



# SAT and SMT Solving

**Sarah Winkler**

KRDB

Department of Computer Science  
Free University of Bozen-Bolzano

lecture 3  
WS 2022

# Outline

- Summary of Last Week
- Maximum Satisfiability
- Algorithms for Minimum Unsatisfiability
- Application: Automotive Configuration
- NP-Completeness

## Definition (Implication Graph)

for derivation  $\parallel F' \implies_B^* M \parallel F$  **implication graph** is constructed as follows:

- ▶ add node labelled  $l$  for every decision literal  $l$  in  $M$
- ▶ repeat until there is no change:
  - if  $\exists$  clause  $l_1 \vee \dots \vee l_m \vee l'$  in  $F$  such that there are already nodes  $l_1^c, \dots, l_m^c$ 
    - ▶ add node  $l'$  if not yet present
    - ▶ add edges  $l_i^c \rightarrow l'$  for all  $1 \leq i \leq m$  if not yet present
  - ▶ if  $\exists$  clause  $l'_1 \vee \dots \vee l'_k$  in  $F$  such that there are nodes  $l_1'^c, \dots, l_k'^c$ 
    - ▶ add conflict node labeled  $C$
    - ▶ add edges  $l_i'^c \rightarrow C$

## Definition (Implication Graph)

for derivation  $\parallel F' \implies_B^* M \parallel F$  **implication graph** is constructed as follows:

- ▶ add node labelled  $l$  for every decision literal  $l$  in  $M$
- ▶ repeat until there is no change:
  - if  $\exists$  clause  $l_1 \vee \dots \vee l_m \vee l'$  in  $F$  such that there are already nodes  $l_1^c, \dots, l_m^c$ 
    - ▶ add node  $l'$  if not yet present
    - ▶ add edges  $l_i^c \rightarrow l'$  for all  $1 \leq i \leq m$  if not yet present
  - if  $\exists$  clause  $l'_1 \vee \dots \vee l'_k$  in  $F$  such that there are nodes  $l_1^c, \dots, l_k^c$ 
    - ▶ add conflict node labeled  $C$
    - ▶ add edges  $l_i^c \rightarrow C$

## Definitions

- ▶ **cut** separates decision literals from conflict node
- ▶ literal  $l$  in implication graph is **unique implication point (UIP)** if all paths from last decision literal to conflict node go through  $l$

## Definition (Implication Graph)

for derivation  $\parallel F' \implies_B^* M \parallel F$  **implication graph** is constructed as follows:

- ▶ add node labelled  $l$  for every decision literal  $l$  in  $M$
- ▶ repeat until there is no change:
  - if  $\exists$  clause  $l_1 \vee \dots \vee l_m \vee l'$  in  $F$  such that there are already nodes  $l_1^c, \dots, l_m^c$ 
    - ▶ add node  $l'$  if not yet present
    - ▶ add edges  $l_i^c \rightarrow l'$  for all  $1 \leq i \leq m$  if not yet present
  - if  $\exists$  clause  $l'_1 \vee \dots \vee l'_k$  in  $F$  such that there are nodes  $l_1^c, \dots, l_k^c$ 
    - ▶ add conflict node labeled  $C$
    - ▶ add edges  $l_i'^c \rightarrow C$

## Definitions

- ▶ **cut** separates decision literals from conflict node
- ▶ literal  $l$  in implication graph is **unique implication point (UIP)** if all paths from last decision literal to conflict node go through  $l$

## Lemma

- ▶ if edges intersected by cut are  $l_1 \rightarrow l'_1, \dots, l_k \rightarrow l'_k$  then  $F' \models l_1^c \vee \dots \vee l_k^c$
- ▶ this clause is backjump clause if some  $l_i$  is UIP

## Backjump clauses by resolution

- ▶ set  $C_0$  to conflict clause
- ▶ let  $l$  be last assigned literal such that  $l^c$  is in  $C_0$
- ▶ while  $l$  is no decision literal:
  - ▶  $C_{i+1}$  is resolvent of  $C_i$  and clause  $D$  that led to assignment of  $l$
  - ▶ let  $l$  be last assigned literal such that  $l^c$  is in  $C_{i+1}$

## Backjump clauses by resolution

- ▶ set  $C_0$  to conflict clause
- ▶ let  $l$  be last assigned literal such that  $l^c$  is in  $C_0$
- ▶ while  $l$  is no decision literal:
  - ▶  $C_{i+1}$  is resolvent of  $C_i$  and clause  $D$  that led to assignment of  $l$
  - ▶ let  $l$  be last assigned literal such that  $l^c$  is in  $C_{i+1}$

## Lemma

*every clause  $C_i$  corresponds to cut in implication graph:*

*there is cut intersecting edges  $l_{i1} \rightarrow l'_{i1}, \dots, l_{ik} \rightarrow l'_{ik}$  such that  $C_i = l_{i1}^c \vee \dots \vee l_{ik}^c$*

## Definition (DPLL with Learning and Restarts)

DPLL with learning and restarts  $\mathcal{R}$  extends system  $\mathcal{B}$  by following three rules:

- ▶ **learn** 
$$M \parallel F \implies M \parallel F, C$$
  
if  $F \models C$  and all atoms of  $C$  occur in  $M$  or  $F$
- ▶ **forget** 
$$M \parallel F, C \implies M \parallel F$$
  
if  $F \models C$
- ▶ **restart** 
$$M \parallel F \implies \parallel F$$



## Definition (DPLL with Learning and Restarts)

DPLL with learning and restarts  $\mathcal{R}$  extends system  $\mathcal{B}$  by following three rules:

- ▶ **learn** 
$$M \parallel F \implies M \parallel F, C$$
if  $F \models C$  and all atoms of  $C$  occur in  $M$  or  $F$
- ▶ **forget** 
$$M \parallel F, C \implies M \parallel F$$
if  $F \models C$
- ▶ **restart** 
$$M \parallel F \implies \parallel F$$

## Theorem (Termination)

any derivation  $\parallel F \implies_{\mathcal{R}} S_1 \implies_{\mathcal{R}} S_2 \implies_{\mathcal{R}} \dots$  is finite if

- ▶ it contains no infinite subderivation of learn and forget steps, and
- ▶ restart is applied with increasing periodicity

## Definition (DPLL with Learning and Restarts)

DPLL with learning and restarts  $\mathcal{R}$  extends system  $\mathcal{B}$  by following three rules:

- ▶ **learn** 
$$M \parallel F \implies M \parallel F, C$$
if  $F \models C$  and all atoms of  $C$  occur in  $M$  or  $F$
- ▶ **forget** 
$$M \parallel F, C \implies M \parallel F$$
if  $F \models C$
- ▶ **restart** 
$$M \parallel F \implies \parallel F$$

## Theorem (Termination)

any derivation  $\parallel F \implies_{\mathcal{R}} S_1 \implies_{\mathcal{R}} S_2 \implies_{\mathcal{R}} \dots$  is finite if

- ▶ it contains no infinite subderivation of learn and forget steps, and
- ▶ restart is applied with increasing periodicity

## Theorem (Correctness)

for  $\parallel F \implies_{\mathcal{R}} S_1 \implies_{\mathcal{R}} S_2 \implies_{\mathcal{R}} \dots \implies_{\mathcal{R}} S_n$  with final state  $S_n$ :

- ▶ if  $S_n = \text{FailState}$  then  $F$  is unsatisfiable
- ▶ if  $S_n = M \parallel F'$  then  $F$  is satisfiable and  $M \models F'$

# Two-Watched Literal Scheme

## Idea

- ▶ maintain two pointers  $p_1$  and  $p_2$  for each clause  $C$
- ▶ each pointer points to a literal in the clause that is: unassigned or true if possible, otherwise false
- ▶ ensure invariant that  $p_1(C) \neq p_2(C)$

## Key properties

- ▶ clause  $C$  enables unit propagation if  $p_1(C)$  is false and  $p_2(C)$  is unassigned or vice versa  $\mathcal{O}(n)$
- ▶ clause  $C$  is conflict clause if  $p_1(C)$  and  $p_2(C)$  are false literals

## Setting pointers

- ▶ initialization: set  $p_1$  and  $p_2$  to different (unassigned) literals in clause
- ▶ decide or unit propagate:  
when assigning literal  $l$  true, redirect all pointers to  $l^c$  to other literal in their clause if possible
- ▶ backjump: no need to change pointers!

# Outline

- Summary of Last Week
- Maximum Satisfiability
- Algorithms for Minimum Unsatisfiability
- Application: Automotive Configuration
- NP-Completeness

## maxSAT Problem

input: propositional formula  $\varphi$  in CNF

output: valuation  $\alpha$  such that  $\alpha$  satisfies maximal number of clauses in  $\varphi$

## maxSAT Problem

input: propositional formula  $\varphi$  in CNF

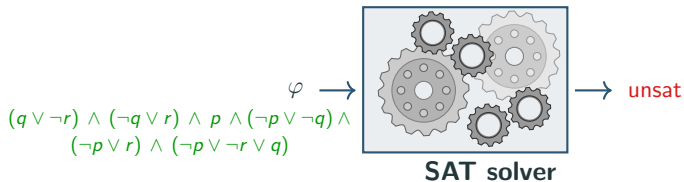
output: valuation  $\alpha$  such that  $\alpha$  satisfies maximal number of clauses in  $\varphi$

# maxSAT

## maxSAT Problem

input: propositional formula  $\varphi$  in CNF

output: valuation  $\alpha$  such that  $\alpha$  satisfies maximal number of clauses in  $\varphi$

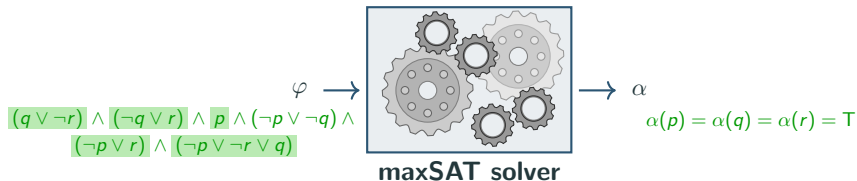


# maxSAT

## maxSAT Problem

input: propositional formula  $\varphi$  in CNF

output: valuation  $\alpha$  such that  $\alpha$  satisfies maximal number of clauses in  $\varphi$



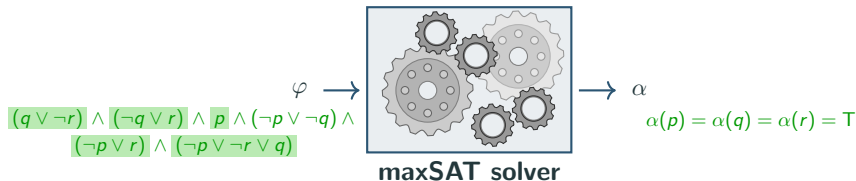


# maxSAT

## maxSAT Problem

input: propositional formula  $\varphi$  in CNF

output: valuation  $\alpha$  such that  $\alpha$  satisfies maximal number of clauses in  $\varphi$



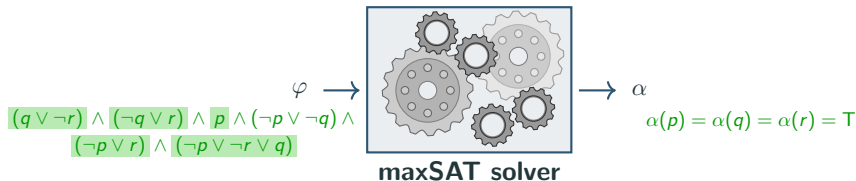
## Terminology

- **optimization problem**  $P$  asks to find “best” solution among all solutions

## maxSAT Problem

input: propositional formula  $\varphi$  in CNF

output: valuation  $\alpha$  such that  $\alpha$  satisfies maximal number of clauses in  $\varphi$



## Terminology

- ▶ optimization problem  $P$  asks to find “best” solution among all solutions
- ▶ **maxSAT encoding** transforms optimization problem  $P$  into formula  $\varphi$  such that optimal solution to  $P$  corresponds to maxSAT solution to  $\varphi$

## Remark

many real world are have optimization problems

## Remark

many real world are have optimization problems

## Examples

- ▶ find **shortest path** to goal state
  - ▶ planning
  - ▶ model checking

## Remark

many real world are have optimization problems

## Examples

- ▶ find shortest path to goal state
  - ▶ planning
  - ▶ model checking
- ▶ find smallest explanation
  - ▶ debugging
  - ▶ configuration

## Remark

many real world are have optimization problems

## Examples

- ▶ find shortest path to goal state
  - ▶ planning
  - ▶ model checking
- ▶ find smallest explanation
  - ▶ debugging
  - ▶ configuration
- ▶ find least resource-consuming schedule
  - ▶ scheduling
  - ▶ logistics

## Remark

many real world are have optimization problems

## Examples

- ▶ find shortest path to goal state
  - ▶ planning
  - ▶ model checking
- ▶ find smallest explanation
  - ▶ debugging
  - ▶ configuration
- ▶ find least resource-consuming schedule
  - ▶ scheduling
  - ▶ logistics
- ▶ find most probable explanation
  - ▶ probabilistic inference
- ▶ ...

## Remark

many real world are have optimization problems

## Examples

- ▶ find shortest path to goal state
  - ▶ planning
  - ▶ model checking
- ▶ find smallest explanation
  - ▶ debugging
  - ▶ configuration
- ▶ find least resource-consuming schedule
  - ▶ scheduling
  - ▶ logistics
- ▶ find most probable explanation
  - ▶ probabilistic inference
- ▶ ...

## Notation

for valuation  $v$  let  $\bar{v}(\varphi) = \begin{cases} 1 & \text{if } v(\varphi) = \text{T} \\ 0 & \text{if } v(\varphi) = \text{F} \end{cases}$



# Maximal Satisfiability

Consider CNF formula  $\varphi$  as set of clauses  $C \in \varphi$

# Maximal Satisfiability

Consider CNF formula  $\varphi$  as set of clauses  $C \in \varphi$

## Maximal Satisfiability (maxSAT)

instance: CNF formula  $\varphi$

question: what is maximal  $\sum_{C \in \varphi} \bar{v}(C)$  for valuation  $v$ ?

# Maximal Satisfiability

Consider CNF formula  $\varphi$  as set of clauses  $C \in \varphi$

## Maximal Satisfiability (maxSAT)

instance: CNF formula  $\varphi$

question: what is maximal  $\sum_{C \in \varphi} \bar{v}(C)$  for valuation  $v$ ?

### Example

$$\varphi = \{ \begin{array}{llllll} 6 \vee 2, & \bar{6} \vee 2, & \bar{2} \vee 1, & \bar{1}, & \bar{6} \vee 8, & 6 \vee \bar{8}, \\ 2 \vee 4, & \bar{4} \vee 5, & 7 \vee 5, & \bar{7} \vee 5, & \bar{3}, & \bar{5} \vee 3 \end{array} \}$$

►  $\text{maxSAT}(\varphi) = 10$ , e.g. for valuation  $\bar{1} \ 2 \ \bar{3} \ 4 \ 5 \ 6 \ \bar{7} \ 8$

# Maximal Satisfiability

Consider CNF formula  $\varphi$  as set of clauses  $C \in \varphi$

## Maximal Satisfiability (maxSAT)

instance: CNF formula  $\varphi$

question: what is maximal  $\sum_{C \in \varphi} \bar{v}(C)$  for valuation  $v$ ?

### Example

$$\varphi = \{ \boxed{6 \vee 2}, \quad \boxed{\bar{6} \vee 2}, \quad \bar{2} \vee 1, \quad \boxed{\bar{1}}, \quad \boxed{\bar{6} \vee 8}, \quad \boxed{6 \vee \bar{8}}, \\ \boxed{2 \vee 4}, \quad \boxed{\bar{4} \vee 5}, \quad \boxed{7 \vee 5}, \quad \boxed{\bar{7} \vee 5}, \quad \boxed{3}, \quad \boxed{\bar{5} \vee 3} \}$$

►  $\text{maxSAT}(\varphi) = 10$ , e.g. for valuation  $\bar{1} \ 2 \ \bar{3} \ 4 \ 5 \ 6 \ \bar{7} \ 8$

# Maximal Satisfiability

Consider CNF formula  $\varphi$  as set of clauses  $C \in \varphi$

## Maximal Satisfiability (maxSAT)

instance: CNF formula  $\varphi$

question: what is maximal  $\sum_{C \in \varphi} \bar{v}(C)$  for valuation  $v$ ?

## Partial Maximal Satisfiability (pmaxSAT)

instance: CNF formulas  $\chi$  and  $\varphi$

question: what is maximal  $\sum_{C \in \varphi} \bar{v}(C)$  for valuation  $v$  with  $v(\chi) = \text{T}$ ?

## Example

$$\varphi = \{ \begin{array}{llllll} 6 \vee 2, & \bar{6} \vee 2, & \bar{2} \vee 1, & \bar{1}, & \bar{6} \vee 8, & 6 \vee \bar{8}, \\ 2 \vee 4, & \bar{4} \vee 5, & 7 \vee 5, & \bar{7} \vee 5, & \bar{3}, & \bar{5} \vee 3 \end{array} \}$$

►  $\text{maxSAT}(\varphi) = 10$ , e.g. for valuation  $\bar{1} \ 2 \ \bar{3} \ 4 \ 5 \ 6 \ \bar{7} \ 8$

# Maximal Satisfiability

Consider CNF formula  $\varphi$  as set of clauses  $C \in \varphi$

## Maximal Satisfiability (maxSAT)

instance: CNF formula  $\varphi$

question: what is maximal  $\sum_{C \in \varphi} \bar{v}(C)$  for valuation  $v$ ?

## Partial Maximal Satisfiability (pmaxSAT)

instance: CNF formulas  $\chi$  and  $\varphi$

question: what is maximal  $\sum_{C \in \varphi} \bar{v}(C)$  for valuation  $v$  with  $v(\chi) = T$ ?

## Example

$$\varphi = \{ \begin{array}{llllll} 6 \vee 2, & \bar{6} \vee 2, & \bar{2} \vee 1, & \bar{1}, & \bar{6} \vee 8, & 6 \vee \bar{8}, \\ 2 \vee 4, & \bar{4} \vee 5, & 7 \vee 5, & \bar{7} \vee 5, & \bar{3}, & \bar{5} \vee 3 \end{array} \}$$

$$\chi = \{ \bar{1} \vee 2, \quad \bar{2} \vee \bar{3}, \quad \bar{5} \vee 1, \quad 3 \}$$

►  $\text{maxSAT}(\varphi) = 10$ , e.g. for valuation  $\bar{1} \ 2 \ \bar{3} \ 4 \ 5 \ 6 \ \bar{7} \ 8$

►  $\text{pmaxSAT}(\chi, \varphi) = 8$ , e.g. for valuation  $\bar{1} \ \bar{2} \ 3 \ 4 \ \bar{5} \ 6 \ 7 \ 8$

# Maximal Satisfiability

Consider CNF formula  $\varphi$  as set of clauses  $C \in \varphi$

## Maximal Satisfiability (maxSAT)

instance: CNF formula  $\varphi$

question: what is maximal  $\sum_{C \in \varphi} \bar{v}(C)$  for valuation  $v$ ?

## Partial Maximal Satisfiability (pmaxSAT)

instance: CNF formulas  $\chi$  and  $\varphi$

question: what is maximal  $\sum_{C \in \varphi} \bar{v}(C)$  for valuation  $v$  with  $v(\chi) = \text{T}$ ?

## Example

$$\begin{aligned}\varphi = \{ & \boxed{6 \vee 2}, & \boxed{\bar{6} \vee 2}, & \boxed{\bar{2} \vee 1}, & \boxed{\bar{1}}, & \boxed{\bar{6} \vee 8}, & \boxed{6 \vee \bar{8}}, \\ & \boxed{2 \vee 4}, & \boxed{\bar{4} \vee 5}, & \boxed{7 \vee 5}, & \boxed{\bar{7} \vee 5}, & \boxed{\bar{3}}, & \boxed{5 \vee 3} \} \\ \chi = \{ & \boxed{\bar{1} \vee 2}, & \boxed{\bar{2} \vee \bar{3}}, & \boxed{5 \vee 1}, & \boxed{3} \}\end{aligned}$$

►  $\text{maxSAT}(\varphi) = 10$ , e.g. for valuation  $\bar{1} \bar{2} \bar{3} 4 5 6 \bar{7} 8$

►  $\text{pmaxSAT}(\chi, \varphi) = 8$ , e.g. for valuation  $\bar{1} \bar{2} \bar{3} 4 \bar{5} 6 \bar{7} 8$

# Maximal Satisfiability

Consider CNF formula  $\varphi$  as set of clauses  $C \in \varphi$

## Maximal Satisfiability (maxSAT)

instance: CNF formula  $\varphi$

question: what is maximal  $\sum_{C \in \varphi} \bar{v}(C)$  for valuation  $v$ ?

## Partial Maximal Satisfiability (pmaxSAT)

instance: CNF formulas  $\chi$  and  $\varphi$

question: what is maximal  $\sum_{C \in \varphi} \bar{v}(C)$  for valuation  $v$  with  $v(\chi) = \text{T}$ ?

## Terminology

- ▶  $\varphi$  are **soft** constraints
- ▶  $\chi$  are **hard** constraints



# Maximal Satisfiability

Consider CNF formula  $\varphi$  as set of clauses  $C \in \varphi$

## Maximal Satisfiability (maxSAT)

instance: CNF formula  $\varphi$

question: what is maximal  $\sum_{C \in \varphi} \bar{v}(C)$  for valuation  $v$ ?

## Partial Maximal Satisfiability (pmaxSAT)

instance: CNF formulas  $\chi$  and  $\varphi$

question: what is maximal  $\sum_{C \in \varphi} \bar{v}(C)$  for valuation  $v$  with  $v(\chi) = \text{T}$ ?

## Terminology

- ▶  $\varphi$  are soft constraints
- ▶  $\chi$  are hard constraints

## Notation

write **maxSAT**( $\varphi$ ) and **pmaxSAT**( $\chi, \varphi$ ) for solutions to these problems

## Weighted Maximal Satisfiability ( $\text{maxSAT}_w$ )

instance: CNF formula  $\varphi$  with weight  $w_C \in \mathbb{Z}$  for all  $C \in \varphi$

## Weighted Maximal Satisfiability ( $\text{maxSAT}_w$ )

instance: CNF formula  $\varphi$  with weight  $w_C \in \mathbb{Z}$  for all  $C \in \varphi$

question: what is maximal  $\sum_{C \in \varphi} w_C \cdot \bar{v}(C)$  for valuation  $v$ ?

## Weighted Maximal Satisfiability ( $\text{maxSAT}_w$ )

instance: CNF formula  $\varphi$  with weight  $w_C \in \mathbb{Z}$  for all  $C \in \varphi$

question: what is maximal  $\sum_{C \in \varphi} w_C \cdot \bar{v}(C)$  for valuation  $v$ ?

### Example

$$\varphi = \{(\neg x, 2), \quad (y, 4), \quad (\neg x \vee \neg y, 5), \quad (x \vee \neg y, 1)\}$$

►  $\text{maxSAT}_w(\varphi) =$

## Weighted Maximal Satisfiability ( $\text{maxSAT}_w$ )

instance: CNF formula  $\varphi$  with weight  $w_C \in \mathbb{Z}$  for all  $C \in \varphi$

question: what is maximal  $\sum_{C \in \varphi} w_C \cdot \bar{v}(C)$  for valuation  $v$ ?

### Example

$$\varphi = \{(\neg x, 2), \quad (y, 4), \quad (\neg x \vee \neg y, 5), \quad (x \vee \neg y, 1)\}$$

►  $\text{maxSAT}_w(\varphi) = 11$  e.g. for valuation  $v(x) = \text{F}$  and  $v(y) = \text{T}$

## Weighted Maximal Satisfiability ( $\text{maxSAT}_w$ )

instance: CNF formula  $\varphi$  with weight  $w_C \in \mathbb{Z}$  for all  $C \in \varphi$

question: what is maximal  $\sum_{C \in \varphi} w_C \cdot \bar{v}(C)$  for valuation  $v$ ?

## Weighted Partial Maximal Satisfiability ( $\text{pmaxSAT}_w$ )

instance: CNF formulas  $\varphi$  and  $\chi$ , with weight  $w_C \in \mathbb{Z}$  for all  $C \in \varphi$

question: what is maximal  $\sum_{C \in \varphi} w_C \cdot \bar{v}(C)$  for valuation  $v$  with  $v(\chi) = \text{T}$ ?

## Example

$$\varphi = \{(\neg x, 2), \quad (y, 4), \quad (\neg x \vee \neg y, 5), \quad (x \vee \neg y, 1)\}$$

►  $\text{maxSAT}_w(\varphi) = 11$  e.g. for valuation  $v(x) = \text{F}$  and  $v(y) = \text{T}$

## Weighted Maximal Satisfiability ( $\text{maxSAT}_w$ )

instance: CNF formula  $\varphi$  with weight  $w_C \in \mathbb{Z}$  for all  $C \in \varphi$

question: what is maximal  $\sum_{C \in \varphi} w_C \cdot \bar{v}(C)$  for valuation  $v$ ?

## Weighted Partial Maximal Satisfiability ( $\text{pmaxSAT}_w$ )

instance: CNF formulas  $\varphi$  and  $\chi$ , with weight  $w_C \in \mathbb{Z}$  for all  $C \in \varphi$

question: what is maximal  $\sum_{C \in \varphi} w_C \cdot \bar{v}(C)$  for valuation  $v$  with  $v(\chi) = \text{T}$ ?

## Example

$$\varphi = \{(\neg x, 2), \quad (y, 4), \quad (\neg x \vee \neg y, 5), \quad (x \vee \neg y, 1)\}$$

$$\chi = \{x\}$$

- ▶  $\text{maxSAT}_w(\varphi) = 11$  e.g. for valuation  $v(x) = \text{F}$  and  $v(y) = \text{T}$
- ▶  $\text{pmaxSAT}_w(\chi, \varphi) =$

## Weighted Maximal Satisfiability ( $\text{maxSAT}_w$ )

instance: CNF formula  $\varphi$  with weight  $w_C \in \mathbb{Z}$  for all  $C \in \varphi$

question: what is maximal  $\sum_{C \in \varphi} w_C \cdot \bar{v}(C)$  for valuation  $v$ ?

## Weighted Partial Maximal Satisfiability ( $\text{pmaxSAT}_w$ )

instance: CNF formulas  $\varphi$  and  $\chi$ , with weight  $w_C \in \mathbb{Z}$  for all  $C \in \varphi$

question: what is maximal  $\sum_{C \in \varphi} w_C \cdot \bar{v}(C)$  for valuation  $v$  with  $v(\chi) = \text{T}$ ?

## Example

$$\varphi = \{(\neg x, 2), \quad (y, 4), \quad (\neg x \vee \neg y, 5), \quad (x \vee \neg y, 1)\}$$

$$\chi = \{x\}$$

- ▶  $\text{maxSAT}_w(\varphi) = 11$  e.g. for valuation  $v(x) = \text{F}$  and  $v(y) = \text{T}$
- ▶  $\text{pmaxSAT}_w(\chi, \varphi) = 6$ , e.g. for valuation  $v(x) = \text{T}$  and  $v(y) = \text{F}$



## Weighted Maximal Satisfiability ( $\text{maxSAT}_w$ )

instance: CNF formula  $\varphi$  with weight  $w_C \in \mathbb{Z}$  for all  $C \in \varphi$

question: what is maximal  $\sum_{C \in \varphi} w_C \cdot \bar{v}(C)$  for valuation  $v$ ?

## Weighted Partial Maximal Satisfiability ( $\text{pmaxSAT}_w$ )

instance: CNF formulas  $\varphi$  and  $\chi$ , with weight  $w_C \in \mathbb{Z}$  for all  $C \in \varphi$

question: what is maximal  $\sum_{C \in \varphi} w_C \cdot \bar{v}(C)$  for valuation  $v$  with  $v(\chi) = \text{T}$ ?

## Notation

write  $\text{maxSAT}_w(\varphi)$  and  $\text{pmaxSAT}_w(\chi, \varphi)$  for solutions to these problems

## Example

$$\varphi = \{(\neg x, 2), \quad (y, 4), \quad (\neg x \vee \neg y, 5), \quad (x \vee \neg y, 1)\}$$

$$\chi = \{x\}$$

- ▶  $\text{maxSAT}_w(\varphi) = 11$  e.g. for valuation  $v(x) = \text{F}$  and  $v(y) = \text{T}$
- ▶  $\text{pmaxSAT}_w(\chi, \varphi) = 6$ , e.g. for valuation  $v(x) = \text{T}$  and  $v(y) = \text{F}$

## Minimum Unsatisfiability (minUNSAT)

instance: CNF formula  $\varphi$

question: what is **minimal**  $\sum_{C \in \varphi} \bar{v}(\neg C)$  for valuation  $v$ ?

## Minimum Unsatisfiability (minUNSAT)

instance: CNF formula  $\varphi$

question: what is minimal  $\sum_{C \in \varphi} \bar{v}(\neg C)$  for valuation  $v$ ?

### Notation

write  $\text{minUNSAT}(\varphi)$  for solution to minimal unsatisfiability problem for  $\varphi$

## Minimum Unsatisfiability (minUNSAT)

instance: CNF formula  $\varphi$

question: what is minimal  $\sum_{C \in \varphi} \bar{v}(\neg C)$  for valuation  $v$ ?

## Notation

write  $\text{minUNSAT}(\varphi)$  for solution to minimal unsatisfiability problem for  $\varphi$

## Lemma

$$|\varphi| = \text{minUNSAT}(\varphi) + \text{maxSAT}(\varphi)$$

## Minimum Unsatisfiability (minUNSAT)

instance: CNF formula  $\varphi$

question: what is minimal  $\sum_{C \in \varphi} \bar{v}(\neg C)$  for valuation  $v$ ?

## Notation

write  $\text{minUNSAT}(\varphi)$  for solution to minimal unsatisfiability problem for  $\varphi$

## Lemma

$$|\varphi| = \text{minUNSAT}(\varphi) + \text{maxSAT}(\varphi)$$

## Example

$$\varphi = \{ \neg x, \quad x \vee y, \quad \neg y \vee \neg z, \quad x, \quad y \vee \neg z \}$$

►  $\text{maxSAT}(\varphi) =$

## Minimum Unsatisfiability (minUNSAT)

instance: CNF formula  $\varphi$

question: what is minimal  $\sum_{C \in \varphi} \bar{v}(\neg C)$  for valuation  $v$ ?

## Notation

write  $\text{minUNSAT}(\varphi)$  for solution to minimal unsatisfiability problem for  $\varphi$

## Lemma

$$|\varphi| = \text{minUNSAT}(\varphi) + \text{maxSAT}(\varphi)$$

## Example

$$\varphi = \{\neg x, \quad x \vee y, \quad \neg y \vee \neg z, \quad x, \quad y \vee \neg z\}$$

using  $v(x) = v(y) = \text{T}$  and  $v(z) = \text{F}$  have

►  $\text{maxSAT}(\varphi) = 4$

## Minimum Unsatisfiability (minUNSAT)

instance: CNF formula  $\varphi$

question: what is minimal  $\sum_{C \in \varphi} \bar{v}(\neg C)$  for valuation  $v$ ?

## Notation

write  $\text{minUNSAT}(\varphi)$  for solution to minimal unsatisfiability problem for  $\varphi$

## Lemma

$$|\varphi| = \text{minUNSAT}(\varphi) + \text{maxSAT}(\varphi)$$

## Example

$$\varphi = \{\neg x, \quad x \vee y, \quad \neg y \vee \neg z, \quad x, \quad y \vee \neg z\}$$

using  $v(x) = v(y) = \text{T}$  and  $v(z) = \text{F}$  have

- ▶  $\text{maxSAT}(\varphi) = 4$
- ▶  $\text{minUNSAT}(\varphi) =$

## Minimum Unsatisfiability (minUNSAT)

instance: CNF formula  $\varphi$

question: what is minimal  $\sum_{C \in \varphi} \bar{v}(\neg C)$  for valuation  $v$ ?

## Notation

write  $\text{minUNSAT}(\varphi)$  for solution to minimal unsatisfiability problem for  $\varphi$

## Lemma

$$|\varphi| = \text{minUNSAT}(\varphi) + \text{maxSAT}(\varphi)$$

## Example

$$\varphi = \{\neg x, \quad x \vee y, \quad \neg y \vee \neg z, \quad x, \quad y \vee \neg z\}$$

using  $v(x) = v(y) = \text{T}$  and  $v(z) = \text{F}$  have

- ▶  $\text{maxSAT}(\varphi) = 4$
- ▶  $\text{minUNSAT}(\varphi) = 1$



## Minimum Unsatisfiability (minUNSAT)

instance: CNF formula  $\varphi$

question: what is minimal  $\sum_{C \in \varphi} \bar{v}(\neg C)$  for valuation  $v$ ?

## Notation

write  $\text{minUNSAT}(\varphi)$  for solution to minimal unsatisfiability problem for  $\varphi$

## Lemma

$$|\varphi| = \text{minUNSAT}(\varphi) + \text{maxSAT}(\varphi)$$

## Example

$$\varphi = \{ \neg x, \quad x \vee y, \quad \neg y \vee \neg z, \quad x, \quad y \vee \neg z \}$$

using  $v(x) = v(y) = \text{T}$  and  $v(z) = \text{F}$  have

- ▶  $\text{maxSAT}(\varphi) = 4$
- ▶  $\text{minUNSAT}(\varphi) = 1$

## Remark

$\text{maxSAT}$  and  $\text{minUNSAT}$  are **dual notions**

- Summary of Last Week
- Maximum Satisfiability
- Algorithms for Minimum Unsatisfiability
  - Branch and Bound
  - Binary Search
- Application: Automotive Configuration
- NP-Completeness

- Summary of Last Week
- Maximum Satisfiability
- Algorithms for Minimum Unsatisfiability
  - Branch and Bound
  - Binary Search
- Application: Automotive Configuration
- NP-Completeness

## Idea

- ▶ gets **list** of clauses  $\varphi$  as input and returns **minUNSAT**( $\varphi$ )

## Idea

- ▶ gets list of clauses  $\varphi$  as input and returns  $\text{minUNSAT}(\varphi)$
- ▶ explores assignments in depth-first search

# Branch & Bound

## Idea

- ▶ gets list of clauses  $\varphi$  as input and returns  $\text{minUNSAT}(\varphi)$
- ▶ explores assignments in depth-first search

## Ingredients

- ▶ **UB** is minimal number of unsatisfied clauses found so far (**upper bound**)

# Branch & Bound

## Idea

- ▶ gets list of clauses  $\varphi$  as input and returns  $\text{minUNSAT}(\varphi)$
- ▶ explores assignments in depth-first search

## Ingredients

- ▶ UB is minimal number of unsatisfied clauses found so far (upper bound)
- ▶  $\varphi_x$  is formula  $\varphi$  with all occurrences of  $x$  replaced by T
- ▶  $\varphi_{\bar{x}}$  is formula  $\varphi$  with all occurrences of  $x$  replaced by F

# Branch & Bound

## Idea

- ▶ gets list of clauses  $\varphi$  as input and returns  $\text{minUNSAT}(\varphi)$
- ▶ explores assignments in depth-first search

## Ingredients

- ▶ UB is minimal number of unsatisfied clauses found so far (upper bound)
- ▶  $\varphi_x$  is formula  $\varphi$  with all occurrences of  $x$  replaced by T
- ▶  $\varphi_{\bar{x}}$  is formula  $\varphi$  with all occurrences of  $x$  replaced by F
- ▶ for list of clauses  $\varphi$ , function  $\text{simp}(\varphi)$ 
  - ▶ replaces  $\neg T$  by F and  $\neg F$  by T
  - ▶ drops all clauses which contain T
  - ▶ removes F from all remaining clauses



# Branch & Bound

## Idea

- ▶ gets list of clauses  $\varphi$  as input and returns  $\text{minUNSAT}(\varphi)$
- ▶ explores assignments in depth-first search

## Ingredients

- ▶ UB is minimal number of unsatisfied clauses found so far (upper bound)
- ▶  $\varphi_x$  is formula  $\varphi$  with all occurrences of  $x$  replaced by  $T$
- ▶  $\varphi_{\bar{x}}$  is formula  $\varphi$  with all occurrences of  $x$  replaced by  $F$
- ▶ for list of clauses  $\varphi$ , function  $\text{simp}(\varphi)$ 
  - ▶ replaces  $\neg T$  by  $F$  and  $\neg F$  by  $T$
  - ▶ drops all clauses which contain  $T$
  - ▶ removes  $F$  from all remaining clauses

## Example

$$\begin{aligned} \varphi &= y \vee \neg F, & x \vee y \vee F, & & F, & & x \vee \neg y \vee T, & & x \vee \neg z \\ \text{simp}(\varphi) &= & x \vee y, & & \square, & & & & x \vee \neg z \end{aligned}$$

# Branch & Bound

## Idea

- ▶ gets list of clauses  $\varphi$  as input and returns  $\text{minUNSAT}(\varphi)$
- ▶ explores assignments in depth-first search

## Ingredients

- ▶ UB is minimal number of unsatisfied clauses found so far (upper bound)
- ▶  $\varphi_x$  is formula  $\varphi$  with all occurrences of  $x$  replaced by  $T$
- ▶  $\varphi_{\bar{x}}$  is formula  $\varphi$  with all occurrences of  $x$  replaced by  $F$
- ▶ for list of clauses  $\varphi$ , function  $\text{simp}(\varphi)$ 
  - ▶ replaces  $\neg T$  by  $F$  and  $\neg F$  by  $T$
  - ▶ drops all clauses which contain  $T$
  - ▶ removes  $F$  from all remaining clauses
- ▶  $\square$  denotes empty clause and  $\#\text{empty}(\varphi)$  number of empty clauses in  $\varphi$

## Example

$$\begin{array}{llllll} \varphi = y \vee \neg F, & x \vee y \vee F, & F, & x \vee \neg y \vee T, & x \vee \neg z \\ \text{simp}(\varphi) = & x \vee y, & \square, & & x \vee \neg z \end{array}$$

## Algorithm (Branch & Bound)

---

```
function BnB( $\varphi$ , UB)
   $\varphi$  = simp( $\varphi$ )
  if  $\varphi$  contains only empty clauses then
    return #empty( $\varphi$ )
  if #empty( $\varphi$ )  $\geq$  UB then
    return UB
  x = selectVariable( $\varphi$ )
  UB' = min(UB, BnB( $\varphi_x$ , UB))
  return min(UB', BnB( $\varphi_{\bar{x}}$ , UB'))
```

---

## Algorithm (Branch & Bound)

---

```
function BnB( $\varphi$ , UB)
   $\varphi$  = simp( $\varphi$ )
  if  $\varphi$  contains only empty clauses then
    return #empty( $\varphi$ )
  if #empty( $\varphi$ )  $\geq$  UB then
    return UB
  x = selectVariable( $\varphi$ )
  UB' = min(UB, BnB( $\varphi_x$ , UB))
  return min(UB', BnB( $\varphi_{\bar{x}}$ , UB'))
```

---

- note that number of clauses falsified by any valuation is  $\leq |\varphi|$

## Algorithm (Branch & Bound)

---

```
function BnB( $\varphi$ , UB)
   $\varphi$  = simp( $\varphi$ )
  if  $\varphi$  contains only empty clauses then
    return #empty( $\varphi$ )
  if #empty( $\varphi$ )  $\geq$  UB then
    return UB
  x = selectVariable( $\varphi$ )
  UB' = min(UB, BnB( $\varphi_x$ , UB))
  return min(UB', BnB( $\varphi_{\bar{x}}$ , UB'))
```

---

- ▶ note that number of clauses falsified by any valuation is  $\leq |\varphi|$
- ▶ start by calling **BnB( $\varphi$ ,  $|\varphi|$ )**

## Algorithm (Branch & Bound)

---

```
function BnB( $\varphi$ , UB)
   $\varphi$  = simp( $\varphi$ )
  if  $\varphi$  contains only empty clauses then
    return #empty( $\varphi$ )
  if #empty( $\varphi$ )  $\geq$  UB then
    return UB
  x = selectVariable( $\varphi$ )
  UB' = min(UB, BnB( $\varphi_x$ , UB))
  return min(UB', BnB( $\varphi_{\bar{x}}$ , UB'))
```

---

- ▶ note that number of clauses falsified by any valuation is  $\leq |\varphi|$
- ▶ start by calling  $\text{BnB}(\varphi, |\varphi|)$
- ▶ **idea:** #empty( $\varphi$ ) is number of clauses falsified by current valuation

## Example

- ▶  $\varphi = x, \neg x \vee y, z \vee \neg y, x \vee z, x \vee y, \neg y$

## Example

- ▶  $\varphi = x, \neg x \vee y, z \vee \neg y, x \vee z, x \vee y, \neg y$
- ▶ call **BnB**( $\varphi$ , 6)



## Example

- ▶  $\varphi = x, \neg x \vee y, z \vee \neg y, x \vee z, x \vee y, \neg y$
- ▶ call  $\text{BnB}(\varphi, 6)$
- ▶  $\text{simp}(\varphi) = \varphi$

$\text{BnB}(\varphi, 6)$

## Example

- ▶  $\varphi = x, \neg x \vee y, z \vee \neg y, x \vee z, x \vee y, \neg y$
- ▶ call  $\text{BnB}(\varphi, 6)$
- ▶  $\text{simp}(\varphi) = \varphi$

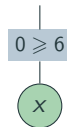
$\text{BnB}(\varphi, 6)$

|  
0  $\geq$  6

## Example

- ▶  $\varphi = x, \neg x \vee y, z \vee \neg y, x \vee z, x \vee y, \neg y$
- ▶ call  $\text{BnB}(\varphi, 6)$
- ▶  $\text{simp}(\varphi) = \varphi$

$\text{BnB}(\varphi, 6)$



## Example

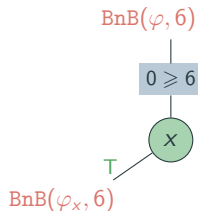
- ▶  $\varphi = x, \neg x \vee y, z \vee \neg y, x \vee z, x \vee y, \neg y$
- ▶ call  $\text{BnB}(\varphi, 6)$
- ▶  $\text{simp}(\varphi) = \varphi$
- ▶  $\varphi_x = \text{T}, \neg \text{T} \vee y, z \vee \neg y, \text{T} \vee z, \text{T} \vee y, \neg y$

$\text{BnB}(\varphi, 6)$



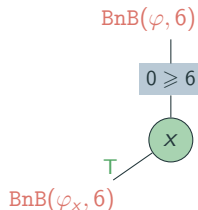
## Example

- ▶  $\varphi = x, \neg x \vee y, z \vee \neg y, x \vee z, x \vee y, \neg y$
- ▶ call  $\text{BnB}(\varphi, 6)$
- ▶  $\text{simp}(\varphi) = \varphi$
- ▶  $\varphi_x = T, \neg T \vee y, z \vee \neg y, T \vee z, T \vee y, \neg y$



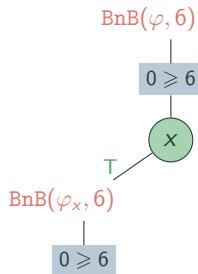
## Example

- ▶  $\varphi = x, \neg x \vee y, z \vee \neg y, x \vee z, x \vee y, \neg y$
- ▶ call  $\text{BnB}(\varphi, 6)$
- ▶  $\text{simp}(\varphi) = \varphi$
- ▶  $\varphi_x = \text{T}, \neg \text{T} \vee y, z \vee \neg y, \text{T} \vee z, \text{T} \vee y, \neg y$   
 $\text{simp}(\varphi_x) = y, z \vee \neg y, \neg y$



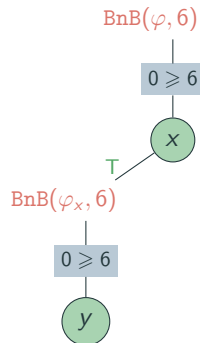
## Example

- ▶  $\varphi = x, \neg x \vee y, z \vee \neg y, x \vee z, x \vee y, \neg y$
- ▶ call  $\text{BnB}(\varphi, 6)$
- ▶  $\text{simp}(\varphi) = \varphi$
- ▶  $\varphi_x = T, \neg T \vee y, z \vee \neg y, T \vee z, T \vee y, \neg y$   
 $\text{simp}(\varphi_x) = y, z \vee \neg y, \neg y$



## Example

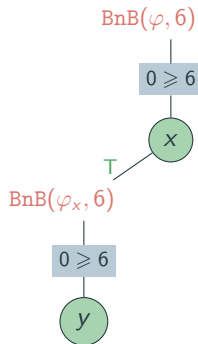
- ▶  $\varphi = x, \neg x \vee y, z \vee \neg y, x \vee z, x \vee y, \neg y$
- ▶ call  $\text{BnB}(\varphi, 6)$
- ▶  $\text{simp}(\varphi) = \varphi$
- ▶  $\varphi_x = \text{T}, \neg \text{T} \vee y, z \vee \neg y, \text{T} \vee z, \text{T} \vee y, \neg y$   
 $\text{simp}(\varphi_x) = y, z \vee \neg y, \neg y$





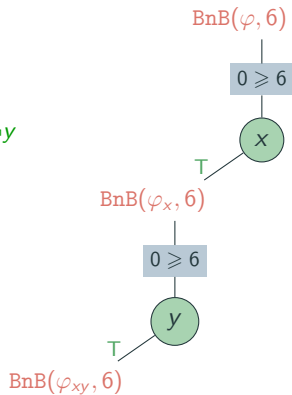
## Example

- ▶  $\varphi = x, \neg x \vee y, z \vee \neg y, x \vee z, x \vee y, \neg y$
- ▶ call  $\text{BnB}(\varphi, 6)$
- ▶  $\text{simp}(\varphi) = \varphi$
- ▶  $\varphi_x = T, \neg T \vee y, z \vee \neg y, T \vee z, T \vee y, \neg y$   
     $\text{simp}(\varphi_x) = y, z \vee \neg y, \neg y$ 
  - ▶  $\varphi_{xy} = T, z \vee \neg T, \neg T$



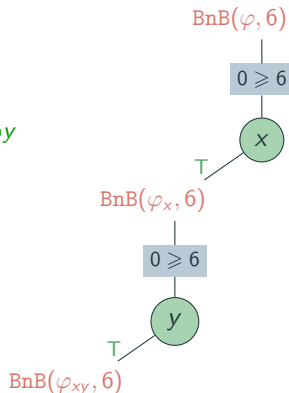
## Example

- ▶  $\varphi = x, \neg x \vee y, z \vee \neg y, x \vee z, x \vee y, \neg y$
  - ▶ call **BnB**( $\varphi$ , 6)
  - ▶ **simp**( $\varphi$ ) =  $\varphi$
- 
- ▶  $\varphi_x = T, \neg T \vee y, z \vee \neg y, T \vee z, T \vee y, \neg y$   
**simp**( $\varphi_x$ ) =  $y, z \vee \neg y, \neg y$ 
    - ▶  $\varphi_{xy} = T, z \vee \neg T, \neg T$



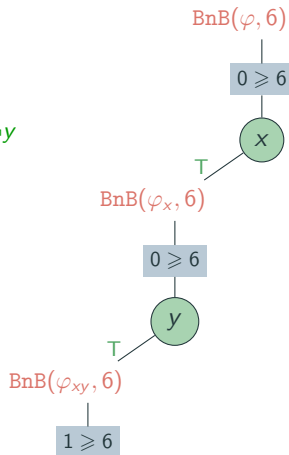
## Example

- ▶  $\varphi = x, \neg x \vee y, z \vee \neg y, x \vee z, x \vee y, \neg y$
- ▶ call  $\text{BnB}(\varphi, 6)$
- ▶  $\text{simp}(\varphi) = \varphi$
- ▶  $\varphi_x = \text{T}, \neg \text{T} \vee y, z \vee \neg y, \text{T} \vee z, \text{T} \vee y, \neg y$   
     $\text{simp}(\varphi_x) = y, z \vee \neg y, \neg y$ 
  - ▶  $\varphi_{xy} = \text{T}, z \vee \neg \text{T}, \neg \text{T}$   
     $\text{simp}(\varphi_{xy}) = z, \square$



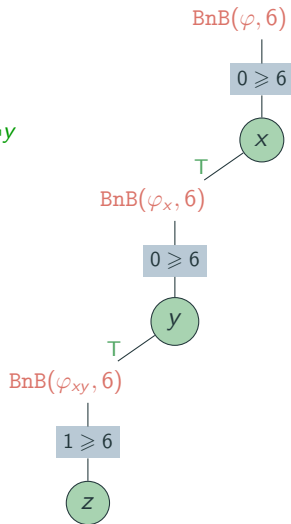
## Example

- ▶  $\varphi = x, \neg x \vee y, z \vee \neg y, x \vee z, x \vee y, \neg y$
- ▶ call  $\text{BnB}(\varphi, 6)$
- ▶  $\text{simp}(\varphi) = \varphi$
- ▶  $\varphi_x = \text{T}, \neg \text{T} \vee y, z \vee \neg y, \text{T} \vee z, \text{T} \vee y, \neg y$   
 $\text{simp}(\varphi_x) = y, z \vee \neg y, \neg y$ 
  - ▶  $\varphi_{xy} = \text{T}, z \vee \neg \text{T}, \neg \text{T}$   
 $\text{simp}(\varphi_{xy}) = z, \square$



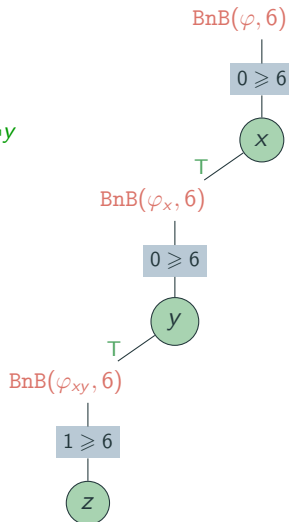
## Example

- ▶  $\varphi = x, \neg x \vee y, z \vee \neg y, x \vee z, x \vee y, \neg y$
- ▶ call  $\text{BnB}(\varphi, 6)$
- ▶  $\text{simp}(\varphi) = \varphi$
- ▶  $\varphi_x = \text{T}, \neg \text{T} \vee y, z \vee \neg y, \text{T} \vee z, \text{T} \vee y, \neg y$   
 $\text{simp}(\varphi_x) = y, z \vee \neg y, \neg y$ 
  - ▶  $\varphi_{xy} = \text{T}, z \vee \neg \text{T}, \neg \text{T}$   
 $\text{simp}(\varphi_{xy}) = z, \square$



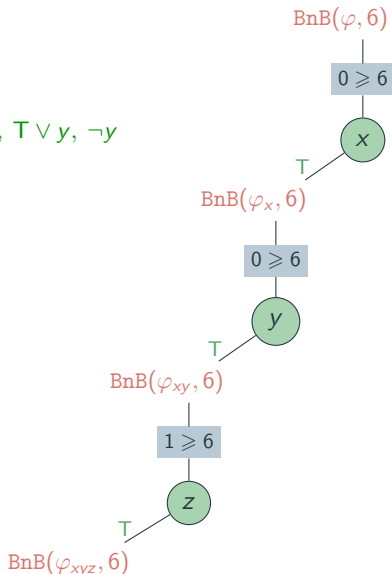
## Example

- ▶  $\varphi = x, \neg x \vee y, z \vee \neg y, x \vee z, x \vee y, \neg y$
- ▶ call  $\text{BnB}(\varphi, 6)$
- ▶  $\text{simp}(\varphi) = \varphi$
- ▶  $\varphi_x = \text{T}, \neg \text{T} \vee y, z \vee \neg y, \text{T} \vee z, \text{T} \vee y, \neg y$   
 $\text{simp}(\varphi_x) = y, z \vee \neg y, \neg y$ 
  - ▶  $\varphi_{xy} = \text{T}, z \vee \neg \text{T}, \neg \text{T}$   
 $\text{simp}(\varphi_{xy}) = z, \square$ 
    - ▶  $\varphi_{xyz} = \text{T}, \square$



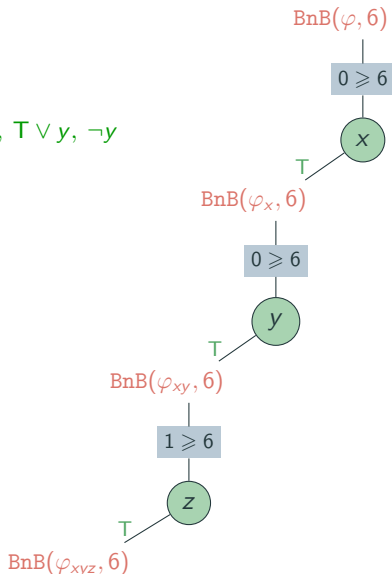
## Example

- ▶  $\varphi = x, \neg x \vee y, z \vee \neg y, x \vee z, x \vee y, \neg y$
- ▶ call  $\text{BnB}(\varphi, 6)$
- ▶  $\text{simp}(\varphi) = \varphi$
- ▶  $\varphi_x = \top, \neg \top \vee y, z \vee \neg y, \top \vee z, \top \vee y, \neg y$   
 $\text{simp}(\varphi_x) = y, z \vee \neg y, \neg y$ 
  - ▶  $\varphi_{xy} = \top, z \vee \neg \top, \neg \top$   
 $\text{simp}(\varphi_{xy}) = z, \square$ 
    - ▶  $\varphi_{xyz} = \top, \square$



## Example

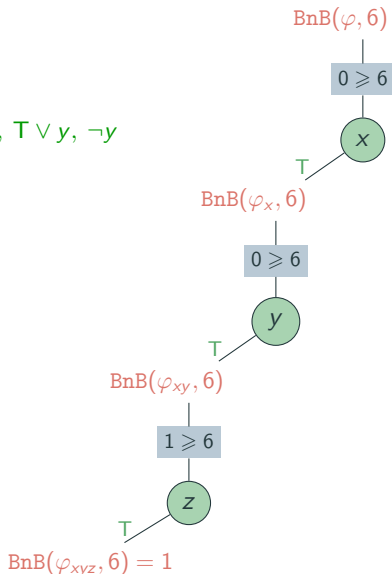
- ▶  $\varphi = x, \neg x \vee y, z \vee \neg y, x \vee z, x \vee y, \neg y$
- ▶ call  $\text{BnB}(\varphi, 6)$
- ▶  $\text{simp}(\varphi) = \varphi$
- ▶  $\varphi_x = \top, \neg \top \vee y, z \vee \neg y, \top \vee z, \top \vee y, \neg y$   
 $\text{simp}(\varphi_x) = y, z \vee \neg y, \neg y$ 
  - ▶  $\varphi_{xy} = \top, z \vee \neg \top, \neg \top$   
 $\text{simp}(\varphi_{xy}) = z, \square$ 
    - ▶  $\varphi_{xyz} = \top, \square$   
 $\text{simp}(\varphi_{xyz}) = \square$





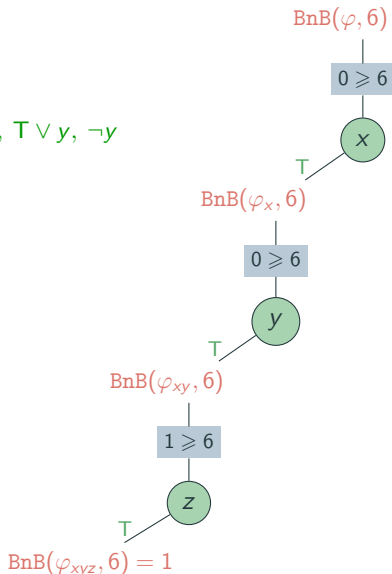
## Example

- ▶  $\varphi = x, \neg x \vee y, z \vee \neg y, x \vee z, x \vee y, \neg y$
- ▶ call  $\text{BnB}(\varphi, 6)$
- ▶  $\text{simp}(\varphi) = \varphi$
- ▶  $\varphi_x = \top, \neg \top \vee y, z \vee \neg y, \top \vee z, \top \vee y, \neg y$   
 $\text{simp}(\varphi_x) = y, z \vee \neg y, \neg y$ 
  - ▶  $\varphi_{xy} = \top, z \vee \neg \top, \neg \top$   
 $\text{simp}(\varphi_{xy}) = z, \square$ 
    - ▶  $\varphi_{xyz} = \top, \square$   
 $\text{simp}(\varphi_{xyz}) = \square$



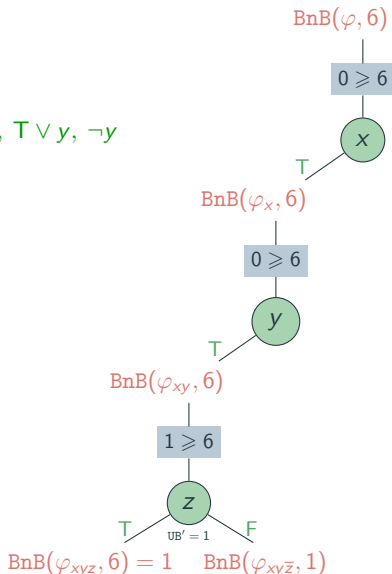
## Example

- ▶  $\varphi = x, \neg x \vee y, z \vee \neg y, x \vee z, x \vee y, \neg y$
- ▶ call  $\text{BnB}(\varphi, 6)$
- ▶  $\text{simp}(\varphi) = \varphi$
- ▶  $\varphi_x = \top, \neg \top \vee y, z \vee \neg y, \top \vee z, \top \vee y, \neg y$   
 $\text{simp}(\varphi_x) = y, z \vee \neg y, \neg y$ 
  - ▶  $\varphi_{xy} = \top, z \vee \neg \top, \neg \top$   
 $\text{simp}(\varphi_{xy}) = z, \square$ 
    - ▶  $\varphi_{xyz} = \top, \square$   
 $\text{simp}(\varphi_{xyz}) = \square$
    - ▶  $\varphi_{xy\bar{z}} = \text{F}, \square$



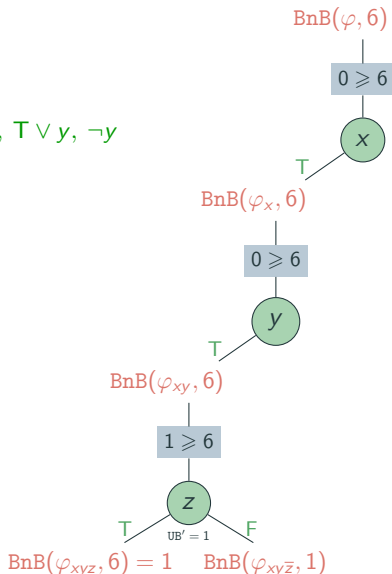
## Example

- ▶  $\varphi = x, \neg x \vee y, z \vee \neg y, x \vee z, x \vee y, \neg y$
- ▶ call  $\text{BnB}(\varphi, 6)$
- ▶  $\text{simp}(\varphi) = \varphi$
- ▶  $\varphi_x = \top, \neg \top \vee y, z \vee \neg y, \top \vee z, \top \vee y, \neg y$   
 $\text{simp}(\varphi_x) = y, z \vee \neg y, \neg y$ 
  - ▶  $\varphi_{xy} = \top, z \vee \neg \top, \neg \top$   
 $\text{simp}(\varphi_{xy}) = z, \square$ 
    - ▶  $\varphi_{xyz} = \top, \square$   
 $\text{simp}(\varphi_{xyz}) = \square$
    - ▶  $\varphi_{xy\bar{z}} = \text{F}, \square$



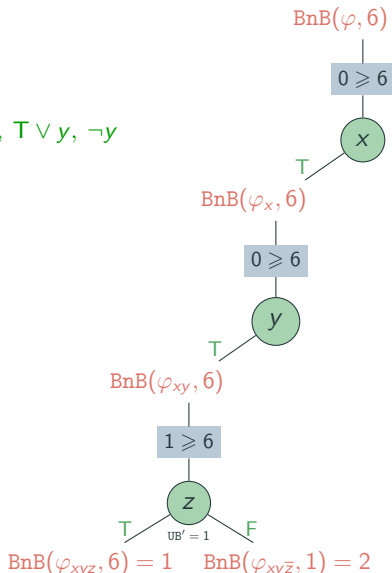
## Example

- ▶  $\varphi = x, \neg x \vee y, z \vee \neg y, x \vee z, x \vee y, \neg y$
- ▶ call  $\text{BnB}(\varphi, 6)$
- ▶  $\text{simp}(\varphi) = \varphi$
- ▶  $\varphi_x = \top, \neg \top \vee y, z \vee \neg y, \top \vee z, \top \vee y, \neg y$   
 $\text{simp}(\varphi_x) = y, z \vee \neg y, \neg y$ 
  - ▶  $\varphi_{xy} = \top, z \vee \neg \top, \neg \top$   
 $\text{simp}(\varphi_{xy}) = z, \square$ 
    - ▶  $\varphi_{xyz} = \top, \square$   
 $\text{simp}(\varphi_{xyz}) = \square$
    - ▶  $\varphi_{xy\bar{z}} = \text{F}, \square$   
 $\text{simp}(\varphi_{xy\bar{z}}) = \square, \square$



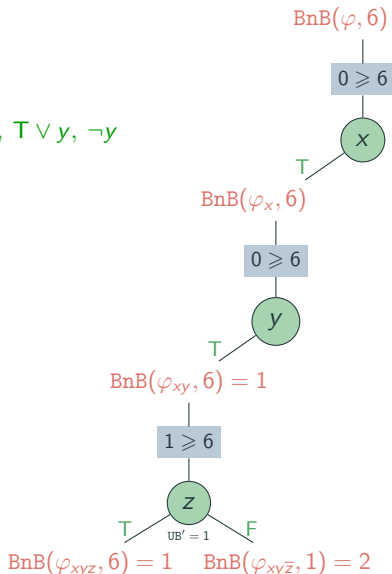
## Example

- ▶  $\varphi = x, \neg x \vee y, z \vee \neg y, x \vee z, x \vee y, \neg y$
- ▶ call  $\text{BnB}(\varphi, 6)$
- ▶  $\text{simp}(\varphi) = \varphi$
  
- ▶  $\varphi_x = \top, \neg \top \vee y, z \vee \neg y, \top \vee z, \top \vee y, \neg y$   
 $\text{simp}(\varphi_x) = y, z \vee \neg y, \neg y$ 
  - ▶  $\varphi_{xy} = \top, z \vee \neg \top, \neg \top$   
 $\text{simp}(\varphi_{xy}) = z, \square$ 
    - ▶  $\varphi_{xyz} = \top, \square$   
 $\text{simp}(\varphi_{xyz}) = \square$
    - ▶  $\varphi_{xy\bar{z}} = \text{F}, \square$   
 $\text{simp}(\varphi_{xy\bar{z}}) = \square, \square$



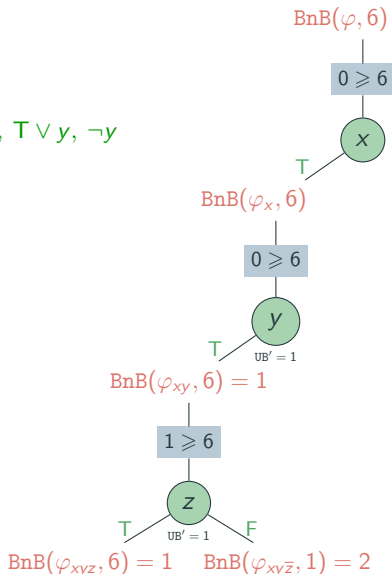
## Example

- ▶  $\varphi = x, \neg x \vee y, z \vee \neg y, x \vee z, x \vee y, \neg y$
- ▶ call  $\text{BnB}(\varphi, 6)$
- ▶  $\text{simp}(\varphi) = \varphi$
  
- ▶  $\varphi_x = \top, \neg \top \vee y, z \vee \neg y, \top \vee z, \top \vee y, \neg y$   
 $\text{simp}(\varphi_x) = y, z \vee \neg y, \neg y$ 
  - ▶  $\varphi_{xy} = \top, z \vee \neg \top, \neg \top$   
 $\text{simp}(\varphi_{xy}) = z, \square$ 
    - ▶  $\varphi_{xyz} = \top, \square$   
 $\text{simp}(\varphi_{xyz}) = \square$
    - ▶  $\varphi_{xy\bar{z}} = \text{F}, \square$   
 $\text{simp}(\varphi_{xy\bar{z}}) = \square, \square$



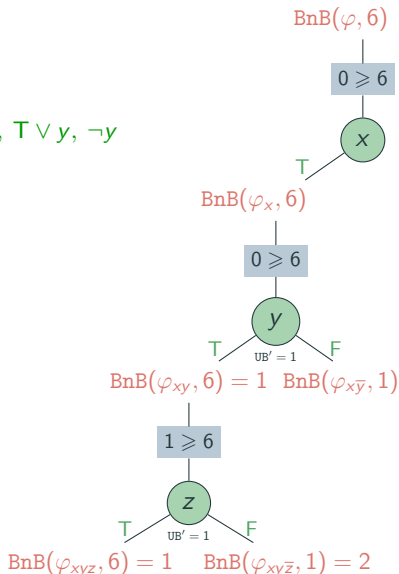
## Example

- ▶  $\varphi = x, \neg x \vee y, z \vee \neg y, x \vee z, x \vee y, \neg y$
- ▶ call  $\text{BnB}(\varphi, 6)$
- ▶  $\text{simp}(\varphi) = \varphi$
  
- ▶  $\varphi_x = \top, \neg \top \vee y, z \vee \neg y, \top \vee z, \top \vee y, \neg y$   
 $\text{simp}(\varphi_x) = y, z \vee \neg y, \neg y$ 
  - ▶  $\varphi_{xy} = \top, z \vee \neg \top, \neg \top$   
 $\text{simp}(\varphi_{xy}) = z, \square$ 
    - ▶  $\varphi_{xyz} = \top, \square$   
 $\text{simp}(\varphi_{xyz}) = \square$
    - ▶  $\varphi_{xy\bar{z}} = \text{F}, \square$   
 $\text{simp}(\varphi_{xy\bar{z}}) = \square, \square$
  - ▶  $\varphi_{x\bar{y}} = \text{F}, z \vee \neg \text{F}, \neg \text{F}$



## Example

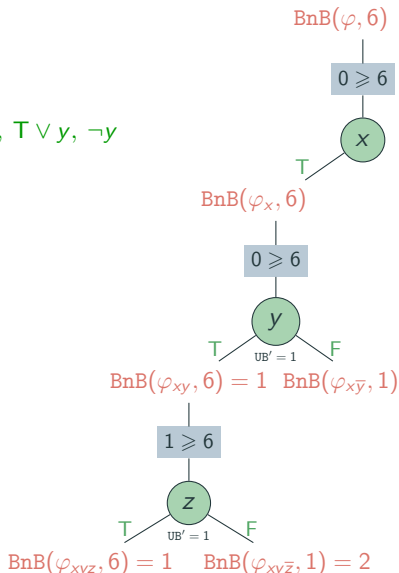
- ▶  $\varphi = x, \neg x \vee y, z \vee \neg y, x \vee z, x \vee y, \neg y$
- ▶ call  $\text{BnB}(\varphi, 6)$
- ▶  $\text{simp}(\varphi) = \varphi$
- ▶  $\varphi_x = \top, \neg \top \vee y, z \vee \neg y, \top \vee z, \top \vee y, \neg y$   
 $\text{simp}(\varphi_x) = y, z \vee \neg y, \neg y$ 
  - ▶  $\varphi_{xy} = \top, z \vee \neg \top, \neg \top$   
 $\text{simp}(\varphi_{xy}) = z, \square$ 
    - ▶  $\varphi_{xyz} = \top, \square$   
 $\text{simp}(\varphi_{xyz}) = \square$
    - ▶  $\varphi_{xy\bar{z}} = \text{F}, \square$   
 $\text{simp}(\varphi_{xy\bar{z}}) = \square, \square$
  - ▶  $\varphi_{x\bar{y}} = \text{F}, z \vee \neg \text{F}, \neg \text{F}$





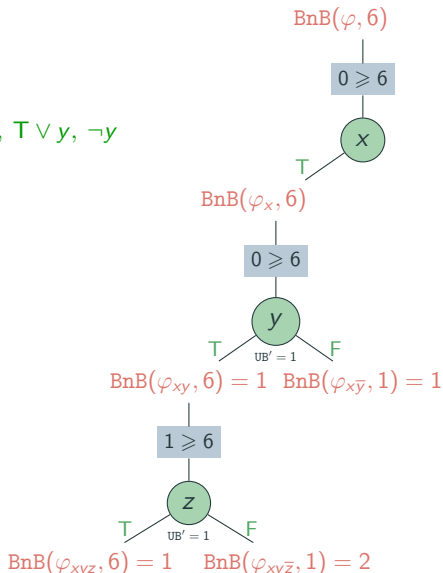
## Example

- ▶  $\varphi = x, \neg x \vee y, z \vee \neg y, x \vee z, x \vee y, \neg y$
- ▶ call  $\text{BnB}(\varphi, 6)$
- ▶  $\text{simp}(\varphi) = \varphi$
  
- ▶  $\varphi_x = \top, \neg \top \vee y, z \vee \neg y, \top \vee z, \top \vee y, \neg y$   
 $\text{simp}(\varphi_x) = y, z \vee \neg y, \neg y$ 
  - ▶  $\varphi_{xy} = \top, z \vee \neg \top, \neg \top$   
 $\text{simp}(\varphi_{xy}) = z, \square$ 
    - ▶  $\varphi_{xyz} = \top, \square$   
 $\text{simp}(\varphi_{xyz}) = \square$
    - ▶  $\varphi_{xy\bar{z}} = \top, \square$   
 $\text{simp}(\varphi_{xy\bar{z}}) = \square, \square$
  - ▶  $\varphi_{x\bar{y}} = \top, z \vee \neg \top, \neg \top$   
 $\text{simp}(\varphi_{x\bar{y}}) = \square$



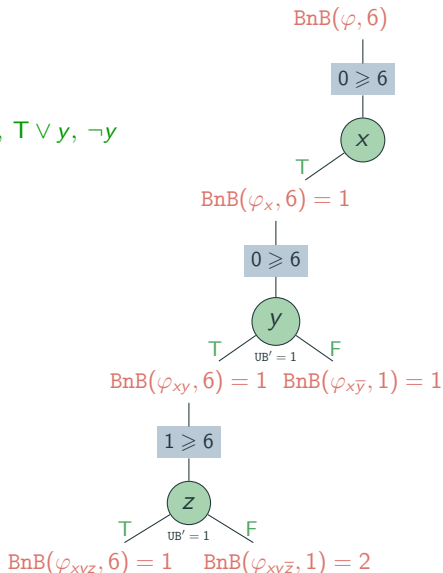
## Example

- ▶  $\varphi = x, \neg x \vee y, z \vee \neg y, x \vee z, x \vee y, \neg y$
- ▶ call  $\text{BnB}(\varphi, 6)$
- ▶  $\text{simp}(\varphi) = \varphi$
- ▶  $\varphi_x = \text{T}, \neg \text{T} \vee y, z \vee \neg y, \text{T} \vee z, \text{T} \vee y, \neg y$   
 $\text{simp}(\varphi_x) = y, z \vee \neg y, \neg y$ 
  - ▶  $\varphi_{xy} = \text{T}, z \vee \neg \text{T}, \neg \text{T}$   
 $\text{simp}(\varphi_{xy}) = z, \square$ 
    - ▶  $\varphi_{xyz} = \text{T}, \square$   
 $\text{simp}(\varphi_{xyz}) = \square$
    - ▶  $\varphi_{xy\bar{z}} = \text{F}, \square$   
 $\text{simp}(\varphi_{xy\bar{z}}) = \square, \square$
  - ▶  $\varphi_{x\bar{y}} = \text{F}, z \vee \neg \text{F}, \neg \text{F}$   
 $\text{simp}(\varphi_{x\bar{y}}) = \square$



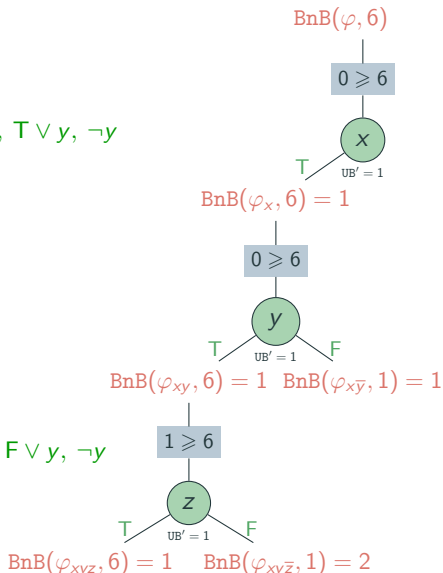
## Example

- ▶  $\varphi = x, \neg x \vee y, z \vee \neg y, x \vee z, x \vee y, \neg y$
- ▶ call  $\text{BnB}(\varphi, 6)$
- ▶  $\text{simp}(\varphi) = \varphi$
  
- ▶  $\varphi_x = \top, \neg \top \vee y, z \vee \neg y, \top \vee z, \top \vee y, \neg y$   
 $\text{simp}(\varphi_x) = y, z \vee \neg y, \neg y$ 
  - ▶  $\varphi_{xy} = \top, z \vee \neg \top, \neg \top$   
 $\text{simp}(\varphi_{xy}) = z, \square$ 
    - ▶  $\varphi_{xyz} = \top, \square$   
 $\text{simp}(\varphi_{xyz}) = \square$
    - ▶  $\varphi_{xy\bar{z}} = \top, \square$   
 $\text{simp}(\varphi_{xy\bar{z}}) = \square, \square$
  - ▶  $\varphi_{x\bar{y}} = \top, z \vee \neg \top, \neg \top$   
 $\text{simp}(\varphi_{x\bar{y}}) = \square$



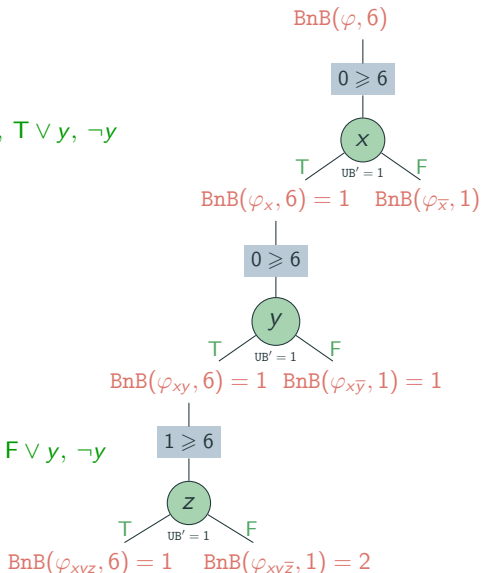
## Example

- ▶  $\varphi = x, \neg x \vee y, z \vee \neg y, x \vee z, x \vee y, \neg y$
- ▶ call  $\text{BnB}(\varphi, 6)$
- ▶  $\text{simp}(\varphi) = \varphi$
- ▶  $\varphi_x = \text{T}, \neg \text{T} \vee y, z \vee \neg y, \text{T} \vee z, \text{T} \vee y, \neg y$   
 $\text{simp}(\varphi_x) = y, z \vee \neg y, \neg y$ 
  - ▶  $\varphi_{xy} = \text{T}, z \vee \neg \text{T}, \neg \text{T}$   
 $\text{simp}(\varphi_{xy}) = z, \square$ 
    - ▶  $\varphi_{xyz} = \text{T}, \square$   
 $\text{simp}(\varphi_{xyz}) = \square$
    - ▶  $\varphi_{xy\bar{z}} = \text{F}, \square$   
 $\text{simp}(\varphi_{xy\bar{z}}) = \square, \square$
  - ▶  $\varphi_{x\bar{y}} = \text{F}, z \vee \neg \text{F}, \neg \text{F}$   
 $\text{simp}(\varphi_{x\bar{y}}) = \square$
- ▶  $\varphi_{\bar{x}} = \text{F}, \neg \text{F} \vee y, z \vee \neg y, \text{F} \vee z, \text{F} \vee y, \neg y$



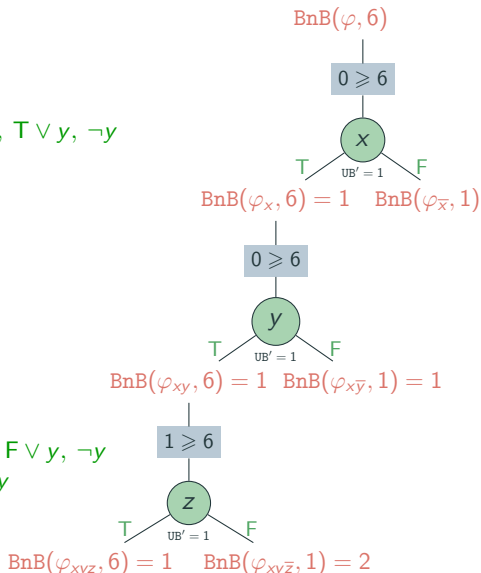
## Example

- ▶  $\varphi = x, \neg x \vee y, z \vee \neg y, x \vee z, x \vee y, \neg y$
- ▶ call  $\text{BnB}(\varphi, 6)$
- ▶  $\text{simp}(\varphi) = \varphi$
- ▶  $\varphi_x = \text{T}, \neg \text{T} \vee y, z \vee \neg y, \text{T} \vee z, \text{T} \vee y, \neg y$   
 $\text{simp}(\varphi_x) = y, z \vee \neg y, \neg y$ 
  - ▶  $\varphi_{xy} = \text{T}, z \vee \neg \text{T}, \neg \text{T}$   
 $\text{simp}(\varphi_{xy}) = z, \square$ 
    - ▶  $\varphi_{xyz} = \text{T}, \square$   
 $\text{simp}(\varphi_{xyz}) = \square$
    - ▶  $\varphi_{xy\bar{z}} = \text{F}, \square$   
 $\text{simp}(\varphi_{xy\bar{z}}) = \square, \square$
  - ▶  $\varphi_{x\bar{y}} = \text{F}, z \vee \neg \text{F}, \neg \text{F}$   
 $\text{simp}(\varphi_{x\bar{y}}) = \square$
- ▶  $\varphi_{\bar{x}} = \text{F}, \neg \text{F} \vee y, z \vee \neg y, \text{F} \vee z, \text{F} \vee y, \neg y$



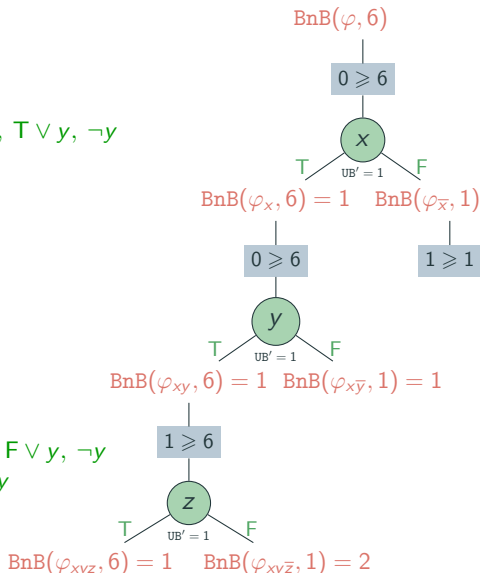
## Example

- ▶  $\varphi = x, \neg x \vee y, z \vee \neg y, x \vee z, x \vee y, \neg y$
- ▶ call  $\text{BnB}(\varphi, 6)$
- ▶  $\text{simp}(\varphi) = \varphi$
- ▶  $\varphi_x = \top, \neg \top \vee y, z \vee \neg y, \top \vee z, \top \vee y, \neg y$   
 $\text{simp}(\varphi_x) = y, z \vee \neg y, \neg y$ 
  - ▶  $\varphi_{xy} = \top, z \vee \neg \top, \neg \top$   
 $\text{simp}(\varphi_{xy}) = z, \square$ 
    - ▶  $\varphi_{xyz} = \top, \square$   
 $\text{simp}(\varphi_{xyz}) = \square$
    - ▶  $\varphi_{xy\bar{z}} = \top, \square$   
 $\text{simp}(\varphi_{xy\bar{z}}) = \square, \square$
  - ▶  $\varphi_{x\bar{y}} = \top, z \vee \neg \top, \neg \top$   
 $\text{simp}(\varphi_{x\bar{y}}) = \square$
- ▶  $\varphi_{\bar{x}} = \bot, \neg \bot \vee y, z \vee \neg y, \bot \vee z, \bot \vee y, \neg y$   
 $\text{simp}(\varphi_{\bar{x}}) = \square, z \vee \neg y, z, y, \neg y$



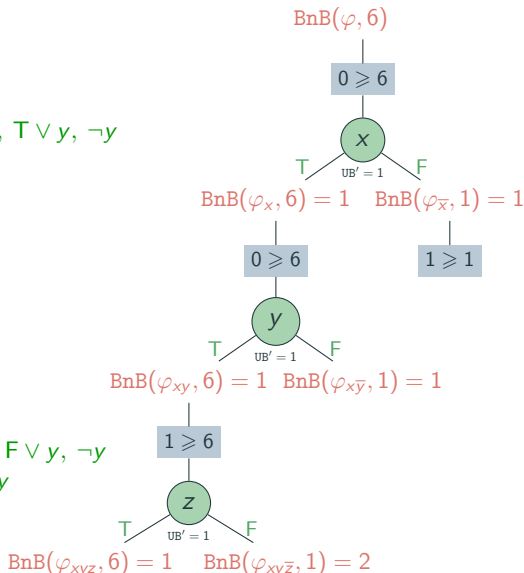
## Example

- ▶  $\varphi = x, \neg x \vee y, z \vee \neg y, x \vee z, x \vee y, \neg y$
- ▶ call  $\text{BnB}(\varphi, 6)$
- ▶  $\text{simp}(\varphi) = \varphi$
- ▶  $\varphi_x = \text{T}, \neg \text{T} \vee y, z \vee \neg y, \text{T} \vee z, \text{T} \vee y, \neg y$   
 $\text{simp}(\varphi_x) = y, z \vee \neg y, \neg y$ 
  - ▶  $\varphi_{xy} = \text{T}, z \vee \neg \text{T}, \neg \text{T}$   
 $\text{simp}(\varphi_{xy}) = z, \square$ 
    - ▶  $\varphi_{xyz} = \text{T}, \square$   
 $\text{simp}(\varphi_{xyz}) = \square$
    - ▶  $\varphi_{xy\bar{z}} = \text{F}, \square$   
 $\text{simp}(\varphi_{xy\bar{z}}) = \square, \square$
  - ▶  $\varphi_{x\bar{y}} = \text{F}, z \vee \neg \text{F}, \neg \text{F}$   
 $\text{simp}(\varphi_{x\bar{y}}) = \square$
- ▶  $\varphi_{\bar{x}} = \text{F}, \neg \text{F} \vee y, z \vee \neg y, \text{F} \vee z, \text{F} \vee y, \neg y$   
 $\text{simp}(\varphi_{\bar{x}}) = \square, z \vee \neg y, z, y, \neg y$



## Example

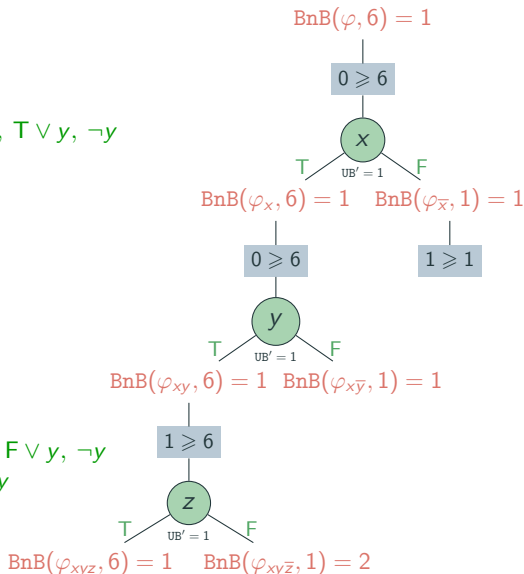
- ▶  $\varphi = x, \neg x \vee y, z \vee \neg y, x \vee z, x \vee y, \neg y$
- ▶ call  $\text{BnB}(\varphi, 6)$
- ▶  $\text{simp}(\varphi) = \varphi$
- ▶  $\varphi_x = \text{T}, \neg \text{T} \vee y, z \vee \neg y, \text{T} \vee z, \text{T} \vee y, \neg y$   
 $\text{simp}(\varphi_x) = y, z \vee \neg y, \neg y$ 
  - ▶  $\varphi_{xy} = \text{T}, z \vee \neg \text{T}, \neg \text{T}$   
 $\text{simp}(\varphi_{xy}) = z, \square$ 
    - ▶  $\varphi_{xyz} = \text{T}, \square$   
 $\text{simp}(\varphi_{xyz}) = \square$
    - ▶  $\varphi_{xy\bar{z}} = \text{F}, \square$   
 $\text{simp}(\varphi_{xy\bar{z}}) = \square, \square$
  - ▶  $\varphi_{x\bar{y}} = \text{F}, z \vee \neg \text{F}, \neg \text{F}$   
 $\text{simp}(\varphi_{x\bar{y}}) = \square$
- ▶  $\varphi_{\bar{x}} = \text{F}, \neg \text{F} \vee y, z \vee \neg y, \text{F} \vee z, \text{F} \vee y, \neg y$   
 $\text{simp}(\varphi_{\bar{x}}) = \square, z \vee \neg y, z, y, \neg y$





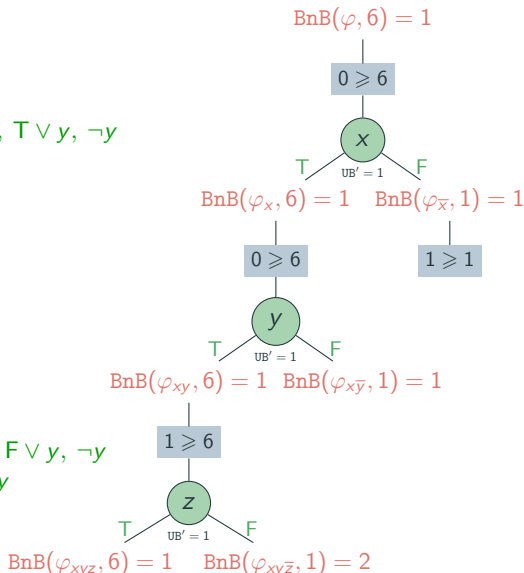
## Example

- ▶  $\varphi = x, \neg x \vee y, z \vee \neg y, x \vee z, x \vee y, \neg y$
- ▶ call  $\text{BnB}(\varphi, 6)$
- ▶  $\text{simp}(\varphi) = \varphi$
  
- ▶  $\varphi_x = \text{T}, \neg \text{T} \vee y, z \vee \neg y, \text{T} \vee z, \text{T} \vee y, \neg y$   
 $\text{simp}(\varphi_x) = y, z \vee \neg y, \neg y$ 
  - ▶  $\varphi_{xy} = \text{T}, z \vee \neg \text{T}, \neg \text{T}$   
 $\text{simp}(\varphi_{xy}) = z, \square$ 
    - ▶  $\varphi_{xyz} = \text{T}, \square$   
 $\text{simp}(\varphi_{xyz}) = \square$
    - ▶  $\varphi_{xy\bar{z}} = \text{F}, \square$   
 $\text{simp}(\varphi_{xy\bar{z}}) = \square, \square$
  - ▶  $\varphi_{x\bar{y}} = \text{F}, z \vee \neg \text{F}, \neg \text{F}$   
 $\text{simp}(\varphi_{x\bar{y}}) = \square$
- ▶  $\varphi_{\bar{x}} = \text{F}, \neg \text{F} \vee y, z \vee \neg y, \text{F} \vee z, \text{F} \vee y, \neg y$   
 $\text{simp}(\varphi_{\bar{x}}) = \square, z \vee \neg y, z, y, \neg y$



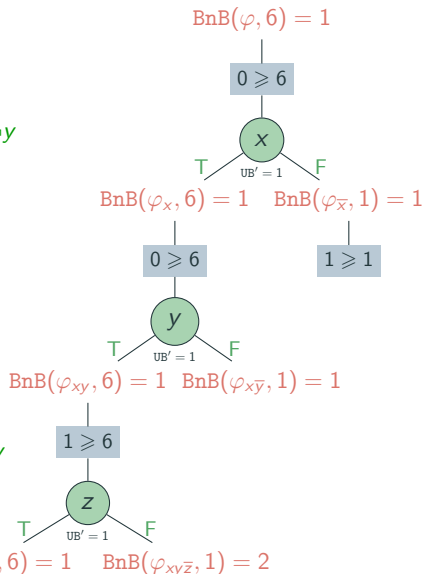
## Example

- ▶  $\varphi = x, \neg x \vee y, z \vee \neg y, x \vee z, x \vee y, \neg y$
- ▶ call  $\text{BnB}(\varphi, 6)$
- ▶  $\text{simp}(\varphi) = \varphi$
- ▶  $\varphi_x = \text{T}, \neg \text{T} \vee y, z \vee \neg y, \text{T} \vee z, \text{T} \vee y, \neg y$   
 $\text{simp}(\varphi_x) = y, z \vee \neg y, \neg y$ 
  - ▶  $\varphi_{xy} = \text{T}, z \vee \neg \text{T}, \neg \text{T}$   
 $\text{simp}(\varphi_{xy}) = z, \square$ 
    - ▶  $\varphi_{xyz} = \text{T}, \square$   
 $\text{simp}(\varphi_{xyz}) = \square$
    - ▶  $\varphi_{xy\bar{z}} = \text{F}, \square$   
 $\text{simp}(\varphi_{xy\bar{z}}) = \square, \square$
  - ▶  $\varphi_{x\bar{y}} = \text{F}, z \vee \neg \text{F}, \neg \text{F}$   
 $\text{simp}(\varphi_{x\bar{y}}) = \square$
- ▶  $\varphi_{\bar{x}} = \text{F}, \neg \text{F} \vee y, z \vee \neg y, \text{F} \vee z, \text{F} \vee y, \neg y$   
 $\text{simp}(\varphi_{\bar{x}}) = \square, z \vee \neg y, z, y, \neg y$
- ▶  $\text{minUNSAT}(\varphi) = 1$



## Example

- ▶  $\varphi = x, \neg x \vee y, z \vee \neg y, x \vee z, x \vee y, \neg y$
- ▶ call  $\text{BnB}(\varphi, 6)$
- ▶  $\text{simp}(\varphi) = \varphi$
- ▶  $\varphi_x = T, \neg T \vee y, z \vee \neg y, T \vee z, T \vee y, \neg y$   
 $\text{simp}(\varphi_x) = y, z \vee \neg y, \neg y$ 
  - ▶  $\varphi_{xy} = T, z \vee \neg T, \neg T$   
 $\text{simp}(\varphi_{xy}) = z, \square$ 
    - ▶  $\varphi_{xyz} = T, \square$   
 $\text{simp}(\varphi_{xyz}) = \square$
    - ▶  $\varphi_{xy\bar{z}} = F, \square$   
 $\text{simp}(\varphi_{xy\bar{z}}) = \square, \square$
  - ▶  $\varphi_{x\bar{y}} = F, z \vee \neg F, \neg F$   
 $\text{simp}(\varphi_{x\bar{y}}) = \square$
- ▶  $\varphi_{\bar{x}} = F, \neg F \vee y, z \vee \neg y, F \vee z, F \vee y, \neg y$   
 $\text{simp}(\varphi_{\bar{x}}) = \square, z \vee \neg y, z, y, \neg y$
- ▶  $\text{minUNSAT}(\varphi) = 1$
- ▶ e.g.  $v(x) = v(y) = v(z) = T$



# Binary Search

## Idea

- ▶ gets list of clauses  $\varphi$  as input and returns  $\text{minUNSAT}(\varphi)$

# Binary Search

## Idea

- ▶ gets list of clauses  $\varphi$  as input and returns  $\text{minUNSAT}(\varphi)$
- ▶ repeatedly call SAT solver in binary search fashion

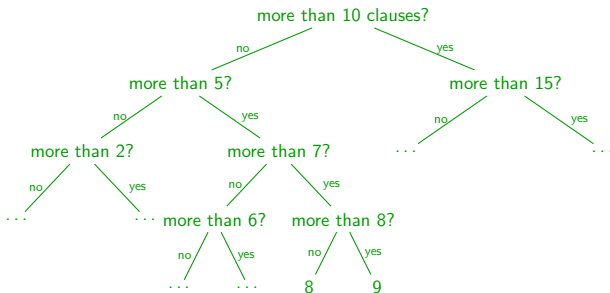
# Binary Search

## Idea

- ▶ gets list of clauses  $\varphi$  as input and returns  $\text{minUNSAT}(\varphi)$
- ▶ repeatedly call SAT solver in binary search fashion

## Example

Suppose given formula with 20 clauses. Can we satisfy ...



# Cardinality Constraints

## Definitions

- ▶ **cardinality constraint** has form  $(\sum_{x \in X} x) \bowtie N$  where  $\bowtie$  is  $=$ ,  $<$ ,  $>$ ,  $\leq$ , or  $\geq$ ,  $X$  is set of propositional variables and  $N \in \mathbb{N}$

# Cardinality Constraints

## Definitions

- ▶ cardinality constraint has form  $(\sum_{x \in X} x) \bowtie N$  where  $\bowtie$  is  $=$ ,  $<$ ,  $>$ ,  $\leq$ , or  $\geq$ ,  $X$  is set of propositional variables and  $N \in \mathbb{N}$

## Example

- ▶  $x + y + z = 1$
- ▶  $x_1 + x_2 + \dots + x_8 \leq 3$



# Cardinality Constraints

## Definitions

- ▶ cardinality constraint has form  $(\sum_{x \in X} x) \bowtie N$  where  $\bowtie$  is  $=$ ,  $<$ ,  $>$ ,  $\leq$ , or  $\geq$ ,  $X$  is set of propositional variables and  $N \in \mathbb{N}$
- ▶ valuation  $v$  satisfies  $(\sum_{x \in X} x) \bowtie N$  iff  $k \bowtie N$   
where  $k$  is number of variables  $x \in X$  such that  $v(x) = \text{T}$

## Example

- ▶  $x + y + z = 1$
- ▶  $x_1 + x_2 + \dots + x_8 \leq 3$

# Cardinality Constraints

## Definitions

- ▶ cardinality constraint has form  $(\sum_{x \in X} x) \bowtie N$  where  $\bowtie$  is  $=$ ,  $<$ ,  $>$ ,  $\leq$ , or  $\geq$ ,  $X$  is set of propositional variables and  $N \in \mathbb{N}$
- ▶ valuation  $v$  satisfies  $(\sum_{x \in X} x) \bowtie N$  iff  $k \bowtie N$  where  $k$  is number of variables  $x \in X$  such that  $v(x) = \text{T}$

## Example

- ▶  $x + y + z = 1$  satisfied by  $v(x) = v(y) = \text{F}$ ,  $v(z) = \text{T}$
- ▶  $x_1 + x_2 + \dots + x_8 \leq 3$  satisfied by  $v(x_1) = \dots = v(x_8) = \text{F}$

# Cardinality Constraints

## Definitions

- ▶ cardinality constraint has form  $(\sum_{x \in X} x) \bowtie N$  where  $\bowtie$  is  $=$ ,  $<$ ,  $>$ ,  $\leq$ , or  $\geq$ ,  $X$  is set of propositional variables and  $N \in \mathbb{N}$
- ▶ valuation  $v$  satisfies  $(\sum_{x \in X} x) \bowtie N$  iff  $k \bowtie N$  where  $k$  is number of variables  $x \in X$  such that  $v(x) = T$

## Remarks

- ▶ cardinality constraints are expressible in CNF

## Example

- ▶  $x + y + z = 1$  satisfied by  $v(x) = v(y) = F$ ,  $v(z) = T$
- ▶  $x_1 + x_2 + \dots + x_8 \leq 3$  satisfied by  $v(x_1) = \dots = v(x_8) = F$

# Cardinality Constraints

## Definitions

- ▶ cardinality constraint has form  $(\sum_{x \in X} x) \bowtie N$  where  $\bowtie$  is  $=$ ,  $<$ ,  $>$ ,  $\leq$ , or  $\geq$ ,  $X$  is set of propositional variables and  $N \in \mathbb{N}$
- ▶ valuation  $v$  satisfies  $(\sum_{x \in X} x) \bowtie N$  iff  $k \bowtie N$  where  $k$  is number of variables  $x \in X$  such that  $v(x) = T$

## Remarks

- ▶ cardinality constraints are expressible in CNF
  - ▶ enumerate all possible subsets

$\mathcal{O}(2^{|X|})$

## Example

- ▶  $x + y + z = 1$  satisfied by  $v(x) = v(y) = F$ ,  $v(z) = T$
- ▶  $x_1 + x_2 + \dots + x_8 \leq 3$  satisfied by  $v(x_1) = \dots = v(x_8) = F$

# Cardinality Constraints

## Definitions

- ▶ cardinality constraint has form  $(\sum_{x \in X} x) \bowtie N$  where  $\bowtie$  is  $=$ ,  $<$ ,  $>$ ,  $\leq$ , or  $\geq$ ,  $X$  is set of propositional variables and  $N \in \mathbb{N}$
- ▶ valuation  $v$  satisfies  $(\sum_{x \in X} x) \bowtie N$  iff  $k \bowtie N$  where  $k$  is number of variables  $x \in X$  such that  $v(x) = T$

## Remarks

- ▶ cardinality constraints are expressible in CNF
  - ▶ enumerate all possible subsets
  - ▶ BDDs

$$\mathcal{O}(2^{|X|})$$
$$\mathcal{O}(N \cdot |X|)$$

## Example

- ▶  $x + y + z = 1$  satisfied by  $v(x) = v(y) = F$ ,  $v(z) = T$
- ▶  $x_1 + x_2 + \dots + x_8 \leq 3$  satisfied by  $v(x_1) = \dots = v(x_8) = F$

# Cardinality Constraints

## Definitions

- ▶ cardinality constraint has form  $(\sum_{x \in X} x) \bowtie N$  where  $\bowtie$  is  $=$ ,  $<$ ,  $>$ ,  $\leq$ , or  $\geq$ ,  $X$  is set of propositional variables and  $N \in \mathbb{N}$
- ▶ valuation  $v$  satisfies  $(\sum_{x \in X} x) \bowtie N$  iff  $k \bowtie N$  where  $k$  is number of variables  $x \in X$  such that  $v(x) = T$

## Remarks

- ▶ cardinality constraints are expressible in CNF
  - ▶ enumerate all possible subsets
  - ▶ BDDs
  - ▶ sorting networks

$$\mathcal{O}(2^{|X|})$$

$$\mathcal{O}(N \cdot |X|)$$

$$\mathcal{O}(|X| \cdot \log^2(|X|))$$

## Example

- ▶  $x + y + z = 1$  satisfied by  $v(x) = v(y) = F$ ,  $v(z) = T$
- ▶  $x_1 + x_2 + \dots + x_8 \leq 3$  satisfied by  $v(x_1) = \dots = v(x_8) = F$

# Cardinality Constraints

## Definitions

- ▶ cardinality constraint has form  $(\sum_{x \in X} x) \bowtie N$  where  $\bowtie$  is  $=$ ,  $<$ ,  $>$ ,  $\leq$ , or  $\geq$ ,  $X$  is set of propositional variables and  $N \in \mathbb{N}$
- ▶ valuation  $v$  satisfies  $(\sum_{x \in X} x) \bowtie N$  iff  $k \bowtie N$  where  $k$  is number of variables  $x \in X$  such that  $v(x) = \text{T}$

## Remarks

- ▶ cardinality constraints are expressible in CNF
  - ▶ enumerate all possible subsets
  - ▶ BDDs
  - ▶ sorting networks
- ▶ write  $\text{CNF}(\sum_{x \in X} x \bowtie N)$  for CNF encoding

$$\mathcal{O}(2^{|X|})$$

$$\mathcal{O}(N \cdot |X|)$$

$$\mathcal{O}(|X| \cdot \log^2(|X|))$$

## Example

- ▶  $x + y + z = 1$  satisfied by  $v(x) = v(y) = \text{F}$ ,  $v(z) = \text{T}$
- ▶  $x_1 + x_2 + \dots + x_8 \leq 3$  satisfied by  $v(x_1) = \dots = v(x_8) = \text{F}$

# Cardinality Constraints

## Definitions

- ▶ cardinality constraint has form  $(\sum_{x \in X} x) \bowtie N$  where  $\bowtie$  is  $=$ ,  $<$ ,  $>$ ,  $\leq$ , or  $\geq$ ,  $X$  is set of propositional variables and  $N \in \mathbb{N}$
- ▶ valuation  $v$  satisfies  $(\sum_{x \in X} x) \bowtie N$  iff  $k \bowtie N$  where  $k$  is number of variables  $x \in X$  such that  $v(x) = \text{T}$

## Remarks

- ▶ cardinality constraints are expressible in CNF
  - ▶ enumerate all possible subsets  $\mathcal{O}(2^{|X|})$
  - ▶ BDDs  $\mathcal{O}(N \cdot |X|)$
  - ▶ sorting networks  $\mathcal{O}(|X| \cdot \log^2(|X|))$
- ▶ write  $\text{CNF}(\sum_{x \in X} x \bowtie N)$  for CNF encoding
- ▶ cardinality constraints occur very frequently! ( $n$ -queens, Minesweeper, ...)

## Example

- ▶  $x + y + z = 1$  satisfied by  $v(x) = v(y) = \text{F}$ ,  $v(z) = \text{T}$
- ▶  $x_1 + x_2 + \dots + x_8 \leq 3$  satisfied by  $v(x_1) = \dots = v(x_8) = \text{F}$



## Algorithm (Binary Search)

---

```
function BinarySearch( $\{C_1, \dots, C_m\}$ )  
   $\varphi := \{C_1 \vee b_1, \dots, C_m \vee b_m\}$   
  return search( $\varphi, 0, m$ )
```

---

## Algorithm (Binary Search)

---

```
function BinarySearch( $\{C_1, \dots, C_m\}$ )
```

```
   $\varphi := \{C_1 \vee b_1, \dots, C_m \vee b_m\}$ 
```

```
  return search( $\varphi$ , 0, m)
```



$b_1, \dots, b_m$  are fresh variables

## Algorithm (Binary Search)

---

```
function BinarySearch( $\{C_1, \dots, C_m\}$ )
```

```
   $\varphi := \{C_1 \vee b_1, \dots, C_m \vee b_m\}$ 
```

```
  return search( $\varphi, 0, m$ )
```



$b_1, \dots, b_m$  are fresh variables

## Algorithm (Binary Search)

---

```
function BinarySearch( $\{C_1, \dots, C_m\}$ )
```

```
   $\varphi := \{C_1 \vee b_1, \dots, C_m \vee b_m\}$ 
```

```
  return search( $\varphi, 0, m$ )
```

$b_1, \dots, b_m$  are fresh variables

---

```
function search( $\varphi, L, U$ )
```

```
  if  $L \geq U$  then
```

```
    return U
```

```
  mid :=  $\lfloor \frac{U+L}{2} \rfloor$ 
```

```
  if SAT( $\varphi \wedge \text{CNF}(\sum_{i=1}^m b_i \leq \text{mid})$ ) then
```

```
    return search( $\varphi, L, \text{mid}$ )
```

```
  else
```

```
    return search( $\varphi, \text{mid} + 1, U$ )
```

---

## Algorithm (Binary Search)

---

```
function BinarySearch( $\{C_1, \dots, C_m\}$ )
```

```
   $\varphi := \{C_1 \vee b_1, \dots, C_m \vee b_m\}$ 
```

```
  return search( $\varphi, 0, m$ )
```

$b_1, \dots, b_m$  are fresh variables

---

```
function search( $\varphi, L, U$ )
```

```
  if  $L \geq U$  then
```

```
    return U
```

```
  mid :=  $\lfloor \frac{U+L}{2} \rfloor$ 
```

```
  if SAT( $\varphi \wedge \text{CNF}(\sum_{i=1}^m b_i \leq \text{mid})$ ) then
```

```
    return search( $\varphi, L, \text{mid}$ )
```

```
  else
```

```
    return search( $\varphi, \text{mid} + 1, U$ )
```

---

## Algorithm (Binary Search)

---

```
function BinarySearch( $\{C_1, \dots, C_m\}$ )
```

```
   $\varphi := \{C_1 \vee b_1, \dots, C_m \vee b_m\}$ 
```

```
  return search( $\varphi, 0, m$ )
```

$b_1, \dots, b_m$  are fresh variables

---

```
function search( $\varphi, L, U$ )
```

```
  if  $L \geq U$  then
```

```
    return U
```

```
  mid :=  $\lfloor \frac{U+L}{2} \rfloor$ 
```

```
  if SAT( $\varphi \wedge \text{CNF}(\sum_{i=1}^m b_i \leq \text{mid})$ ) then
```

```
    return search( $\varphi, L, \text{mid}$ )
```

```
  else
```

```
    return search( $\varphi, \text{mid} + 1, U$ )
```

---

## Algorithm (Binary Search)

---

```
function BinarySearch( $\{C_1, \dots, C_m\}$ )
```

```
   $\varphi := \{C_1 \vee b_1, \dots, C_m \vee b_m\}$ 
```

```
  return search( $\varphi, 0, m$ )
```

$b_1, \dots, b_m$  are fresh variables

---

```
function search( $\varphi, L, U$ )
```

```
  if  $L \geq U$  then
```

```
    return U
```

```
   $\text{mid} := \lfloor \frac{U+L}{2} \rfloor$ 
```

```
  if SAT( $\varphi \wedge \text{CNF}(\sum_{i=1}^m b_i \leq \text{mid})$ ) then
```

```
    return search( $\varphi, L, \text{mid}$ )
```

```
  else
```

```
    return search( $\varphi, \text{mid} + 1, U$ )
```

---

## Algorithm (Binary Search)

---

```
function BinarySearch( $\{C_1, \dots, C_m\}$ )
```

```
   $\varphi := \{C_1 \vee b_1, \dots, C_m \vee b_m\}$ 
```

```
  return search( $\varphi, 0, m$ )
```

$b_1, \dots, b_m$  are fresh variables

---

```
function search( $\varphi, L, U$ )
```

```
  if  $L \geq U$  then
```

```
    return U
```

```
  mid :=  $\lfloor \frac{U+L}{2} \rfloor$ 
```

```
  if SAT( $\varphi \wedge \text{CNF}(\sum_{i=1}^m b_i \leq \text{mid})$ ) then
```

```
    return search( $\varphi, L, \text{mid}$ )
```

```
  else
```

```
    return search( $\varphi, \text{mid} + 1, U$ )
```

---

## Theorem

$\text{BinarySearch}(\psi) = \min \text{UNSAT}(\psi)$



## Example

$$\varphi = \{ \begin{array}{l} 6 \vee 2 \quad , \quad \bar{6} \vee 2 \quad , \quad \bar{2} \vee 1 \quad , \quad \bar{1} \quad , \quad \bar{6} \vee 8 \quad , \\ 6 \vee \bar{8} \quad , \quad 2 \vee 4 \quad , \quad \bar{4} \vee 5 \quad , \quad 7 \vee 5 \quad , \quad \bar{7} \vee 5 \quad , \\ \bar{3} \quad , \quad \bar{5} \vee 3 \quad \end{array} \}$$

## Example

$$\varphi = \{ \begin{array}{lllll} 6 \vee 2 \vee b_1, & \bar{6} \vee 2 \vee b_2, & \bar{2} \vee 1 \vee b_3, & \bar{1} \vee b_4, & \bar{6} \vee 8 \vee b_5, \\ 6 \vee \bar{8} \vee b_6, & 2 \vee 4 \vee b_7, & \bar{4} \vee 5 \vee b_8, & 7 \vee 5 \vee b_9, & \bar{7} \vee 5 \vee b_{10}, \\ \bar{3} \vee b_{11}, & \bar{5} \vee 3 \vee b_{12} \end{array} \}$$

## Example

$$\varphi = \{ 6 \vee 2 \vee b_1, \quad \bar{6} \vee 2 \vee b_2, \quad \bar{2} \vee 1 \vee b_3, \quad \bar{1} \vee b_4, \quad \bar{6} \vee 8 \vee b_5, \\ 6 \vee \bar{8} \vee b_6, \quad 2 \vee 4 \vee b_7, \quad \bar{4} \vee 5 \vee b_8, \quad 7 \vee 5 \vee b_9, \quad \bar{7} \vee 5 \vee b_{10}, \\ \bar{3} \vee b_{11}, \quad \bar{5} \vee 3 \vee b_{12} \}$$

►  $L = 0, U = 12, \text{mid} = 6$        $\text{SAT}(\varphi \wedge \text{CNF}(\sum_{i=1}^m b_i \leq 6))?$       ✓

## Example

$$\varphi = \{ 6 \vee 2 \vee b_1, \quad \bar{6} \vee 2 \vee b_2, \quad \bar{2} \vee 1 \vee b_3, \quad \bar{1} \vee b_4, \quad \bar{6} \vee 8 \vee b_5, \\ 6 \vee \bar{8} \vee b_6, \quad 2 \vee 4 \vee b_7, \quad \bar{4} \vee 5 \vee b_8, \quad 7 \vee 5 \vee b_9, \quad \bar{7} \vee 5 \vee b_{10}, \\ \bar{3} \vee b_{11}, \quad \bar{5} \vee 3 \vee b_{12} \}$$

- ▶  $L = 0, U = 12, \text{mid} = 6$        $\text{SAT}(\varphi \wedge \text{CNF}(\sum_{i=1}^m b_i \leq 6))?$       ✓
- ▶  $L = 0, U = 6, \text{mid} = 3$        $\text{SAT}(\varphi \wedge \text{CNF}(\sum_{i=1}^m b_i \leq 3))?$       ✓

## Example

$$\varphi = \{ 6 \vee 2 \vee b_1, \quad \bar{6} \vee 2 \vee b_2, \quad \bar{2} \vee 1 \vee b_3, \quad \bar{1} \vee b_4, \quad \bar{6} \vee 8 \vee b_5, \\ 6 \vee \bar{8} \vee b_6, \quad 2 \vee 4 \vee b_7, \quad \bar{4} \vee 5 \vee b_8, \quad 7 \vee 5 \vee b_9, \quad \bar{7} \vee 5 \vee b_{10}, \\ \bar{3} \vee b_{11}, \quad \bar{5} \vee 3 \vee b_{12} \}$$

- ▶  $L = 0, U = 12, \text{mid} = 6$        $\text{SAT}(\varphi \wedge \text{CNF}(\sum_{i=1}^m b_i \leq 6))?$       ✓
- ▶  $L = 0, U = 6, \text{mid} = 3$        $\text{SAT}(\varphi \wedge \text{CNF}(\sum_{i=1}^m b_i \leq 3))?$       ✓
- ▶  $L = 0, U = 3, \text{mid} = 1$        $\text{SAT}(\varphi \wedge \text{CNF}(\sum_{i=1}^m b_i \leq 1))?$       ✗

## Example

$$\varphi = \{ 6 \vee 2 \vee b_1, \quad \bar{6} \vee 2 \vee b_2, \quad \bar{2} \vee 1 \vee b_3, \quad \bar{1} \vee b_4, \quad \bar{6} \vee 8 \vee b_5, \\ 6 \vee \bar{8} \vee b_6, \quad 2 \vee 4 \vee b_7, \quad \bar{4} \vee 5 \vee b_8, \quad 7 \vee 5 \vee b_9, \quad \bar{7} \vee 5 \vee b_{10}, \\ \bar{3} \vee b_{11}, \quad \bar{5} \vee 3 \vee b_{12} \}$$

- |                          |  |   |
|--------------------------|--|---|
| ► L = 0, U = 12, mid = 6 | SAT( $\varphi \wedge \text{CNF}(\sum_{i=1}^m b_i \leq 6)$ )? | ✓ |
| ► L = 0, U = 6, mid = 3  | SAT( $\varphi \wedge \text{CNF}(\sum_{i=1}^m b_i \leq 3)$ )? | ✓ |
| ► L = 0, U = 3, mid = 1  | SAT( $\varphi \wedge \text{CNF}(\sum_{i=1}^m b_i \leq 1)$ )? | ✗ |
| ► L = 2, U = 3, mid = 2  | SAT( $\varphi \wedge \text{CNF}(\sum_{i=1}^m b_i \leq 2)$ )? | ✓ |

## Example

$$\varphi = \{ 6 \vee 2 \vee b_1, \quad \bar{6} \vee 2 \vee b_2, \quad \bar{2} \vee 1 \vee b_3, \quad \bar{1} \vee b_4, \quad \bar{6} \vee 8 \vee b_5, \\ 6 \vee \bar{8} \vee b_6, \quad 2 \vee 4 \vee b_7, \quad \bar{4} \vee 5 \vee b_8, \quad 7 \vee 5 \vee b_9, \quad \bar{7} \vee 5 \vee b_{10}, \\ \bar{3} \vee b_{11}, \quad \bar{5} \vee 3 \vee b_{12} \}$$

- |                          |  |   |
|--------------------------|--|---|
| ► L = 0, U = 12, mid = 6 | SAT( $\varphi \wedge \text{CNF}(\sum_{i=1}^m b_i \leq 6)$ )? | ✓ |
| ► L = 0, U = 6, mid = 3  | SAT( $\varphi \wedge \text{CNF}(\sum_{i=1}^m b_i \leq 3)$ )? | ✓ |
| ► L = 0, U = 3, mid = 1  | SAT( $\varphi \wedge \text{CNF}(\sum_{i=1}^m b_i \leq 1)$ )? | ✗ |
| ► L = 2, U = 3, mid = 2  | SAT( $\varphi \wedge \text{CNF}(\sum_{i=1}^m b_i \leq 2)$ )? | ✓ |
| ► L = 2, U = 2           | return 2   |   |

# Cardinality Constraints in Z3

```
from z3 import *

xs = [ Bool("x"+str(i)) for i in range (0,10)]
ys = [ Bool("y"+str(i)) for i in range (0,10)]

def card(ps):
    return sum([If(x, 1, 0) for x in ps])

solver = Solver()
solver.add(card(xs) == 5, card(ys) > 2, card(ys) <= 4)

if solver.check() == sat:
    model = solver.model()
    for i in range(0,10):
        print(xs[i], "=", model[xs[i]], ys[i], "=", model[ys[i]])
```



# MaxSAT in Z3

```
from z3 import *

vs = [Bool("v" + str(i)) for i in range(0,5)]
opt = Optimize() # like solver, but can maximize
# add hard constraints directly
opt.add(Or(Not(vs[2]), vs[3], vs[4]))
opt.add(Or(Not(vs[3]), vs[0]))
# now the soft constraints
c0 = Or(vs[2], vs[1])
c1 = Or(Not(vs[2]), vs[1])
c2 = Or(Not(vs[1]), vs[0])
c3 = Not(vs[0])
c4 = Or(Not(vs[3]), vs[1])
# build cost: If(c0,1,0) + If(c1, 1, 0) + If(c2, 1, 0) + ...
cost = sum([ If(c, 1, 0) for c in [c0, c1, c2, c3, c4] ])
opt.maximize(cost)
res = opt.check()
if res == z3.sat:
    model = opt.model() # get valuation
    print(model.eval(cost)) # number of satisfied clauses
    print(model) # assignment
```

# Application: Automotive Configuration (1)

## Manufacturer constraints on components

component family	components
engine	$E_1, E_2, E_3$
gearbox	$G_1, G_2, G_3$
control unit	$C_1, \dots, C_5$
dashboard	$D_1, \dots, D_4$
navigation system	$N_1, N_2, N_3$
air conditioner	$AC_1, AC_2, AC_3$
alarm system	$AS_1, AS_2$
radio	$R_1, \dots, R_5$

## Component families with limitations

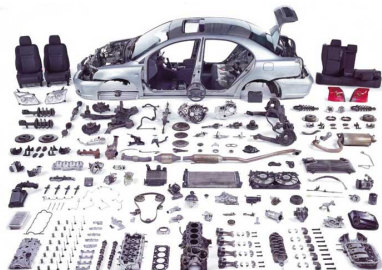


# Application: Automotive Configuration (1)

## Manufacturer constraints on components

component family	components limit
engine	$E_1, E_2, E_3 = 1$
gearbox	$G_1, G_2, G_3 = 1$
control unit	$C_1, \dots, C_5 = 1$
dashboard	$D_1, \dots, D_4 = 1$
navigation system	$N_1, N_2, N_3 \leq 1$
air conditioner	$AC_1, AC_2, AC_3 \leq 1$
alarm system	$AS_1, AS_2 \leq 1$
radio	$R_1, \dots, R_5 \leq 1$

## Component families with limitations



# Application: Automotive Configuration (1)

## Manufacturer constraints on components

component family	components limit
engine	$E_1, E_2, E_3 = 1$
gearbox	$G_1, G_2, G_3 = 1$
control unit	$C_1, \dots, C_5 = 1$
dashboard	$D_1, \dots, D_4 = 1$
navigation system	$N_1, N_2, N_3 \leq 1$
air conditioner	$AC_1, AC_2, AC_3 \leq 1$
alarm system	$AS_1, AS_2 \leq 1$
radio	$R_1, \dots, R_5 \leq 1$

## Component families with limitations

$G_1$	$\rightarrow E_1 \vee E_2$
$N_1 \vee N_2$	$\rightarrow D_1$
$N_3$	$\rightarrow D_2 \vee D_3$
$AC_1 \vee AC_3$	$\rightarrow D_1 \vee D_2$
$AS_1$	$\rightarrow D_2 \vee D_3$
$R_1 \vee R_2 \vee R_5$	$\rightarrow D_1 \vee D_4$

## Component dependencies

# Application: Automotive Configuration (1)

## Manufacturer constraints on components

component family	components limit
engine	$E_1, E_2, E_3 = 1$
gearbox	$G_1, G_2, G_3 = 1$
control unit	$C_1, \dots, C_5 = 1$
dashboard	$D_1, \dots, D_4 = 1$
navigation system	$N_1, N_2, N_3 \leq 1$
air conditioner	$AC_1, AC_2, AC_3 \leq 1$
alarm system	$AS_1, AS_2 \leq 1$
radio	$R_1, \dots, R_5 \leq 1$

## Component families with limitations

## Encoding

- ▶ for every component  $c$  use variable  $x_c$  which is assigned T iff  $c$  is used
- ▶ require limitations and dependencies  $\varphi_{car}$  by adding respective clauses

$G_1$	$\rightarrow E_1 \vee E_2$
$N_1 \vee N_2$	$\rightarrow D_1$
$N_3$	$\rightarrow D_2 \vee D_3$
$AC_1 \vee AC_3$	$\rightarrow D_1 \vee D_2$
$AS_1$	$\rightarrow D_2 \vee D_3$
$R_1 \vee R_2 \vee R_5$	$\rightarrow D_1 \vee D_4$

## Component dependencies

# Application: Automotive Configuration (1)

## Manufacturer constraints on components

component family	components limit
engine	$E_1, E_2, E_3 = 1$
gearbox	$G_1, G_2, G_3 = 1$
control unit	$C_1, \dots, C_5 = 1$
dashboard	$D_1, \dots, D_4 = 1$
navigation system	$N_1, N_2, N_3 \leq 1$
air conditioner	$AC_1, AC_2, AC_3 \leq 1$
alarm system	$AS_1, AS_2 \leq 1$
radio	$R_1, \dots, R_5 \leq 1$

### Component families with limitations

## Encoding

- ▶ for every component  $c$  use variable  $x_c$  which is assigned T iff  $c$  is used
- ▶ require limitations and dependencies  $\varphi_{\text{car}}$  by adding respective clauses

## Problem 1: Validity of configuration

- ▶ is desired configuration valid?

$G_1$	$\rightarrow E_1 \vee E_2$
$N_1 \vee N_2$	$\rightarrow D_1$
$N_3$	$\rightarrow D_2 \vee D_3$
$AC_1 \vee AC_3$	$\rightarrow D_1 \vee D_2$
$AS_1$	$\rightarrow D_2 \vee D_3$
$R_1 \vee R_2 \vee R_5$	$\rightarrow D_1 \vee D_4$

### Component dependencies

SAT encoding

# Application: Automotive Configuration (1)

## Manufacturer constraints on components

component family	components limit
engine	$E_1, E_2, E_3 = 1$
gearbox	$G_1, G_2, G_3 = 1$
control unit	$C_1, \dots, C_5 = 1$
dashboard	$D_1, \dots, D_4 = 1$
navigation system	$N_1, N_2, N_3 \leq 1$
air conditioner	$AC_1, AC_2, AC_3 \leq 1$
alarm system	$AS_1, AS_2 \leq 1$
radio	$R_1, \dots, R_5 \leq 1$

### Component families with limitations

## Encoding

- ▶ for every component  $c$  use variable  $x_c$  which is assigned T iff  $c$  is used
- ▶ require limitations and dependencies  $\varphi_{\text{car}}$  by adding respective clauses

## Problem 1: Validity of configuration

- ▶ is desired configuration valid?

e.g.  $E_1 \wedge G_1 \wedge C_5 \wedge (D_2 \vee D_3)$  ✓

$G_1$	$\rightarrow E_1 \vee E_2$
$N_1 \vee N_2$	$\rightarrow D_1$
$N_3$	$\rightarrow D_2 \vee D_3$
$AC_1 \vee AC_3$	$\rightarrow D_1 \vee D_2$
$AS_1$	$\rightarrow D_2 \vee D_3$
$R_1 \vee R_2 \vee R_5$	$\rightarrow D_1 \vee D_4$

### Component dependencies

$E_3 \wedge G_1 \wedge C_5 \wedge D_2 \vee AC_1$  ✗

SAT encoding

## Application: Automotive Configuration (2)

### Problem 2: Maximize number of desired components

- find maximal valid subset of configuration  $c_1, \dots, c_n$

partial maxSAT

$$\underbrace{\varphi_{\text{car}}}_{\text{hard clauses}} \wedge \underbrace{x_{c_1} \wedge \dots \wedge x_{c_n}}_{\text{soft clauses}}$$



## Application: Automotive Configuration (2)

### Problem 2: Maximize number of desired components

- find maximal valid subset of configuration  $c_1, \dots, c_n$

partial maxSAT

$$\underbrace{\varphi_{\text{car}}}_{\text{hard clauses}} \wedge \underbrace{x_{c_1} \wedge \dots \wedge x_{c_n}}_{\text{soft clauses}}$$

### Example

component family	choice	result
engine	$E_1$	$E_1$
gearbox	$G_3$	$G_3$
control unit	$C_2$	$C_2$
dashboard	$D_1 \vee D_3$	$D_1$
navigation system	$N_2$	$N_2$
air conditioner	$AC_1$	$AC_1$
alarm system	$AS_2$	—
radio	$R_5$	$R_5$

## Application: Automotive Configuration (2)

### Problem 2: Maximize number of desired components

- ▶ find maximal valid subset of configuration  $c_1, \dots, c_n$  partial maxSAT
- ▶ possibly with priorities  $p_i$  for component  $c_i$  weighted partial maxSAT

$$\underbrace{\varphi_{\text{car}}}_{\text{hard clauses}} \wedge \underbrace{(x_{c_1}, p_1) \wedge \dots \wedge (x_{c_n}, p_n)}_{\text{soft clauses}}$$

### Example

component family	choice	result
engine	$E_1$	$E_1$
gearbox	$G_3$	$G_3$
control unit	$C_2$	$C_2$
dashboard	$D_1 \vee D_3$	$D_1$
navigation system	$N_2$	$N_2$
air conditioner	$AC_1$	$AC_1$
alarm system	$AS_2$	—
radio	$R_5$	$R_5$

## Application: Automotive Configuration (2)

### Problem 2: Maximize number of desired components

- ▶ find maximal valid subset of configuration  $c_1, \dots, c_n$  partial maxSAT
- ▶ possibly with priorities  $p_i$  for component  $c_i$  weighted partial maxSAT

$$\underbrace{\varphi_{\text{car}}}_{\text{hard clauses}} \wedge \underbrace{(x_{c_1}, p_1) \wedge \dots \wedge (x_{c_n}, p_n)}_{\text{soft clauses}}$$

## Application: Automotive Configuration (2)

### Problem 2: Maximize number of desired components

- ▶ find maximal valid subset of configuration  $c_1, \dots, c_n$  partial maxSAT
- ▶ possibly with priorities  $p_i$  for component  $c_i$  weighted partial maxSAT

$$\underbrace{\varphi_{\text{car}}}_{\text{hard clauses}} \wedge \underbrace{(x_{c_1}, p_1) \wedge \dots \wedge (x_{c_n}, p_n)}_{\text{soft clauses}}$$

### Problem 3: Minimization of cost

- ▶ given cost  $q_i$  for each component  $c_i$ , find cheapest valid configuration weighted partial maxSAT

$$\underbrace{\varphi_{\text{car}}}_{\text{hard clauses}} \wedge \underbrace{(c_1, -q_1) \wedge \dots \wedge (c_n, -q_n)}_{\text{soft clauses}}$$

## Application: Automotive Configuration (2)

### Problem 2: Maximize number of desired components

- ▶ find maximal valid subset of configuration  $c_1, \dots, c_n$  partial maxSAT
- ▶ possibly with priorities  $p_i$  for component  $c_i$  weighted partial maxSAT

$$\underbrace{\varphi_{\text{car}}}_{\text{hard clauses}} \wedge \underbrace{(x_{c_1}, p_1) \wedge \dots \wedge (x_{c_n}, p_n)}_{\text{soft clauses}}$$

### Problem 3: Minimization of cost

- ▶ given cost  $q_i$  for each component  $c_i$ , find cheapest valid configuration weighted partial maxSAT

$$\underbrace{\varphi_{\text{car}}}_{\text{hard clauses}} \wedge \underbrace{(c_1, -q_1) \wedge \dots \wedge (c_n, -q_n)}_{\text{soft clauses}}$$

### Result

collaboration with BMW: evaluated on configuration formulas of 2013 product line

## Remark

maxSAT is not a decision problem

## Remark

maxSAT is not a decision problem

## Definition

$FP^{NP}$  is class of functions computable in polynomial time with access to NP oracle

## Remark

maxSAT is not a decision problem

## Definition

$FP^{NP}$  is class of functions computable in polynomial time with access to NP oracle

## Theorem

maxSAT is  $FP^{NP}$ -complete



## Remark

maxSAT is not a decision problem

## Definition

$FP^{NP}$  is class of functions computable in polynomial time with access to NP oracle

## Theorem

maxSAT is  $FP^{NP}$ -complete

## Remarks

- ▶  $FP^{NP}$  allows polynomial number of oracle calls (which is e.g. SAT solver)
- ▶ other members of  $FP^{NP}$ :  
optimization versions of travelling salesperson and Knapsack

# Outline

- Summary of Last Week
- Maximum Satisfiability
- Algorithms for Minimum Unsatisfiability
- Application: Automotive Configuration
- NP-Completeness

## Theorem

SAT is NP-complete.

(Cook 1971, Levin 1973)

# NP-Completeness

## Theorem

SAT is NP-complete.

(Cook 1971, Levin 1973)

## Proof.

- ▶ SAT is in NP

easy

# NP-Completeness

## Theorem

SAT is NP-complete.

(Cook 1971, Levin 1973)

## Proof.

► SAT is in NP

easy

► SAT is NP-hard

hard

# NP-Completeness

## Theorem

(Cook 1971, Levin 1973)

SAT is NP-complete.

## Proof.

- ▶ SAT is in NP
  - ▶ given  $\varphi$ , guess nondeterministically an assignment  $v$
  - ▶ can check whether  $v$  satisfies  $\varphi$  (in time linear in size of  $\varphi$ )
- ▶ SAT is NP-hard

easy

hard

# NP-Completeness

## Theorem

(Cook 1971, Levin 1973)

SAT is NP-complete.

## Proof.

- ▶ SAT is in NP easy
  - ▶ given  $\varphi$ , guess nondeterministically an assignment  $v$
  - ▶ can check whether  $v$  satisfies  $\varphi$  (in time linear in size of  $\varphi$ )
- ▶ SAT is NP-hard hard
  - ▶ show that any problem in NP can be **reduced** to a SAT problem

# NP-Completeness

## Theorem

(Cook 1971, Levin 1973)

SAT is NP-complete.

## Proof.

- ▶ SAT is in NP easy
  - ▶ given  $\varphi$ , guess nondeterministically an assignment  $v$
  - ▶ can check whether  $v$  satisfies  $\varphi$  (in time linear in size of  $\varphi$ )
- ▶ SAT is NP-hard hard
  - ▶ show that any problem in NP can be reduced to a SAT problem
  - ▶ more precisely:
    - ▶ given nondeterministic Turing machine  $\mathcal{N}$  and input  $w$  such that  $\mathcal{N}$  runs in polynomial time
    - ▶ construct formula  $\varphi$  such that

$$\mathcal{N} \text{ accepts } w \iff \varphi \text{ is satisfiable}$$



# Reminder: Turing Machines

## Definition

**Turing machine (TM)** is 8-tuple  $\mathcal{N} = (Q, \Sigma, \Gamma, \vdash, \sqsubset, \delta, s, t)$  with

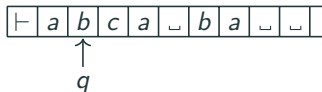
- ▶  $Q$ : finite set of states

# Reminder: Turing Machines

## Definition

**Turing machine (TM)** is 8-tuple  $\mathcal{N} = (Q, \Sigma, \Gamma, \vdash, \sqcup, \delta, s, t)$  with

- ▶  $Q$ : finite set of states
- ▶  $\Sigma$ : input alphabet

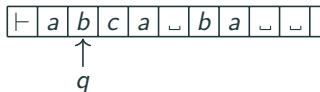


# Reminder: Turing Machines

## Definition

**Turing machine (TM)** is 8-tuple  $\mathcal{N} = (Q, \Sigma, \Gamma, \vdash, \sqcup, \delta, s, t)$  with

- ▶  $Q$ : finite set of states
- ▶  $\Sigma$ : input alphabet
- ▶  $\Gamma \supseteq \Sigma$ : **tape** alphabet

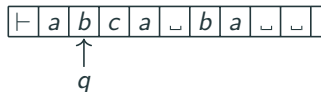


# Reminder: Turing Machines

## Definition

**Turing machine (TM)** is 8-tuple  $\mathcal{N} = (Q, \Sigma, \Gamma, \vdash, \sqcup, \delta, s, t)$  with

- ▶  $Q$ : finite set of states
- ▶  $\Sigma$ : input alphabet
- ▶  $\Gamma \supseteq \Sigma$ : tape alphabet
- ▶  $\vdash \in \Gamma - \Sigma$ : **left endmarker**

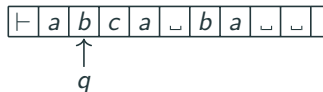


# Reminder: Turing Machines

## Definition

**Turing machine (TM)** is 8-tuple  $\mathcal{N} = (Q, \Sigma, \Gamma, \vdash, \sqcup, \delta, s, t)$  with

- ▶  $Q$ : finite set of states
- ▶  $\Sigma$ : input alphabet
- ▶  $\Gamma \supseteq \Sigma$ : tape alphabet
- ▶  $\vdash \in \Gamma - \Sigma$ : left endmarker
- ▶  $\sqcup \in \Gamma - \Sigma$ : blank symbol

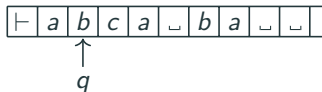


# Reminder: Turing Machines

## Definition

**Turing machine (TM)** is 8-tuple  $\mathcal{N} = (Q, \Sigma, \Gamma, \vdash, \sqcup, \delta, s, t)$  with

- ▶  $Q$ : finite set of states
- ▶  $\Sigma$ : input alphabet
- ▶  $\Gamma \supseteq \Sigma$ : tape alphabet
- ▶  $\vdash \in \Gamma - \Sigma$ : left endmarker
- ▶  $\sqcup \in \Gamma - \Sigma$ : blank symbol
- ▶  $\delta: Q \times \Gamma \rightarrow Q \times \Gamma \times \{L, R\}$ : transition function

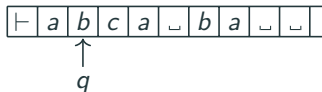


# Reminder: Turing Machines

## Definition

**Turing machine (TM)** is 8-tuple  $\mathcal{N} = (Q, \Sigma, \Gamma, \vdash, \sqcup, \delta, s, t)$  with

- ▶  $Q$ : finite set of states
- ▶  $\Sigma$ : input alphabet
- ▶  $\Gamma \supseteq \Sigma$ : tape alphabet
- ▶  $\vdash \in \Gamma - \Sigma$ : left endmarker
- ▶  $\sqcup \in \Gamma - \Sigma$ : blank symbol
- ▶  $\delta: Q \times \Gamma \rightarrow Q \times \Gamma \times \{L, R\}$ : transition function
- ▶  $s \in Q$ : start state

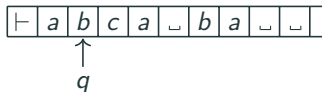


# Reminder: Turing Machines

## Definition

**Turing machine (TM)** is 8-tuple  $\mathcal{N} = (Q, \Sigma, \Gamma, \vdash, \sqcup, \delta, s, t)$  with

- ▶  $Q$ : finite set of states
- ▶  $\Sigma$ : input alphabet
- ▶  $\Gamma \supseteq \Sigma$ : tape alphabet
- ▶  $\vdash \in \Gamma - \Sigma$ : left endmarker
- ▶  $\sqcup \in \Gamma - \Sigma$ : blank symbol
- ▶  $\delta: Q \times \Gamma \rightarrow Q \times \Gamma \times \{L, R\}$ : transition function
- ▶  $s \in Q$ : start state
- ▶  $t \in Q$ : **accept** state



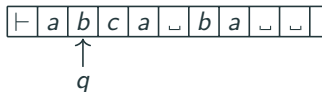


# Reminder: Turing Machines

## Definition

**Turing machine (TM)** is 8-tuple  $\mathcal{N} = (Q, \Sigma, \Gamma, \vdash, \sqcup, \delta, s, t)$  with

- ▶  $Q$ : finite set of states
- ▶  $\Sigma$ : input alphabet
- ▶  $\Gamma \supseteq \Sigma$ : tape alphabet
- ▶  $\vdash \in \Gamma - \Sigma$ : left endmarker
- ▶  $\sqcup \in \Gamma - \Sigma$ : blank symbol
- ▶  $\delta: Q \times \Gamma \rightarrow Q \times \Gamma \times \{L, R\}$ : transition function
- ▶  $s \in Q$ : start state
- ▶  $t \in Q$ : accept state



such that

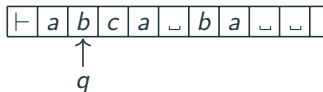
$$\forall a \in \Gamma \exists b, b' \in \Gamma \exists d, d' \in \{L, R\}: \delta(t, a) = (t, b, d)$$

# Reminder: Turing Machines

## Definition

**Turing machine (TM)** is 8-tuple  $\mathcal{N} = (Q, \Sigma, \Gamma, \vdash, \sqcup, \delta, s, t)$  with

- ▶  $Q$ : finite set of states
- ▶  $\Sigma$ : input alphabet
- ▶  $\Gamma \supseteq \Sigma$ : tape alphabet
- ▶  $\vdash \in \Gamma - \Sigma$ : left endmarker
- ▶  $\sqcup \in \Gamma - \Sigma$ : blank symbol
- ▶  $\delta: Q \times \Gamma \rightarrow Q \times \Gamma \times \{L, R\}$ : transition function
- ▶  $s \in Q$ : start state
- ▶  $t \in Q$ : accept state



such that

$$\forall a \in \Gamma \exists b, b' \in \Gamma \exists d, d' \in \{L, R\}: \delta(t, a) = (t, b, d)$$

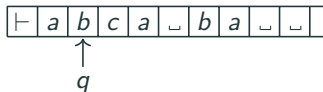
$$\forall p \in Q \exists q \in Q: \delta(p, \vdash) = (q, \vdash, R)$$

# Reminder: Turing Machines

## Definition

**Turing machine (TM)** is 8-tuple  $\mathcal{N} = (Q, \Sigma, \Gamma, \vdash, \sqcup, \delta, s, t)$  with

- ▶  $Q$ : finite set of states
- ▶  $\Sigma$ : input alphabet
- ▶  $\Gamma \supseteq \Sigma$ : tape alphabet
- ▶  $\vdash \in \Gamma - \Sigma$ : left endmarker
- ▶  $\sqcup \in \Gamma - \Sigma$ : blank symbol
- ▶  $\delta: Q \times \Gamma \rightarrow Q \times \Gamma \times \{L, R\}$ : transition function
- ▶  $s \in Q$ : start state
- ▶  $t \in Q$ : accept state



such that

$$\forall a \in \Gamma \exists b, b' \in \Gamma \exists d, d' \in \{L, R\}: \delta(t, a) = (t, b, d)$$

$$\forall p \in Q \exists q \in Q: \delta(p, \vdash) = (q, \vdash, R)$$

## Definition

$\mathcal{N}$  accepts  $w$  if there is accepting run  $(s, \vdash w, 0) \xrightarrow[\mathcal{N}]{}^* (t, \dots)$

## Example (Turing machine to recognize palindromes)



$\mathcal{N} = (\mathcal{Q}, \Sigma, \Gamma, \vdash, \sqcup, \delta, q_{init}, q_{acc})$  with

►  $\mathcal{Q} = \{q_{init}, q_{read0}, q_{read1}, q_{acc}, q_{search0}, q_{search1}, q_{back}\}$

## Example (Turing machine to recognize palindromes)



$\mathcal{N} = (\mathcal{Q}, \Sigma, \Gamma, \vdash, \sqcup, \delta, q_{init}, q_{acc})$  with

- ▶  $\mathcal{Q} = \{q_{init}, q_{read0}, q_{read1}, q_{acc}, q_{search0}, q_{search1}, q_{back}\}$
- ▶  $\Sigma = \{0, 1\}$

## Example (Turing machine to recognize palindromes)



$\mathcal{N} = (\mathcal{Q}, \Sigma, \Gamma, \vdash, \sqcup, \delta, q_{init}, q_{acc})$  with

- ▶  $\mathcal{Q} = \{q_{init}, q_{read0}, q_{read1}, q_{acc}, q_{search0}, q_{search1}, q_{back}\}$
- ▶  $\Sigma = \{0, 1\}$
- ▶  $\Gamma = \{0, 1, \vdash, \sqcup\}$

## Example (Turing machine to recognize palindromes)



$\mathcal{N} = (\mathcal{Q}, \Sigma, \Gamma, \vdash, \sqcup, \delta, q_{init}, q_{acc})$  with

- ▶  $\mathcal{Q} = \{q_{init}, q_{read0}, q_{read1}, q_{acc}, q_{search0}, q_{search1}, q_{back}\}$
- ▶  $\Sigma = \{0, 1\}$
- ▶  $\Gamma = \{0, 1, \vdash, \sqcup\}$
- ▶ start state  $q_{init}$ , accept state  $q_{acc}$

## Example (Turing machine to recognize palindromes)



$\mathcal{N} = (\mathcal{Q}, \Sigma, \Gamma, \vdash, \sqcup, \delta, q_{init}, q_{acc})$  with

- ▶  $\mathcal{Q} = \{q_{init}, q_{read0}, q_{read1}, q_{acc}, q_{search0}, q_{search1}, q_{back}\}$
- ▶  $\Sigma = \{0, 1\}$
- ▶  $\Gamma = \{0, 1, \vdash, \sqcup\}$
- ▶ start state  $q_{init}$ , accept state  $q_{acc}$

$\delta$	$\vdash$	0	1	$\sqcup$
$q_{init}$	$(q_{init}, \vdash, R)$	$(q_{read0}, \vdash, R)$	$(q_{read1}, \vdash, R)$	$(q_{acc}, \sqcup, R)$
$q_{read0}$		$(q_{read0}, 0, R)$	$(q_{read0}, 1, R)$	$(q_{search0}, \sqcup, L)$
$q_{read1}$		$(q_{read1}, 0, R)$	$(q_{read1}, 1, R)$	$(q_{search1}, \sqcup, L)$
$q_{search0}$	$(q_{acc}, \vdash, R)$	$(q_{back}, \sqcup, L)$		
$q_{search1}$	$(q_{acc}, \vdash, R)$		$(q_{back}, \sqcup, L)$	
$q_{back}$	$(q_{init}, \vdash, R)$	$(q_{back}, 0, L)$	$(q_{back}, 1, L)$	



## Proof: SAT is NP hard

- ▶ given nondeterministic Turing machine  $\mathcal{N}$  running in polynomial time
- ▶ i.e. there is some polynomial  $p(n)$  such that for any input  $w$  of size  $n$ ,  $\mathcal{N}$  needs at most  $p(n)$  steps

## Proof: SAT is NP hard

- ▶ given nondeterministic Turing machine  $\mathcal{N}$  running in polynomial time
- ▶ i.e. there is some polynomial  $p(n)$  such that for any input  $w$  of size  $n$ ,  $\mathcal{N}$  needs at most  $p(n)$  steps
- ▶ in  $p(n)$  steps,  $\mathcal{N}$  can write at most  $p(n)$  tape cells

## Proof: SAT is NP hard

- ▶ given nondeterministic Turing machine  $\mathcal{N}$  running in polynomial time
- ▶ i.e. there is some polynomial  $p(n)$  such that for any input  $w$  of size  $n$ ,  $\mathcal{N}$  needs at most  $p(n)$  steps
- ▶ in  $p(n)$  steps,  $\mathcal{N}$  can write at most  $p(n)$  tape cells
- ▶ represent run of  $\mathcal{N}$  as **computation table** of size  $(p(n) + 1) \times (p(n) + 1)$

## Proof: SAT is NP hard

- ▶ given nondeterministic Turing machine  $\mathcal{N}$  running in polynomial time
- ▶ i.e. there is some polynomial  $p(n)$  such that for any input  $w$  of size  $n$ ,  $\mathcal{N}$  needs at most  $p(n)$  steps
- ▶ in  $p(n)$  steps,  $\mathcal{N}$  can write at most  $p(n)$  tape cells
- ▶ represent run of  $\mathcal{N}$  as computation table of size  $(p(n) + 1) \times (p(n) + 1)$ 
  - ▶ every cell contains a symbol in  $\Gamma$
  - ▶ the first row represents the initial configuration
  - ▶ all other rows are configuration that follows from the previous one

## Example (TM $\mathcal{N}$ for palindromes)

- ▶ needs at most  $p(n) = (n+1)(n+2)/2 + 1$  steps on input of length  $n$

## Example (TM $\mathcal{N}$ for palindromes)

- needs at most  $p(n) = (n+1)(n+2)/2 + 1$  steps on input of length  $n$
- for input 010, have computation table

$q_{init}$	┐	0	1	0	┐	┐	┐	┐	┐	┐
$q_{init}$	┐	0	1	0	┐	┐	┐	┐	┐	┐
$q_{read0}$	┐	┐	1	0	┐	┐	┐	┐	┐	┐
$q_{read0}$	┐	┐	1	0	┐	┐	┐	┐	┐	┐
$q_{read0}$	┐	┐	1	0	┐	┐	┐	┐	┐	┐
$q_{search0}$	┐	┐	1	0	┐	┐	┐	┐	┐	┐
$q_{back}$	┐	┐	1	┐	┐	┐	┐	┐	┐	┐
$q_{back}$	┐	┐	1	┐	┐	┐	┐	┐	┐	┐
$q_{init}$	┐	┐	1	┐	┐	┐	┐	┐	┐	┐
$q_{search1}$	┐	┐	┐	┐	┐	┐	┐	┐	┐	┐
$q_{search1}$	┐	┐	┐	┐	┐	┐	┐	┐	┐	┐
$q_{acc}$	┐	┐	┐	┐	┐	┐	┐	┐	┐	┐

## Proof: SAT is NP hard

- ▶ given nondeterministic Turing machine  $\mathcal{N}$  running in polynomial time
- ▶ i.e. there is some polynomial  $p(n)$  such that for any input  $w$  of size  $n$ ,  $\mathcal{N}$  needs at most  $p(n)$  steps
- ▶ in  $p(n)$  steps,  $\mathcal{N}$  can write at most  $p(n)$  tape cells
- ▶ represent run of  $\mathcal{N}$  as computation table of size  $(p(n) + 1) \times (p(n) + 1)$ 
  - ▶ every cell contains a symbol in  $\Gamma$
  - ▶ the first row represents the initial configuration
  - ▶ all other rows are configuration that follows from the previous one
- ▶ encode in huge (but polynomial-size) formula that table models accepting run

## Proof: SAT is NP hard

- ▶ given nondeterministic Turing machine  $\mathcal{N}$  running in polynomial time
- ▶ i.e. there is some polynomial  $p(n)$  such that for any input  $w$  of size  $n$ ,  $\mathcal{N}$  needs at most  $p(n)$  steps
- ▶ in  $p(n)$  steps,  $\mathcal{N}$  can write at most  $p(n)$  tape cells
- ▶ represent run of  $\mathcal{N}$  as computation table of size  $(p(n) + 1) \times (p(n) + 1)$ 
  - ▶ every cell contains a symbol in  $\Gamma$
  - ▶ the first row represents the initial configuration
  - ▶ all other rows are configuration that follows from the previous one
- ▶ encode in huge (but polynomial-size) formula that table models accepting run

### Encoding: Variables

$T_{i,j,s}$      $0 \leq i, j \leq p(n), s \in \Gamma$     in  $i$ th configuration,  $j$ th symbol on tape is  $s$



# Proof: SAT is NP hard

- ▶ given nondeterministic Turing machine  $\mathcal{N}$  running in polynomial time
- ▶ i.e. there is some polynomial  $p(n)$  such that for any input  $w$  of size  $n$ ,  $\mathcal{N}$  needs at most  $p(n)$  steps
- ▶ in  $p(n)$  steps,  $\mathcal{N}$  can write at most  $p(n)$  tape cells
- ▶ represent run of  $\mathcal{N}$  as computation table of size  $(p(n) + 1) \times (p(n) + 1)$ 
  - ▶ every cell contains a symbol in  $\Gamma$
  - ▶ the first row represents the initial configuration
  - ▶ all other rows are configuration that follows from the previous one
- ▶ encode in huge (but polynomial-size) formula that table models accepting run

## Encoding: Variables

$T_{i,j,s}$      $0 \leq i, j \leq p(n), s \in \Gamma$     in  $i$ th configuration,  $j$ th symbol on tape is  $s$

$H_{i,j}$      $0 \leq i, j \leq p(n)$     in  $i$ th configuration, read head is at position  $j$

# Proof: SAT is NP hard

- ▶ given nondeterministic Turing machine  $\mathcal{N}$  running in polynomial time
- ▶ i.e. there is some polynomial  $p(n)$  such that for any input  $w$  of size  $n$ ,  $\mathcal{N}$  needs at most  $p(n)$  steps
- ▶ in  $p(n)$  steps,  $\mathcal{N}$  can write at most  $p(n)$  tape cells
- ▶ represent run of  $\mathcal{N}$  as computation table of size  $(p(n) + 1) \times (p(n) + 1)$ 
  - ▶ every cell contains a symbol in  $\Gamma$
  - ▶ the first row represents the initial configuration
  - ▶ all other rows are configuration that follows from the previous one
- ▶ encode in huge (but polynomial-size) formula that table models accepting run

## Encoding: Variables

$T_{i,j,s}$	$0 \leq i, j \leq p(n), s \in \Gamma$	in $i$ th configuration, $j$ th symbol on tape is $s$
$H_{i,j}$	$0 \leq i, j \leq p(n)$	in $i$ th configuration, read head is at position $j$
$Q_{i,q}$	$0 \leq i \leq p(n), q \in Q$	state is $q$ in $i$ th configuration

# Proof: SAT is NP hard

- ▶ given nondeterministic Turing machine  $\mathcal{N}$  running in polynomial time
- ▶ i.e. there is some polynomial  $p(n)$  such that for any input  $w$  of size  $n$ ,  $\mathcal{N}$  needs at most  $p(n)$  steps
- ▶ in  $p(n)$  steps,  $\mathcal{N}$  can write at most  $p(n)$  tape cells
- ▶ represent run of  $\mathcal{N}$  as computation table of size  $(p(n) + 1) \times (p(n) + 1)$ 
  - ▶ every cell contains a symbol in  $\Gamma$
  - ▶ the first row represents the initial configuration
  - ▶ all other rows are configuration that follows from the previous one
- ▶ encode in huge (but polynomial-size) formula that table models accepting run

## Encoding: Variables

how many?

$T_{i,j,s}$	$0 \leq i, j \leq p(n), s \in \Gamma$	in $i$ th configuration, $j$ th symbol on tape is $s$	$\mathcal{O}(p(n)^2)$
$H_{i,j}$	$0 \leq i, j \leq p(n)$	in $i$ th configuration, read head is at position $j$	$\mathcal{O}(p(n)^2)$
$Q_{i,q}$	$0 \leq i \leq p(n), q \in \mathcal{Q}$	state is $q$ in $i$ th configuration	$\mathcal{O}(p(n))$

## Encoding: Constraints (1)

- ▶ initial state of TM is  $q_{init}$ , initial head position is 0

$\mathcal{O}(1)$

$$Q_{0,q_{init}} \wedge H_{0,0}$$

## Encoding: Constraints (1)

- ▶ initial state of TM is  $q_{init}$ , initial head position is 0

$\mathcal{O}(1)$

$$Q_{0,q_{init}} \wedge H_{0,0}$$

- ▶ initial tape content is  $w$

$\mathcal{O}(p(n))$

$$T_{0,0,\vdash} \wedge \bigwedge_{1 \leq j \leq n} T_{0,j,w_j} \wedge \bigwedge_{n < j \leq p(n)} T_{0,j,\sqcup}$$

## Encoding: Constraints (1)

- ▶ initial state of TM is  $q_{init}$ , initial head position is 0

$\mathcal{O}(1)$

$$Q_{0,q_{init}} \wedge H_{0,0}$$

- ▶ initial tape content is  $w$

$\mathcal{O}(p(n))$

$$T_{0,0,\vdash} \wedge \bigwedge_{1 \leq j \leq n} T_{0,j,w_j} \wedge \bigwedge_{n < j \leq p(n)} T_{0,j,\sqcup}$$

- ▶ at least one symbol in every tape cell in every configuration

$\mathcal{O}(p(n)^2)$

$$\bigwedge_{0 \leq i,j \leq p(n)} \bigvee_{s \in \Gamma} T_{i,j,s}$$

## Encoding: Constraints (1)

- ▶ initial state of TM is  $q_{init}$ , initial head position is 0  $\mathcal{O}(1)$   
$$Q_{0,q_{init}} \wedge H_{0,0}$$
- ▶ initial tape content is  $w$   $\mathcal{O}(p(n))$   
$$T_{0,0,\vdash} \wedge \bigwedge_{1 \leq j \leq n} T_{0,j,w_j} \wedge \bigwedge_{n < j \leq p(n)} T_{0,j,\sqcup}$$
- ▶ at least one symbol in every tape cell in every configuration  $\mathcal{O}(p(n)^2)$   
$$\bigwedge_{0 \leq i,j \leq p(n)} \bigvee_{s \in \Gamma} T_{i,j,s}$$
- ▶ at most one symbol in every tape cell in every configuration  $\mathcal{O}(p(n)^2)$   
$$\bigwedge_{0 \leq i,j \leq p(n)} \bigwedge_{s \neq s' \in \Gamma} \neg T_{i,j,s} \vee \neg T_{i,j,s'}$$

## Encoding: Constraints (1)

- ▶ initial state of TM is  $q_{init}$ , initial head position is 0  $\mathcal{O}(1)$

$$Q_{0,q_{init}} \wedge H_{0,0}$$

- ▶ initial tape content is  $w$   $\mathcal{O}(p(n))$

$$T_{0,0,\vdash} \wedge \bigwedge_{1 \leq j \leq n} T_{0,j,w_j} \wedge \bigwedge_{n < j \leq p(n)} T_{0,j,\sqcup}$$

- ▶ at least one symbol in every tape cell in every configuration  $\mathcal{O}(p(n)^2)$

$$\bigwedge_{0 \leq i,j \leq p(n)} \bigvee_{s \in \Gamma} T_{i,j,s}$$

- ▶ at most one symbol in every tape cell in every configuration  $\mathcal{O}(p(n)^2)$

$$\bigwedge_{0 \leq i,j \leq p(n)} \bigwedge_{s \neq s' \in \Gamma} \neg T_{i,j,s} \vee \neg T_{i,j,s'}$$

- ▶ at most one state at a time  $\mathcal{O}(p(n))$

$$\bigwedge_{0 \leq i,j \leq p(n)} \bigwedge_{q \neq q' \in Q} \neg Q_{i,q} \vee \neg Q_{i,q'}$$



## Encoding: Constraints (1)

- ▶ initial state of TM is  $q_{init}$ , initial head position is 0  $\mathcal{O}(1)$

$$Q_{0,q_{init}} \wedge H_{0,0}$$
- ▶ initial tape content is  $w$   $\mathcal{O}(p(n))$

$$T_{0,0,\vdash} \wedge \bigwedge_{1 \leq j \leq n} T_{0,j,w_j} \wedge \bigwedge_{n < j \leq p(n)} T_{0,j,\sqcup}$$
- ▶ at least one symbol in every tape cell in every configuration  $\mathcal{O}(p(n)^2)$

$$\bigwedge_{0 \leq i,j \leq p(n)} \bigvee_{s \in \Gamma} T_{i,j,s}$$
- ▶ at most one symbol in every tape cell in every configuration  $\mathcal{O}(p(n)^2)$

$$\bigwedge_{0 \leq i,j \leq p(n)} \bigwedge_{s \neq s' \in \Gamma} \neg T_{i,j,s} \vee \neg T_{i,j,s'}$$
- ▶ at most one state at a time  $\mathcal{O}(p(n))$

$$\bigwedge_{0 \leq i,j \leq p(n)} \bigwedge_{q \neq q' \in Q} \neg Q_{i,q} \vee \neg Q_{i,q'}$$
- ▶ read head is in at most one position at a time  $\mathcal{O}(p(n)^3)$

$$\bigwedge_{0 \leq i \leq p(n)} \bigwedge_{0 \leq j < j' \leq p(n)} \neg H_{i,j} \vee \neg H_{i,j'}$$

## Encoding: Constraints (2)

- ▶ possible transitions\*

$\mathcal{O}(p(n)^2)$

$$\begin{aligned} \bigwedge_{0 \leq i, j \leq p(n)} \bigwedge_{q \in Q} \bigwedge_{s \in \Gamma} (H_{i,j} \wedge Q_{i,q} \wedge T_{i,j,s}) \rightarrow \\ \bigvee_{(q', s', L) \in \delta(q, s)} (H_{i+1, j-1} \wedge Q_{i+1, q'} \wedge T_{i+1, j, s'}) \vee \\ \bigvee_{(q', s', R) \in \delta(q, s)} (H_{i+1, j+1} \wedge Q_{i+1, q'} \wedge T_{i+1, j+1, s'}) \end{aligned}$$

## Encoding: Constraints (2)

- ▶ possible transitions\*

$\mathcal{O}(p(n)^2)$

$$\begin{aligned} \bigwedge_{0 \leq i, j \leq p(n)} \bigwedge_{q \in Q} \bigwedge_{s \in \Gamma} (H_{i,j} \wedge Q_{i,q} \wedge T_{i,j,s}) \rightarrow \\ \bigvee_{(q', s', L) \in \delta(q, s)} (H_{i+1, j-1} \wedge Q_{i+1, q'} \wedge T_{i+1, j, s'}) \vee \\ \bigvee_{(q', s', R) \in \delta(q, s)} (H_{i+1, j+1} \wedge Q_{i+1, q'} \wedge T_{i+1, j+1, s'}) \end{aligned}$$

\* needs some adjustments for  $j = 0$  and  $j = p(n)$

## Encoding: Constraints (2)

- ▶ possible transitions\*

$\mathcal{O}(p(n)^2)$

$$\begin{aligned} \bigwedge_{0 \leq i, j \leq p(n)} \bigwedge_{q \in Q} \bigwedge_{s \in \Gamma} (H_{i,j} \wedge Q_{i,q} \wedge T_{i,j,s}) \rightarrow \\ \bigvee_{(q', s', L) \in \delta(q, s)} (H_{i+1, j-1} \wedge Q_{i+1, q'} \wedge T_{i+1, j, s'}) \vee \\ \bigvee_{(q', s', R) \in \delta(q, s)} (H_{i+1, j+1} \wedge Q_{i+1, q'} \wedge T_{i+1, j+1, s'}) \end{aligned}$$

\* needs some adjustments for  $j = 0$  and  $j = p(n)$

- ▶ at some point accepting state  $q_{acc}$  is reached

$\mathcal{O}(p(n)^2)$

$$\bigwedge_{0 \leq i \leq p(n)} Q_{i, q_{acc}}$$

## Encoding: Constraints (2)

- ▶ possible transitions\*

$\mathcal{O}(p(n)^2)$

$$\bigwedge_{0 \leq i, j \leq p(n)} \bigwedge_{q \in Q} \bigwedge_{s \in \Gamma} (H_{i,j} \wedge Q_{i,q} \wedge T_{i,j,s}) \rightarrow \\ \bigvee_{(q', s', L) \in \delta(q, s)} (H_{i+1, j-1} \wedge Q_{i+1, q'} \wedge T_{i+1, j, s'}) \vee \\ \bigvee_{(q', s', R) \in \delta(q, s)} (H_{i+1, j+1} \wedge Q_{i+1, q'} \wedge T_{i+1, j+1, s'})$$

\* needs some adjustments for  $j = 0$  and  $j = p(n)$

- ▶ at some point accepting state  $q_{acc}$  is reached

$\mathcal{O}(p(n)^2)$

$$\bigwedge_{0 \leq i \leq p(n)} Q_{i, q_{acc}}$$

## Conclusion

- ▶ conjunction of constraints  $\varphi$  is satisfiable iff  $\mathcal{N}$  admits accepting run on  $w$
- ▶ size of  $\varphi$  is polynomial in  $n$
- ▶ so problem in NP reduced to SAT





Rouven Walter, Christoph Zengler and Wolfgang Küchlin.

**Applications of MaxSAT in Automotive Configuration.**

Proc. International Configuration Workshop 2013, pp. 21-28, 2013.



André Abramé and Djamal Habet.

**ahmaxsat: Description and Evaluation of a Branch and Bound Max-SAT Solver.**

Journal on Satisfiability, Boolean Modeling and Computation 9, pp. 89–128, 2015.



Chu-Min Li and Felip Manyà.

**MaxSAT, hard and soft constraints.**

In: Handbook of Satisfiability, IOS Press, pp. 613–631, 2009.



Zhaohui Fu and Sharad Malik.

**On solving the partial MAX-SAT problem.**

In Proc. Theory and Applications of Satisfiability Testing, pp. 252–265, 2006