Algorithms and Complexity

Spring 2018
Aaram Yun

This page is intentionally left blank

Today

- NP-completeness

Circuit SAT

- Consider boolean circuits with a single bit output
- Circuit satisfiability
  - $\texttt{CSAT} := \{\langle C \rangle : \exists z \text{ s.t. } C(z) = 1\}$
  - $R_{\texttt{CSAT}} := \{\langle C, z \rangle : C(z) = 1\}$
- $R_{\texttt{CSAT}}$ is $\mathcal{PC}$-complete, and $\texttt{CSAT}$ is $\mathcal{NP}$-complete

Satisfiability problem:

given a boolean function f, decide whether there's a satisfying assignment.

This depends on how f is given. (If as a function table, this would be easy)

So, many different versions of SAT.

Circuit SAT

- Let's focus on $R_{\mathtt{CSAT}}$
- It is in $\mathcal{PC}$, clearly
- It is $\mathcal{PC}$-complete? Proving R_{CSAT} is $\mathcal{PC}$-hard.

Circuit SAT

- R: any $\mathcal{PC}$ problem
  - Do we have a Levin reduction from R to $R_{\texttt{CSAT}}$?
- M: any polytime checker for R
  - Idea: given any x, produce a circuit C_x, using M, so that, C_x is satisfiable iff $R(x) \neq \emptyset$, and,
  - Also, C_x has enough information about M so that, when we have $C_x(z) = 1$, then, using z, it is possible to obtain y such that $M(x, y) = 1$ (that is, $y \in R(x)$)

Circuit SAT

$R(x) \neq \emptyset$ iff $\exists y \in R(x)$

iff $M(x, y) = 1$

iff $C_x(y) = 1$

iff C_x is satisfiable.

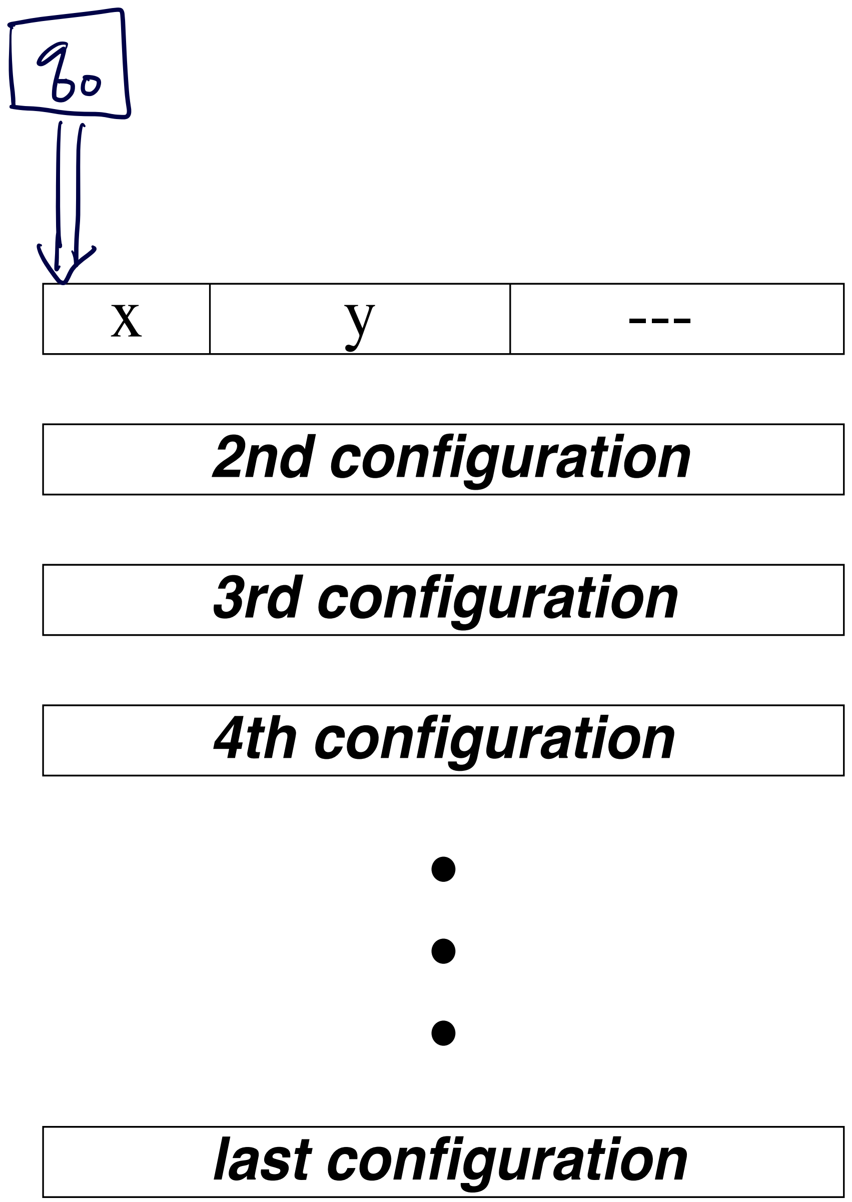
- Consider $A_x(y) := M(x, y)$. Here, $A_x : \{0,1\}^* \to \{0,1\}$
- We will construct a circuit C_x computing A_x
- Finally, we need to show that the mapping $x \mapsto C_x$ is polynomial-time computable

High-level idea: A_x itself is a boolean function

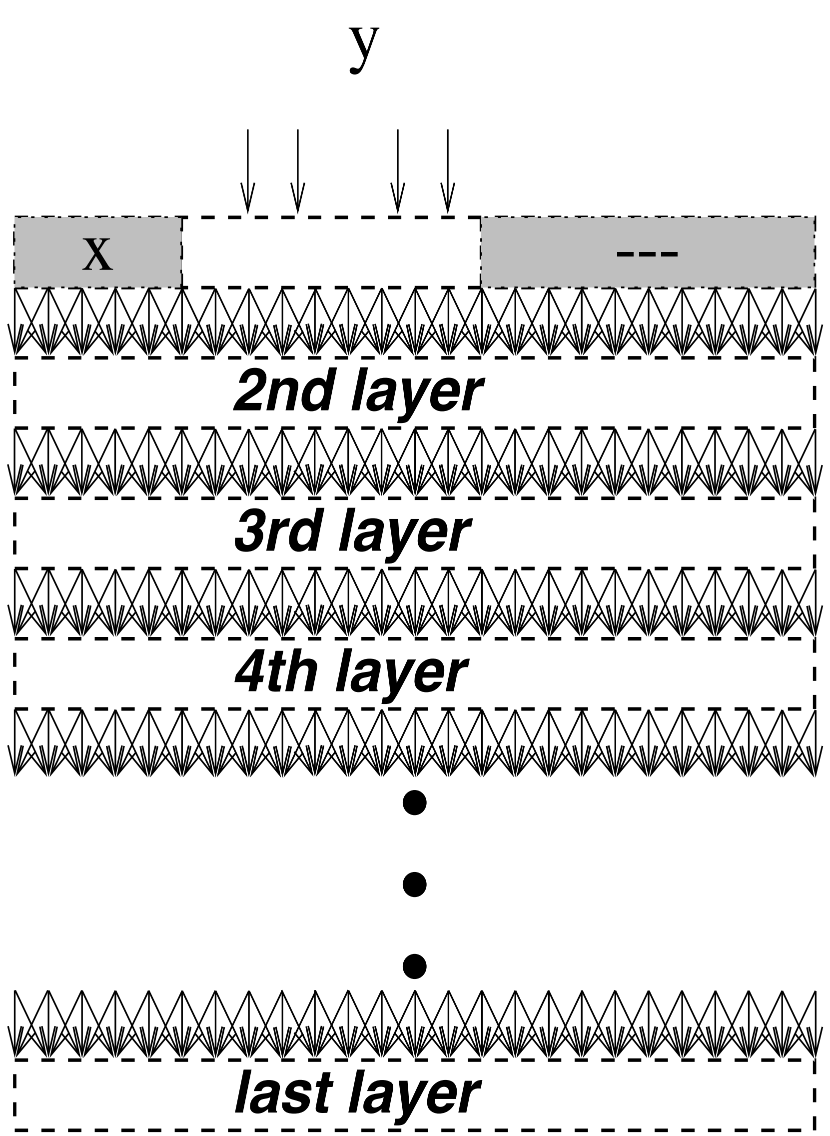
moreover, $R(x) \neq \emptyset$ iff A_x is 'satisfiable'.

Circuit SAT

- First, since we are talking about polynomially-bounded relation R, $|y| \le p(|x|)$ for some polynomial p, so $A_x : \{0,1\}^{p(|x|)} \to \{0,1\}$
  - Really? WLOG, using padding, we can assume that $|y| = p(|x|)$.
- Circuit consists of layers
  - Each layer represents an instantaneous configuration of M
  - For $M(x, y)$, x is hard-wired, but y varies


q_0
x
y

2nd configuration
3rd configuration
4th configuration
last configuration


y
x

2nd layer
3rd layer
4th layer
last layer

$(1,a)$	$(1,\perp)$	$(0,\perp)$	$(y_1,\perp)$	$(y_2,\perp)$	$(\_,\perp)$	$(\_,\perp)$	$(\_,\perp)$	$(\_,\perp)$	$(\_,\perp)$
$(3,\perp)$	$(1,b)$	$(0,\perp)$	$(y_1,\perp)$	$(y_2,\perp)$	$(\_,\perp)$	$(\_,\perp)$	$(\_,\perp)$	$(\_,\perp)$	$(\_,\perp)$
$(3,\perp)$	$(1,\perp)$	$(0,b)$	$(y_1,\perp)$	$(y_2,\perp)$	$(\_,\perp)$	$(\_,\perp)$	$(\_,\perp)$	$(\_,\perp)$	$(\_,\perp)$
$(3,\perp)$	$(1,c)$	$(0,\perp)$							
$(3,c)$	$(1,\perp)$	$(0,\perp)$							
$(1,\perp)$	$(1,f)$	$(0,\perp)$							

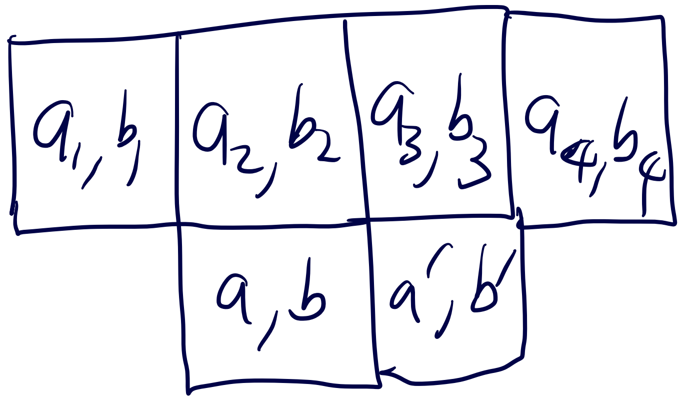
initial configuration
(with input $110y_1y_2$))

$\delta(a,1) = (b,3,+1)$

$(a,b) \in \Sigma \times (Q \cup \{\perp\})$

$(1, q_{halt})$

last configuration



$$a = f(a_1, b_1, a_2, b_2, a_3, b_3)$$

$$b = g(a_1, b_1, a_2, b_2, a_3, b_3)$$

$$\delta : Q \times \Sigma \to Q \times \Sigma \times \{-1, 0, 1\}$$

Encode each square as a bitstring $\in \{0,1\}^c$

(f, g) : can be understood as a function $\{0,1\}^{3c} \longrightarrow \{0,1\}^c$.

$$a' = f(a_2, b_2, a_3, b_3, a_4, b_4)$$

$$b' = g(a_2, b_2, a_3, b_3, a_4, b_4)$$

Circuit SAT

- Circuit consists of layers
  - Given a layer representing an i.c. for M, the circuit computes the next layer representing the next i.c.
  - For each layer, another gadget checks if the layer represents an accepting i.c.
  - Take the AND of all such acceptance checking: this is the output of the circuit

Self-reducibility of NP-complete problems

- $R \in \mathcal{PC}$
- If S_R is $\mathcal{NP}$-complete, then R is self-reducible
  - Step 1: R is reducible to $S'_R := \{\langle x, z\rangle \,:\, \exists y, (x, zy) \in R\}$
  - Step 2: S'_R is reducible to S_R

NP-completeness of SAT

- Cook-Levin theorem: Circuit SAT is NP-complete
- If you want to prove that a problem Π is NP-complete, then
  - First, prove that Π is in $\mathcal{NP}$ (or, in $\mathcal{PC}$ if it is a search problem), and
  - Second, pick your favorite NP-complete problem Π', and reduce Π' to Π
- Now, you have CSAT
- Reduce CSAT to SAT

NP-completeness of SAT

- $\mathtt{SAT}$ is the set of all satisfiable CNF formulae
- $\mathtt{SAT} = \{\phi : \phi \text{ is a satisfiable CNF}\}$
- $R_{\mathtt{SAT}} = \{(\phi, \tau) : \phi(\tau) = 1\}$
- Theorem: $\mathtt{SAT}$ is $\mathcal{NP}$-complete and $R_{\mathtt{SAT}}$ is $\mathcal{PC}$-complete

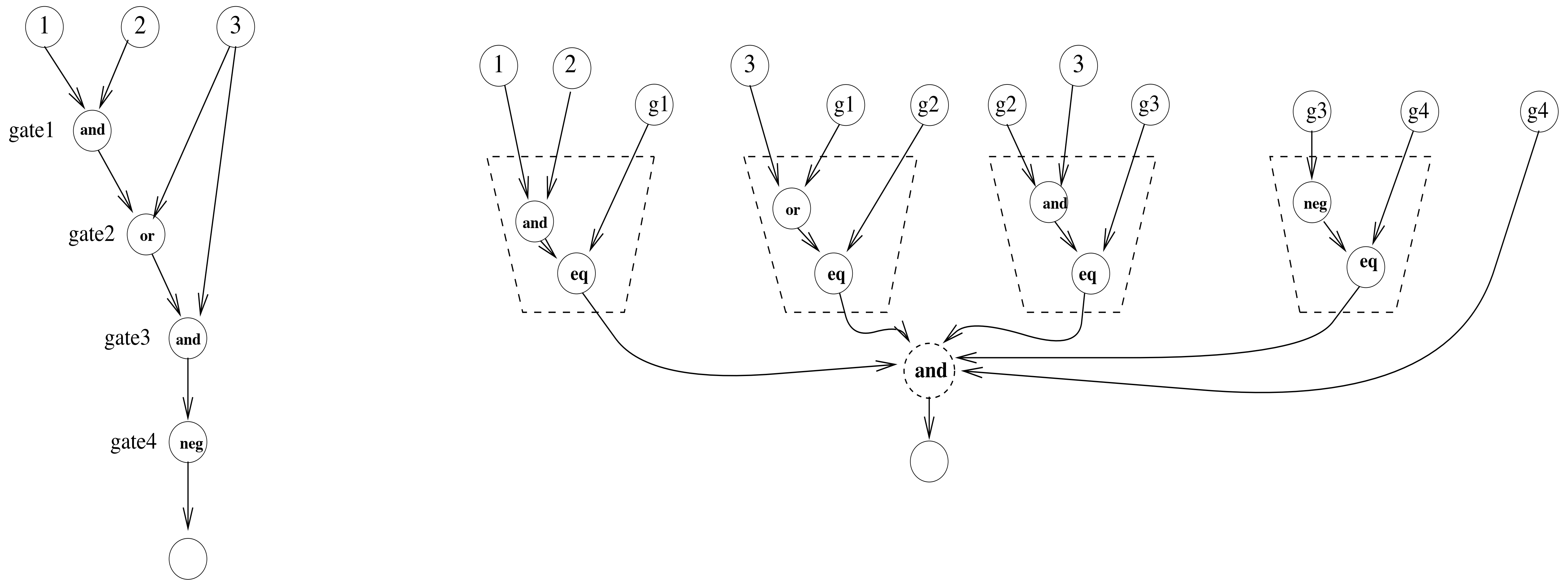
$C \longmapsto \phi$

NP-completeness of SAT

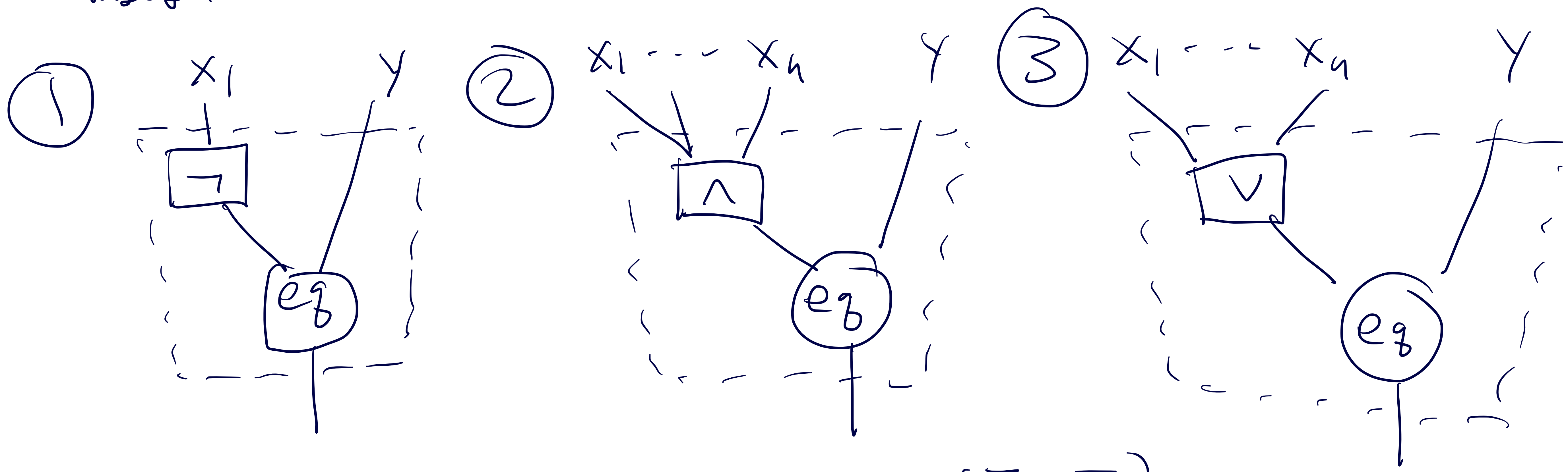
- Theorem: $\mathtt{SAT}$ is $\mathcal{NP}$-complete and $R_{\mathtt{SAT}}$ is $\mathcal{PC}$-complete
  - Proving that $\mathtt{SAT} \in \mathcal{NP}$ and $R_{\mathtt{SAT}} \in \mathcal{PC}$ is easy
  - To prove that $\mathtt{SAT}$ is $\mathcal{NP}$-hard and $R_{\mathtt{SAT}}$ is $\mathcal{PC}$-hard, reduce from $\mathtt{CSAT}$

NP-completeness of SAT

- Idea: introduce new variables corresponding to gate values



What remains: make sure that the following can be expressed as clauses:



Case 1 $\quad eq(x,y) = \neg(x \oplus y) = (x \wedge y) \vee (\bar{x} \wedge \bar{y})$

Then, $eq(\bar{x}_1, y) = (\bar{x}_1 \wedge y) \vee (x_1 \wedge \bar{y})$

$$= ((\bar{x}_1 \wedge y) \vee x_1) \wedge ((\bar{x}_1 \wedge y) \vee \bar{y})$$

$$= \cancel{(\bar{x}_1 \vee x_1)} \wedge (y \vee x_1) \wedge (\bar{x}_1 \vee \bar{y}) \wedge \cancel{(y \vee \bar{y})}$$

Case 1) $eq(\overline{x_1}, y) = (y \vee x_1) \wedge (\overline{x_1} \vee \overline{y})$

Case 2) $eq(x, y) = (y \vee \overline{x}) \wedge (x \vee \overline{y})$, so,

$$eq(x_1 \wedge \cdots \wedge x_n, y) = (\overline{x_1 \wedge \cdots \wedge x_n} \vee y) \wedge ((x_1 \wedge \cdots \wedge x_n) \vee \overline{y})$$

$$= (\overline{x_1} \vee \cdots \vee \overline{x_n} \vee y) \wedge (x_1 \vee \overline{y}) \wedge (x_2 \vee \overline{y}) \wedge \cdots (x_n \vee \overline{y})$$

Case 3) $eq(x_1 \vee \cdots \vee x_n, y) = (\overline{x_1 \vee \cdots \vee x_n} \vee y) \wedge (x_1 \vee \cdots \vee x_n \vee \overline{y})$

$$= ((\overline{x_1} \wedge \cdots \wedge \overline{x_n}) \vee y) \wedge (x_1 \vee \cdots \vee x_n \vee \overline{y})$$

$$= (\overline{x_1} \vee y) \wedge \cdots \wedge (\overline{x_n} \vee y) \wedge (x_1 \vee \cdots \vee x_n \vee \overline{y})$$

NP-completeness of 3SAT

- $\mathtt{3SAT} = \{\phi : \phi \text{ is a satisfiable 3CNF}\}$
- $R_{\mathtt{3SAT}} = \{(\phi, \tau) : \phi(\tau) = 1 \text{ and } \phi \text{ is a 3CNF}\}$
- Theorem $\mathtt{3SAT}$ is $\mathcal{NP}$-complete and $R_{\mathtt{3SAT}}$ is $\mathcal{PC}$-complete
  - Proof: reduce from $\mathtt{SAT}$

proof is really easy: ϕ: CNF $\rightsquigarrow$ ϕ': 3CNF.

$\phi = C_1 \wedge \cdots \wedge C_m$

$C_1 = l_1 \vee l_2 \vee l_3 \vee \cdots \vee l_k = (l_1 \vee l_2) \vee (l_3 \vee \cdots \vee l_k)$

$(l_1 \vee l_2 \vee u) \wedge (\bar{u} \vee l_3 \vee \cdots l_k)$