# CS3383 Unit 5.1: SAT

David Bremner

December 4, 2020

# Outline

## Combinatorial Search

- SAT
- Tractable kinds of SAT

# The SAT Problem

## Conjunctive Normal Form (CNF)

Variables $\{ x_1 ... x_n \}$

Literals $L = \{ x_i, \bar{x}_i \mid \text{variable } x_i \}$

Clauses $\{ z_1, ..., z_k \} \subset L$

# The SAT Problem

## Conjunctive Normal Form (CNF)

**Variables** $\{ x_1 \ldots x_n \}$

**Literals** $L = \{ x_i, \bar{x}_i \mid \text{variable } x_i \}$

**Clauses** $\{ z_1, \ldots, z_k \} \subset L$

## Propositional Satisfiability (SAT)

**Instance** Set of clauses $S$

**Question** $\exists$ setting of variables to $0$, $1$ such that each clause has at least one true literal?

# SAT Example

$$\{\{1,2,3\},\{-1,-2,-3\}\} = \{\{x_1,x_2,x_3\},\{\bar{x}_1,\bar{x}_2,\bar{x}_3\}\}$$

(A) $=$

## Truth Table

| $x_1$ | $x_2$ | $x_3$ | $A$ |
|---|---|---|---|
| 0 | 0 | 0 | 0 |
| 0 | 0 | 1 | 1 |
| 0 | 1 | 0 | |
| | $\vdots$ | | |

# SAT Example

$$\{\{1,2,3\},\{-1,-2,-3\}\} = \{\{x_1,x_2,x_3\},\{\bar{x}_1,\bar{x}_2,\bar{x}_3\}\}$$

(A) $$= (x_1 \vee x_2 \vee x_3) \wedge (\bar{x}_1 \vee \bar{x}_2 \vee \bar{x}_3)$$

## Truth Table

| $x_1$ | $x_2$ | $x_3$ | $A$ |
|---|---|---|---|
| 0 | 0 | 0 | 0 |
| 0 | 0 | 1 | 1 |
| 0 | 1 | 0 | |
| | $\vdots$ | | |

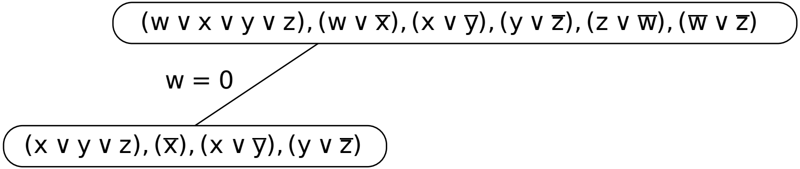
# Backtracking for SAT

**representation** (reduced) clauses

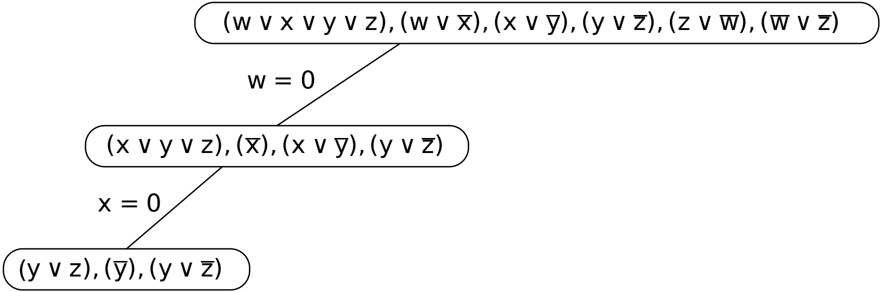
**test** if empty clause, return `False`. If no clauses, return `True`. Otherwise return `None` (UNKNOWN)

**expand** $P_0 = \text{reduce}(P, j, 0)$, $P_1 = \text{reduce}(P, j, 1)$ for some $j$.

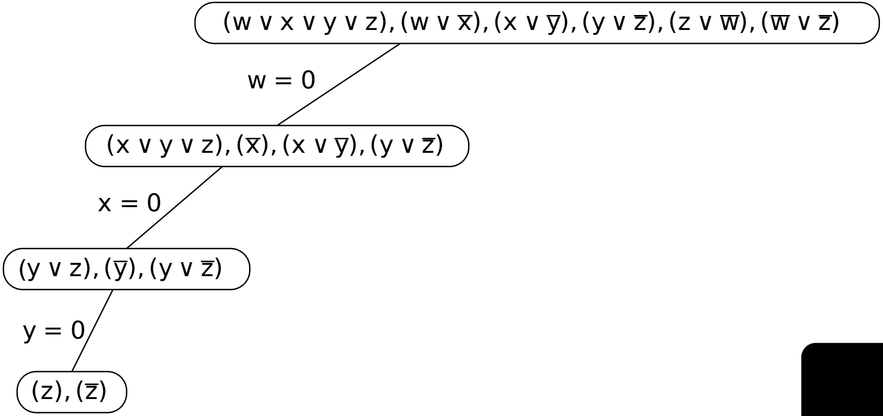
# Backtracking SAT Example II



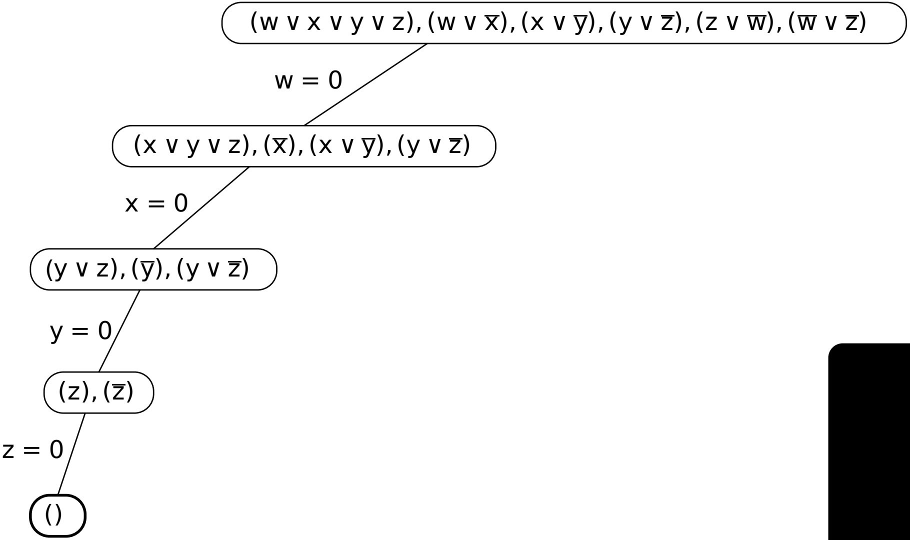
# Backtracking SAT Example II



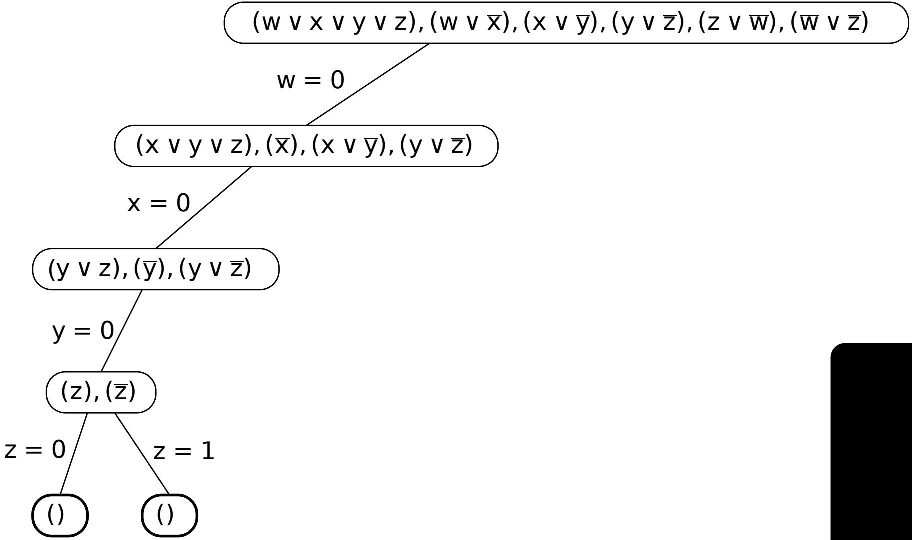
# Backtracking SAT Example II



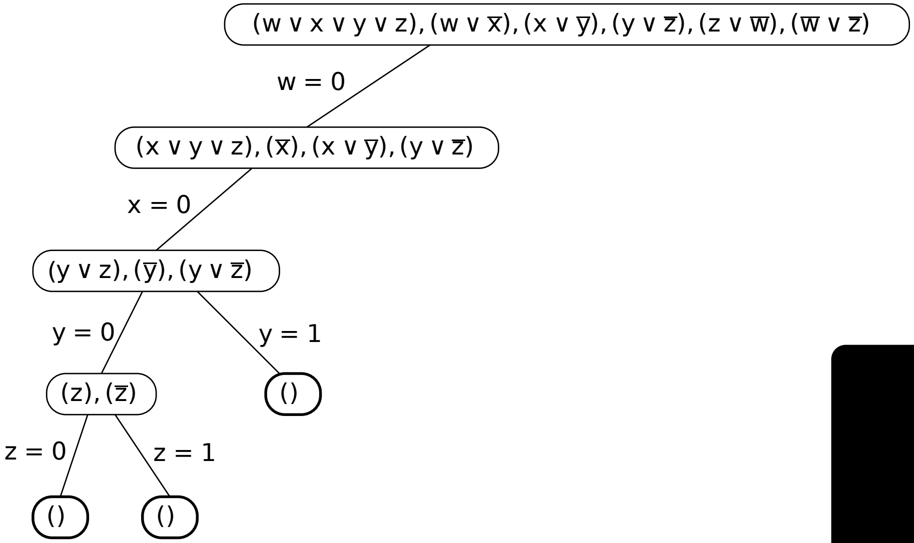
# Backtracking SAT Example II



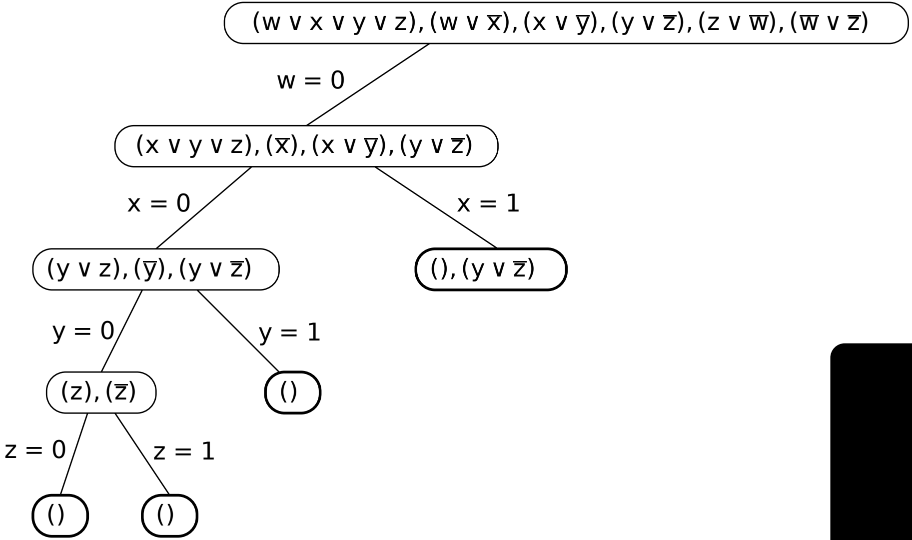
# Backtracking SAT Example II



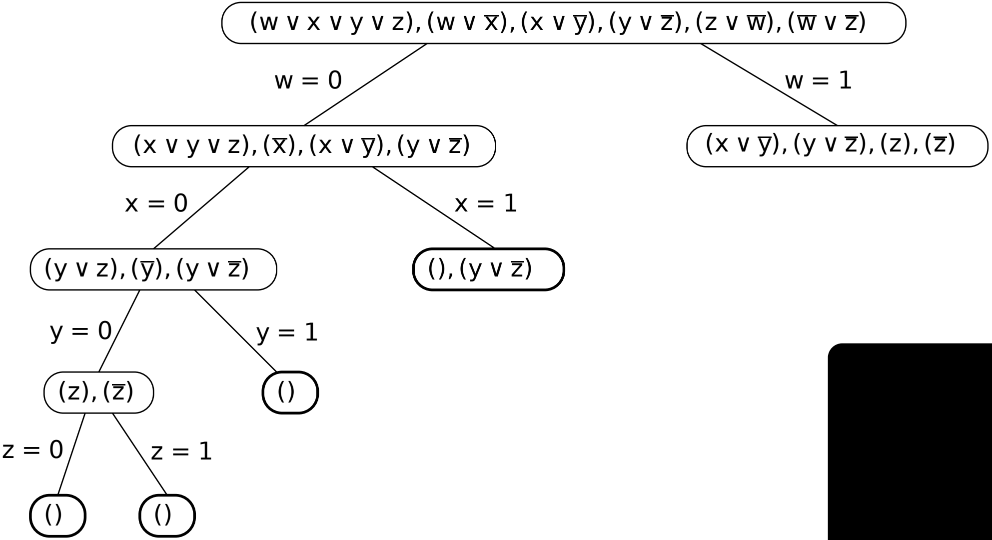
# Backtracking SAT Example II



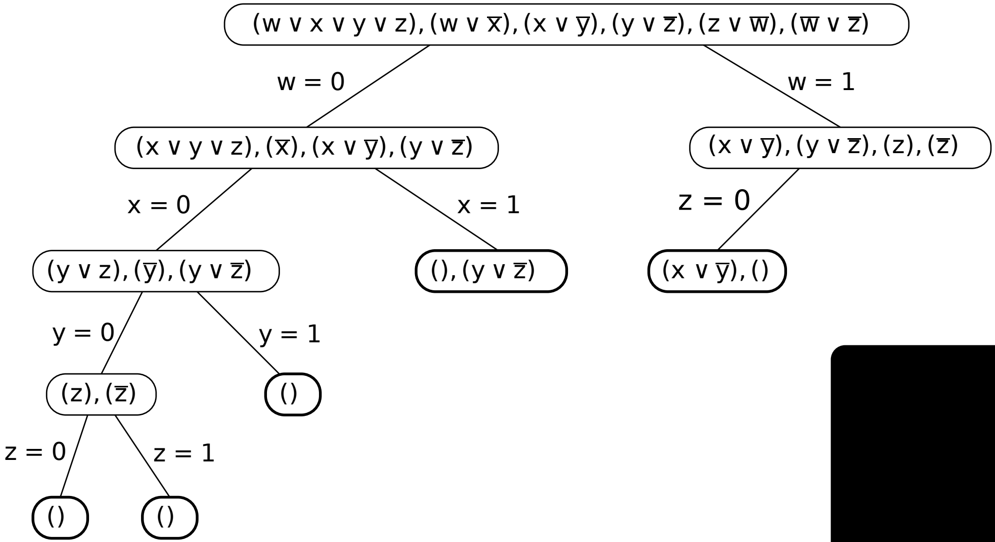
# Backtracking SAT Example II



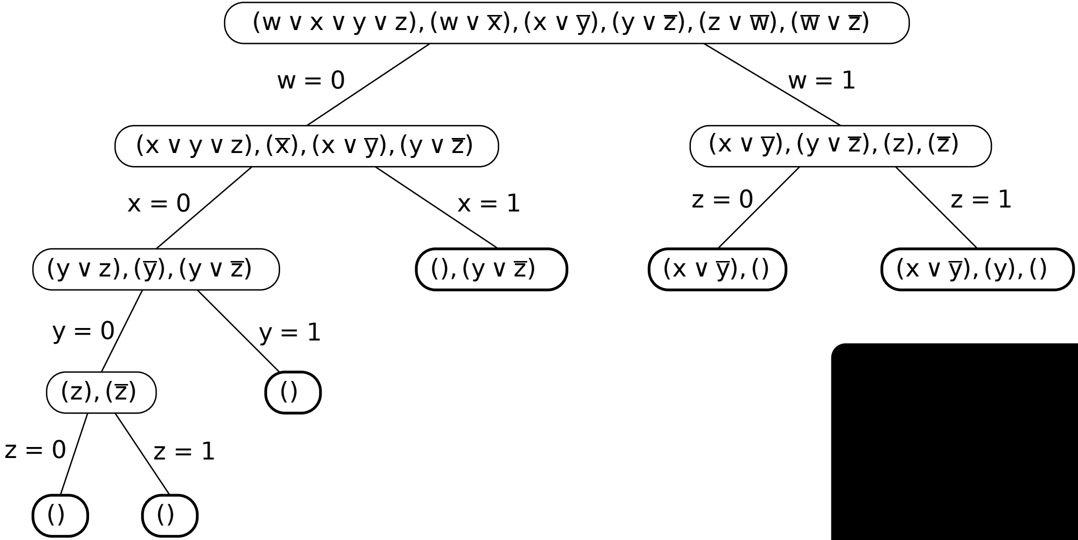
# Backtracking SAT Example II



# Backtracking SAT Example II

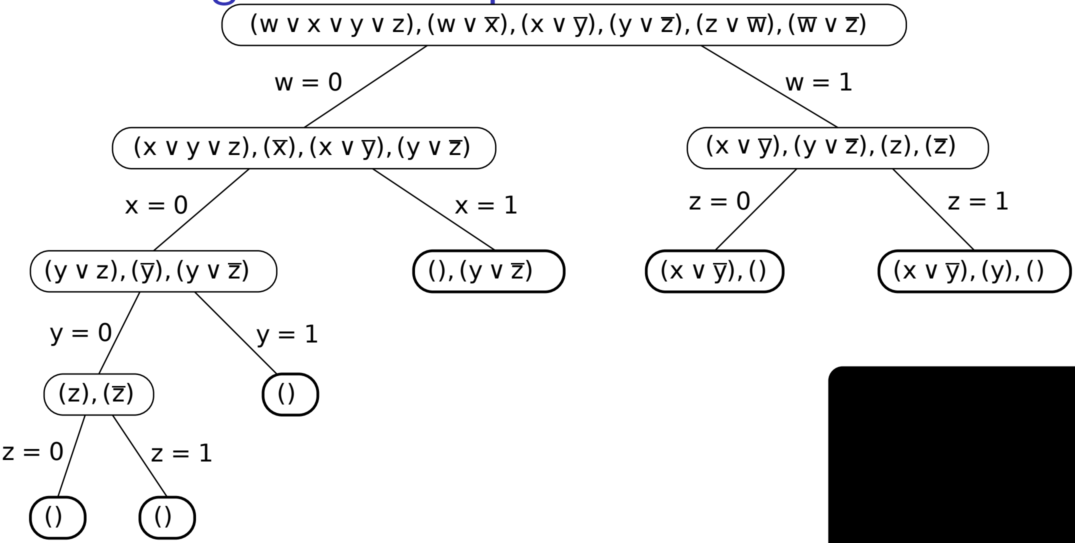


# Backtracking SAT Example II




[redacted]

# Backtracking SAT Example II




We tried 11 possibilities. Maximum?

# 2SAT

- In 2SAT problem every clause has at most 2 elements

# 2SAT

- In 2SAT problem every clause has at most 2 elements
- 2SAT is solvable in polynomial time, but not quite trivially.

# 2SAT

- In 2SAT problem every clause has at most 2 elements
- 2SAT is solvable in polynomial time, but not quite trivially.
- Greedy fails on

$$(x_1 \vee x_2) \wedge (x_1 \vee x_3) \wedge (\bar{x}_1 \vee x_4) \wedge (\bar{x}_4 \vee x_5) \wedge (\bar{x}_4 \vee x_6)$$

# 2SAT

- In 2SAT problem every clause has at most 2 elements
- 2SAT is solvable in polynomial time, but not quite trivially.
- Greedy fails on

$$(x_1 \vee x_2) \wedge (x_1 \vee x_3) \wedge (\bar{x}_1 \vee x_4) \wedge (\bar{x}_4 \vee x_5) \wedge (\bar{x}_4 \vee x_6)$$

- to maximize number of clauses satisfied, choose $x_1 \leftarrow 1$, $x_4 \leftarrow 0$

# 2SAT

- In 2SAT problem every clause has at most 2 elements
- 2SAT is solvable in polynomial time, but not quite trivially.
- Greedy fails on

$$(x_1 \vee x_2) \wedge (x_1 \vee x_3) \wedge (\bar{x}_1 \vee x_4) \wedge (\bar{x}_4 \vee x_5) \wedge (\bar{x}_4 \vee x_6)$$

- to maximize number of clauses satisfied, choose $x_1 \leftarrow 1$, $x_4 \leftarrow 0$
- solvable with *unit propagation*

# Horn SAT

## Horn formulas

implication $(z \wedge w \wedge q) \Rightarrow u$. LHS is all positive, RHS one positive literal

negative clauses $(\bar{x} \vee \bar{w} \vee \bar{y})$. All literals negated.

# Horn formulas as CNF

- negative clauses are already CNF

# Horn formulas as CNF

- negative clauses are already CNF
- implications use the following transformations

$$(\bigwedge_{i=1}^{k} x_i) \Rightarrow y$$

$$\neg(\bigwedge_{i=1}^{k} x_i) \vee y$$

$$(\bigvee_{i=1}^{k} \bar{x}_i) \vee y$$

# Horn formulas as CNF

- negative clauses are already CNF
- implications use the following transformations

$$(\bigwedge_{i=1}^{k} x_i) \Rightarrow y$$

$$\neg(\bigwedge_{i=1}^{k} x_i) \vee y$$

$$(\bigvee_{i=1}^{k} \bar{x}_i) \vee y$$

- special CNF with at most one positive literal.

# Unit propagation

```
def UnitProp(S):
  Q = [ c for c in S if len(c)==1 ]; V = []
  while len(Q)>0:
    z = Q.pop()[0]; T = []
    V.append(z)
    for C in S:
      C = [j for j in C if j!=-z]
      if len(C)==0: return (False,V)
      if len(C)==1: Q.append(C)
      if not z in C: T.append(C)
    if len(T)==0: return (True,V)
    S = T
  return S
```

# Solving Horn SAT with Unit Propagation

1. Apply unit propagation
2. If no contradiction is detected, set the remaining variables to false.

**Claim 1**

If the procedure detects a contradiction, the instance is unsatisfiable

# Solving Horn SAT with Unit Propagation

1. Apply unit propagation
2. If no contradiction is detected, set the remaining variables to false.

**Claim 1**

If the procedure detects a contradiction, the instance is unsatisfiable

**Claim 2**

If no contradiction is detected, the resulting assignment is valid

# Solving Horn SAT with Unit Propagation

1. Apply unit propagation
2. If no contradiction is detected, set the remaining variables to false.

**Claim 1**

If the procedure detects a contradiction, the instance is unsatisfiable

**Claim 2**

If no contradiction is detected, the resulting assignment is valid

**Proof**

any remaining clause has at least one negative literal