# Computational Logic and Satisfiability

## IN4077

# Introduction

**Marijn J.H. Heule**

September 9, 2008



**Delft University of Technology**

TU Delft
Delft University of Technology

# Motivation satisfiability solving

From 100 variables, 200 constraints (early 90’s)

to 1,000,000 vars. and 5,000,000 clauses in 15 years.

# Motivation satisfiability solving

From 100 variables, 200 constraints (early 90's)
to 1,000,000 vars. and 5,000,000 clauses in 15 years.

Applications:

Hardware and Software Verification, Planning, Scheduling, Optimal Control, Protocol Design, Routing, Combinatorial problems, Equivalence Checking, etc.

# Motivation satisfiability solving

From 100 variables, 200 constraints (early 90's)
to 1,000,000 vars. and 5,000,000 clauses in 15 years.

Applications:

Hardware and Software Verification, Planning, Scheduling, Optimal Control, Protocol Design, Routing, Combinatorial problems, Equivalence Checking, etc.

SAT used to solve many other problems!

# Defying NP-Completeness

Current state of the art complete SAT solvers can handle very large problem instances of real-world combinatorial:

*We are dealing with formidable search spaces of exponential size — to prove optimality we have to implicitly search the entire search, HOW?*

The problems we are able to solve are much larger than would predict given that these are in general NP complete

Disclaimer: **A hard random unsat 3-SAT formula with 1000 vars. cannot be solved while real-world sat and unsat instances with over 1,000,000 vars. are solved in a few minutes.**

# Overview

- Introduction
- The Satisfiability problem
- Terminology
- SAT solving
- SAT benchmarks

# Introduction

"You are chief of protocol for the embassy ball. The crown prince instructs you either to invite *Peru* or to exclude *Qatar*. The queen asks you to invite either *Qatar* or *Romania* or both. The king, in a spiteful mood, wants to snub either *Romania* or *Peru* or both. Is there a guest list that will satisfy the whims of the entire royal family?"

# Introduction

"You are chief of protocol for the embassy ball. The crown prince instructs you either to invite *Peru* or to exclude *Qatar*. The queen asks you to invite either *Qatar* or *Romania* or both. The king, in a spiteful mood, wants to snub either *Romania* or *Peru* or both. Is there a guest list that will satisfy the whims of the entire royal family?"

$$(P \vee \neg Q) \wedge (Q \vee R) \wedge (\neg R \vee \neg P)$$

# Introduction: SAT question

Given *formula*,
does there exist an *assignment*
to the *Boolean variables*
that satisfies all *clauses*?

# Terminology: Variables and literals

Boolean variable

- can be assigned the Boolean values `0` or `1`

Literal

- refers either to $x_i$ or its complement $\neg x_i$
- literals $x_i$ are satisfied if variable $x_i$ is assigned to `1` (true)
- literals $\neg x_i$ are satisfied if variable $x_i$ is assigned to `0` (false)

# Terminology: Clauses

Clause

- Disjunction of literals: E.g. $C_j = l_1 \vee l_2 \vee l_3$
- Can be falsified with only $one$ assignment to its literals: All literals assigned to false
- Can be satisfied with $2^k - 1$ assignment to its $k$ literals
- One special clause - the empty clause (denoted by $\emptyset$) - which is always falsified

# Terminology: Formulae

Formula

- Conjunction of clauses: E.g. $\mathcal{F} = C_1 \wedge C_2 \wedge C_3$
- Is *satisfiable* if there exists an assignment satisfying all clauses, otherwise *unsatisfiable*
- Formulae are not only defined in *Conjunction Normal Form* (CNF), but generally also stored as such - also learned information

# Terminology: Assignments

Assignment

- Mapping of the values `0` and `1` to the variables
- $\varphi \circ \mathcal{F}$ results in a reduced formula $\mathcal{F}_{\text{reduced}}$:
  - all satisfied clauses are removed
  - all falsified literals are removed
- *satisfying assignment* $\leftrightarrow \mathcal{F}_{\text{reduced}}$ is empty
- *falsifying assignment* $\leftrightarrow \mathcal{F}_{\text{reduced}}$ contains $\emptyset$
- *partial assignment* versus *full assignment*

# SAT solving: Unit propagation

A *unit clause* is a clause of size 1

UnitPropagation $(\varphi, \mathcal{F})$:

**while** $\emptyset \notin \mathcal{F}$ **and** unit clause $y$ exists **do**
    expand $\varphi$ and simplify $\mathcal{F}$
**end while**
**return** $\varphi, \mathcal{F}$

# Unit propagation: Example

$$\mathcal{F}_{\text{unit}} := (\neg x_1 \vee \neg x_3 \vee x_4) \wedge (\neg x_1 \vee \neg x_2 \vee x_3) \\ (\neg x_1 \vee x_2) \wedge (x_1 \vee x_3 \vee x_6) \wedge (\neg x_1 \vee x_4 \vee \neg x_5) \\ (x_1 \vee \neg x_6) \wedge (x_4 \vee x_5 \vee x_6) \wedge (x_5 \vee \neg x_6)$$

# Unit propagation: Example

$$\mathcal{F}_{\text{unit}} := (\neg x_1 \vee \neg x_3 \vee x_4) \wedge (\neg x_1 \vee \neg x_2 \vee x_3)$$
$$(\neg x_1 \vee x_2) \wedge (x_1 \vee x_3 \vee x_6) \wedge (\neg x_1 \vee x_4 \vee \neg x_5)$$
$$(x_1 \vee \neg x_6) \wedge (x_4 \vee x_5 \vee x_6) \wedge (x_5 \vee \neg x_6)$$

$$\varphi = \{x_1{=}1\}$$

# Unit propagation: Example

$$\mathcal{F}_{\text{unit}} := (\neg x_1 \vee \neg x_3 \vee x_4) \wedge (\neg x_1 \vee \neg x_2 \vee x_3) \\ (\neg x_1 \vee x_2) \wedge (x_1 \vee x_3 \vee x_6) \wedge (\neg x_1 \vee x_4 \vee \neg x_5) \\ (x_1 \vee \neg x_6) \wedge (x_4 \vee x_5 \vee x_6) \wedge (x_5 \vee \neg x_6)$$

$$\varphi = \{x_1{=}1, x_2{=}1\}$$

# Unit propagation: Example

$$\mathcal{F}_{\text{unit}} := (\neg x_1 \vee \neg x_3 \vee x_4) \wedge (\neg x_1 \vee \neg x_2 \vee x_3)$$
$$(\neg x_1 \vee x_2) \wedge (x_1 \vee x_3 \vee x_6) \wedge (\neg x_1 \vee x_4 \vee \neg x_5)$$
$$(x_1 \vee \neg x_6) \wedge (x_4 \vee x_5 \vee x_6) \wedge (x_5 \vee \neg x_6)$$

$$\varphi = \{x_1\text{=}1, x_2\text{=}1, x_3\text{=}1\}$$

# Unit propagation: Example

$$\mathcal{F}_{\text{unit}} := (\neg x_1 \vee \neg x_3 \vee x_4) \wedge (\neg x_1 \vee \neg x_2 \vee x_3)$$
$$(\neg x_1 \vee x_2) \wedge (x_1 \vee x_3 \vee x_6) \wedge (\neg x_1 \vee x_4 \vee \neg x_5)$$
$$(x_1 \vee \neg x_6) \wedge (x_4 \vee x_5 \vee x_6) \wedge (x_5 \vee \neg x_6)$$

$$\varphi = \{x_1{=}1, x_2{=}1, x_3{=}1, x_4{=}1\}$$

# SAT solving: DPLL

Davis Putnam Logemann Loveland [DP60,DLL62]

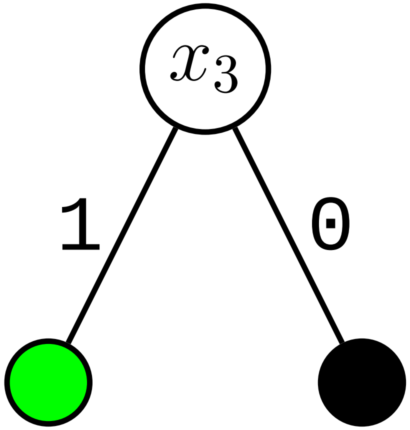
- Simplify (Unit Propagation)
- Split the formula
  - Variable Selection Heuristics
  - Direction heuristics

# DPLL: Example

$$\mathcal{F}_{\text{DPLL}} := (x_1 \vee x_2 \vee \neg x_3) \wedge (\neg x_1 \vee x_2 \vee x_3)$$
$$(\neg x_1 \vee \neg x_2 \vee x_3) \wedge (x_1 \vee x_3) \wedge (\neg x_1 \vee \neg x_3)$$

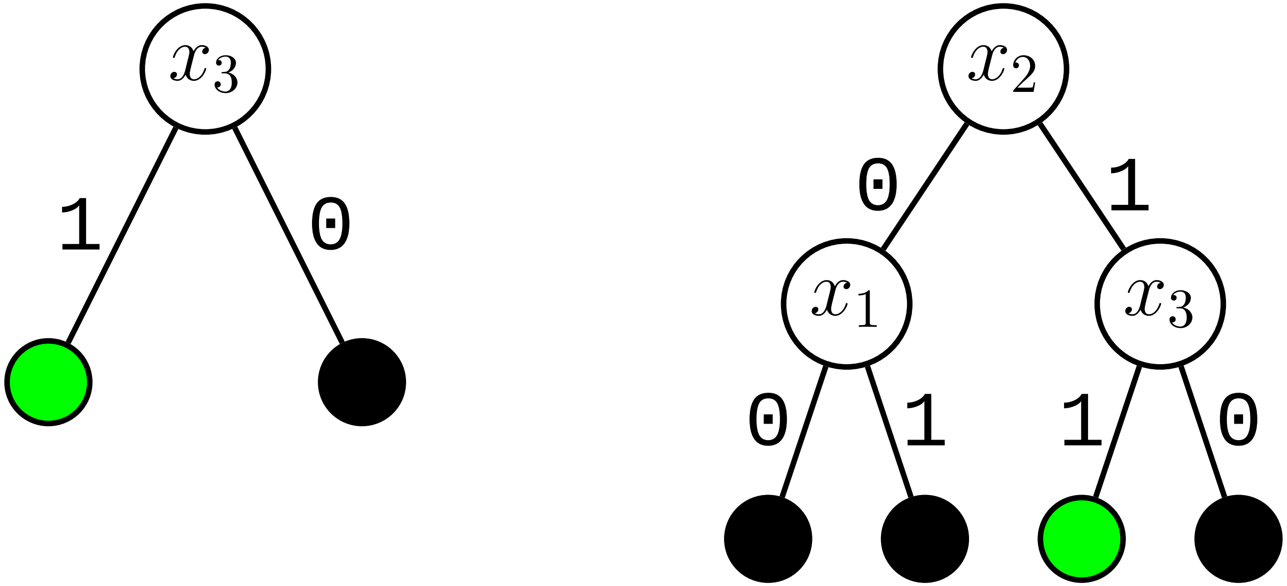
# DPLL: Example

$$\mathcal{F}_{\mathrm{DPLL}} := (x_1 \vee x_2 \vee \neg x_3) \wedge (\neg x_1 \vee x_2 \vee x_3)$$
$$(\neg x_1 \vee \neg x_2 \vee x_3) \wedge (x_1 \vee x_3) \wedge (\neg x_1 \vee \neg x_3)$$



# DPLL: Example

$$\mathcal{F}_{\text{DPLL}} := (x_1 \vee x_2 \vee \neg x_3) \wedge (\neg x_1 \vee x_2 \vee x_3) \wedge$$
$$(\neg x_1 \vee \neg x_2 \vee x_3) \wedge (x_1 \vee x_3) \wedge (\neg x_1 \vee \neg x_3)$$



# SAT solving: Decisions, implications

Decision variables

- Selected by the heuristics
- Play a crucial role in performance

Implied variables

- Assigned by reasoning (e.g. unit propagation)
- Maximizing the number of implied variables is an important aspect of look-ahead SAT solvers

# SAT Solving: Clauses $\leftrightarrow$ assignments

- A clause $C$ represents a set of falsified assignments, i.e. those assignments that falsify all literals in $C$
- A falsifying assignment $\varphi$ for a given formula represents a set of clauses that follow from the formula
  - For instance with all decision variables
  - Important feature of conflict-driven SAT solvers

# SAT solvers

## Conflict-driven

- "brute-force", complete
- examples: zchaff, minisat, rsat

## Look-ahead

- lots of reasoning, complete
- examples: march, OKsolver, kcnfs

## Local search

- local optimazations, incomplete
- examples: WalkSAT, UnitWalk

# Applications: Industrial

- Model Checking
  - Turing award ’07: Edmund M. Clarke
- Software Verification
- Hardware Verification
- Equivalence Checking Problems

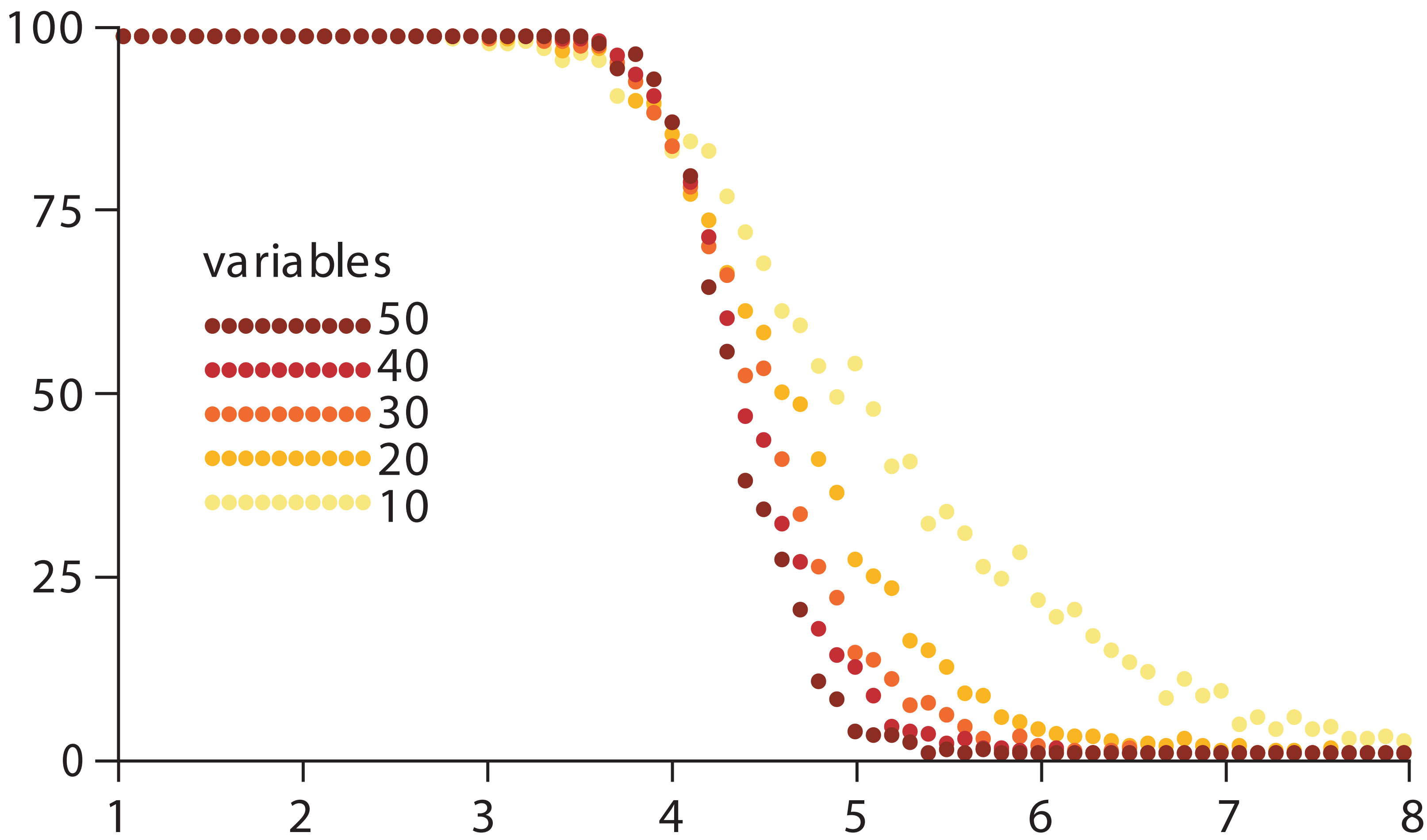
# Applications: Crafted

- Combinatorial problems
- Sudoku
- Factorization problems

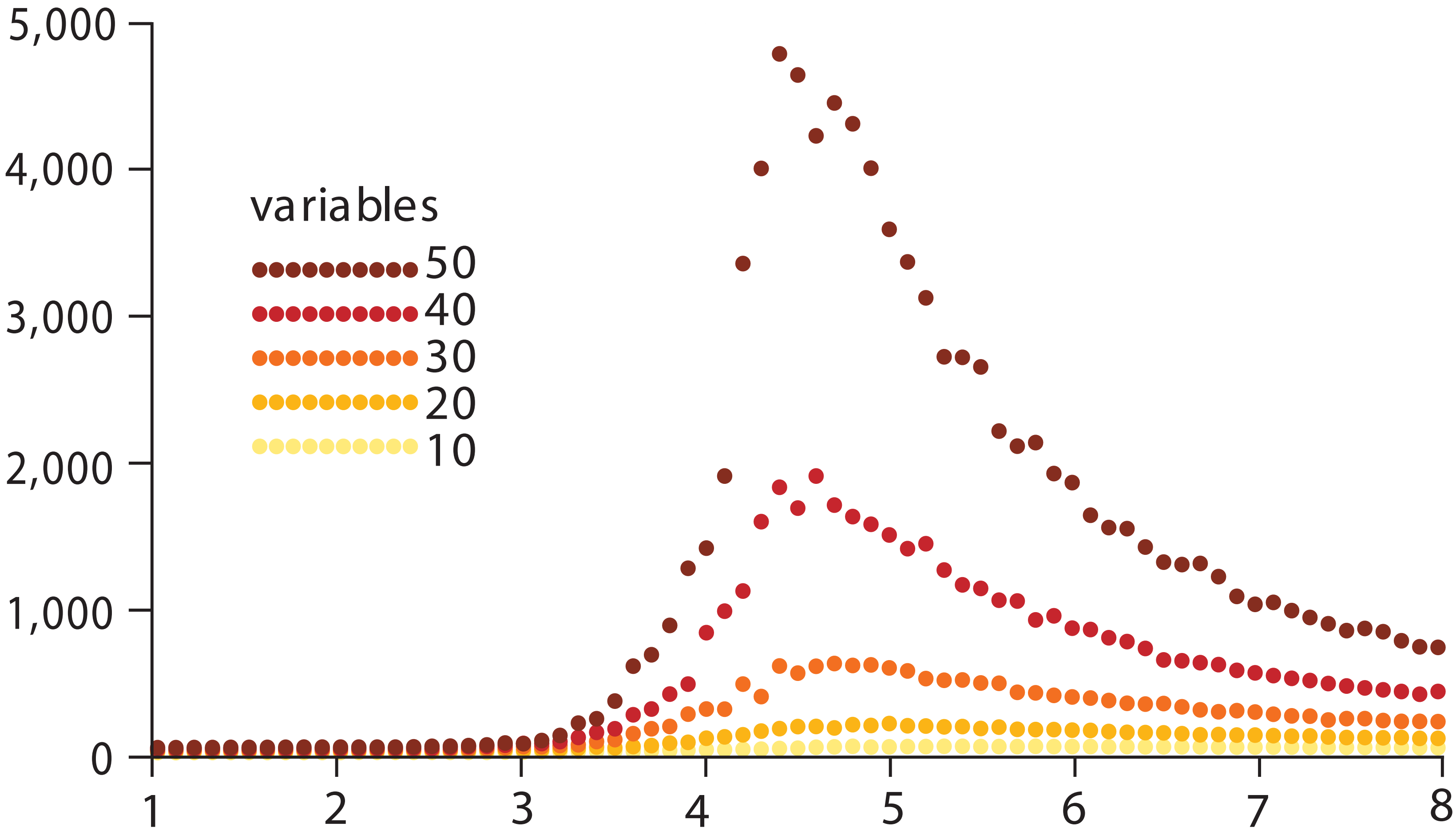
# Random $k$-SAT: Introduction

- All clauses have length $k$
- Variables have the same probability to occur
- Each literal is negated with probability of 50%
- Density is ratio Clauses to Variables

# Random 3-SAT: phase transition



# Random 3-SAT: threshold



# SAT game

SAT game

# Computational Logic and Satisfiability

## IN4077

# Introduction

**Marijn J.H. Heule**

September 9, 2008





TU Delft

Delft University of Technology