

# CS 433 Automated Reasoning 2024

## Lecture 10: Encoding into SAT problem

Instructor: Ashutosh Gupta

IITB India

Compile date: 2024-02-08

## Topic 10.1

# Understanding encoding in SAT

# SAT encoding

Since SAT is a NP-complete problem, therefore any NP-complete problem can be encoded into SAT in polynomial size.

Therefore, we can **solve hard problems** using SAT solvers.

We will look into a few interesting examples.

Objective of an encoding.

- ▶ Compact encoding (linear if possible)
- ▶ Redundant clauses may help the solver
- ▶ Encoding should be “compatible” with CDCL

## Encoding into CNF

CNF is the form of choice

- ▶ Most problems specify collection of restrictions on solutions
- ▶ Each restriction is usually of the form

if-this  $\Rightarrow$  then-this

The above constraints are naturally in CNF.

“Even if the system has hundreds and thousands of formulas, it can be put into CNF **piece by piece** without any **multiplying out**”

– Martin Davis and Hilary Putnam

### Exercise 10.1

*Which of the following two encodings of  $\text{ite}(p, q, r)$  is/are in CNF?*

1.  $(p \wedge q) \vee (\neg p \wedge r)$
2.  $(p \Rightarrow q) \wedge (\neg p \Rightarrow r)$

# Graph Coloring

## Problem:

color a graph  $(\{v_1, \dots, v_n\}, E)$  with at most  $d$  colors such that if  $(v_i, v_j) \in E$  then the colors of  $v_i$  and  $v_j$  are different.

## SAT encoding

Variables:  $p_{ij}$  for  $i \in 1..n$  and  $j \in 1..d$ .  $p_{ij}$  is true iff  $v_i$  is assigned  $j$ th color.

Clauses:

- ▶ Each vertex has at least one color

$$\text{for each } i \in 1..n \quad (p_{i1} \vee \dots \vee p_{id})$$

- ▶ if  $(v_i, v_j) \in E$  then color of  $v_i$  is different from  $v_j$ .

$$(\neg p_{ik} \vee \neg p_{jk}) \quad \text{for each } k \in 1..d, \quad (v_i, v_j) \in E$$

## Exercise 10.2

a. Encode: "every vertex has at most one color."

b. Do we need this constraint to solve the problem?

## Pigeon hole principle

### Prove:

if we place  $n + 1$  pigeons in  $n$  holes then there is a hole with at least 2 pigeons

The theorem holds true for any  $n$ , but we can prove it for a **fixed**  $n$ .

### SAT encoding

Variables:  $p_{ij}$  for  $i \in 0..n$  and  $j \in 1..n$ .  $p_{ij}$  is true iff pigeon  $i$  sits in hole  $j$ .

Clauses:

- ▶ Each pigeon sits in at least one hole

$$\text{for each } i \in 0..n \quad (p_{i1} \vee \dots \vee p_{in})$$

- ▶ There is at most one pigeon in each hole.

$$(\neg p_{ik} \vee \neg p_{jk}) \quad \text{for each } k \in 1..n, \quad i < j \in 0..n$$

## Topic 10.2

### Cardinality constraints

## Cardinality constraints

$$p_1 + \dots + p_n \bowtie k$$

where  $\bowtie \in \{<, >, \leq, \geq, =, \neq\}$

Cardinality constraints occur in various contexts. For example, graph coloring.

Let us study efficient encodings for them.



## Encoding $p_1 + \dots + p_n = 1$

- ▶ At least one of  $p_i$ s are true

$$(p_1 \vee \dots \vee p_n)$$

- ▶ Not more than one  $p_i$ s are true

$$(\neg p_i \vee \neg p_j) \quad i, j \in \{1, \dots, n\}$$

### Exercise 10.3

- What is the complexity of at least one constraints?*
- What is the complexity of at most one constraints?*

## Sequential encoding of $p_1 + .. + p_n \leq 1$

The earlier encoding of at most one is **quadratic**. We can do better by introducing fresh variables.

Let  $s_i$  be a fresh variable to indicate that  $p_1 + \dots + p_i$  has reached 1.

The following constraints encode  $p_1 + .. + p_n \leq 1$ .

$$\begin{aligned} & (p_1 \Rightarrow s_1) \quad \wedge \\ & \bigwedge_{1 < i < n} ( (p_i \vee s_{i-1}) \Rightarrow s_i ) \quad \wedge \quad (s_{i-1} \Rightarrow \neg p_i) \quad ) \\ & \wedge \quad (s_{n-1} \Rightarrow \neg p_n) \end{aligned}$$

If  $p_i = 1$ , for each  $j \geq i$ ,  $s_j = 1$ .

If already seen a one, no more ones.

### Exercise 10.4

- Give a satisfying assignment when  $p_3 = 1$  and all other  $p_i$ s are 0.
- Give a satisfying assignment of  $s_i$ s when all  $p_i$ s are 0.
- Why use **strict** upper bound ( $< n$ ) in the iterative conjunction? What if we use non-strict?
- Convert the constraints into CNF.

## Encoding $p_1 + \dots + p_n \leq k$

There are several encodings

- ▶ Generalized pairwise
- ▶ Sequential counter
- ▶ Operational encoding
- ▶ Sorting networks
- ▶ Cardinality networks

### Exercise 10.5

*Given the above encodings, how to encode  $p_1 + \dots + p_n \geq k$ ?*

## Generalized pairwise encoding for $p_1 + \dots + p_n \leq k$

No  $k + 1$  variables must be true at the same time.

For each  $i_1, \dots, i_{k+1} \in 1..n$ , we add the following clause

$$(\neg p_{i_1} \vee \dots \vee \neg p_{i_{k+1}})$$

### Exercise 10.6

*How many clauses are added for the encoding?*

# Sequential counter encoding for $p_1 + \dots + p_n \leq k$

**Commentary:**  $s_{ij}$  does not exactly encode as suggested. If  $s_{ij}$  is false then  $p_1 + \dots + p_j < i$ . Otherwise, the encoding does not guarantee that  $p_1 + \dots + p_j \geq i$ .

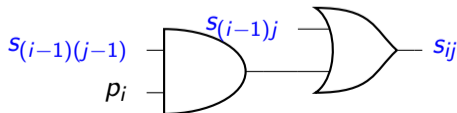
Let variable  $s_{ij}$  indicate that the sum upto  $p_i$  has reached to  $j$  or not.

- ▶ Constraints for first variable  $p_1$

$$(p_1 \Rightarrow s_{11}) \wedge \bigwedge_{j \in [2, k]} \neg s_{1j}$$

- ▶ Constraints for  $p_i$ , where  $i > 1$

$$((p_i \vee s_{(i-1)1}) \Rightarrow s_{i1}) \wedge \bigwedge_{j \in [2, k]} \underbrace{(((p_i \wedge s_{(i-1)(j-1)}) \vee s_{(i-1)j}) \Rightarrow s_{ij})}_{\text{add } +1}$$



## Sequential counter encoding for $p_1 + \dots + p_n \leq k$ (II)

- ▶ More constraints for  $p_i$ , if the sum has reached to  $k$  at  $i - 1$ , no more ones

$$(s_{(i-1)k} \Rightarrow \neg p_i)$$

### Exercise 10.7

- What is the variable/clause complexity?
- What if we drop constraints  $\bigwedge_{j \in [2, k]} \neg s_{1j}$ ?

## Topic 10.3

More problems

# Solving Sudoku using SAT solvers

## Example 10.1

4	2	6	5	7	1	3	9	8
8	5	7	2	9	3	1	4	6
1	3	9	4	6	8	2	7	5
9	7	1	3	8	5	6	2	4
5	4	3	7	2	6	8	1	9
6	8	2	1	4	9	7	5	3
7	9	4	6	3	2	5	8	1
2	6	5	8	1	4	9	3	7
3	1	8	9	5	7	4	6	2

Sudoku

► Variables:

$v_{i,j,k} \in \mathcal{B}$  where  $i, j, k \in [1, 9]$

- $v_{i,j,k} = 1$ , if  
column  $i$  and row  $j$  contains  $k$ .

► Value in each cell is valid:

$$\sum_{k=1}^9 v_{i,j,k} = 1 \quad i, j \in \{1, \dots, 9\}$$

► Each value used exactly once in each row:

$$\sum_{i=1}^9 v_{i,j,k} = 1 \quad j, k \in \{1, \dots, 9\}$$

► Each value used exactly once in each column:

$$\sum_{j=1}^9 v_{i,j,k} = 1 \quad i, k \in \{1, \dots, 9\}$$

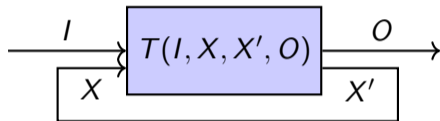
► Each value used exactly once in each  $3 \times 3$  grid

$$\sum_{s=1}^3 \sum_{r=1}^3 v_{3i+r, 3j+s, k} = 1 \quad i, j \in \{0, 1, 2\}, k \in \{1, \dots, 9\}$$



## Bounded model checking

Consider a Mealy machine



- ▶  $I$  is a vector of variables representing input
- ▶  $O$  is a vector of variables representing output
- ▶  $X$  is a vector of variables representing current state
- ▶  $X'$  is a vector of variables representing next state

Prove: After  $n$  steps, the machines always produces output  $O$  that satisfies some formula  $F(O)$ .

# Bounded model checking encoding

SAT encoding:

Variables:

- ▶  $I_0, \dots, I_{n-1}$  representing input at every step
- ▶  $O_1, \dots, O_n$  representing output at every step
- ▶  $X_0, \dots, X_n$  representing internal state at every step

Clauses:

- ▶ Encoding system runs:  $T(I_0, X_0, X_1, O_1) \wedge \dots \wedge T(I_{n-1}, X_{n-1}, X_n, O_n)$
- ▶ Encoding property:  $\neg F(O_1, \dots, O_n)$

If the encoding is unsat the property holds.

## Example: bounded model checking

### Example 10.2

Consider the following 2-Bit counter with two bits  $p$  and  $q$ .

$$p' := \neg p$$

$$q' := p \vee \neg q$$

where  $p'$  and  $q'$  are the next value for the bits. How many steps the above counter counts?

Let us suppose if we claim that that it is a mod 3 counter (may not be in the order of 00,01,11). We can *use a SAT solver* to find it out.

We can construct the following constraints to encode a single transition.

$$T(p', q', p, q) \triangleq (p' \Leftrightarrow \neg p) \wedge (q' \Leftrightarrow (p \vee \neg q))$$

## Example: bounded model checking

We encode the three step execution of the counter as follows.

$$Trs = T(p_3, q_3, p_2, q_2) \wedge T(p_2, q_2, p_1, q_1) \wedge T(p_1, q_1, p_0, q_0)$$

$p_i$ s and  $q_i$ s are fresh names to encode the intermediate states. If we expand  $T$  in  $Trs$ , we obtain.

$$(p_3 \Leftrightarrow \neg p_2) \wedge (q_3 \Leftrightarrow (p_2 \vee \neg q_2)) \wedge (p_2 \Leftrightarrow \neg p_1) \wedge (q_2 \Leftrightarrow (p_1 \vee \neg q_1)) \wedge (p_1 \Leftrightarrow \neg p_0) \wedge (q_1 \Leftrightarrow (p_0 \vee \neg q_0))$$

Property: distinct values for the intermediate steps and finally repeat the first value.

$$F = \underbrace{((p_0 \oplus p_1) \vee (q_0 \oplus q_1)) \wedge ((p_0 \oplus p_2) \vee (q_0 \oplus q_2)) \wedge ((p_2 \oplus p_1) \vee (q_2 \oplus q_1))}_{\text{distinct intermediate values}} \wedge \underbrace{((p_0 \Leftrightarrow p_3) \vee (q_0 \Leftrightarrow q_3))}_{\text{repeat}}$$

SAT solver can check satisfiability of

$$Trs \wedge \neg F$$

## Topic 10.4

### Input Format

# DIMACS Input format

## Example 10.3

*Input CNF*

```
c
c this is a comment
c
p cnf 4 6
-2 3 0
1 3 0
-1 2 3 -4 0
-1 -2 0
1 -2 0
2 -3 0
```

Declares number of variables and clauses.

Each row is a clause ending with 0

Clause is  $p_2 \vee \neg p_3$

## Topic 10.5

### Pseudo-Boolean constraints

# Pseudo-Boolean constraints

Let  $p_1, \dots, p_n$  be Boolean variables.

The following is a pseudo-Boolean constraint.

$$c_1 p_1 + \dots + c_n p_n \leq c,$$

where  $c_1, \dots, c_n, c \in \mathbb{Z}$ .

How should we solve them?

- ▶ Using Boolean reasoning
- ▶ Using arithmetic reasoning (Not covered in these slides)

Here we will see the Boolean encoding for the constraints.

**Commentary:** Pseudo-Boolean constraints are very important class. Several hybrid approaches are being developed. For in-depth please look at [http://www.it.uu.se/research/group/optimisation/NordConsNet2018/10\\_roundingsat-seminar.pdf](http://www.it.uu.se/research/group/optimisation/NordConsNet2018/10_roundingsat-seminar.pdf) by Jan Elffers, KTH Royal Institute of Technology  
<https://sat-smt.in/assets/slides/daniel1.pdf> by Daniel Le Berre



## Observations on pseudo-Boolean constraints

- ▶ Replacing negative coefficients to positive

$$t - c_i p_i \leq c \quad \rightsquigarrow \quad t + c_i (\neg p_i) \leq c + c_i$$

- ▶ Divide the whole constraints by  $d := \gcd(c_1, \dots, c_n)$ .

$$c_1 p_1 + \dots + c_n p_n \leq c \quad \rightsquigarrow \quad (c_1/d)p_1 + \dots + (c_n/d)p_n \leq \lfloor c/d \rfloor$$

- ▶ Trim large coefficients to  $c + 1$ . Let us suppose  $c_i > c$ .

$$t + c_i p_i \leq c \quad \rightsquigarrow \quad t + (c + 1)p_i \leq c$$

## Observations on pseudo-Boolean constraints

- ▶ Trivially true are replaced by  $\top$ . If  $c \geq c_1 + \dots + c_n$

$$c_1 p_1 + \dots + c_n p_n \leq c \quad \rightsquigarrow \quad \top$$

- ▶ Trivially false are replaced by  $\perp$ . If  $c < 0$

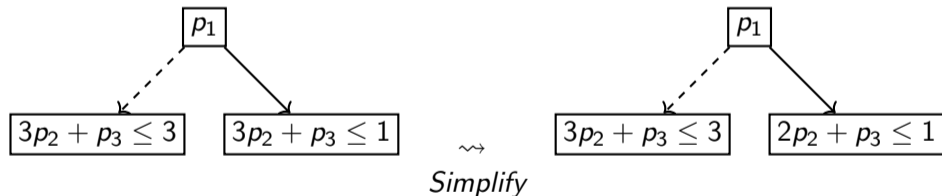
$$c_1 p_1 + \dots + c_n p_n \leq c \quad \rightsquigarrow \quad \perp$$

## Translating to decision diagrams

We choose a 0 and 1 for each variable to split cases and simplify.

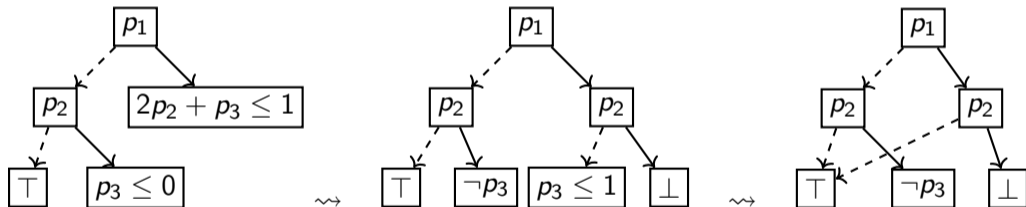
### Example 10.4

Consider  $2p_1 + 3p_2 + p_3 \leq 3$

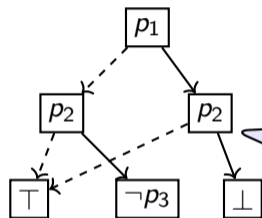


## Example: translating to decision diagrams

We can split node left node  $3p_2 + p_3 \leq 3$  further on  $p_2$ .



## Example: decision diagrams to clauses



$$\rightsquigarrow (\neg p_1 \Rightarrow temp1) \wedge (temp1 \wedge \neg p_2 \Rightarrow \top) \wedge (temp1 \wedge p_2 \Rightarrow \neg p_3) \wedge$$

$$(p_1 \Rightarrow temp2) \wedge (temp2 \wedge \neg p_2 \Rightarrow \top) \wedge (temp2 \wedge p_2 \Rightarrow \perp)$$

A fresh variable for each internal node

### Exercise 10.8

- Simplify the clauses
- Complexity of the translation from pseudo-Boolean constraints?

## Exercise: Pseudo-Boolean constraints

### Exercise 10.9

Let  $p_1$ ,  $p_2$ , and  $p_3$  be Boolean variables. Convert the following pseudo-Boolean inequalities into BDDs while applying simplifications eagerly, and thereafter into equisatisfiable CNF clauses.

- ▶  $2p_1 + 6p_3 + p_2 \leq 3$
- ▶  $2p_1 + 6p_3 + p_2 \geq 3$
- ▶  $2p_1 + 3p_3 + 5p_2 \leq 6$

# Exponential sized BDDs for Pseudo-Boolean constraints

Consider the following pseudo-Boolean constraint

$$\sum_{i=1}^{2n} \sum_{j=1}^{2n} (2^{j-1} + 2^{2n+i-1}) p_{ij} \leq (2^{4n} - 1)n$$

Any BDD representing the above constraints have at least  $2^n$  nodes.

Proof in : A New Look at BDDs for Pseudo-Boolean Constraints, <https://www.cs.upc.edu/~oliveras/espai/papers/JAIR-bdd.pdf>

## API for pseudo-Boolean constraints in Z3

```
from z3 import *
p = Bool("p")
q = Bool("q") # declare a Boolean variable

c1 = PbLe([(p,1),(q,2)], 3) # encodes  $p+2q \leq 3$ 
c2 = PbGe([(p,1),(q,-1)], 4) # encodes  $p-q \geq 4$ 

s = Solver()
s.add( And(c1,c2) )
s.check()
```



## Exercise: equivalent ranges in pseudo-Boolean constraints

### Exercise 10.10

Let  $p_1$ ,  $p_2$ , and  $p_3$  be Boolean variables. Let us consider pseudo-Boolean constraint  $2p_1 + 3p_3 + 5p_2 \leq K$ , for some non-negative integer  $K$ . For which of the following ranges of  $K$ , the constraint has same set of satisfying models?

- ▶  $[0, 2]$
- ▶  $[3, 4]$
- ▶  $[7, 7]$
- ▶  $[10, 12]$

# Pseudo-Boolean constraints

## Exercise 10.11

Let  $a$ ,  $b$ , and  $n$  be positive integers such that  $\sum_{i=1}^n b^i < a$ . Let  $w_i = a + b^i$  for each  $i \in 1..n$ . Show that the following pseudo-Boolean constraints are equivalent.

$$w_1 p_1 + \dots + w_n p_n \leq (an/2)$$

and

$$p_1 + \dots + p_n \leq (n/2) - 1$$

# Topic 10.6

## Problems

# SAT encoding: $n$ queens

## Exercise 10.12

*Encode  $N$ -queens problem in a SAT problem.*

*$N$ -queens problem: Place  $n$  queens in  $n \times n$  chess such that none of the queens threaten each other.*

# SAT encoding: peaceable queens

## Exercise 10.13

*Encode the problem of finding the maximal number of placing the same number of white queens and black queens on an  $n$  by  $n$  chess board so that no queen attacks any queen of the opposite color*

Interesting video on peaceable queens <https://www.youtube.com/watch?v=IN1fPtY9jYg>

# SAT encoding: overlapping subsets

## Exercise 10.14

*For a set of size  $n$ , find a maximal collection of  $k$  sized subsets such that any pair of the sets have exactly one common element.*

# SAT encoding: setting a question paper

## Exercise 10.15

*There is a database of questions with the following properties:*

- ▶ *Hardness level*  $\in \{Easy, Medium, Hard\}$
- ▶ *Marks*  $\in \mathbb{N}$
- ▶ *Topic*  $\in \{T_1, \dots, T_t\}$
- ▶ *LastAsked*  $\in \text{Years}$

*Make a question paper with the following properties*

- ▶ *It must contain  $x\%$  easy,  $y\%$  medium, and  $z\%$  difficult marks.*
- ▶ *The total marks of the paper are given.*
- ▶ *The number of problems in the paper are given.*
- ▶ *All topics must be covered.*
- ▶ *No question that was asked in last five years must be asked.*

*Write an encoding into SAT problem that finds such a solution. Test your encoding on reasonably sized input database. Choose a strategy to evaluate your tool and report plots to demonstrate the*

# SAT encoding: finding a schedule

## Exercise 10.16

*An institute is offering  $m$  courses.*

- ▶ *Each has a number of contact hours  $\implies$  credits*

*The institute has  $r$  rooms.*

- ▶ *Each room has a maximum student capacity*

*The institute has  $s$  weekly slots to conduct the courses.*

- ▶ *Each slot has either 1 or 1.5 hour length*

*There are  $n$  students.*

- ▶ *Each student have to take minimum number of credits*
- ▶ *Each student has a set of preferred courses.*

*Assign each course slots and a room such that all student can take courses from their preferred courses that meet their minimum credit criteria.*

*Write an encoding into SAT problem that finds such an assignment . Test your encoding on reasonably sized input. Choose a strategy to evaluate your tool and report plots to demonstrate*



## SAT encoding: synthesis by examples

### Exercise 10.17

Consider an unknown function  $f : \mathcal{B}^N \rightarrow \mathcal{B}$ . Let us suppose for inputs  $I_1, \dots, I_m \in \mathcal{B}^N$ , we know the values of  $f(I_1), \dots, f(I_m)$ .

- Write a SAT encoding of finding a  $k$ -sat formula containing  $\ell$  clauses that represents the function.
- Write a SAT encoding of finding an NNF (negation normal form, i.e.,  $\neg$  is only allowed on atoms) formula of height  $k$  and width  $\ell$  that represents the function. (Let us not count negation in the height.)
- Write a SAT encoding of finding a binary decision diagram of height  $k$  and maximum width  $\ell$  that represents the function.

Test your encoding on reasonably sized input. Choose a strategy to evaluate your tool and report plots to demonstrate the performance.

# SAT encoding: Rubik's cube

## Exercise 10.18

*Write a Rubik's cube solver using a SAT solver*

- ▶ *Input:*
  - ▶ *start state,*
  - ▶ *final state, and*
  - ▶ *number of operations  $k$*
- ▶ *Output:*
  - ▶ *sequence of valid operations or*
  - ▶ *"impossible to solve within  $k$  operations"*

*Test your encoding on reasonably many inputs. Choose a strategy to evaluate your tool and report plots to demonstrate the performance.*

# SAT encoding: TickTacToe

## Exercise 10.19

*Write an encoding for synthesizing always winning strategy for the TickTacToe player 1.*

*Since it is a game one needs to give a grammar for the Boolean function, from a space of functions.*

# SAT encoding: square of squares

## Exercise 10.20

*Squaring the square problem: “Tiling an integral square using only other smaller integral squares such that all tiles have different sizes.”*

*Consider a square of size  $n \times n$ , find a solution of above problem using a SAT solver using tiles less than  $k$ .*

*Test your encoding on reasonably sized  $n$  and  $k$ . Choose an strategy to evaluate your tool and report plots to demonstrate the performance.*

## SAT encoding: Mondrian art

### Exercise 10.21

*Mondrian art problem: "Divide an integer square into non-congruent rectangles. If all the sides are integers, what is the smallest possible difference in area between the largest and smallest rectangles?"*

*Consider a square of size  $n \times n$ , find a Mondrian solution above  $k$  using a SAT solver.*

## Example : make mastermind player

### Exercise 10.22

*Mastermind is a two player game. There are  $n$  colors. Let  $k < n$  be a positive number.*

- 1. Player one chooses a hidden sequence of  $k$  colors (colors may repeat)*
- 2. The game proceeds iteratively as follows until player two has guessed the sequence correctly.*
  - ▶ Player two makes a guess of sequence of  $k$  colors*
  - ▶ Player one gives feedback to player two by giving*
    - ▶ (red response) the number of correct colors in the correct positions, and*
    - ▶ (white response) the number of correct colors in the wrong positions.*

*Play the game here <http://www.webgamesonline.com/mastermind/>*

*Create player two using a SAT solver that is tolerant to unreliable player one, i.e., sometimes player one gives wrong answer.*

## Example: make RushHour solver using SAT solvers

### Exercise 10.23

*The RushHour game consists of a collection of cars of length two, which are set either horizontally or vertically on a  $n \times n$ . Two cars cannot overlap each other. The grid that has a single exit, which is a node on the grid (usually at one side). Each car can move forward or backward in the direction of the car if another car is not blocking the car. Cars cannot change their orientation. There is one special car, which is positioned in the grid such that it can reach to the exit. The objective of the rush hour game is to find a series of moves such that the special car reaches to the exit point.*

Example:

### Exercise 10.24

*Consider an undirected connected graph  $G = (V, E)$  and nodes  $s, t \in V$ . Give a SAT encoding of removing the minimal set of edges such that  $s$  and  $t$  are not connected.*



# Removing edges to be acyclic graph

## Exercise 10.25

*Give a SAT encoding for removing minimum number of edges in a (un)directed graphs such that the graph becomes acyclic.*

## Topic 10.7

Extra section : cardinality constraints other encodings

## Bitwise encoding of $p_1 + \dots + p_n \leq 1$

Let  $m = \lceil \log_2 n \rceil$ .

- ▶ Consider bits  $r_1, \dots, r_m$
- ▶ For each  $i \in 1 \dots n$ , let  $b_1, \dots, b_m$  be the binary encoding of  $(i - 1)$ .  
We add the following constraints for  $p_i$  to be 1.

$$(p_i \Rightarrow (r_1 = b_1 \wedge \dots \wedge r_m = b_m))$$

### Example 10.5

Consider  $p_1 + p_2 + p_3 \leq 1$ .

$$m = \lceil \log_2 n \rceil = 2.$$

We get the following constraints.

$$(p_1 \Rightarrow (r_1 = 0 \wedge r_2 = 0))$$

$$(p_2 \Rightarrow (r_1 = 0 \wedge r_2 = 1))$$

$$(p_3 \Rightarrow (r_1 = 1 \wedge r_2 = 0))$$

$\rightsquigarrow$

*Simplified*

$$(p_1 \Rightarrow (\neg r_1 \wedge \neg r_2))$$

$$(p_2 \Rightarrow (\neg r_1 \wedge r_2))$$

$$(p_3 \Rightarrow (r_1 \wedge \neg r_2))$$

### Exercise 10.26

What are the variable and clause size complexities?

## Operational encoding for $p_1 + \dots + p_n \leq k$

Sum the bits using full adders. Compare the resulting bits against  $k$ .

Produces  $O(n)$  encoding, however, the encoding is not considered good for SAT solvers, since it is **not arc-consistent**.

**Commentary:** To learn more about arc-consistent encoding look at the extra slides

## Topic 10.8

Extra section : cardinality constraints via merge sort

## Cardinality constraints via sorted variables $O(n \ln^2 n)$

Let us suppose we have a circuit that produces sorted bits in decreasing order.

$$([y_1, \dots, y_n], Cs) := \text{sort}(p_1, \dots, p_n)$$

We can encode the cardinality constraints as follows

$$\begin{array}{ll} p_1 + \dots + p_n \leq k & \{\neg y_{k+1}\} \cup Cs \\ p_1 + \dots + p_n \geq k & \{y_k\} \cup Cs \end{array}$$

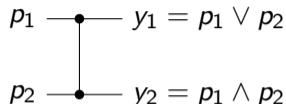
### Exercise 10.27

- How to encode  $p_1 + \dots + p_n < k$
- How to encode  $p_1 + \dots + p_n > k$
- How to encode  $p_1 + \dots + p_n = k$

For details : look at the extra slides at the end of the lecture.

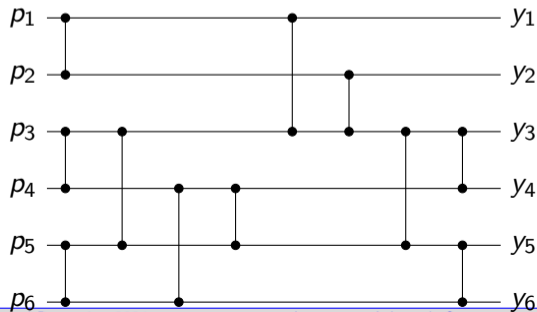
## Sorting networks

The following circuit sorts two bits  $p_1$  and  $p_2$ .



We can sort any number of bits by composing the circuit according to a sorting algorithm.

**Example 10.6** *Sorting 6 bits using merge sort.*



# Formal definition of sorting networks

**Base case:**

$$n = 1$$

$$\text{sort}(p_1, p_2) \triangleq \text{merge}([p_1], [p_2]);$$

**Induction step:**

$$2n > 2$$

Let,

sort/merge returns a vector of signals and a set of clauses.

$$([p'_1, \dots, p'_n], Cs_1) := \text{sort}(p_1, \dots, p_n)$$

$$([p'_{n+1}, \dots, p'_{2n}], Cs_2) := \text{sort}(p_{n+1}, \dots, p_{2n})$$

$$([y_1, \dots, y_{2n}], Cs_M) := \text{merge}([p'_1, \dots, p'_n], [p'_{n+1}, \dots, p'_{2n}])$$

Then,

$$\text{sort}(p_1, \dots, p_{2n}) \triangleq ([y_1, \dots, y_{2n}], Cs_1 \cup Cs_2 \cup Cs_M)$$



## Formally merge: odd-even merging network

Merge assumes that the input vectors are sorted.

**Base case:**

$$\text{merge}([p_1], [p_2]) \triangleq ([y_1, y_2], \{y_1 \Leftrightarrow p_1 \wedge p_2, y_2 \Leftrightarrow p_1 \vee p_2\});$$

**Induction step:**

Let

$$([z_1, \dots, z_n], Cs_1) := \text{merge}([p_1, p_3, \dots, p_{n-1}], [y_1, y_3, \dots, y_{n-1}])$$

$$([z'_1, \dots, z'_n], Cs_2) := \text{merge}([p_2, p_4, \dots, p_n], [y_2, y_4, \dots, y_n])$$

$$([c_{2i}, c_{2i+1}], CS_M^i) := \text{merge}([z_{i+1}], [z'_i]) \quad \text{for each } i \in [1, n-1]$$

Then,

$$\text{merge}([p_1, \dots, p_n], [y_1, \dots, y_n]) \triangleq ([z_1, c_1, \dots, c_{2n-1}, z'_n], Cs_1 \cup Cs_2 \cup \bigcup_i CS_M^i)$$

## Topic 10.9

Extra section : arc-consistency

## Arc-consistency

Let  $C(Ps)$  be a problem with variables  $Ps = p_1, \dots, p_n$ .

Let  $E(Ps, Ts)$  be an encoding of the problem, where variables  $Ts = t_1, \dots, t_k$  are introduced by the encoding.

### Definition 10.1

We say  $E(Ps, Ts)$  is *arc-consistent* if for any partial model  $m$  of  $E$

1. If  $m|_{Ps}$  is inconsistent with  $C$ , then *unit propagation* in  $E$  causes conflict.
2. If  $m|_{Ps}$  is extendable to  $m'$  by *local reasoning* in  $C$ , then *unit propagation* in  $E$  obtains  $m''$  such that  $m''|_{Ps} = m'$ .

Unit propagation == Local reasoning

**Commentary:** Constraint satisfaction problem (CSP) is a general problem. [https://en.wikipedia.org/wiki/Constraint\\_satisfaction\\_problem](https://en.wikipedia.org/wiki/Constraint_satisfaction_problem)  
There you have a formal meaning of local reasoning or local consistency. But, in our context, we have not defined it. The user who is encoding his problem into SAT will choose what (s)he thinks is the local reasoning in his problem.

## Example: arc-consistency

### Example 10.7

Consider problem  $p_1 + \dots + p_n \leq 1$

An encoding is arc-consistent if

1. If at any time two  $p_i$ s are made true, unit propagation should trigger unsatisfiability
2. If at any time  $p_i$  is made true, unit propagation should make all other  $p_j$ s false

In the problem, the above two points of reasoning are *defined to be local reasoning by us*.

One may choose some other definition. Often, there is a natural choice.

**Commentary:** The unit propagation in the encoding must mimic the local reasoning of the problem. Intuitively, the encoding must not make the life of a solver harder. If the local reasoning can deduce something, then unit propagation must also deduce it. For more discussion, look in <http://minisat.se/downloads/MiniSat+.pdf> page5

## Example: non arc-consistent encoding

### Example 10.8

Consider problem  $p_1 + p_2 + p_3 \leq 0$

Let us use full adder encoding

$$\underbrace{\neg(p_1 \oplus p_2 \oplus p_3)}_{\text{sum}} \wedge \underbrace{\neg((p_1 \wedge p_2) \vee (p_2 \wedge p_3) \vee (p_1 \wedge p_3))}_{\text{carry}}$$

Clearly  $p_1, p_2, p_3$  are 0. But, the unit propagation without any decisions does not give the model.  
*Local reasoning*

However, we can encode the problem into an arc-consistent encoding as follows.

$$\neg p_1 \wedge \neg p_2 \wedge \neg p_3$$

End of Lecture 10