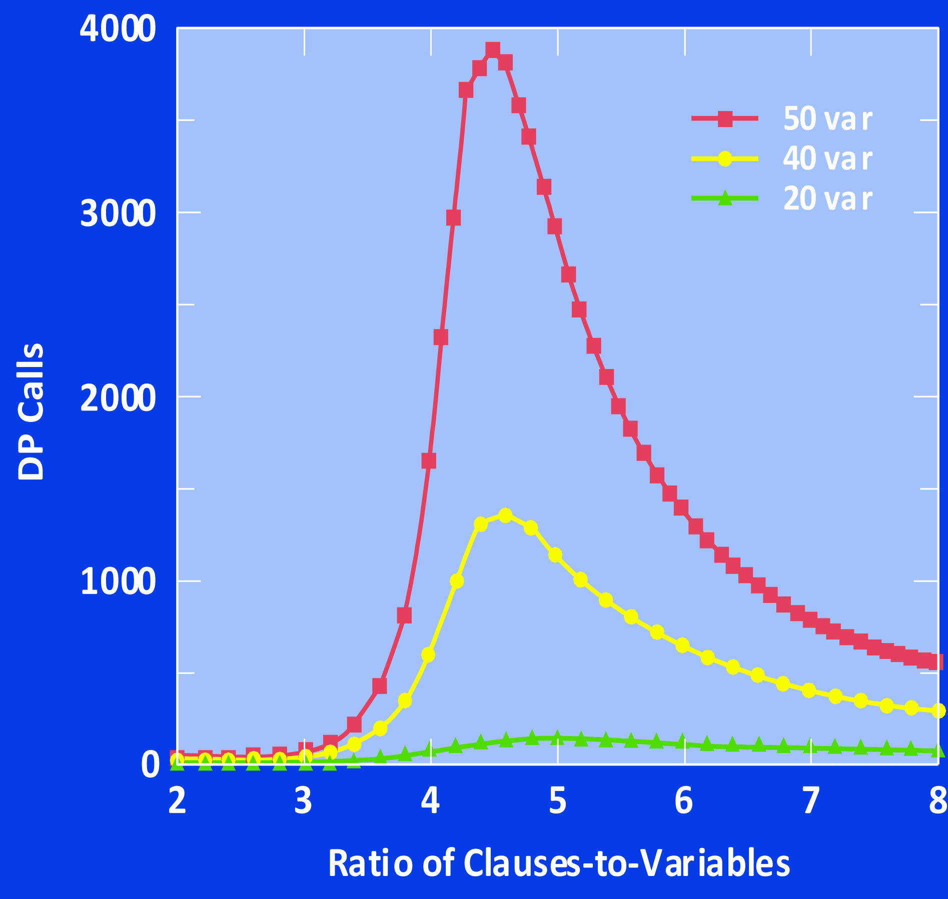
Procedure: DPLL(Φ, A)
Input: A *cnf* theory Φ,
An assignment A to propositions in Φ
Output: A decision of whether Φ is satisfiable.

A' = propagate(Φ);
If a clause in Φ is violated, given A' Return false;
Else If all propositions of Φ are assigned in A', Return true;
Else Q = some unassigned proposition in Φ;
Return (DPLL(Φ, A' [Q = True]) or
DPLL(Φ, A' [Q = False]))

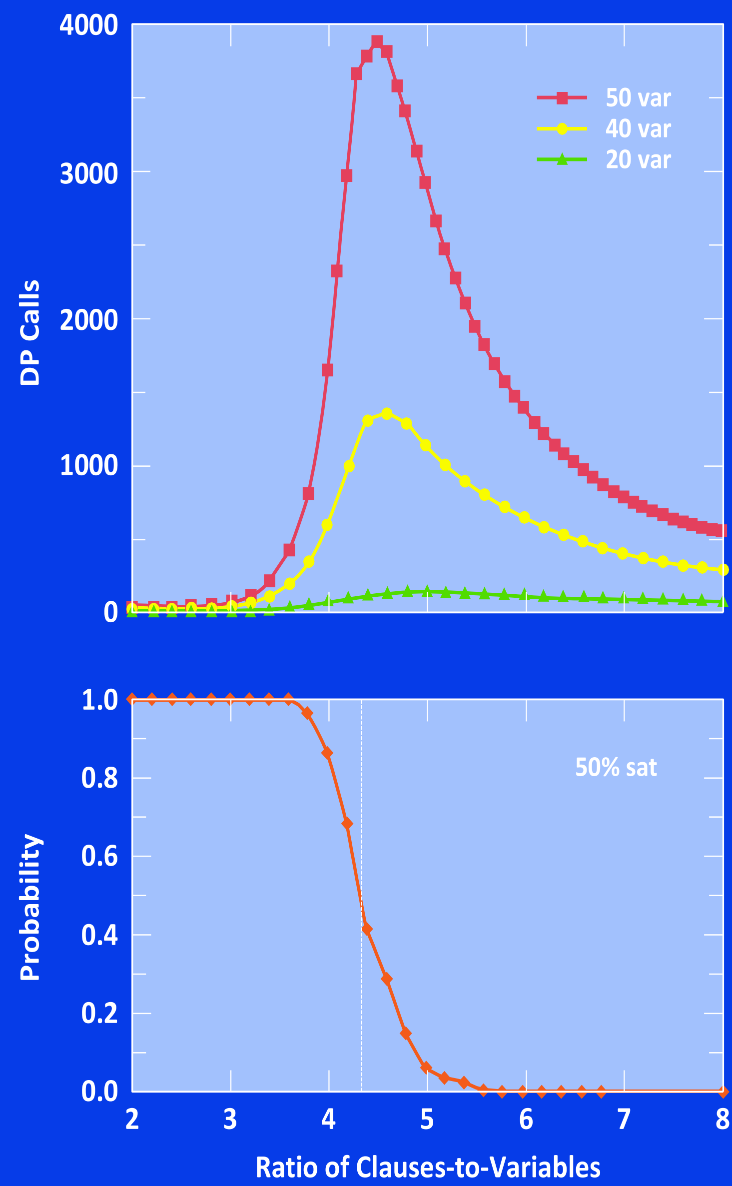
Outline

- Review
- Propositional Satisfiability with Inference
  - Unit Propagation
  - DPLL: Unit Propagation + Backtrack Search
- Empirical, Average Case Analysis
- Model-based Diagnosis

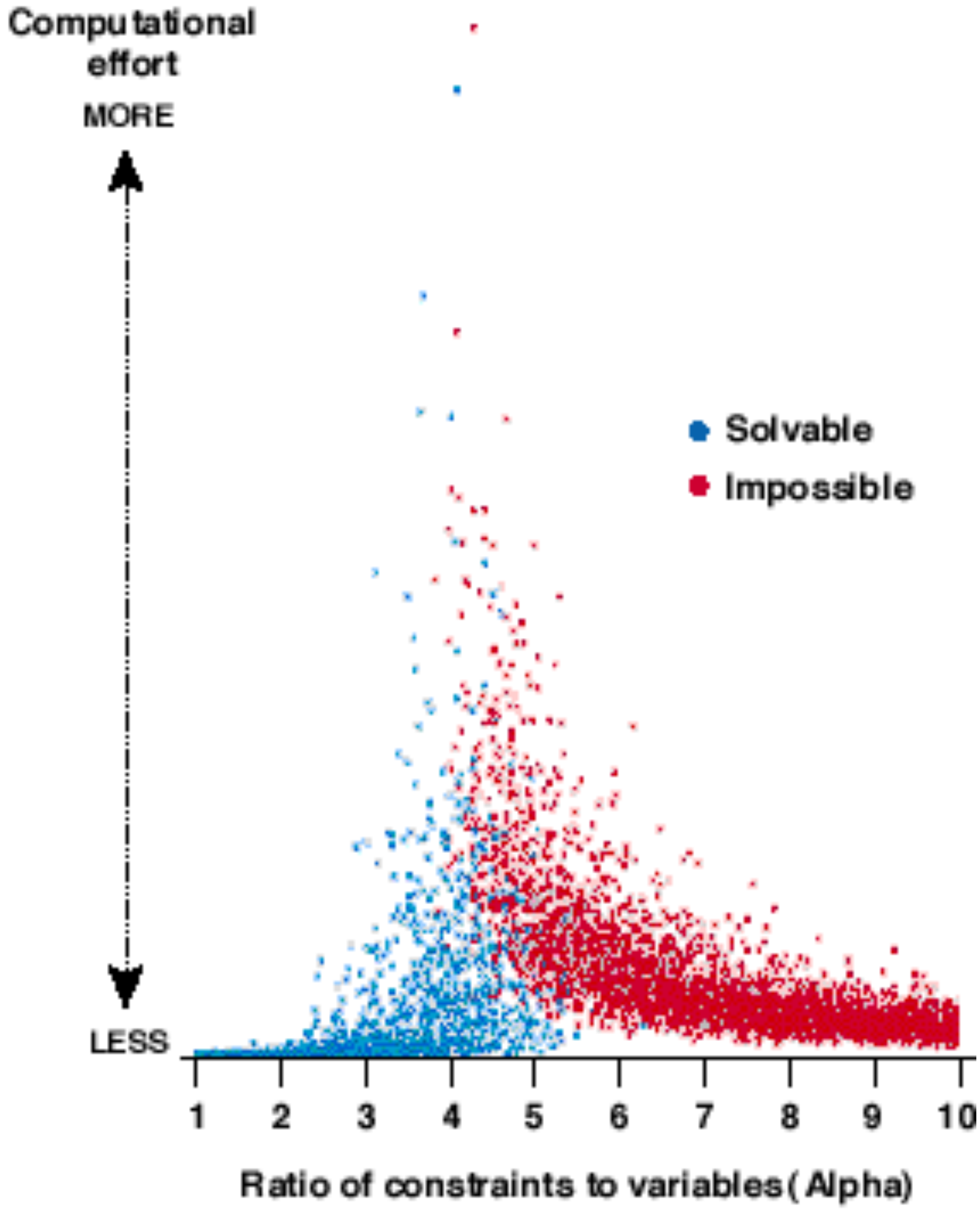
Hardness of 3SAT



The 4.3 Point



Mitchell, Selman, and Levesque 1991



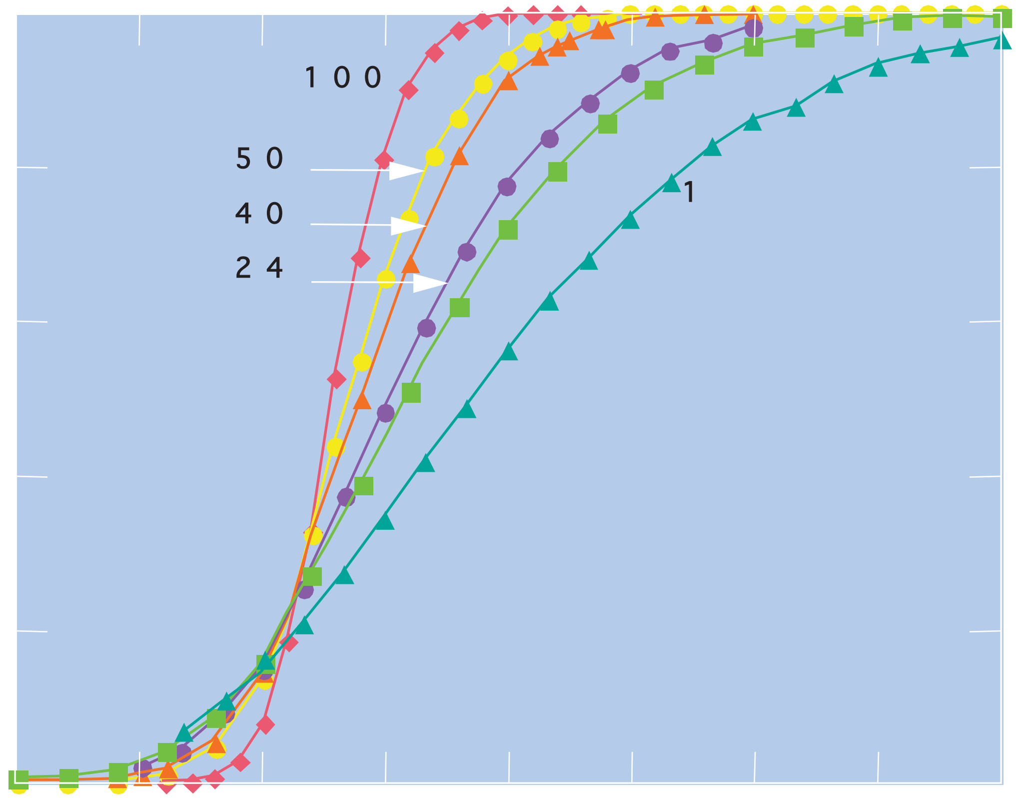
Computational effort
MORE
LESS
Solvable
Impossible
1
2
3
4
5
6
7
8
9
10
Ratio of constraints to variables (Alpha)

Intuition

- At low ratios:
  - few clauses (constraints),
  - many assignments,
  - easily found.

- At high ratios:
  - many clauses,
  - inconsistencies easily detected.

Phase Transitions for Different Numbers of Variables



Phase Transitions: 2, 3 4, 5 and 6-SAT