Lecture 18

CSE 431
Intro to Theory of Computation

Previously $A \le_m^P B$ mapping reduction s.t. map f is polytime computable

- $A \le_m^P B$ and $B \in P \Rightarrow A \in P$
- $A \le_m^P B$ and $B \in NP \Rightarrow A \in NP$

Def^n B is NP-hard iff $\forall A \in NP \; A \le_m^P B$

Def^n B is NP-complete iff

(1) $B \in NP$ and
(2) B is NP-hard

Thm [Cook-Levin] 3SAT is NP-complete

We prove this by first showing a different direct NP-completeness result

CIRCUIT-SAT = $\{\langle C\rangle$: C is a Boolean Circuit with input y s.t. $C(y)=1\}$

Thm CIRCUIT-SAT is NP-complete

Proof (1) CIRCUIT-SAT $\in$ NP

Given $\langle C\rangle$:

certificate: String y for input assignment length $\le |\langle C\rangle|$ ✓

verify: Evaluate C on input y and accept iff value = 1 polytime ✓

(2) Show all $A \in NP$, $A \le_m^p$ CIRCUIT-SAT

Let $A \in NP$

Goal: reduction f s.t.

$$\begin{array}{ccc} A & & \text{CIRCUIT-SAT} \\ x & \xmapsto{f} & \langle C_{A,x}\rangle \end{array}$$

↑ circuit depending on A and x

s.t. for all x : $x \in A \Longleftrightarrow \exists y$ st $\underbrace{C_{A,x}(y)=1}$

Since $A \in NP$
$\exists$ verifier V_A (1-tape TM)
s.t.
- V_A is polytime, say, running time $T(n)$ that is $O(n^k)$
- $\forall x$ $x \in A \Leftrightarrow \exists y$, $|y|$ is $O(n^k)$ s.t. V_A accepts $\langle x,y\rangle$

Idea: create $C_{A,x}$ s.t. $C_{A,x}$ on input x should simulate V_A on input $\langle x,y\rangle$

w.l.o.g. $y \in \{0,1\}^*$ "bits"

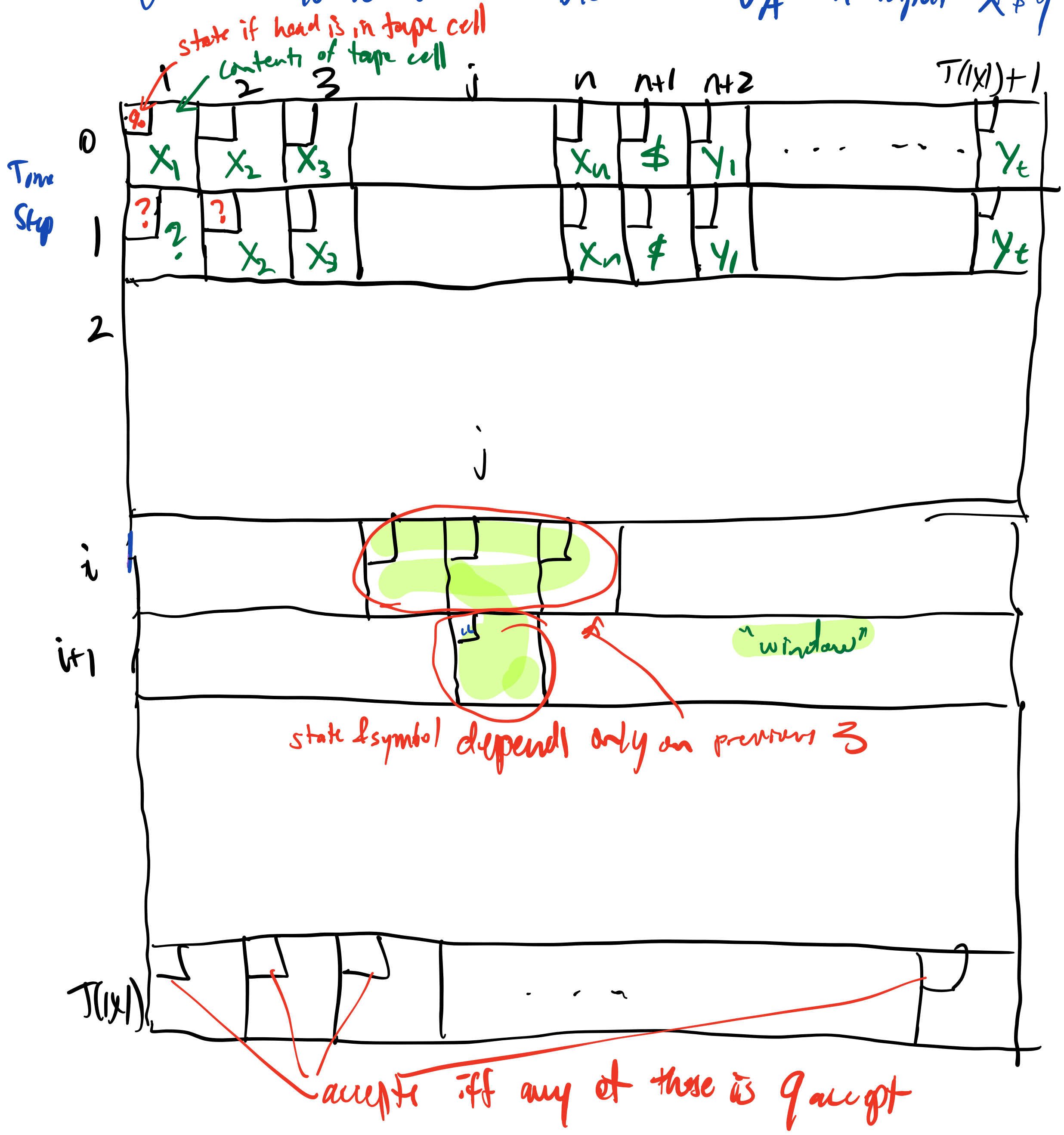
and

$\langle x,y\rangle = x\$y$ $\quad \$ \notin \Gamma$

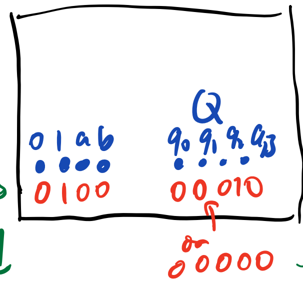
time bound for V_A

$|y| \le T(|x|) - |x|$ (any longer y wouldn't be looked at)

We now look at the "tableau" of V_A on input $x\$y$



Representing each cell in a circuit:

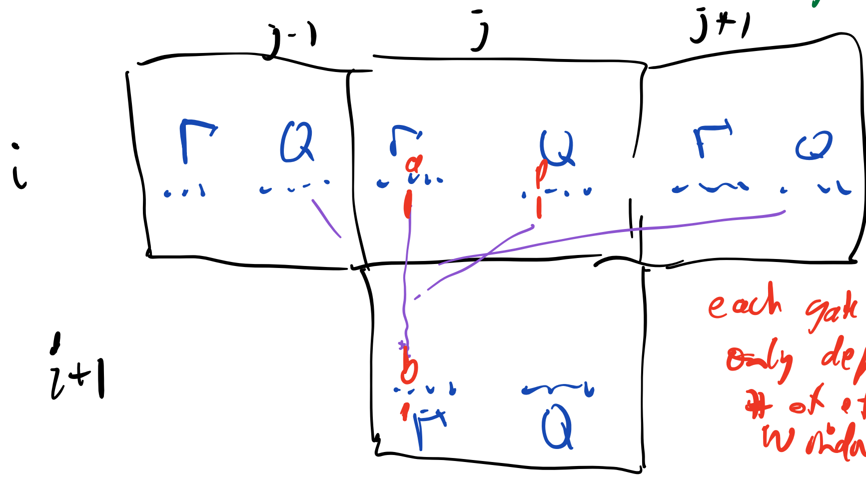


one gate for each symbol in Γ
for each state in Q

"1-hot encoding"

exactly one gate evaluates to 1

at most one gate evaluates to 1.



Use δ function to build circuitry

each gate value only depends on a constant # of other values from the window

e.g. contents are b iff either

- Q gates are all 0's in cell above and cell above has b

or

- Q gate for p in cell above has a 1
  Γ gate for a in cell above has a 1
  and $(p,a) = (q,b,R)$ or (q,b,L) for some $q \in Q$

(for some $p \in Q$, $a \in \Gamma$)

e.g. state is q iff for some $p \in Q$, $a \in \Gamma$, $b \in \Gamma$ either

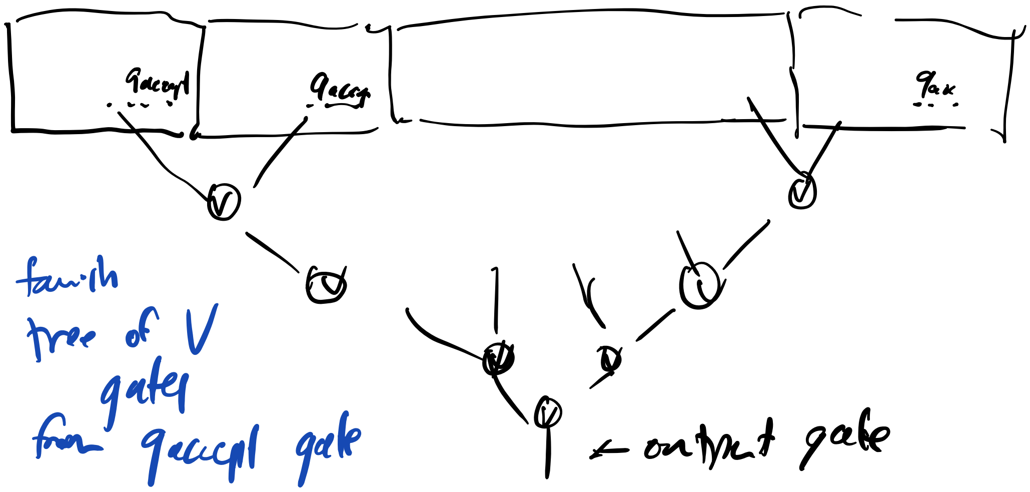
- cell above and to left has gates for $p \in Q$ and $a \in \Gamma$ have value 1 and $\delta(p,a) = (q,b,R)$
- cell above and to right has gates for $p \in Q$ and $a \in \Gamma$ have value 1 and $\delta(p,a) = (q,b,L)$

$C_{A,x}$

The circuit has this same constant-sized piece repeated and linked in an entire grid of $(T/|x|)+1) \times (T/|x|)+1)$ cells (slight change at left end)

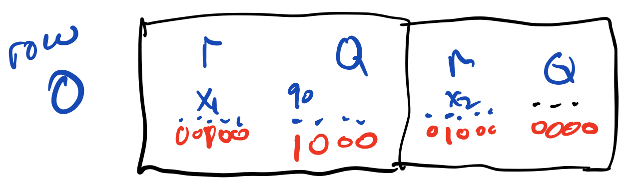
Output: We can assume wlog that q_{accept} values just get copied down to the bottom row (if they exist) as part of this circuit

Want output to be 1 iff V_A accepts $\langle x,y \rangle$
iff $\exists\, q_{accept}$ in final row

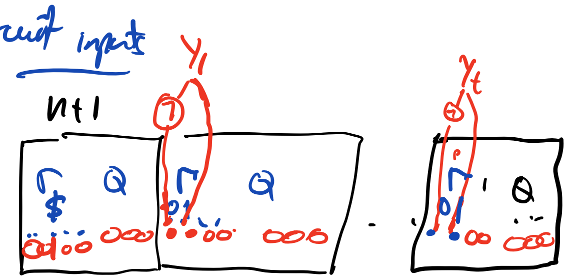


Big fan-in tree of $\vee$ gates from q_{accept} gate

Inputs: x$y :

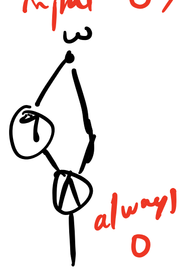


circuit inputs



wlog input 0/1 constants allowed

0:



1:

Note: only symbols possible are 0 and 1

result:

$\neg y_i \; y_i \; 0\,0\,0$

Resulting circuit satisfies $C_{A,x}(y)=1$ iff V_A accepts x & y
and is easy to compute:
Size for $|x|=n$ is $O(T^2(n))$ which is $O(n^{2k})$
polynomial.

$\therefore \forall A \in NP$, $A \leq_m^P$ CIRCUIT-SAT $\blacksquare$

This is the hard work that makes it easier to prove
NP-completeness of everything else:

<u>Claim</u> CIRCUIT-SAT $\leq_m^P C \Rightarrow C$ is NP-hard

<u>Proof</u> We showed $A \leq_m^P B$ and $B \leq_m^P C$
$\Rightarrow A \leq_m^P C$
We simply use CIRCUIT-SAT for B $\blacksquare$

We now prove

<u>Thm</u> 3SAT is NP-complete

<u>Proof</u>: 1. 3SAT $\in$ NP ✓ prev. class

2. <u>Claim</u> CIRCUIT-SAT $\leq_m^P$ 3SAT

Want f: $\langle C \rangle \xmapsto{f} \langle$ 3CNF formula $\varphi \rangle$
s.t. C is SAT $\Longleftrightarrow$ φ is SAT

Now $C(y)=1 \Longleftrightarrow \exists$ values for each gate of C consistent
with input y such that
output gate has value 1.

Design of φ: variables for y
+ variables for each gate of C

clauses represent constraints for gate values being correct

• say output value is 1

Note: gate values are carried on wires: we describe constraints for each gate type

NOT

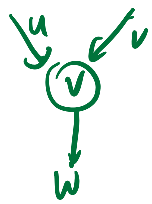
Want $\neg u \Leftrightarrow v$

i.e. $\neg u \rightarrow v$ i.e. $\neg\neg u \vee v$

$v \rightarrow \neg u$ etc

$\boxed{\text{Clauses} \bullet\ u \vee v \quad \bullet\ \neg u \vee \neg v}$

OR



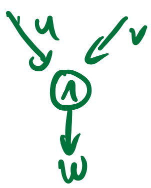
Want $w \Leftrightarrow (u \vee v)$

i.e. $w \rightarrow (u \vee v)$

$(u \vee v) \rightarrow w$ i.e. $u \rightarrow w$, $v \rightarrow w$

$\boxed{\text{Clauses} \bullet\ \neg w \vee u \vee v \quad \bullet\ \neg u \vee w \quad \bullet\ \neg v \vee w}$

AND



Want $w \Leftrightarrow (u \wedge v)$

i.e. $w \rightarrow u$, $w \rightarrow v$

$(u \wedge v) \rightarrow w$

$\boxed{\text{Clauses} \bullet\ \neg w \vee u \quad \bullet\ \neg w \vee v \quad \bullet\ \neg u \vee \neg v \vee w}$

Final formula has clauses like this for each gate plus clause of length 1 for output gate var.
Easy to compute. Clearly correct $\square$

Note: The formula above has ≤ 3 variables in each clause.

<u>Def</u>

Def^n EXACT-3SAT is like 3SAT but every clause has length $= 3$

<u>Thm</u> EXACT3SAT is NP-complete

$$3SAT \leq_m^P EXACT3SAT$$

Idea: for every clause of size 2

logically equivalent

$$(a \vee b) \longmapsto (a \vee b \vee z)(a \vee b \vee \bar{z})$$

for any variable z.

for clause of size 1:

logically equivalent

$$a \mapsto (a \vee z_1 \vee z_2)(a \vee z_1 \vee \bar{z}_2)(a \vee \bar{z}_1 \vee z_2)(a \vee \bar{z}_1 \vee \bar{z}_2)$$

for any two vars z_1, z_2