Introduction to Formal Methods

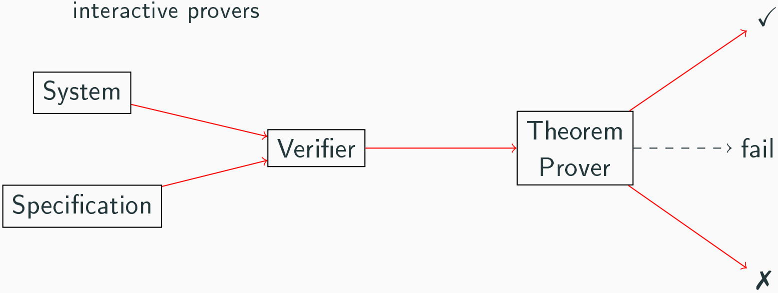
Lecture 2
Boolean Satisfiability (SAT) Solving

Hossein Hojjat & Fatemeh Ghassemi

September 25, 2018

(Deductive) Formal Verification Steps

1. **Modeling:** Create a mathematical model of the system
    - A modeling error can introduce false bugs or mask real bugs
    - For many systems, this step can be done automatically
2. **Specification:** Specify the correctness properties in a formal language
    - Challenge to translate informal specifications into formal ones
    - Many languages: UML, CTL, PSL, Spec#, etc.
3. **Proof:** Prove that the model satisfies the specification
    - Use a theorem prover for generated conditions
    - SAT solving, SMT solving, resolution-based theorem proving, rewriting, interactive provers



Automatic Theorem Provers

- Many real-world verification efforts require human expertise to complete the proofs
- If a computer can do the proof automatically, this greatly improves the feasibility of formal verification
- Automatic theorem provers have improved significantly in recent years
    - Enables formal verification of larger and more complex systems
- In this lecture we will look at one of the techniques for automated theorem proving: SAT solvers

Project

- Boolean Satisfiability is a well-known NP-complete decision problem
  - First NP-complete problem (Cook, 1971)
- Many practical applications in different areas of computer science
  - e.g. SMT solving, Bounded model checking
    (will discuss later in this course)
- Your first project: implement SAT solver
- This lecture: an overview of two SAT-solving algorithms:
  1. Truth Tables
  2. DPLL Algorithm

Propositional Logic

Boolean variable: variable with two possible values: **True** or **False**

Boolean Formula

- **True** and **False** are Boolean formulas
- Any Boolean variable x is a Boolean formula
- If ψ is a Boolean formula then $\overline{\psi}$ is a Boolean formula
- If ψ_1 and ψ_2 are Boolean formulas then $(\psi_1 \circ \psi_2)$ is a Boolean formula
    - $\circ \in \{\wedge, \vee, \rightarrow, \leftrightarrow\}$

Conjunctive Normal Form (CNF)

- Literal: Boolean variable or a negated Boolean variable
- Clause: Disjunction of literals
- CNF: (Conjunctive Normal Form) Conjunction of clauses

Example: CNF formula

$$(x_1 \vee x_2) \wedge (x_1 \vee \overline{x_2} \vee x_3) \wedge (\overline{x_1})$$

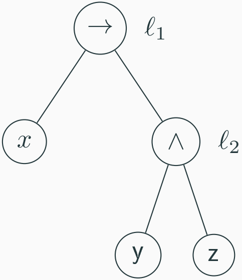
- Boolean variables: $\{x_1, x_2, x_3\}$
- Literals: $\{x_1, \overline{x_1}, x_2, \overline{x_2}, x_3\}$
- Clauses: $\{(x_1 \vee x_2), (x_1 \vee \overline{x_2} \vee x_3), (\overline{x_1})\}$

Tseitin Transformation:
Efficient transformation of an arbitrary Boolean formula to a CNF formula

Tseitin Transformation

1. Build a derivation tree with variables as leaves
2. Introduce a fresh variable for every internal node
3. Encode the meaning of the fresh variables with clauses
4. Conjoin the root with all the encoding clauses

Example: $\phi = (x \to (y \land z))$

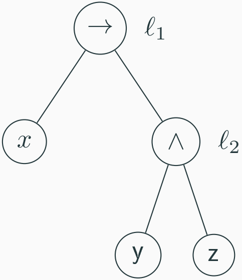


$$(\ell_1 \leftrightarrow (x \to \ell_2)) \land$$
$$(\ell_2 \leftrightarrow (y \land z)) \land$$
$$(\ell_1)$$

Tseitin Transformation

1. Build a derivation tree with variables as leaves
2. Introduce a fresh variable for every internal node
3. Encode the meaning of the fresh variables with clauses
4. Conjoin the root with all the encoding clauses

Example: $\phi = (x \to (y \wedge z))$

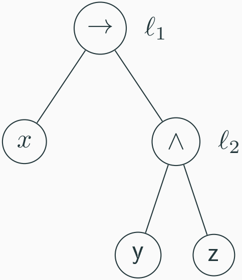


$$(\ell_1 \to (x \to \ell_2)) \wedge ((x \to \ell_2) \to \ell_1) \wedge$$
$$(\ell_2 \leftrightarrow (y \wedge z)) \wedge$$
$$(\ell_1)$$

Tseitin Transformation

1. Build a derivation tree with variables as leaves
2. Introduce a fresh variable for every internal node
3. Encode the meaning of the fresh variables with clauses
4. Conjoin the root with all the encoding clauses

Example: $\phi = (x \rightarrow (y \wedge z))$

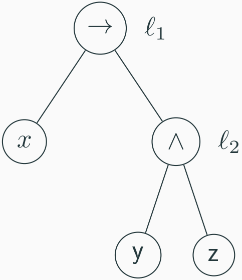


$$(\overline{\ell_1} \vee \overline{x} \vee \ell_2) \wedge (x \vee \ell_1) \wedge (\ell_1 \vee \overline{\ell_2}) \wedge$$
$$(\ell_2 \leftrightarrow (y \wedge z)) \wedge$$
$$(\ell_1)$$

Tseitin Transformation

1. Build a derivation tree with variables as leaves
2. Introduce a fresh variable for every internal node
3. Encode the meaning of the fresh variables with clauses
4. Conjoin the root with all the encoding clauses

Example: $\phi = (x \rightarrow (y \wedge z))$



$$(\overline{\ell_1} \vee \overline{x} \vee \ell_2) \wedge (x \vee \ell_1) \wedge (\ell_1 \vee \overline{\ell_2}) \wedge$$
$$(\overline{\ell_2} \vee y) \wedge (\overline{\ell_2} \vee z) \wedge (\overline{y} \vee \overline{z} \vee \ell_2) \wedge$$
$$(\ell_1)$$

Satisfiability

- A truth assignment assigns a truth value (**True** or **False**) to each Boolean variable

Boolean Satisfiability Problem:

- Given a Boolean formula find:
  - Variable assignment such that the formula evaluates to **True** (Satisfiable)
  - Prove that no such assignment exists (Unsatisfiable)

SAT Solver:

- Program to decide whether a given Boolean formula instance is satisfiable or unsatisfiable
- Usually takes input in Conjunctive Normal Form (CNF)

Satisfiable or Unsatisfiable?

- $(x_1 \vee \overline{x_3}) \wedge (x_2 \vee x_1 \vee x_3) \wedge (\overline{x_1})$

Satisfiable or Unsatisfiable?

- $(x_1 \vee \overline{x_3}) \wedge (x_2 \vee x_1 \vee x_3) \wedge (\overline{x_1})$
- Satisfiable: $\phi = \{x_1 \to \mathbf{F}, x_2 \to \mathbf{T}, x_3 \to \mathbf{F}\}$

Satisfiable or Unsatisfiable?

- $(x_1 \vee \overline{x_3}) \wedge (x_2 \vee x_1 \vee x_3) \wedge (\overline{x_1})$
- Satisfiable: $\phi = \{x_1 \to \mathbf{F}, x_2 \to \mathbf{T}, x_3 \to \mathbf{F}\}$

- $(x_1 \vee x_2) \wedge (\overline{x_1} \vee x_2) \wedge (x_1 \vee \overline{x_2}) \wedge (\overline{x_1} \vee \overline{x_2})$

Satisfiable or Unsatisfiable?

- $(x_1 \vee \overline{x_3}) \wedge (x_2 \vee x_1 \vee x_3) \wedge (\overline{x_1})$
- Satisfiable: $\phi = \{x_1 \to \mathbf{F}, x_2 \to \mathbf{T}, x_3 \to \mathbf{F}\}$

- $(x_1 \vee x_2) \wedge (\overline{x_1} \vee x_2) \wedge (x_1 \vee \overline{x_2}) \wedge (\overline{x_1} \vee \overline{x_2})$
- Unsatisfiable

Satisfiable or Unsatisfiable?

- $(x_1 \vee \overline{x_3}) \wedge (x_2 \vee x_1 \vee x_3) \wedge (\overline{x_1})$
- Satisfiable: $\phi = \{x_1 \rightarrow \mathbf{F}, x_2 \rightarrow \mathbf{T}, x_3 \rightarrow \mathbf{F}\}$

- $(x_1 \vee x_2) \wedge (\overline{x_1} \vee x_2) \wedge (x_1 \vee \overline{x_2}) \wedge (\overline{x_1} \vee \overline{x_2})$
- Unsatisfiable

- $(\overline{x_1} \vee x_2 \vee \overline{x_3}) \wedge (x_1 \vee x_2) \wedge (\overline{x_1} \vee x_2 \vee x_3) \wedge (\overline{x_2})$

Satisfiable or Unsatisfiable?

- $(x_1 \vee \overline{x_3}) \wedge (x_2 \vee x_1 \vee x_3) \wedge (\overline{x_1})$
- Satisfiable: $\phi = \{x_1 \rightarrow \mathbf{F}, x_2 \rightarrow \mathbf{T}, x_3 \rightarrow \mathbf{F}\}$

- $(x_1 \vee x_2) \wedge (\overline{x_1} \vee x_2) \wedge (x_1 \vee \overline{x_2}) \wedge (\overline{x_1} \vee \overline{x_2})$
- Unsatisfiable

- $(\overline{x_1} \vee x_2 \vee \overline{x_3}) \wedge (x_1 \vee x_2) \wedge (\overline{x_1} \vee x_2 \vee x_3) \wedge (\overline{x_2})$
- Unsatisfiable

Satisfiable or Unsatisfiable?

- $(x_1 \lor \overline{x_3}) \land (x_2 \lor x_1 \lor x_3) \land (\overline{x_1})$
- Satisfiable: $\phi = \{x_1 \to \mathbf{F}, x_2 \to \mathbf{T}, x_3 \to \mathbf{F}\}$

- $(x_1 \lor x_2) \land (\overline{x_1} \lor x_2) \land (x_1 \lor \overline{x_2}) \land (\overline{x_1} \lor \overline{x_2})$
- Unsatisfiable

- $(\overline{x_1} \lor x_2 \lor \overline{x_3}) \land (x_1 \lor x_2) \land (\overline{x_1} \lor x_2 \lor x_3) \land (\overline{x_2})$
- Unsatisfiable

- $(\overline{x_1} \lor x_2 \lor \overline{x_3}) \land (x_1 \lor \overline{x_2}) \land (\overline{x_1} \lor x_2 \lor x_3) \land (\overline{x_2} \lor \overline{x_3})$

Satisfiable or Unsatisfiable?

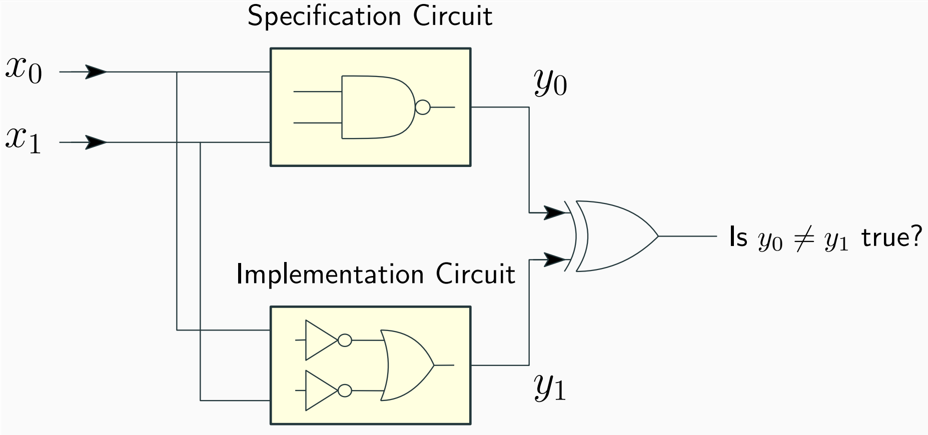
- $(x_1 \vee \overline{x_3}) \wedge (x_2 \vee x_1 \vee x_3) \wedge (\overline{x_1})$
- Satisfiable: $\phi = \{x_1 \to \mathbf{F}, x_2 \to \mathbf{T}, x_3 \to \mathbf{F}\}$

- $(x_1 \vee x_2) \wedge (\overline{x_1} \vee x_2) \wedge (x_1 \vee \overline{x_2}) \wedge (\overline{x_1} \vee \overline{x_2})$
- Unsatisfiable

- $(\overline{x_1} \vee x_2 \vee \overline{x_3}) \wedge (x_1 \vee x_2) \wedge (\overline{x_1} \vee x_2 \vee x_3) \wedge (\overline{x_2})$
- Unsatisfiable

- $(\overline{x_1} \vee x_2 \vee \overline{x_3}) \wedge (x_1 \vee \overline{x_2}) \wedge (\overline{x_1} \vee x_2 \vee x_3) \wedge (\overline{x_2} \vee \overline{x_3})$
- Satisfiable: $\phi = \{x_1 \to \mathbf{F}, x_2 \to \mathbf{F}, x_3 \to \mathbf{F}\}$

Example: Equivalence Verification



Truth Tables

Tabulate values of Boolean formula for all possible values of its Boolean variables

x	$\overline{x}$
F	
T	

x_2	x_1	$x_1 \wedge x_2$
F	**F**	
F	**T**	
T	**F**	
T	**T**	

x_2	x_1	$x_1 \vee x_2$
F	**F**	
F	**T**	
T	**F**	
T	**T**	

Truth Tables

Tabulate values of Boolean formula for all possible values of its Boolean variables

x	$\overline{x}$
F	**T**
T	**F**

x_2	x_1	$x_1 \wedge x_2$
F	**F**	
F	**T**	
T	**F**	
T	**T**	

x_2	x_1	$x_1 \vee x_2$
F	**F**	
F	**T**	
T	**F**	
T	**T**	

Truth Tables

Tabulate values of Boolean formula for all possible values of its Boolean variables

x	$\overline{x}$
F	**T**
T	**F**

x_2	x_1	$x_1 \wedge x_2$
F	**F**	**F**
F	**T**	**F**
T	**F**	**F**
T	**T**	**T**

x_2	x_1	$x_1 \vee x_2$
F	**F**	
F	**T**	
T	**F**	
T	**T**	

Truth Tables

Tabulate values of Boolean formula for all possible values of its Boolean variables

x	$\overline{x}$
F	**T**
T	**F**

x_2	x_1	$x_1 \wedge x_2$
F	**F**	**F**
F	**T**	**F**
T	**F**	**F**
T	**T**	**T**

x_2	x_1	$x_1 \vee x_2$
F	**F**	**F**
F	**T**	**T**
T	**F**	**T**
T	**T**	**T**

Truth Tables

Tabulate values of Boolean formula for all possible values of its Boolean variables

x_3	x_2	x_1	x_2	$\wedge$	$(x_1$	$\vee$	$\overline{x_3})$
F	**F**	**F**					
F	**F**	**T**					
F	**T**	**F**					
F	**T**	**T**					
T	**F**	**F**					
T	**F**	**T**					
T	**T**	**F**					
T	**T**	**T**					

Truth Tables

Tabulate values of Boolean formula for all possible values of its Boolean variables

x_3	x_2	x_1	x_2	$\wedge$	$(x_1$	$\vee$	$\overline{x_3})$
F	**F**	**F**	**F**		**F**		
F	**F**	**T**	**F**		**T**		
F	**T**	**F**	**T**		**F**		
F	**T**	**T**	**T**		**T**		
T	**F**	**F**	**F**		**F**		
T	**F**	**T**	**F**		**T**		
T	**T**	**F**	**T**		**F**		
T	**T**	**T**	**T**		**T**		

Truth Tables

Tabulate values of Boolean formula for all possible values of its Boolean variables

x_3	x_2	x_1	x_2	$\wedge$	$(x_1$	$\vee$	$\overline{x_3})$
F	**F**	**F**	**F**		**F**		**T**
F	**F**	**T**	**F**		**T**		**T**
F	**T**	**F**	**T**		**F**		**T**
F	**T**	**T**	**T**		**T**		**T**
T	**F**	**F**	**F**		**F**		**F**
T	**F**	**T**	**F**		**T**		**F**
T	**T**	**F**	**T**		**F**		**F**
T	**T**	**T**	**T**		**T**		**F**

Truth Tables

Tabulate values of Boolean formula for all possible values of its Boolean variables

x_3	x_2	x_1	x_2	$\wedge$	$(x_1$	$\vee$	$\overline{x_3})$
F	**F**	**F**	**F**		**F**	**T**	**T**
F	**F**	**T**	**F**		**T**	**T**	**T**
F	**T**	**F**	**T**		**F**	**T**	**T**
F	**T**	**T**	**T**		**T**	**T**	**T**
T	**F**	**F**	**F**		**F**	**F**	**F**
T	**F**	**T**	**F**		**T**	**T**	**F**
T	**T**	**F**	**T**		**F**	**F**	**F**
T	**T**	**T**	**T**		**T**	**T**	**F**

Truth Tables

Tabulate values of Boolean formula for all possible values of its Boolean variables

x_3	x_2	x_1	x_2	$\wedge$	$(x_1$	$\vee$	$\overline{x_3})$
F	**F**	**F**	**F**	**F**	**F**	**T**	**T**
F	**F**	**T**	**F**	**F**	**T**	**T**	**T**
F	**T**	**F**	**T**	**T**	**F**	**T**	**T**
F	**T**	**T**	**T**	**T**	**T**	**T**	**T**
T	**F**	**F**	**F**	**F**	**F**	**F**	**F**
T	**F**	**T**	**F**	**F**	**T**	**T**	**F**
T	**T**	**F**	**T**	**F**	**F**	**F**	**F**
T	**T**	**T**	**T**	**T**	**T**	**T**	**F**

Truth Tables

Tabulate values of Boolean formula for all possible values of its Boolean variables

x_3	x_2	x_1	x_2	$\wedge$	$(x_1$	$\vee$	$\overline{x_3})$
F	**F**	**F**	**F**	**F**	**F**	**T**	**T**
F	**F**	**T**	**F**	**F**	**T**	**T**	**T**
F	**T**	**F**	**T**	**T**	**F**	**T**	**T**
F	**T**	**T**	**T**	**T**	**T**	**T**	**T**
T	**F**	**F**	**F**	**F**	**F**	**F**	**F**
T	**F**	**T**	**F**	**F**	**T**	**T**	**F**
T	**T**	**F**	**T**	**F**	**F**	**F**	**F**
T	**T**	**T**	**T**	**T**	**T**	**T**	**F**

Algorithm

- To check whether a Boolean formula α is satisfiable, form the truth table for α:
- If there is a row in which **T** appears as the value for α, then α is satisfiable
- Otherwise, α is unsatisfiable

Run-time Complexity

- What is the complexity of the truth table algorithm?

Run-time Complexity

- What is the complexity of the truth table algorithm?
- 2^n where n is the number of Boolean variables

Run-time Complexity

- What is the complexity of the truth table algorithm?
- 2^n where n is the number of Boolean variables
- Can we do better?

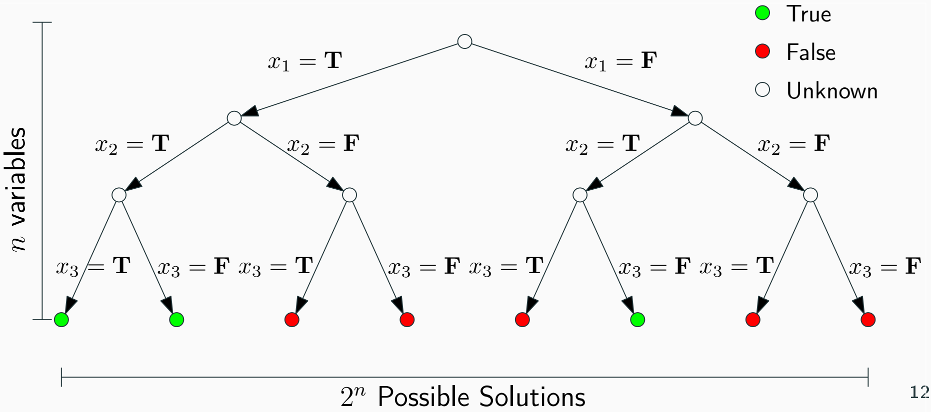
Run-time Complexity

- What is the complexity of the truth table algorithm?
- 2^n where n is the number of Boolean variables
- Can we do better?
- SAT was the first problem shown to be NP-complete
- In worst case, we need to spend the exponential time
- However, we can use heuristics to solve many formulas faster
- Modern SAT solvers are extremely fast most of the time!

Search Tree

- Binary search tree: at each node Boolean variable is set to a value
- SAT solver performs backtrack search in the tree

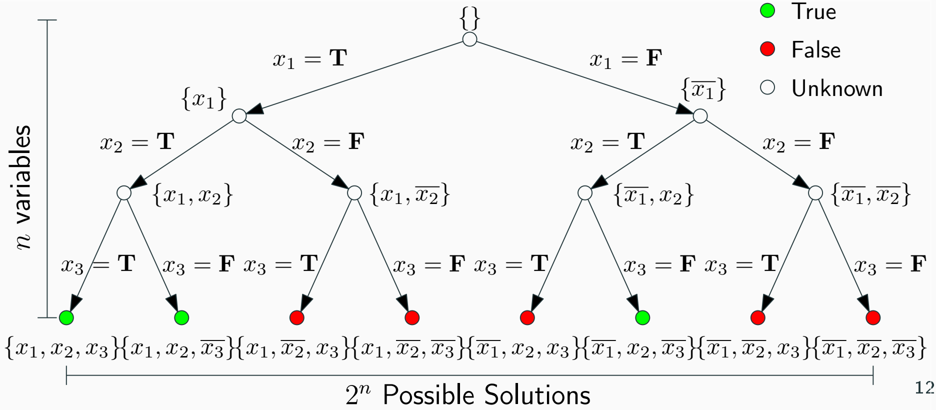
$$x_2 \wedge (x_1 \vee \overline{x_3})$$



Search Tree

- Partial Truth Assignment: assignment to a subset of the Boolean variables
- Search algorithm gradually fills out a partial assignment until:
  - find a satisfying full assignment (if any)
  - backtrack to another partial assignment

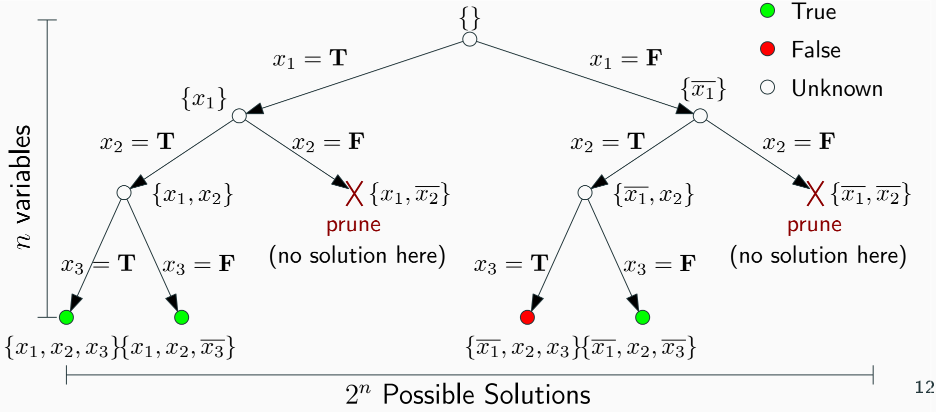
$$x_2 \wedge (x_1 \vee \overline{x_3})$$



Search Tree

- We can use heuristics to prune the search tree

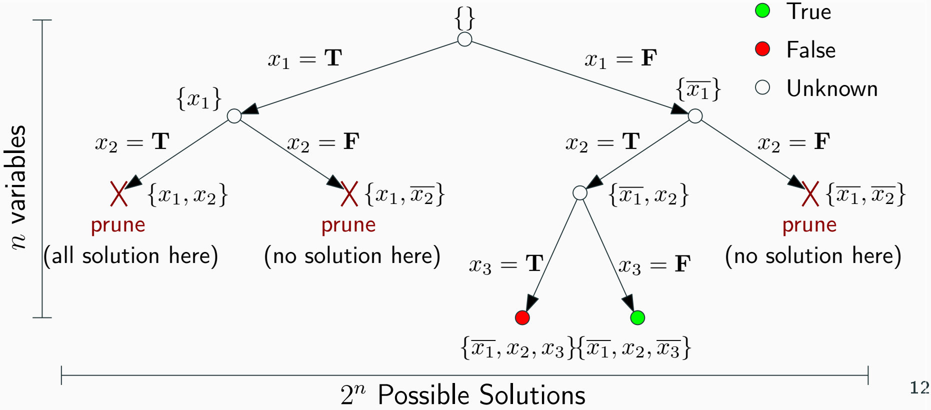
$$x_2 \wedge (x_1 \vee \overline{x_3})$$



Search Tree

- We can use heuristics to prune the search tree

$$x_2 \wedge (x_1 \vee \overline{x_3})$$



Heuristic: Early termination

- A clause becomes $\mathbf{T}$ when one of its literals is $\mathbf{T}$
  - e.g. if x_2 is $\mathbf{T}$ then $(\overline{x_1} \vee x_2 \vee \overline{x_3})$ is $\mathbf{T}$
- A formula becomes $\mathbf{F}$ if any of its clauses is $\mathbf{F}$
  - e.g. if x_2 is $\mathbf{F}$ then $x_2 \wedge (x_1 \vee \overline{x_3})$ is $\mathbf{F}$

During the search if the partial assignment:

- Makes a literal $\mathbf{T}$ then:
  simplify the formula by removing all the clauses that have that literal
- Makes a clause $\mathbf{F}$ then:
  stop the search and backtrack

Heuristic: Pure Variables

- Pure variable: always appears with the same "sign" in all clauses
- e.g., in the three clauses $(x_1 \vee x_2) \wedge (\overline{x_3} \vee x_1) \wedge (\overline{x_2} \vee \overline{x_3})$
  x_1 and x_3 are pure, x_2 is impure
- Make literals with pure symbols $\mathbf{T}$ for satisfiability
- Let x_1 and $\overline{x_3}$ be both $\mathbf{T}$ in example above

Heuristic: Unit Propagation

- Unit Clause: only one literal in the clause, e.g. (x_1)
- The only literal in a unit clause must be $\mathbf{T}$
- e.g., x_1 must be $\mathbf{T}$ in example above
- Also includes clauses where all but one literal is $\mathbf{F}$
- e.g. $(x_1 \vee x_2 \vee x_3)$ where x_2 and x_3 are $\mathbf{F}$

- Unit Propagation (a.k.a "Boolean Constraint Propagation" or BCP): the key component in modern SAT solvers
- Iteratively apply unit propagation until there is no unit clause

Exercise

Apply unit propagation to the following formula:

$$(x_1) \wedge (\overline{x_1} \vee x_2 \vee x_3) \wedge (\overline{x_2}) \wedge (x_1 \vee x_4 \vee x_5) \wedge (\overline{x_3} \vee x_5)$$

Exercise

Apply unit propagation to the following formula:

$$(x_1) \wedge (\overline{x_1} \vee x_2 \vee x_3) \wedge (\overline{x_2}) \wedge (x_1 \vee x_4 \vee x_5) \wedge (\overline{x_3} \vee x_5)$$

$\boxed{x_1 = \textbf{True}}$

$$(x_2 \vee x_3) \wedge (\overline{x_2}) \wedge (\overline{x_3} \vee x_5)$$

$\boxed{x_2 = \textbf{False}}$

$$(x_3) \wedge (\overline{x_3} \vee x_5)$$

$\boxed{x_3 = \textbf{True}}$

$$(x_5)$$

$\boxed{x_5 = \textbf{True}}$

$$\textbf{True}$$

DPLL

- DPLL: popular complete satisfiability checking algorithms
  - There are incomplete approaches such as stochastic search as well
- Davis-Putnam procedure was introduced in 1960 by Martin Davis and Hilary Putnam
- Two years later, Martin Davis, George Logemann, and Donald W. Loveland introduced a refined version of the algorithm
- Nowadays, the later version of the algorithm is often referred to as DPLL procedure
  - **D**avis - **P**utnam - **L**ogemann - **L**oveland procedure

DPLL

DPLL(ϕ):

- Apply unit propagation
- If $\{x, \overline{x}\} \in \text{clauses}(\phi)$ for some x, return **UNSAT**
- Apply pure literal rule
- If ϕ is satisfied (empty), return **SAT**
- Select decision variable x
    - If DPLL($\phi \wedge x$) $=$ **SAT** return **SAT**
    - return DPLL($\phi \wedge \overline{x}$)

DPLL

DPLL(ϕ):

- Apply unit propagation
- If $\{x, \overline{x}\} \in$ clauses(ϕ) for some x, return **UNSAT**
- Apply pure literal rule
- If ϕ is satisfied (empty), return **SAT**
- Select decision variable x
  - If DPLL($\phi \wedge x$) = **SAT** return **SAT**
  - return DPLL($\phi \wedge \overline{x}$)

Example:

$x_1 \vee x_2$

$x_1 \vee \overline{x_2}$

$\overline{x_1} \vee x_3 \vee x_4$

$\overline{x_1} \vee \overline{x_3} \vee x_4$

$\overline{x_1} \vee \overline{x_4}$

$\overline{x_1} \vee x_4 \vee x_5$

DPLL

DPLL(ϕ):

- Apply unit propagation
- If $\{x, \overline{x}\} \in$ clauses(ϕ) for some x, return **UNSAT**
- $\rightarrow$ Apply pure literal rule
- If ϕ is satisfied (empty), return **SAT**
- Select decision variable x
  - If DPLL($\phi \wedge x$) = **SAT** return **SAT**
  - return DPLL($\phi \wedge \overline{x}$)

Example:

$x_1 \vee x_2$

$x_1 \vee \overline{x_2}$

$\overline{x_1} \vee x_3 \vee x_4$

$\overline{x_1} \vee \overline{x_3} \vee x_4$

$\overline{x_1} \vee \overline{x_4}$

~~$\overline{x_1} \vee x_4 \vee x_5$~~

(Pure Literal Rule)

DPLL

DPLL(ϕ):

- Apply unit propagation
- If $\{x, \overline{x}\} \in \text{clauses}(\phi)$ for some x, return **UNSAT**
- $\rightarrow$ Apply pure literal rule
- If ϕ is satisfied (empty), return **SAT**
- Select decision variable x
  - If DPLL($\phi \wedge x$) = **SAT** return **SAT**
  - return DPLL($\phi \wedge \overline{x}$)

Example:

$x_1 \vee x_2$

$x_1 \vee \overline{x_2}$

$\overline{x_1} \vee x_3 \vee x_4$

$\overline{x_1} \vee \overline{x_3} \vee x_4$

$\overline{x_1} \vee \overline{x_4}$

(Pure Literal Rule)

DPLL

DPLL(ϕ):

- Apply unit propagation
- If $\{x, \overline{x}\} \in$ clauses(ϕ) for some x, return **UNSAT**
- Apply pure literal rule
- If ϕ is satisfied (empty), return **SAT**
- Select decision variable x

$\rightarrow$
  - If DPLL($\phi \wedge x$) = **SAT** return **SAT**
  - return DPLL($\phi \wedge \overline{x}$)

Example:

$x_1 \vee x_2$

$x_1 \vee \overline{x_2}$

$\overline{x_1} \vee x_3 \vee x_4$

$\overline{x_1} \vee \overline{x_3} \vee x_4$

$\overline{x_1} \vee \overline{x_4}$

x_4

(Select x_4)

DPLL

DPLL(ϕ):

- → Apply unit propagation
- If $\{x, \overline{x}\} \in \text{clauses}(\phi)$ for some x, return **UNSAT**
- Apply pure literal rule
- If ϕ is satisfied (empty), return **SAT**
- Select decision variable x
  - If DPLL($\phi \wedge x$) = **SAT** return **SAT**
  - return DPLL($\phi \wedge \overline{x}$)

Example:

$x_1 \vee x_2$

$x_1 \vee \overline{x_2}$

$\overline{x_1}$

(Select x_4)

(Unit Propagation)

DPLL

DPLL(ϕ):

→ • Apply unit propagation
- If $\{x, \overline{x}\} \in \text{clauses}(\phi)$ for some x, return **UNSAT**
- Apply pure literal rule
- If ϕ is satisfied (empty), return **SAT**
- Select decision variable x
  - If DPLL($\phi \wedge x$) = **SAT** return **SAT**
  - return DPLL($\phi \wedge \overline{x}$)

Example:

~~$x_1 \vee$~~ x_2

~~$x_1 \vee$~~ $\overline{x_2}$

(Select x_4)

(Unit Propagation)

(Unit Propagation)

DPLL

DPLL(ϕ):

- Apply unit propagation
- $\rightarrow$ If $\{x, \overline{x}\} \in$ clauses(ϕ) for some x, return **UNSAT**
- Apply pure literal rule
- If ϕ is satisfied (empty), return **SAT**
- Select decision variable x
  - If DPLL($\phi \wedge x$) = **SAT** return **SAT**
  - return DPLL($\phi \wedge \overline{x}$)

Example:

~~$x_1 \vee$~~ x_2

~~$x_1 \vee$~~ $\overline{x_2}$ Conflict!

Backtrack $\longrightarrow$ (Select x_4)

(Unit Propagation)

(Unit Propagation)

DPLL

DPLL(ϕ):

- Apply unit propagation
- If $\{x, \overline{x}\} \in \text{clauses}(\phi)$ for some x, return **UNSAT**
- Apply pure literal rule
- If ϕ is satisfied (empty), return **SAT**
- Select decision variable x
  - If DPLL($\phi \wedge x$) = **SAT** return **SAT**
  - $\rightarrow$ return DPLL($\phi \wedge \overline{x}$)

Example:

$x_1 \vee x_2$

$x_1 \vee \overline{x_2}$

$\overline{x_1} \vee x_3 \vee x_4$

$\overline{x_1} \vee \overline{x_3} \vee x_4$

$\overline{x_1} \vee \overline{x_4}$

$\overline{x_4}$

(Select $\overline{x_4}$)

DPLL

DPLL(ϕ):

$\rightarrow$
- Apply unit propagation
- If $\{x, \overline{x}\} \in \text{clauses}(\phi)$ for some x, return **UNSAT**
- Apply pure literal rule
- If ϕ is satisfied (empty), return **SAT**
- Select decision variable x
  - If DPLL($\phi \wedge x$) = **SAT** return **SAT**
  - return DPLL($\phi \wedge \overline{x}$)

Example:

$x_1 \vee x_2$

$x_1 \vee \overline{x_2}$

$\overline{x_1} \vee x_3$

$\overline{x_1} \vee \overline{x_3}$

(Select $\overline{x_4}$)

(Unit Propagation)

DPLL

DPLL(ϕ):

- Apply unit propagation
- If $\{x, \overline{x}\} \in$ clauses(ϕ) for some x, return **UNSAT**
- Apply pure literal rule
- If ϕ is satisfied (empty), return **SAT**
- Select decision variable x
  - $\rightarrow$ If DPLL($\phi \wedge x$) = **SAT** return **SAT**
  - return DPLL($\phi \wedge \overline{x}$)

Example:

$x_1 \vee x_2$

$x_1 \vee \overline{x_2}$

$\overline{x_1} \vee x_3$

$\overline{x_1} \vee \overline{x_3}$

x_1

(Select $\overline{x_4}$)

(Unit Propagation)

(Select x_1)

DPLL

DPLL(ϕ):

$\rightarrow$ • Apply unit propagation
- If $\{x, \overline{x}\} \in$ clauses(ϕ) for some x, return **UNSAT**
- Apply pure literal rule
- If ϕ is satisfied (empty), return **SAT**
- Select decision variable x
  - If DPLL($\phi \wedge x$) = **SAT** return **SAT**
  - return DPLL($\phi \wedge \overline{x}$)

Example:

~~$\overline{x_1} \vee$~~ x_3

~~$\overline{x_1} \vee$~~ $\overline{x_3}$

(Select $\overline{x_4}$)
(Unit Propagation)
(Select x_1)
(Unit Propagation)

DPLL

Example:

DPLL(ϕ):

- Apply unit propagation

→ • If $\{x, \overline{x}\} \in$ clauses(ϕ) for some x, return **UNSAT**

- Apply pure literal rule
- If ϕ is satisfied (empty), return **SAT**
- Select decision variable x
  - If DPLL($\phi \wedge x$) = **SAT** return **SAT**
  - return DPLL($\phi \wedge \overline{x}$)

~~$\overline{x_1} \vee$~~ x_3

~~$\overline{x_1} \vee$~~ $\overline{x_3}$

Conflict!

(Select $\overline{x_4}$)

(Unit Propagation)

Backtrack ⟶ (Select x_1)

(Unit Propagation)

DPLL

DPLL(ϕ):

- Apply unit propagation
- If $\{x, \overline{x}\} \in$ clauses(ϕ) for some x, return **UNSAT**
- Apply pure literal rule
- If ϕ is satisfied (empty), return **SAT**
- Select decision variable x
    - If DPLL($\phi \wedge x$) = **SAT** return **SAT**
$\rightarrow$
    - return DPLL($\phi \wedge \overline{x}$)

$x_1 \vee x_2$

$x_1 \vee \overline{x_2}$

$\overline{x_1} \vee x_3$

$\overline{x_1} \vee \overline{x_3}$

$\overline{x_1}$

(Select $\overline{x_4}$)
(Unit Propagation)
(Select $\overline{x_1}$)

DPLL

DPLL(ϕ):

$\rightarrow$
- Apply unit propagation
- If $\{x, \overline{x}\} \in$ clauses(ϕ) for some x, return **UNSAT**
- Apply pure literal rule
- If ϕ is satisfied (empty), return **SAT**
- Select decision variable x
  - If DPLL($\phi \wedge x$) = **SAT** return **SAT**
  - return DPLL($\phi \wedge \overline{x}$)

~~$x_1 \vee$~~ x_2

~~$x_1 \vee$~~ $\overline{x_2}$

(Select $\overline{x_4}$)

(Unit Propagation)

(Select $\overline{x_1}$)

(Unit Propagation)

DPLL

DPLL(ϕ):

- Apply unit propagation
- → If $\{x, \overline{x}\} \in$ clauses(ϕ) for some x, return **UNSAT**
- Apply pure literal rule
- If ϕ is satisfied (empty), return **SAT**
- Select decision variable x
  - If DPLL($\phi \wedge x$) = **SAT** return **SAT**
  - return DPLL($\phi \wedge \overline{x}$)

~~$x_1 \vee$~~ x_2

~~$x_1 \vee$~~ $\overline{x_2}$

Conflict!

Nowhere to backtrack to now, DPLL returns **UNSAT**

(Select $\overline{x_4}$)

(Unit Propagation)

(Select $\overline{x_1}$)

(Unit Propagation)

Modern SAT Solvers

CDCL = conflict-driven clause learning

- Smart unit-clause preference
- Deterministic and randomized search restarts
- Boolean constraint propagation using lazy data structures
- Conflict-based adaptive branching
- ...

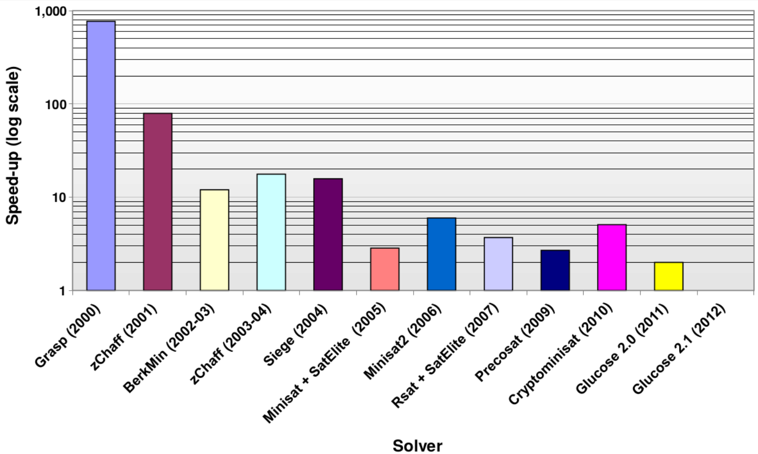
Key Tools: GRASP (1996); Chaff (2001)

Current Capacity: millions of variables

Competition:

- International SAT Solver Competition
- `http://www.satcompetition.org/`

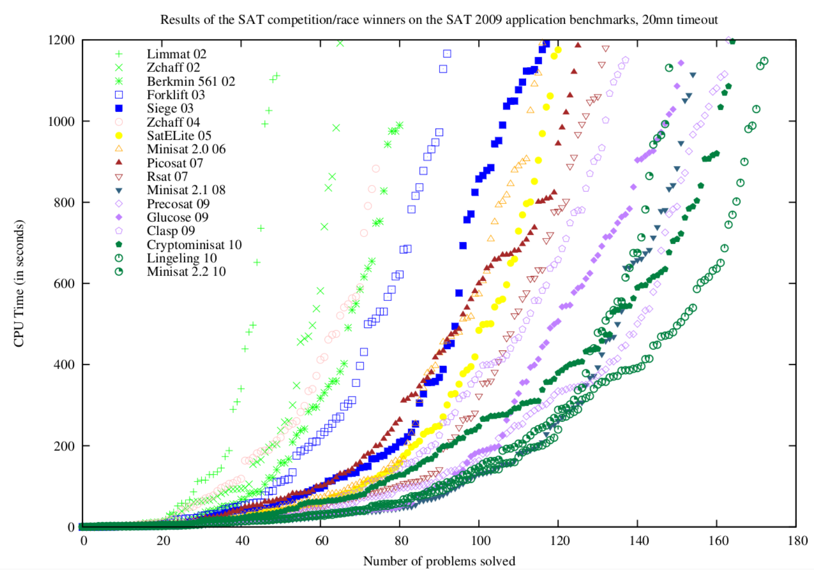
Speed-up of 2012 Solver over other Solvers



from Moshe Vardi

https://www.cs.rice.edu/~vardi/papers/highlights15.pdf

SAT Solver Comparison



(Daniel Le Berre)