



# CSCI 740 - Programming Language Theory

---

## Lecture 5

### Boolean Satisfiability (SAT) Solving

Instructor: Hossein Hojjat

September 8, 2017

- 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 in Scala
- This lecture: an overview of two SAT-solving algorithms:
  1. Truth Tables
  2. DPLL Algorithm

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  $\psi$  is a Boolean formula then  $\overline{\psi}$  is a Boolean formula
- If  $\psi_1$  and  $\psi_2$  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

# 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:  $\phi = \{x_1 \rightarrow \mathbf{F}, x_2 \rightarrow \mathbf{T}, x_3 \rightarrow \mathbf{F}\}$

## Satisfiable or Unsatisfiable?

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



## 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

## 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

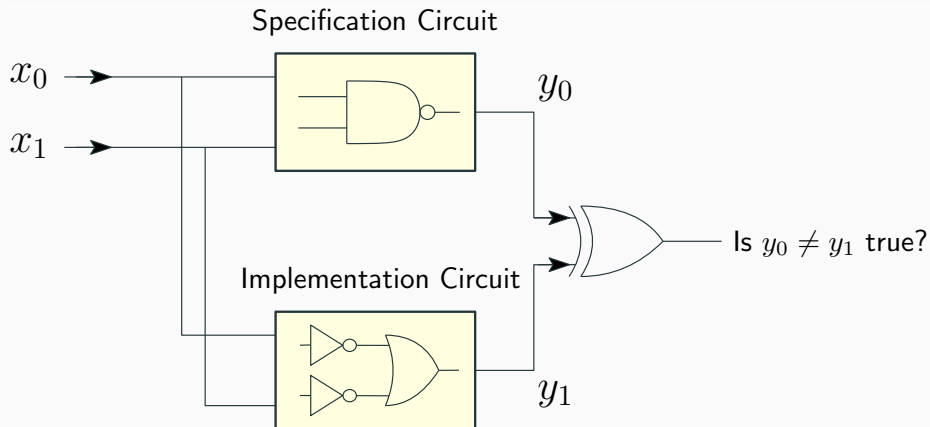
## Satisfiable or Unsatisfiable?

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

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

## Example: Equivalence Verification



# Truth Tables

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

$x$	$\bar{x}$
<b>F</b>	
<b>T</b>	

$x_2$	$x_1$	$x_1 \wedge x_2$
<b>F</b>	<b>F</b>	
<b>F</b>	<b>T</b>	
<b>T</b>	<b>F</b>	
<b>T</b>	<b>T</b>	

$x_2$	$x_1$	$x_1 \vee x_2$
<b>F</b>	<b>F</b>	
<b>F</b>	<b>T</b>	
<b>T</b>	<b>F</b>	
<b>T</b>	<b>T</b>	

# Truth Tables

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

$x$	$\bar{x}$
<b>F</b>	<b>T</b>
<b>T</b>	<b>F</b>

$x_2$	$x_1$	$x_1 \wedge x_2$
<b>F</b>	<b>F</b>	
<b>F</b>	<b>T</b>	
<b>T</b>	<b>F</b>	
<b>T</b>	<b>T</b>	

$x_2$	$x_1$	$x_1 \vee x_2$
<b>F</b>	<b>F</b>	
<b>F</b>	<b>T</b>	
<b>T</b>	<b>F</b>	
<b>T</b>	<b>T</b>	



# Truth Tables

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

$x$	$\bar{x}$
<b>F</b>	<b>T</b>
<b>T</b>	<b>F</b>

$x_2$	$x_1$	$x_1 \wedge x_2$
<b>F</b>	<b>F</b>	<b>F</b>
<b>F</b>	<b>T</b>	<b>F</b>
<b>T</b>	<b>F</b>	<b>F</b>
<b>T</b>	<b>T</b>	<b>T</b>

$x_2$	$x_1$	$x_1 \vee x_2$
<b>F</b>	<b>F</b>	
<b>F</b>	<b>T</b>	
<b>T</b>	<b>F</b>	
<b>T</b>	<b>T</b>	

# Truth Tables

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

$x$	$\bar{x}$
<b>F</b>	<b>T</b>
<b>T</b>	<b>F</b>

$x_2$	$x_1$	$x_1 \wedge x_2$
<b>F</b>	<b>F</b>	<b>F</b>
<b>F</b>	<b>T</b>	<b>F</b>
<b>T</b>	<b>F</b>	<b>F</b>
<b>T</b>	<b>T</b>	<b>T</b>

$x_2$	$x_1$	$x_1 \vee x_2$
<b>F</b>	<b>F</b>	<b>F</b>
<b>F</b>	<b>T</b>	<b>T</b>
<b>T</b>	<b>F</b>	<b>T</b>
<b>T</b>	<b>T</b>	<b>T</b>

# Truth Tables

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

$x_3$	$x_2$	$x_1$	$x_2 \wedge (x_1 \vee \overline{x_3})$
<b>F</b>	<b>F</b>	<b>F</b>	
<b>F</b>	<b>F</b>	<b>T</b>	
<b>F</b>	<b>T</b>	<b>F</b>	
<b>F</b>	<b>T</b>	<b>T</b>	
<b>T</b>	<b>F</b>	<b>F</b>	
<b>T</b>	<b>F</b>	<b>T</b>	
<b>T</b>	<b>T</b>	<b>F</b>	
<b>T</b>	<b>T</b>	<b>T</b>	

# Truth Tables

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

$x_3$	$x_2$	$x_1$	$x_2 \wedge (x_1 \vee \overline{x_3})$
<b>F</b>	<b>F</b>	<b>F</b>	<b>F</b>
<b>F</b>	<b>F</b>	<b>T</b>	<b>F</b>
<b>F</b>	<b>T</b>	<b>F</b>	<b>T</b>
<b>F</b>	<b>T</b>	<b>T</b>	<b>T</b>
<b>T</b>	<b>F</b>	<b>F</b>	<b>F</b>
<b>T</b>	<b>F</b>	<b>T</b>	<b>F</b>
<b>T</b>	<b>T</b>	<b>F</b>	<b>T</b>
<b>T</b>	<b>T</b>	<b>T</b>	<b>T</b>

# Truth Tables

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

$x_3$	$x_2$	$x_1$	$x_2 \wedge (x_1 \vee \overline{x_3})$
<b>F</b>	<b>F</b>	<b>F</b>	<b>F</b>
<b>F</b>	<b>F</b>	<b>T</b>	<b>F</b>
<b>F</b>	<b>T</b>	<b>F</b>	<b>T</b>
<b>F</b>	<b>T</b>	<b>T</b>	<b>T</b>
<b>T</b>	<b>F</b>	<b>F</b>	<b>F</b>
<b>T</b>	<b>F</b>	<b>T</b>	<b>F</b>
<b>T</b>	<b>T</b>	<b>F</b>	<b>F</b>
<b>T</b>	<b>T</b>	<b>T</b>	<b>T</b>

# Truth Tables

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

$x_3$	$x_2$	$x_1$	$x_2 \wedge (x_1 \vee \overline{x_3})$
<b>F</b>	<b>F</b>	<b>F</b>	<b>F</b>
<b>F</b>	<b>F</b>	<b>T</b>	<b>F</b>
<b>F</b>	<b>T</b>	<b>F</b>	<b>T</b>
<b>F</b>	<b>T</b>	<b>T</b>	<b>T</b>
<b>T</b>	<b>F</b>	<b>F</b>	<b>F</b>
<b>T</b>	<b>F</b>	<b>T</b>	<b>F</b>
<b>T</b>	<b>T</b>	<b>F</b>	<b>F</b>
<b>T</b>	<b>T</b>	<b>T</b>	<b>T</b>

# Truth Tables

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

$x_3$	$x_2$	$x_1$	$x_2$	$\wedge$	$(x_1$	$\vee$	$\overline{x_3})$
<b>F</b>	<b>F</b>	<b>F</b>	<b>F</b>	<b>F</b>	<b>F</b>	<b>T</b>	<b>T</b>
<b>F</b>	<b>F</b>	<b>T</b>	<b>F</b>	<b>F</b>	<b>T</b>	<b>T</b>	<b>T</b>
<b>F</b>	<b>T</b>	<b>F</b>	<b>T</b>	<b>T</b>	<b>F</b>	<b>T</b>	<b>T</b>
<b>F</b>	<b>T</b>	<b>T</b>	<b>T</b>	<b>T</b>	<b>T</b>	<b>T</b>	<b>T</b>
<b>T</b>	<b>F</b>	<b>F</b>	<b>F</b>	<b>F</b>	<b>F</b>	<b>F</b>	<b>F</b>
<b>T</b>	<b>F</b>	<b>T</b>	<b>F</b>	<b>F</b>	<b>T</b>	<b>T</b>	<b>F</b>
<b>T</b>	<b>T</b>	<b>F</b>	<b>T</b>	<b>F</b>	<b>F</b>	<b>F</b>	<b>F</b>
<b>T</b>	<b>T</b>	<b>T</b>	<b>T</b>	<b>T</b>	<b>T</b>	<b>T</b>	<b>F</b>

# 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  $\alpha$  is satisfiable, form the truth table for  $\alpha$ :
- If there is a row in which **T** appears as the value for  $\alpha$ , then  $\alpha$  is satisfiable
- Otherwise,  $\alpha$  is unsatisfiable



- What is the complexity of the truth table algorithm?

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

# Run-time Complexity

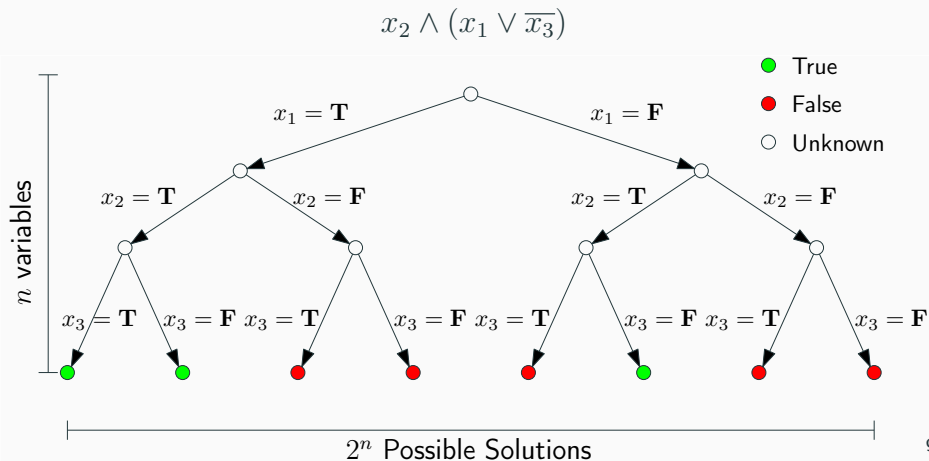
- What is the complexity of the truth table algorithm?
- $2^n$  where  $n$  is the number of Boolean variables
- Can we do better?

# 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

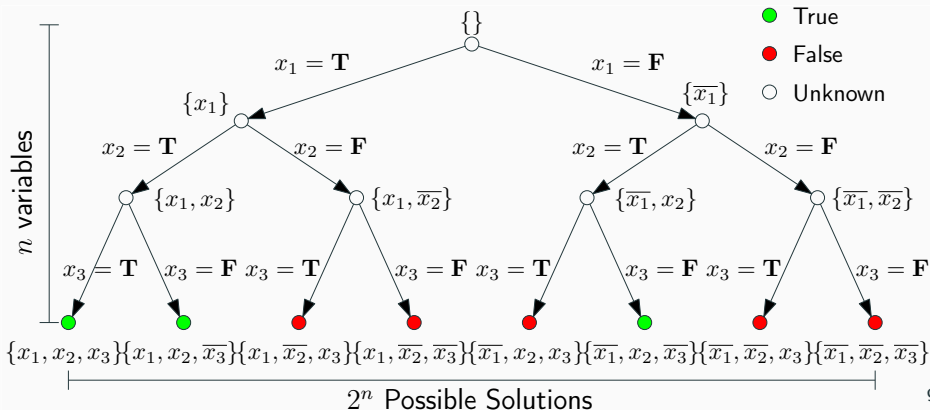
- Binary search tree: at each node Boolean variable is set to a value
- SAT solver performs a backtrack search in the search tree



# Search Tree

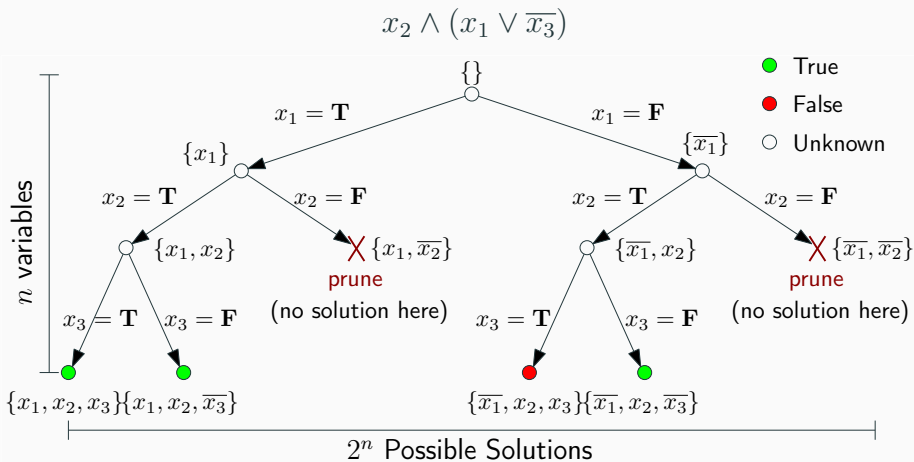
- Partial Truth Assignment: assignment to a subset of the Boolean variables
- Search algorithm gradually fills out a partial assignment until:
  - find a satisfying full assignment (if any)
  - backtrack to another partial assignment

$$x_2 \wedge (x_1 \vee \bar{x}_3)$$



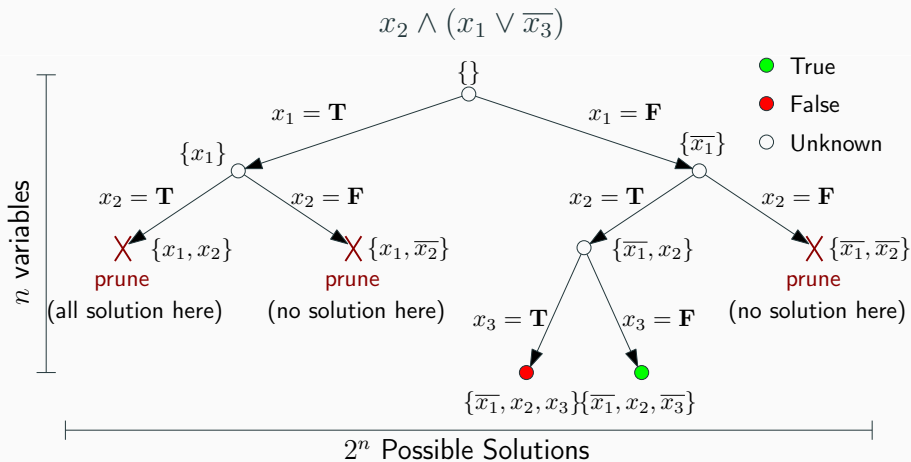
# Search Tree

- We can use heuristics to prune the search tree



# Search Tree

- We can use heuristics to prune the search tree





## Heuristic: Early termination

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

During the search if the partial assignment:

- Makes a literal **T** then:  
simplify the formula by removing all the clauses that have that literal
- Makes a clause **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 **T** for satisfiability
- Let  $x_1$  and  $\overline{x_3}$  be both **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 **T**
- e.g.,  $x_1$  must be **T** in example above
- Also includes clauses where all but one literal is **F**
- e.g.  $(x_1 \vee x_2 \vee x_3)$  where  $x_2$  and  $x_3$  are **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:

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

$$x_1 = \mathbf{True}$$

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

$$x_2 = \mathbf{False}$$

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

$$x_3 = \mathbf{True}$$

$$(x_5)$$

$$x_5 = \mathbf{True}$$

**True**

- 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
  - **Davis - Putnam - Logemann - Loveland** procedure

## DPLL( $\phi$ ):

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

## DPLL( $\phi$ ):

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

## Example:

$$x_1 \vee x_2$$

$$x_1 \vee \bar{x}_2$$

$$\bar{x}_1 \vee x_3 \vee x_4$$

$$\bar{x}_1 \vee \bar{x}_3 \vee x_4$$

$$\bar{x}_1 \vee \bar{x}_4$$

$$\bar{x}_1 \vee x_4 \vee x_5$$



## DPLL( $\phi$ ):

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

## Example:

$$x_1 \vee x_2$$

$$x_1 \vee \bar{x}_2$$

$$\bar{x}_1 \vee x_3 \vee x_4$$

$$\bar{x}_1 \vee \bar{x}_3 \vee x_4$$

$$\bar{x}_1 \vee \bar{x}_4$$

$$\bar{x}_1 \vee x_4 \vee x_5$$

(Pure Literal Rule)

## DPLL( $\phi$ ):

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

## Example:

$$x_1 \vee x_2$$

$$x_1 \vee \bar{x}_2$$

$$\bar{x}_1 \vee x_3 \vee x_4$$

$$\bar{x}_1 \vee \bar{x}_3 \vee x_4$$

$$\bar{x}_1 \vee \bar{x}_4$$

(Pure Literal Rule)

## DPLL( $\phi$ ):

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

## Example:

$$x_1 \vee x_2$$

$$x_1 \vee \bar{x}_2$$

$$\bar{x}_1 \vee x_3 \vee x_4$$

$$\bar{x}_1 \vee \bar{x}_3 \vee x_4$$

$$\bar{x}_1 \vee \bar{x}_4$$

$$x_4$$

(Select  $x_4$ )

DPLL( $\phi$ ):

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

## Example:

$$x_1 \vee x_2$$

$$x_1 \vee \bar{x}_2$$

$$\bar{x}_1$$

(Select  $x_4$ )

(Unit Propagation)

## DPLL( $\phi$ ):

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

## Example:

$$\begin{array}{l} \overline{x_1} \vee x_2 \\ \overline{x_1} \vee \overline{x_2} \end{array}$$

(Select  $x_4$ )  
(Unit Propagation)  
(Unit Propagation)

## DPLL( $\phi$ ):

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

## Example:

$$\begin{array}{l} \overline{x_1} \vee x_2 \\ \overline{x_1} \vee \overline{x_2} \end{array} \quad \text{Conflict!}$$

Backtrack → (Select  $x_4$ )  
 (Unit Propagation)  
 (Unit Propagation)

## DPLL( $\phi$ ):

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

## Example:

$$x_1 \vee x_2$$

$$x_1 \vee \bar{x}_2$$

$$\bar{x}_1 \vee x_3 \vee x_4$$

$$\bar{x}_1 \vee \bar{x}_3 \vee x_4$$

$$\bar{x}_1 \vee \bar{x}_4$$

$$\bar{x}_4$$

(Select  $\bar{x}_4$ )

DPLL( $\phi$ ):

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

## Example:

$$x_1 \vee x_2$$

$$x_1 \vee \bar{x}_2$$

$$\bar{x}_1 \vee x_3$$

$$\bar{x}_1 \vee \bar{x}_3$$

(Select  $\bar{x}_4$ )

(Unit Propagation)



## DPLL( $\phi$ ):

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

## Example:

$$x_1 \vee x_2$$

$$x_1 \vee \bar{x}_2$$

$$\bar{x}_1 \vee x_3$$

$$\bar{x}_1 \vee \bar{x}_3$$

$$x_1$$

(Select  $\bar{x}_4$ )

(Unit Propagation)

(Select  $x_1$ )

## DPLL( $\phi$ ):

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

## Example:

$$\bar{x}_1 \vee x_3$$

$$\bar{x}_1 \vee \bar{x}_3$$

(Select  $\bar{x}_4$ )

(Unit Propagation)

(Select  $x_1$ )

(Unit Propagation)<sub>15</sub>

## DPLL( $\phi$ ):

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

## Example:

$$\begin{array}{l} \bar{x}_1 \vee x_3 \\ \bar{x}_1 \vee \bar{x}_3 \end{array} \quad \text{Conflict!}$$

(Select  $\bar{x}_4$ )

(Unit Propagation)

Backtrack → (Select  $x_1$ )

(Unit Propagation)<sub>15</sub>

DPLL( $\phi$ ):

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

$$x_1 \vee x_2$$

$$x_1 \vee \bar{x}_2$$

$$\bar{x}_1 \vee x_3$$

$$\bar{x}_1 \vee \bar{x}_3$$

$$\bar{x}_1$$

(Select  $\bar{x}_4$ )

(Unit Propagation)

(Select  $\bar{x}_1$ )

## DPLL( $\phi$ ):

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

$$\begin{array}{l} \overline{x_1} \vee x_2 \\ \overline{x_1} \vee \overline{x_2} \end{array}$$

(Select  $\overline{x_4}$ )  
(Unit Propagation)  
(Select  $\overline{x_1}$ )  
(Unit Propagation)<sub>15</sub>

## DPLL( $\phi$ ):

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

$$\begin{array}{l} \bar{x}_1 \vee x_2 \\ \bar{x}_1 \vee \bar{x}_2 \end{array} \quad \text{Conflict!}$$

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

(Select  $\bar{x}_4$ )  
(Unit Propagation)  
(Select  $\bar{x}_1$ )  
(Unit Propagation)<sub>15</sub>

Several performance techniques:

- Conflict-driven clause learning
- Random search restarts
- Boolean constraint propagation using lazy data structures
- Conflict-based adaptive branching
- ...

Competition:

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