RECAP

- Classes P and NP. $P \subseteq NP$
- Poly-time reducibility: $L_1 \leq_P L_2$
  "relative difficulty" of problems:

$$(L_1 \leq_P L_2) \wedge (L_2 \in P) \Rightarrow L_1 \in P$$

("LESS DIFFICULT")

- Classes NPH and NPC

$$L' \in NPC : (L' \in NP) \wedge (\forall L \in NP : L \leq_P L')$$

- Facts:

$$(P \cap NPC \neq \emptyset) \Rightarrow P = NP$$

$$L_1, L_2 \in NPC : (L_1 \leq_P L_2) \wedge (L_2 \leq_P L_1)$$

- Membership in NPC: use transitivity of $\leq_P$ by proving:

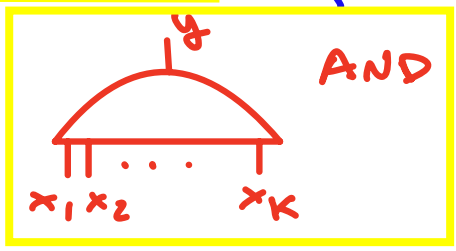
$$L_{KNOWN\text{-}NPC} \leq_P L_{CANDIDATE}$$

$\forall L \in NP,\ L \leq_P L_{KNOWN\text{-}NP}$

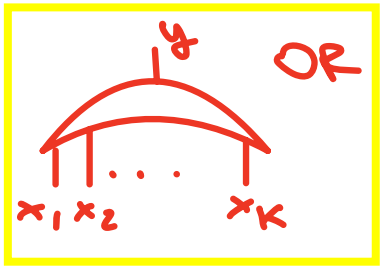
- First language in NPC requires direct application of definition: ∞ reductions!

In 1971 Cook proved the first NP-completeness result on a problem in Boolean logic. We'll work with an equivalent variant defined on Boolean circuits

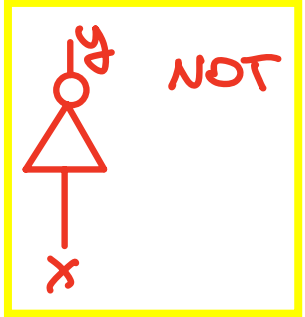
DEF A gate is an elementary switching circuit of three types



Output wire $y = 1$
$\Leftrightarrow$ input wires $x_1, x_2, \ldots, x_k$ all carry 1. Thus $y = \bigwedge_{i=1}^{k} x_i$

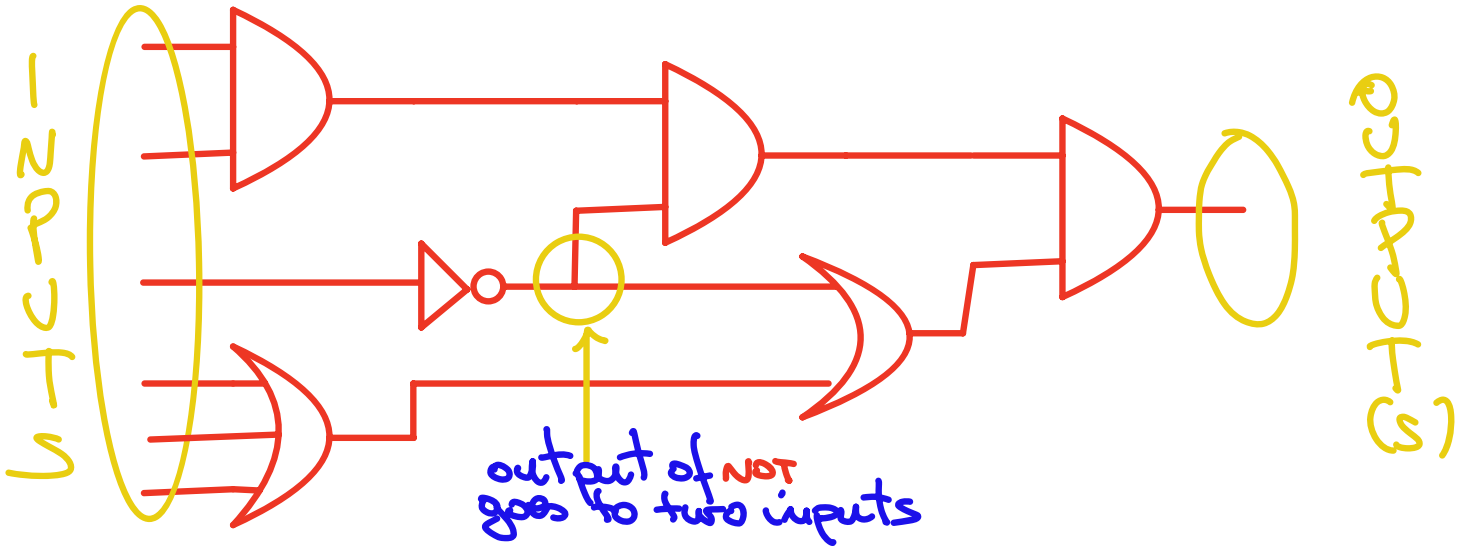


Output wire $y = 1$
$\Leftrightarrow$ at least one input wire carries 1. Thus $y = \bigvee_{i=1}^{k} x_i$



Output wire $y = 1$
$\Leftrightarrow$ input wire $x = 0$
Thus $y = \neg(x)$ $(= \bar{x})$

DEF A Boolean Circuit (BC) is an acyclic interconnection of gates, where the output of a gate can become input to one or more other gates



Wires that do not originate from the output of a gate are called **inputs** of the circuit

Wires that do not enter into a gate are called **outputs** of the circuit.

We will mostly consider circuits with a **single output**

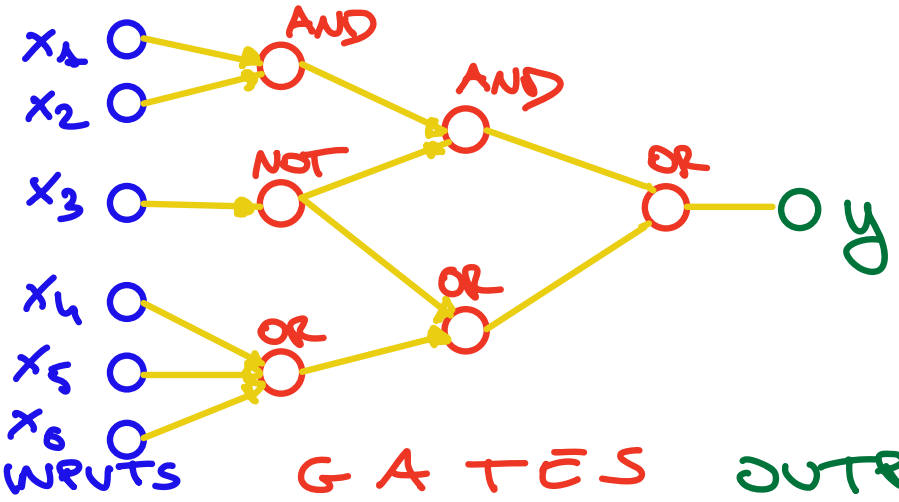
FAN-IN: max # wires entering a gate

FAN-OUT: max # gates fed by the same wire

A circuit can be represented by a directed graph

$$C = (I \cup G \cup O, W)$$

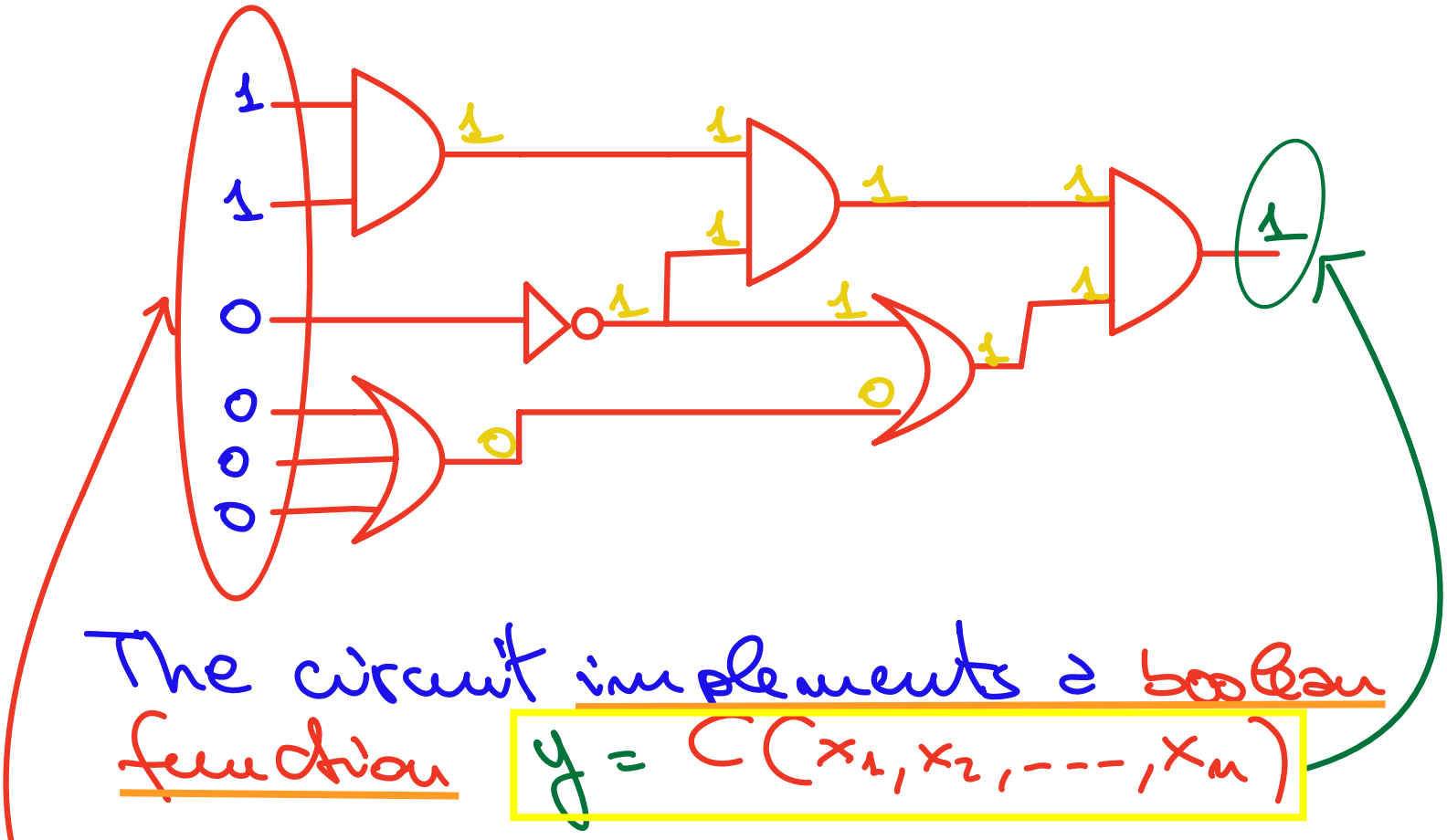
- one labeled node per input, gate, output
- one edge for each wire



no loop ⇒ no memorion

DEF A truth-assignment to the inputs of a circuit associates a boolean value to each input.

Under a truth-assignment each wire in the circuit will switch to the correct boolean value according to the gates traversed:

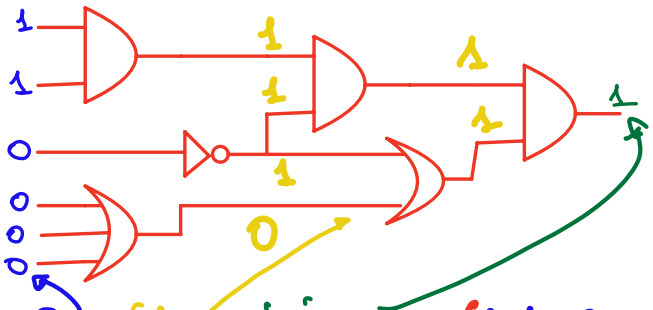


The circuit implements a boolean function $y = C(x_1, x_2, \dots, x_n)$

For a circuit with n inputs, a truth assignment is $\vec{b} \in \{0,1\}^n$

DEF The set of boolean values carried by all wires of a circuit C under a truth-assignment $\vec{b}$ is called a configuration of C under $\vec{b}$

DEF A boolean circuit $C(x_1, x_2, \dots, x_n)$ is satisfiable if there exists a truth assignment $b \in \{0,1\}^n : C(\vec{b}) = 1$.



C is satisfied by the truth assignment

$\vec{b} = (1,1,0,0,0,0)$

Configuration: $(1,1,0,0,0,0,1,1,1,0,1,1,1) \equiv y_C$

(in some prespecified order)

BOOLEAN CIRCUIT SATISFIABILITY (BC-SAT)

I: $\langle C(x_1,\dots,x_m)\rangle = \langle (I \cup G \cup O, W) \rangle$, boolean circuit (usiamo grafo di circuito)

Q: C is satisfiable?

$\exists \vec{b} \in \{0,1\}^n : C(\vec{b}) = 1$?

BC-SAT is an important problem in practice (e.g.: testing, circuit equivalence ...)

It is easy to prove that

THEOREM: BC-SAT $\in$ NP

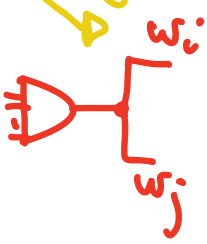
Proof:

VERIFY_BCSAT (x, y) {high-level}

if $x \neq \langle (I \cup G \cup \{y_o\}, W) \rangle$ then return 0

if $y \neq \langle w_1, w_2, \dots, w_{|W|} \rangle \in \{0,1\}^{|W|}$

then return 0
{ "candidate" certificate looks like a configuration y_c (1 value / wire) }
if $\exists$ $w_i \neq w_j$ outgoing from same gate then return 0 { incoherent configuration! }



for each gate $g \in G$
if CI/O relation for g incorrect) then return 0

{ check that the configuration is coherent }
return wire value associated to y_o
{ $=1 \Rightarrow y=y_c$ certifies that C is satisfiable }

circuito ⇑ configurazione ⇑

$T_{VB}(|x|,|y|) = \Theta(|x|+|y|)$

BC-SAT $\in$ NP

Difficult part:

THEOREM (Cook, 1971)

BC-SAT $\in$ NPH

The proof of Cook's theorem is complex in the details but relies on a simple intuition

INFORMAL PROOF:

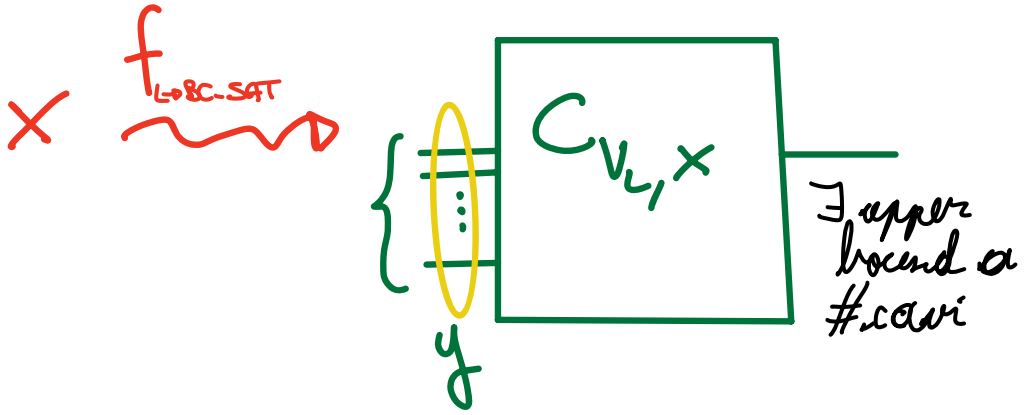
We need to show that given an arbitrary $L \in NP$, $L \leq_P L_{BC\text{-}SAT}$

Given L, the only thing that we know is that $\exists V_L(x,y)$, poly-time verifier for L.

We must define a reduction function

$$f_{L \to BC\text{-}SAT}(x) : \{0,1\}^* \to \{0,1\}^*$$

The reduction function builds a boolean circuit starting from V_L: as follows



The shape of $C_{V_L,x}$ depends on V_L and x.
Its inputs "represent" input y of V_L.

We will make sure that:

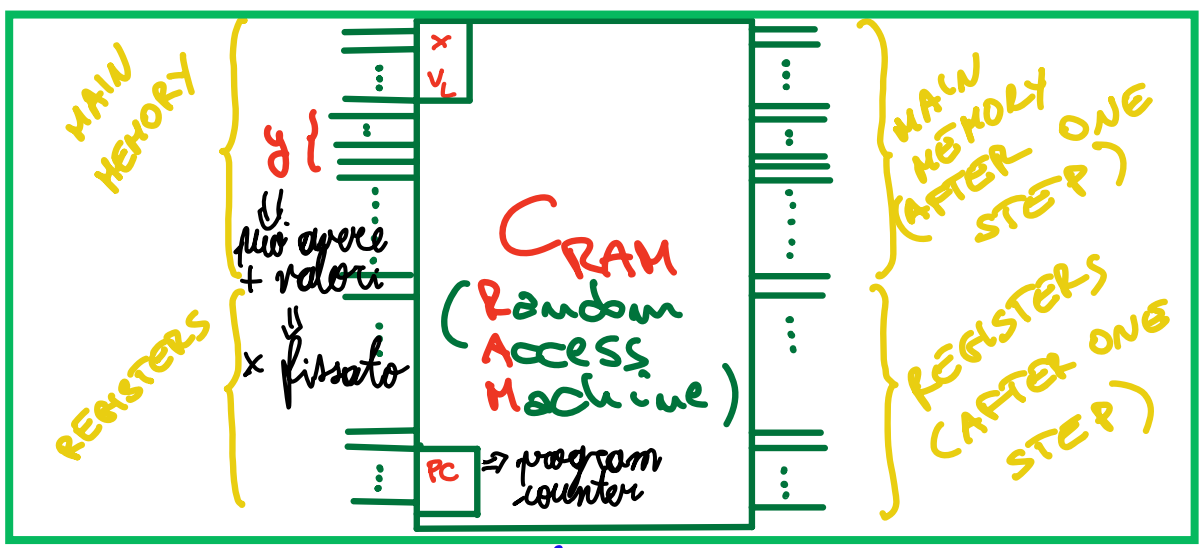
$f(x) = \langle C_{V_L,x} \rangle \in L_{BC\text{-}SAT}$ (is satisfiable)

$\Leftrightarrow$ $\exists$ certificate y of poly-length for x

$\Leftrightarrow$ $x \in L$

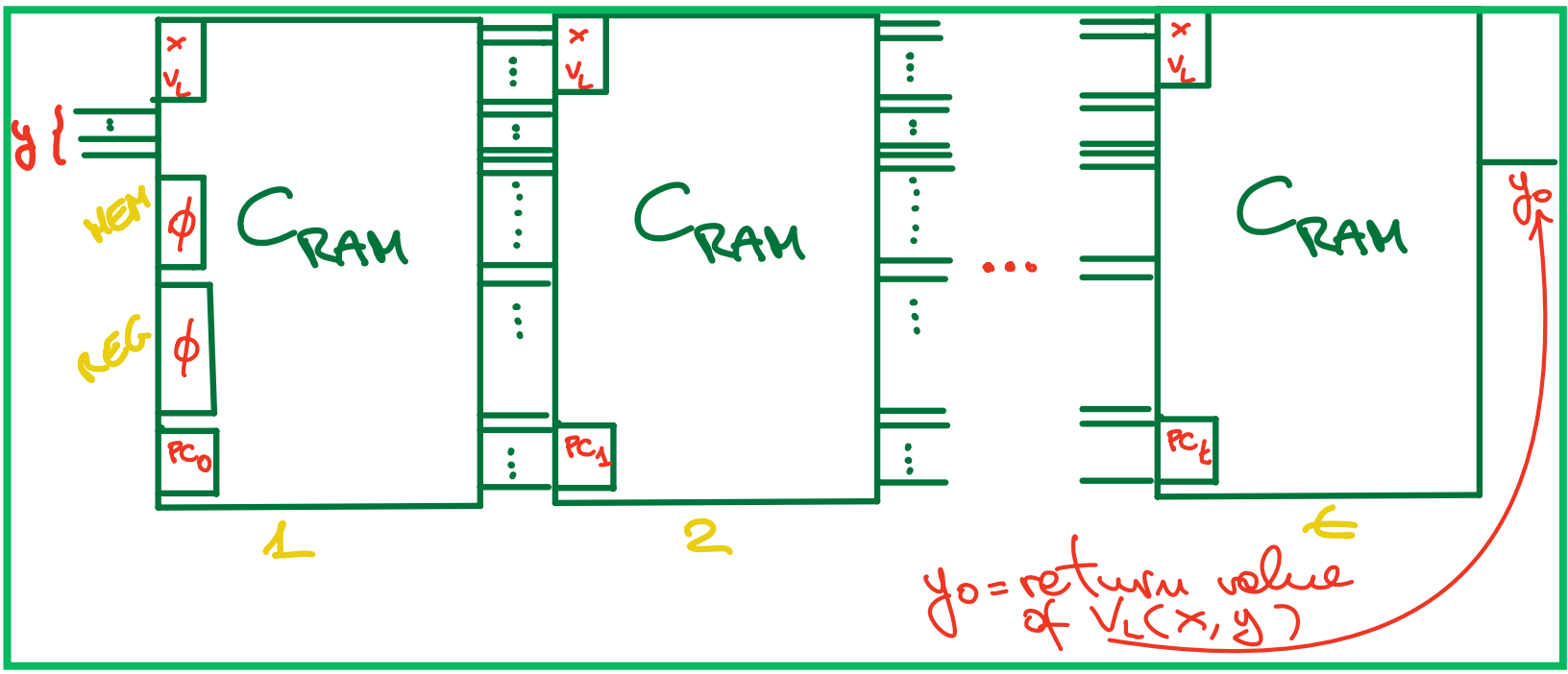
IDEA: $C_{V_L,x}(y)$ "simulates" the computation performed by $V_L(x,y)$

$C_{V_L,x}$ is built starting from a combinatorial circuit executing a single step of a Random Access Machine (RAM)



C_{RAM} implements the standard circuitry of a RAM (ALU, branches, etc...). Memory is "simulated" by passing along stored values on output wires

An algorithm executing t steps can be simulated by cascading t copies of C_{RAM}:



Crucial points:

1. $|C_{RAM}|$ (singolo step) is polynomial in x and y
   - circuitry polynomial in the memory size
2. It suffices to set $|y| = \Theta(|x|^{k_1})$ ⇒ può avere dim. arbitr.
3. It suffices to consider $t = \Theta((|x|+|y|)^{k_2})$
   (k_1, k_2 from definition of poly-time verifier)

⇒ $C_{V_L, x}$ has size polynomial in x!

Therefore $f_{L \to BC\text{-}SAT}$ is ptc

We have

$$x \in L \Leftrightarrow \exists \bar{y},\ |\bar{y}| = O(|x|^{k_1}) : V_L(x, \bar{y}) = 1$$

$$\Leftrightarrow C_{V_L, x}(\bar{y}) = 1$$

$$\{ C_{V_L, x} \text{ simulates } V_L(x, y) \}$$

$$\Leftrightarrow \langle C_{V_L, x} \rangle \in \text{BC-SAT}$$

$$\Leftrightarrow f_{L \to \text{BCSAT}}(x) \in \text{BC-SAT}$$

Thus $L \leq_p$ BC-SAT

This proves that BC-SAT $\in$ NPH

In what follows we will prove that a number of problems are in NPC. The details of the reductions will teach us generic approaches and highlight pitfalls to avoid

FORMULA SATISFIABILITY (SAT)

A **boolean formula** $\phi(x_1, x_2, \dots x_m) : \{0,1\}^m \to \{0,1\}$ is a (parenthesized) expression built on

- m variables $x_1, x_2, \dots, x_m$
- binary operators : $\vee$ OR, $\wedge$ AND, $\to$ IMPLIES, $\leftrightarrow$ EQUALS
- unary operator : $\neg$ NOT

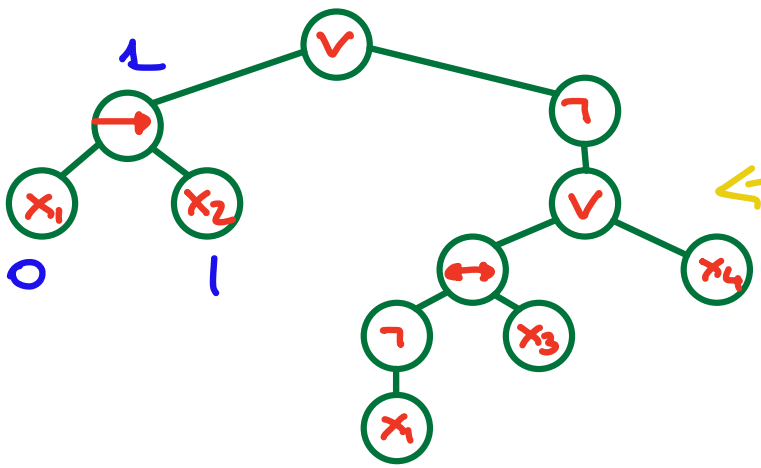
x,y	$x \to y$	$x \leftrightarrow y$
00	1	1
01	1	0
10	0	0
11	1	1

$(\neg x) \vee y$ $\quad$ $(x \wedge y) \vee (\neg x \wedge \neg y)$

EXAMPLE: $\phi(x_1, x_2, x_3, x_4) =$

$= (x_1 \to x_2) \vee (\neg((\neg x_1 \leftrightarrow x_3) \vee x_4))$

Can be represented by the parenthesization tree (leafs: variables ; internal nodes: operators)



A truth-assignment is a vector of boolean values $b \in \{0,1\}^m$, one for each variable

The value $\phi(\vec{b})$ can be computed in time linear in the formula's size by using the tree

A satisfying assignment $\vec{b}$ is such that $\phi(\vec{b})=1$. If $\vec{b}$ exists, ϕ is said satisfiable.

FORMULA SATISFIABILITY (SAT)

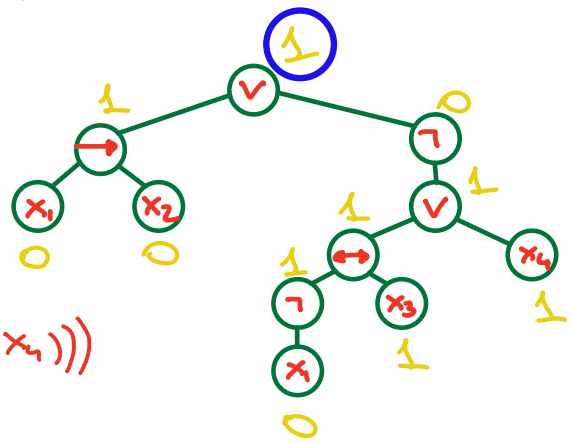
$\begin{cases} I: \langle \phi(x_1,\dots,x_n)\rangle, \ \phi \text{ boolean formula} \\ Q: \text{Is } \phi \text{ satisfiable?} \end{cases}$

EXAMPLE:

$\vec{b}=(0,0,1,1)$

$\phi(x_1,x_2,x_3,x_4) =$

$= (x_1 \to x_2) \vee (\neg((\neg x_1 \leftrightarrow x_3) \vee x_4))$



ϕ is satisfiable $[\phi(\vec{b})=1]$

Very similar to BC-SAT (in fact, this is Cook's original problem)

Nonetheless, the reduction is nontrivial

1. SAT ∈ NP: Trivial: for $\langle \phi(x_1,\dots,x_n)\rangle$ a candidate certificate is $b \in \{0,1\}^n$. $\phi(\vec{b})$ can be computed in linear time using the tree representation. (write code of VERIFY_SAT(x,y) as an exercise)

2. SAT ∈ NPH We show that

$BC\_SAT \leq_p SAT$

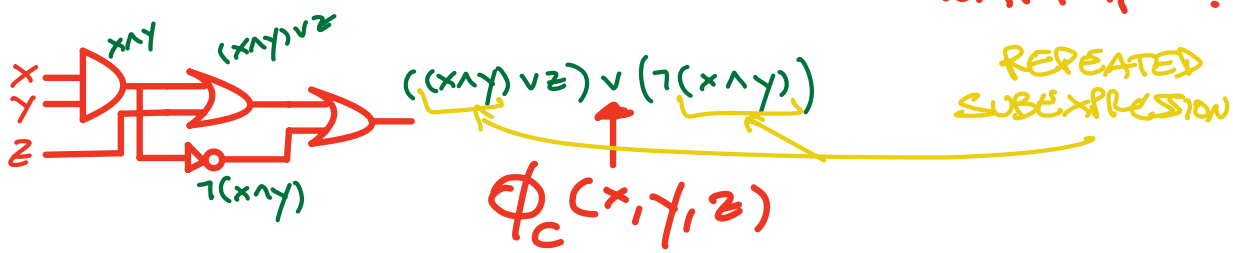
Reduction function f: must transform a circuit C into a formula ϕ_C:

C is satisfiable $\Leftrightarrow$ $f(C) = \phi_C$ is satisfiable

$\langle C \rangle \in BC\_SAT \Leftrightarrow \langle \phi_C \rangle = f(\langle C \rangle) \in SAT$

Seems easy: straightforward method: compute the subexpression associated to each wire. ϕ_C = expression associated to output wire

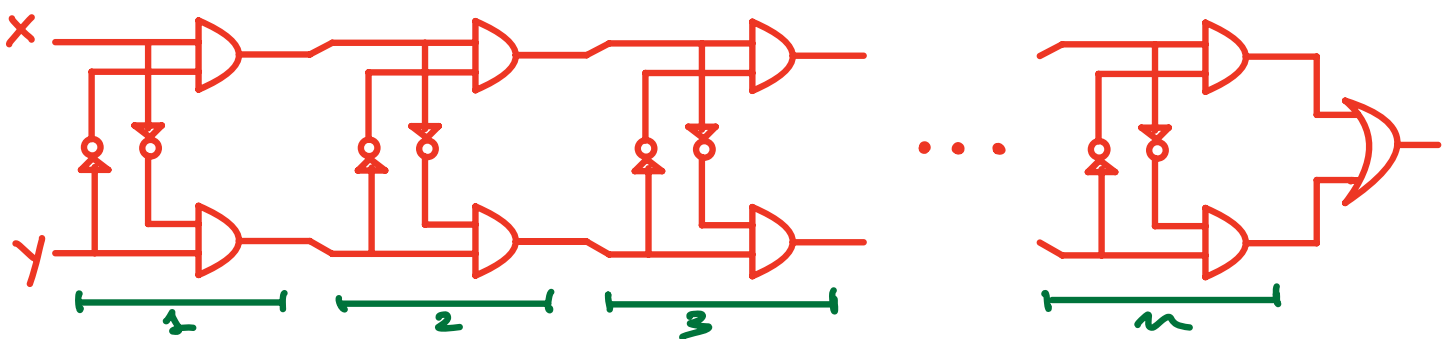
EXAMPLE:



WARNING!

PROBLEM: Repeated subexpressions may blow-up the length of the formula exponentially!

CONSIDER:



Two-input circuit made of n stages

$|C(x_1, x_2)| = \Theta(n)$

Outputs of Stage 1: $(x \wedge (\neg y))$, $((\neg x) \wedge y)$

Outputs of Stage 2: $(x \wedge (\neg y)) \wedge (\neg((\neg x) \wedge y))$

$(\neg(x \wedge (\neg y))) \wedge ((\neg x) \wedge y))$

etc

The number of symbols in the subexpressions of Stage i is more than double than that of Stage $i-1$

$$\Rightarrow \begin{cases} Size(i) \geq 2\, Size(i-1) \\ size(1) = c \end{cases}$$

$$\Rightarrow |\phi_C(x,y)| \geq Size(n) \geq c \cdot 2^{n-1} !$$

OBSERVE: $|C| = \Theta(n)$ but $|\phi_C(x,y)| = \Omega(2^n)$

This reduction cannot be ptc! (I can't generate an exponential number of symbols in polynomial time!)

We need a more sophisticated approach!

$C = (I \cup G \cup \{0\}, W)$

x_i ← I ; y_g $\forall g \in G$ ← G ; y_0 ← $\{0\}$

New variables $y_g : g \in G$