Lecture 11

Cook-Levin Theorem (contd.), Search vs Decision

Constructing the ϕ_{χ}

Constructing the ϕ_{χ} $x \in L \iff \exists u \in \{0,1\}^{p(|x|)}$, s.t. M(x,u) = 1

Constructing the ϕ_{χ}

$x \in L \iff \exists y \in \{0,1\}^{|x|+p(|x|)}$ and $ID = (ID_1, ID_2, \dots, ID_{p'(|x|)})$, where $|ID_i| = c$, such that:

Constructing the ϕ_x

1) First |x| bits of y = x.

$x \in L \iff \exists y \in \{0,1\}^{|x|+p(|x|)}$ and $ID = (ID_1, ID_2, \dots, ID_{p'(|x|)})$, where $|ID_i| = c$, such that:

Constructing the ϕ_x $x \in L \iff \exists y \in \{0,1\}^{|x|+p(|x|)}$ and $ID = (ID_1, ID_2, ..., ID_{p'(|x|)})$, where $|ID_i| = c$, such that: 1) First |x| bits of y = x. 2) $ID_1 = (q_{start}, \triangleright, \triangleright).$

Constructing the ϕ_x

1) First |x| bits of y = x.

2) $ID_1 = (q_{start}, \triangleright, \triangleright).$

3) $\forall i \in [2,p'(|x|)],$

$x \in L \iff \exists y \in \{0,1\}^{|x|+p(|x|)}$ and $ID = (ID_1, ID_2, \dots, ID_{p'(|x|)})$, where $|ID_i| = c$, such that:

Constructing the ϕ_x

1) First |x| bits of y = x.

2) $ID_1 = (q_{start}, \triangleright, \triangleright).$

3) $\forall i \in [2,p'(|x|)], F_i(ID_{i-1}, y_{inputpos(i)}, ID_{prev(i)}, ID_i) = 1.$

$x \in L \iff \exists y \in \{0,1\}^{|x|+p(|x|)}$ and $ID = (ID_1, ID_2, ..., ID_{p'(|x|)})$, where $|ID_i| = c$, such that:

Constructing the ϕ_r

1) First |x| bits of y = x. 2) $ID_1 = (q_{start}, \triangleright, \triangleright).$ 3) $\forall i \in [2,p'(|x|)], F_i(ID_{i-1}, y_{inputpos(i)}, ID_{prev(i)}, ID_i) = 1.$

4) $ID_{p'(|x|)} = (q_{halt}, _, 1).$

$x \in L \iff \exists y \in \{0,1\}^{|x|+p(|x|)}$ and $ID = (ID_1, ID_2, ..., ID_{p'(|x|)})$, where $|ID_i| = c$, such that:

Constructing ϕ_x from x:

Constructing ϕ_x from x:

• Compute prev(i) and inputpos(i) by running M on $0^{|x|+p(|x|)}$.

Constructing ϕ_x from x:

- Compute prev(i) and inputpos(i) by running M on $0^{|x|+p(|x|)}$.
- Construct boolean formula ϕ_x :

Constructing ϕ_x from x:

- Compute prev(i) and inputpos(i) by running M on $0^{|x|+p(|x|)}$.
- Construct boolean formula ϕ_x :
 - With variables $Y_1, Y_2, \ldots Y_{|x|+p(|x|)}$

Constructing ϕ_x from x:

- Compute prev(i) and inputpos(i) by running M on $0^{|x|+p(|x|)}$.
- Construct boolean formula ϕ_x :

• With variables $Y_1, Y_2, ..., Y_{|x|+p(|x|)}$ and $ID_{i1}, ID_{i2}, ..., ID_{ic}$ for every $i \in [1, p'(|x|)]$.

Constructing ϕ_x from x:

- Compute prev(i) and inputpos(i) by running M on $0^{|x|+p(|x|)}$.
- Construct boolean formula ϕ_x :

• With variables $Y_1, Y_2, ..., Y_{|x|+p(|x|)}$ and $ID_{i1}, ID_{i2}, ..., ID_{ic}$ for every $i \in [1, p'(|x|)]$. • That checks whether y and ID satisfy the AND of conditions 1), 2), 3), and 4).

Constructing ϕ_x from x:

- Compute prev(i) and inputpos(i) by running M on $0^{|x|+p(|x|)}$.
- Construct boolean formula ϕ_x :

1) First |x| bits of y = x. (Linear size ϕ_1 . If x = 101, then $\phi_1 = (Y_1) \land (\neg Y_2) \land (Y_3)$)

• With variables $Y_1, Y_2, ..., Y_{|x|+p(|x|)}$ and $ID_{i1}, ID_{i2}, ..., ID_{ic}$ for every $i \in [1, p'(|x|)]$. • That checks whether y and ID satisfy the AND of conditions 1), 2), 3), and 4).



Constructing ϕ_x from x:

- Compute prev(i) and inputpos(i) by running M on $0^{|x|+p(|x|)}$.
- Construct boolean formula ϕ_x :

 - With variables $Y_1, Y_2, ..., Y_{|x|+p(|x|)}$ and $ID_{i1}, ID_{i2}, ..., ID_{ic}$ for every $i \in [1, p'(|x|)]$. • That checks whether y and ID satisfy the AND of conditions 1), 2), 3), and 4).

- 1) First |x| bits of y = x. (Linear size ϕ_1 . If x = 101, then $\phi_1 = (Y_1) \land (\neg Y_2) \land (Y_3)$)



Constructing the ϕ_{r} $x \in L \iff \exists y \in \{0,1\}^{|x|+p(|x|)}$ and $ID = (ID_1, ID_2, ..., ID_{p'(|x|)})$, where $|ID_i| = c$, such that: 2) $ID_1 = (q_{start}, \triangleright, \triangleright)$. (Constant size ϕ_2) 3) $\forall i \in [2,p'(|x|)], F_i(ID_{i-1}, y_{inputpos(i)}, ID_{prev(i)}, ID_i) = 1.$ 4) $ID_{p'(|x|)} = (q_{halt}, -, 1)$. (Constant size ϕ_4)

Constructing ϕ_x from x:

- Compute prev(i) and inputpos(i) by running M on $0^{|x|+p(|x|)}$.
- Construct boolean formula ϕ_x :

 - With variables $Y_1, Y_2, ..., Y_{|x|+p(|x|)}$ and $ID_{i1}, ID_{i2}, ..., ID_{ic}$ for every $i \in [1, p'(|x|)]$. • That checks whether y and ID satisfy the AND of conditions 1), 2), 3), and 4).

1) First |x| bits of y = x. (Linear size ϕ_1 . If x = 101, then $\phi_1 = (Y_1) \land (\neg Y_2) \land (Y_3)$)



Constructing the ϕ_x $x \in L \iff \exists y \in \{0,1\}^{|x|+p(|x|)}$ and $ID = (ID_1, ID_2, ..., ID_{p'(|x|)})$, where $|ID_i| = c$, such that: 2) $ID_1 = (q_{start}, \triangleright, \triangleright)$. (Constant size ϕ_2) 4) $ID_{p'(|x|)} = (q_{halt}, -, 1)$. (Constant size ϕ_4)

Constructing ϕ_x from x:

- Compute prev(i) and inputpos(i) by running M on $0^{|x|+p(|x|)}$.
- Construct boolean formula ϕ_x :

 - With variables $Y_1, Y_2, ..., Y_{|x|+p(|x|)}$ and $ID_{i1}, ID_{i2}, ..., ID_{ic}$ for every $i \in [1, p'(|x|)]$. • That checks whether y and ID satisfy the AND of conditions 1), 2), 3), and 4).

- 1) First |x| bits of y = x. (Linear size ϕ_1 . If x = 101, then $\phi_1 = (Y_1) \land (\neg Y_2) \land (Y_3)$) 3) $\forall i \in [2, p'(|x|)], F_i(ID_{i-1}, y_{inputpos(i)}, ID_{prev(i)}, ID_i) = 1.$ ((3c + 1).2^(3c+1) size ϕ_3)





Constructing the ϕ_x $x \in L \iff \exists y \in \{0,1\}^{|x|+p(|x|)}$ and $ID = (ID_1, ID_2, ..., ID_{p'(|x|)})$, where $|ID_i| = c$, such that: 2) $ID_1 = (q_{start}, \triangleright, \triangleright)$. (Constant size ϕ_2) 4) $ID_{p'(|x|)} = (q_{halt}, -, 1)$. (Constant size ϕ_4)

Constructing ϕ_x from x:

- Compute prev(i) and inputpos(i) by running M on $0^{|x|+p(|x|)}$.
- Construct boolean formula ϕ_x :

 - With variables $Y_1, Y_2, ..., Y_{|x|+p(|x|)}$ and $ID_{i1}, ID_{i2}, ..., ID_{ic}$ for every $i \in [1, p'(|x|)]$. • That checks whether y and ID satisfy the AND of conditions 1), 2), 3), and 4).

- 1) First |x| bits of y = x. (Linear size ϕ_1 . If x = 101, then $\phi_1 = (Y_1) \land (\neg Y_2) \land (Y_3)$) 3) $\forall i \in [2, p'(|x|)], F_i(ID_{i-1}, y_{inputpos(i)}, ID_{prev(i)}, ID_i) = 1.$ ((3c + 1).2^(3c+1) size ϕ_3)





Idea:

Idea: Reduce SAT to 3SAT by repeatedly breaking down clauses of k > 3 literals into two clauses of almost k/2 many literals.

Idea: Reduce SAT to 3SAT by repeatedly breaking down clauses of k > 3 literals into two clauses of almost k/2 many literals.

 $\phi =$

Idea: Reduce SAT to 3SAT by repeatedly breaking down clauses of k > 3 literals into two clauses of almost k/2 many literals.

 $\boldsymbol{\phi} = (u_1 \lor u_2 \lor \ldots \lor u_k)$

Idea: Reduce SAT to 3SAT by repeatedly breaking down clauses of k > 3 literals into two clauses of almost k/2 many literals.

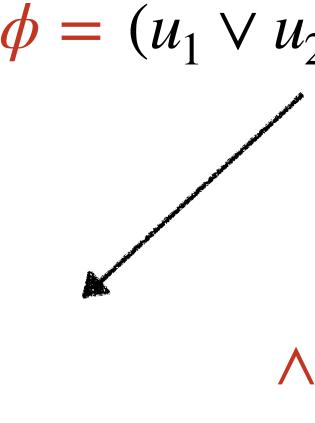
 $\boldsymbol{\phi} = (u_1 \lor u_2 \lor \ldots \lor u_k)$

Idea: Reduce SAT to 3SAT by repeatedly breaking down clauses of k > 3 literals into two clauses of almost k/2 many literals.

 $\phi = (u_1 \lor u_2 \lor \ldots \lor u_k)$

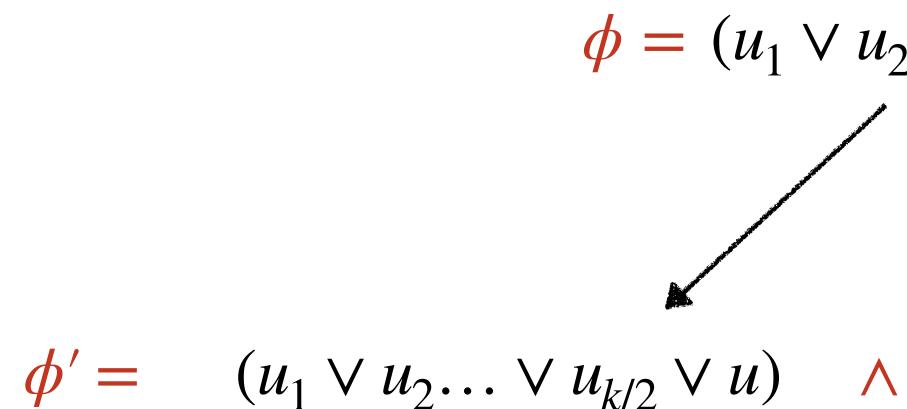
 $\phi' =$

two clauses of almost k/2 many literals.



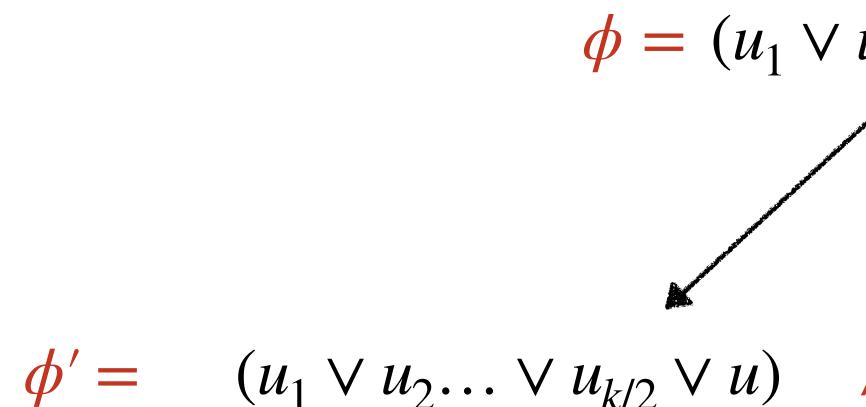
 $\boldsymbol{\phi} = (u_1 \lor u_2 \lor \ldots \lor u_k)$

two clauses of almost k/2 many literals.



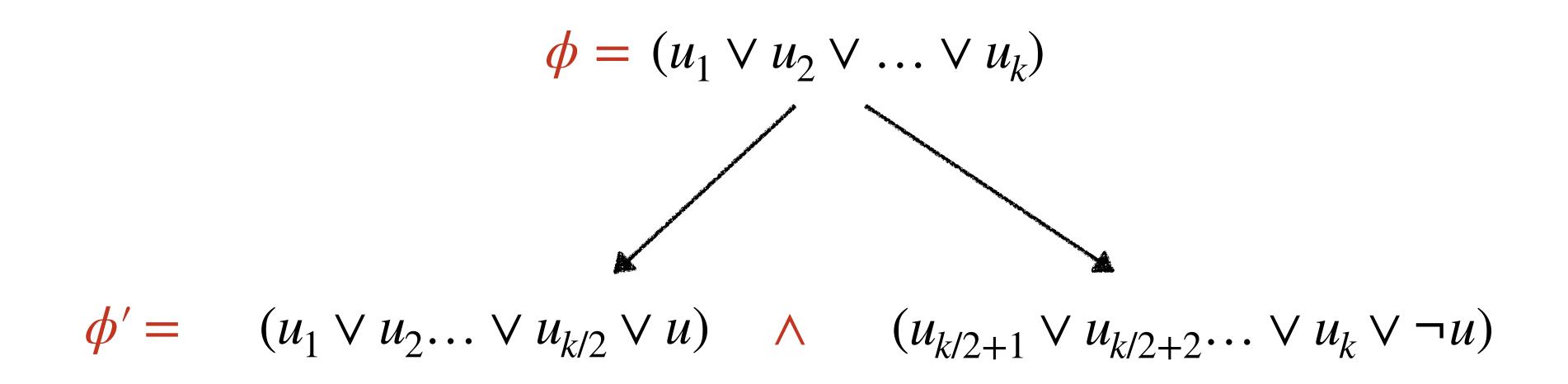
 $\boldsymbol{\phi} = (u_1 \lor u_2 \lor \ldots \lor u_k)$

two clauses of almost k/2 many literals.



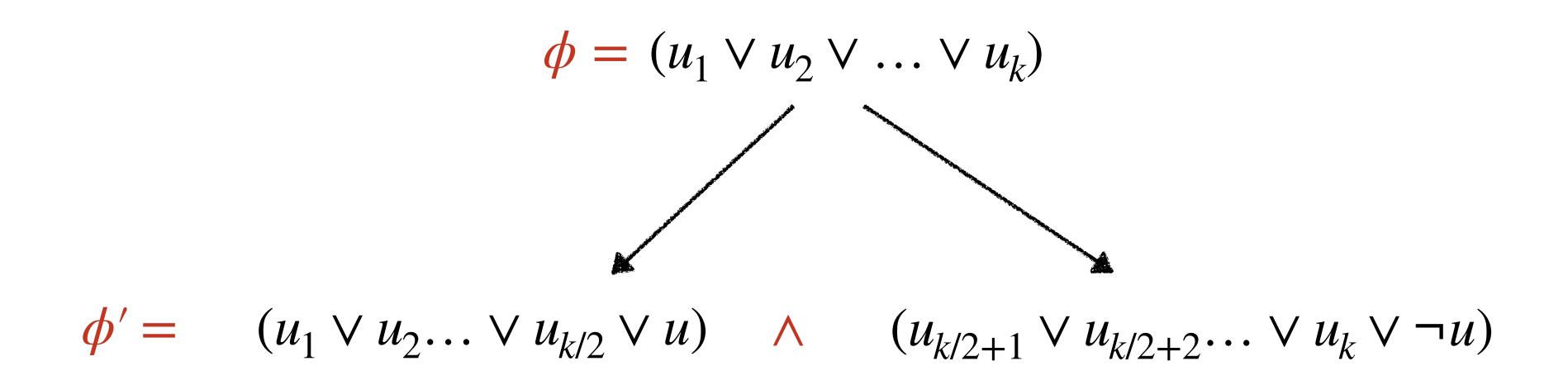
 $\boldsymbol{\phi} = (u_1 \lor u_2 \lor \ldots \lor u_k)$ $\phi' = (u_1 \lor u_2 \dots \lor u_{k/2} \lor u) \land (u_{k/2+1} \lor u_{k/2+2} \dots \lor u_k \lor \neg u)$

two clauses of almost k/2 many literals.



Time to break a clause of k literals into a 3CNF formula:

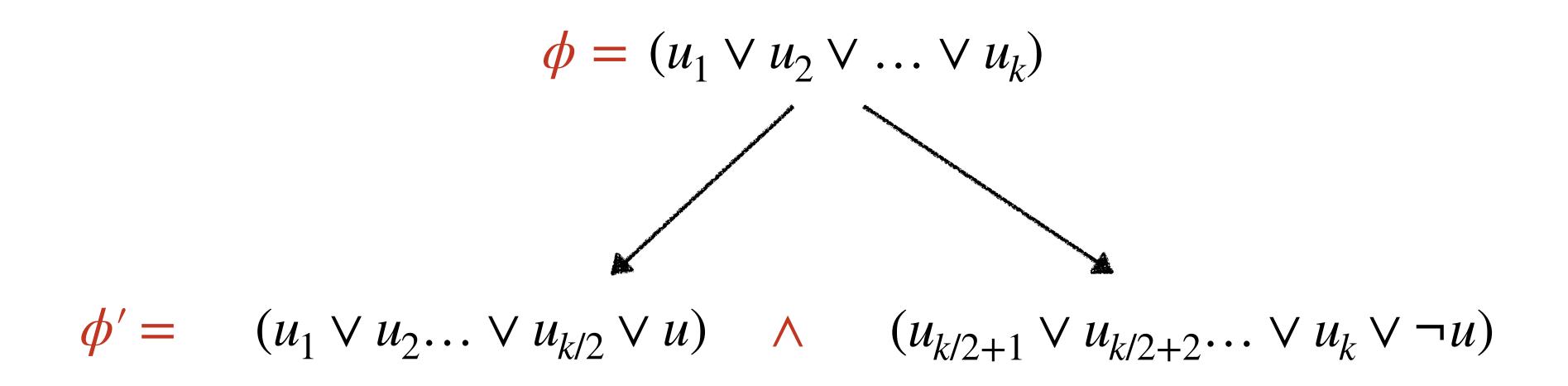
two clauses of almost k/2 many literals.



Time to break a clause of k literals into a 3CNF formula:

• T(k) =

two clauses of almost k/2 many literals.

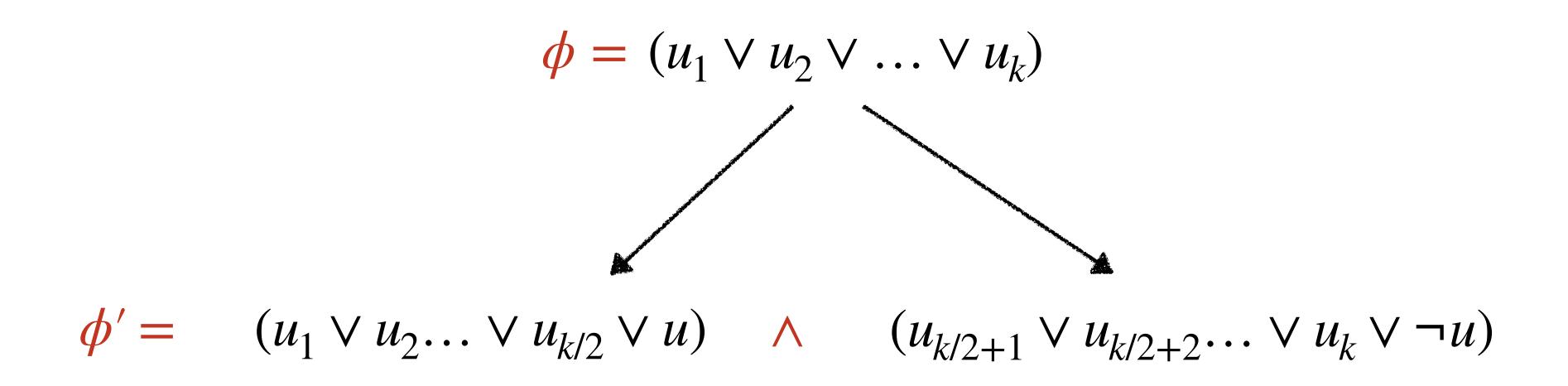


Time to break a clause of k literals into a 3CNF formula:

•
$$T(k) =$$

•
$$T(3) = c$$

two clauses of almost k/2 many literals.



Time to break a clause of k literals into a 3CNF formula:

• T(k) = 2.T(k/2 + 1) + O(k)

•
$$T(3) = c$$

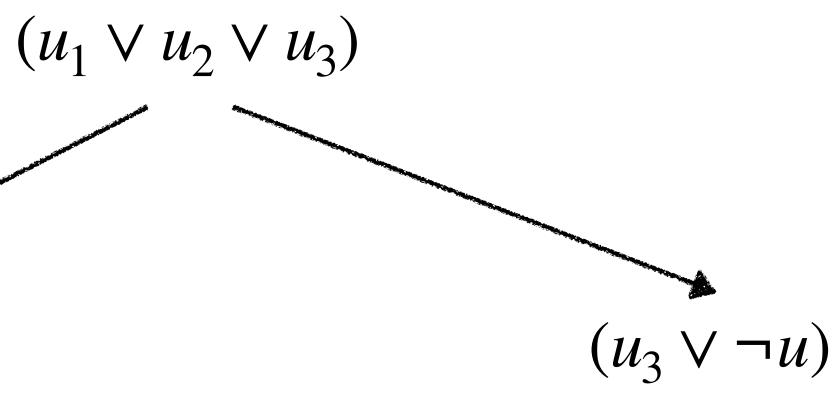
Isn't 2SAT also NP-Complete?

Isn't 2SAT also NP-Complete?

 $(u_1 \lor u_2 \lor u_3)$

Isn't 2SAT also NP-Complete?

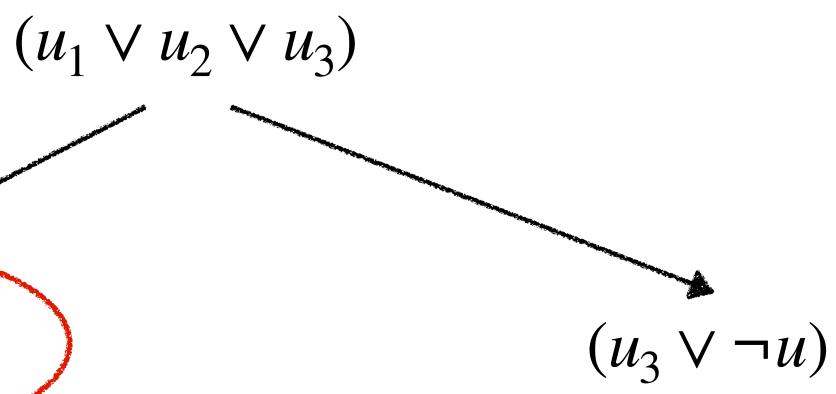
 $(u_1 \land u_1 \land u_1 \land u_2 \lor u)$



Isn't 2SAT also NP-Complete?

$(u_1 \lor u_2 \lor u)$

Further breakdown isn't possible.



Theorem: Suppose that P = NP. Then, for every $L \in NP$ and a verifier TM M for L, there is



Theorem: Suppose that P = NP. Then, for every $L \in NP$ and a verifier TM M for L, there is a polytime TM B that on input $x \in L$ outputs a certificate for x w.r.t L and M, if $x \in L$.



Theorem: Suppose that P = NP. Then, for every $L \in NP$ and a verifier TM M for L, there is a polytime TM B that on input $x \in L$ outputs a certificate for x w.r.t L and M, if $x \in L$. **Proof:** Let L = SAT and A be a polytime TM that decides SAT.



Proof: Let L = SAT and A be a polytime TM that decides SAT.

- **Theorem:** Suppose that P = NP. Then, for every $L \in NP$ and a verifier TM M for L, there is a polytime TM B that on input $x \in L$ outputs a certificate for x w.r.t L and M, if $x \in L$.
- Create a polytime TM B that on input ϕ , finds a satisfying assignment for ϕ if it exists.



Proof: Let L = SAT and A be a polytime TM that decides SAT.

Idea:

- **Theorem:** Suppose that P = NP. Then, for every $L \in NP$ and a verifier TM M for L, there is a polytime TM B that on input $x \in L$ outputs a certificate for x w.r.t L and M, if $x \in L$.
- Create a polytime TM B that on input ϕ , finds a satisfying assignment for ϕ if it exists.



Proof: Let L = SAT and A be a polytime TM that decides SAT.

Idea: Let $\phi = (u_1 \lor u_2) \land (\neg u_2 \lor \neg u_3) \land (\neg u_1 \lor u_3)$

- **Theorem:** Suppose that P = NP. Then, for every $L \in NP$ and a verifier TM M for L, there is a polytime TM B that on input $x \in L$ outputs a certificate for x w.r.t L and M, if $x \in L$.
- Create a polytime TM B that on input ϕ , finds a satisfying assignment for ϕ if it exists.



Proof: Let L = SAT and A be a polytime TM that decides SAT.

Idea: Let $\phi = (u_1 \lor u_2) \land (\neg u_2 \lor \neg u_3) \land (\neg u_1 \lor u_3)$

 $\phi_{u_1=0} = (0 \lor u_2) \land (\neg u_2 \lor \neg u_3) \land (1 \lor u_3)$

- **Theorem:** Suppose that P = NP. Then, for every $L \in NP$ and a verifier TM M for L, there is a polytime TM B that on input $x \in L$ outputs a certificate for x w.r.t L and M, if $x \in L$.
- Create a polytime TM B that on input ϕ , finds a satisfying assignment for ϕ if it exists.



Proof: Let L = SAT and A be a polytime TM that decides SAT.

Idea: Let
$$\phi = (u_1 \lor u_2) \land (\neg u_2 \lor \neg u_3) \land (\neg u_1 \lor u_3)$$

$$\phi_{u_1=0} = (0 \lor u_2) \land (\neg u_2 \lor \neg u_2)$$

$$\phi_{u_1=1} = (1 \lor u_2) \land (\neg u_2 \lor \neg u_2)$$

- **Theorem:** Suppose that P = NP. Then, for every $L \in NP$ and a verifier TM M for L, there is a polytime TM B that on input $x \in L$ outputs a certificate for x w.r.t L and M, if $x \in L$.
- Create a polytime TM B that on input ϕ , finds a satisfying assignment for ϕ if it exists.

 $(u_3) \wedge (1 \vee u_3)$

 $(u_3) \wedge (0 \vee u_3)$



Proof: Let L = SAT and A be a polytime TM that decides SAT.

Idea: Let
$$\phi = (u_1 \lor u_2) \land (\neg u_2 \lor \neg u_3) \land (\neg u_1 \lor u_3)$$

$$\phi_{u_1=0} = (0 \lor u_2) \land (\neg u_2 \lor \neg u_2)$$

$$\phi_{u_1=1} = (1 \lor u_2) \land (\neg u_2 \lor \neg u_2)$$

- **Theorem:** Suppose that P = NP. Then, for every $L \in NP$ and a verifier TM M for L, there is a polytime TM B that on input $x \in L$ outputs a certificate for x w.r.t L and M, if $x \in L$.
- Create a polytime TM B that on input ϕ , finds a satisfying assignment for ϕ if it exists.

 $u_3) \wedge (1 \vee u_3) = (u_2) \wedge (\neg u_2 \vee \neg u_3)$ $(u_3) \wedge (0 \vee u_3)$



Proof: Let L = SAT and A be a polytime TM that decides SAT.

Idea: Let
$$\phi = (u_1 \lor u_2) \land (\neg u_2 \lor \neg u_3) \land (\neg u_1 \lor u_3)$$

$$\phi_{u_1=0} = (0 \lor u_2) \land (\neg u_2 \lor \neg u_2)$$

$$\phi_{u_1=1} = (1 \lor u_2) \land (\neg u_2 \lor \neg u_2)$$

- **Theorem:** Suppose that P = NP. Then, for every $L \in NP$ and a verifier TM M for L, there is a polytime TM B that on input $x \in L$ outputs a certificate for x w.r.t L and M, if $x \in L$.
- Create a polytime TM B that on input ϕ , finds a satisfying assignment for ϕ if it exists.

 $u_3) \wedge (1 \vee u_3) = (u_2) \wedge (\neg u_2 \vee \neg u_3)$ $(u_3) \wedge (0 \vee u_3) = (\neg u_2 \vee \neg u_3) \wedge (u_2)$



Proof: Let L = SAT and A be a polytime TM that decides SAT.

Idea: Let
$$\phi = (u_1 \lor u_2) \land (\neg u_2 \lor \neg u_3) \land (\neg u_1 \lor u_3)$$

$$\phi_{u_1=0} = (0 \lor u_2) \land (\neg u_2 \lor \neg u_3) \land (1 \lor u_3) = (u_2) \land (\neg u_2 \lor \neg u_3)$$

$$\phi_{u_1=1} = (1 \lor u_2) \land (\neg u_2 \lor \neg u_3) \land (0 \lor u_3) = (\neg u_2 \lor \neg u_3) \land (u_2)$$

$$\phi_{u_1=1} = (1 \lor u_2) \land (\neg u_2 \lor \neg u_2)$$

If ϕ is satisfiable then either $\phi_{u_1=0}$ or $\phi_{u_1=1}$ is satisfiable.

- **Theorem:** Suppose that P = NP. Then, for every $L \in NP$ and a verifier TM M for L, there is a polytime TM B that on input $x \in L$ outputs a certificate for x w.r.t L and M, if $x \in L$.
- Create a polytime TM B that on input ϕ , finds a satisfying assignment for ϕ if it exists.



Proof: Let L = SAT and A be a polytime TM that decides SAT.

- **Theorem:** Suppose that P = NP. Then, for every $L \in NP$ and a verifier TM M for L, there is a polytime TM B that on input $x \in L$ outputs a certificate for x w.r.t L and M, if $x \in L$.
- Create a polytime TM B that on input ϕ , finds a satisfying assignment for ϕ if it exists.



Proof: Let L = SAT and A be a polytime TM that decides SAT. $B(\phi)$:

- **Theorem:** Suppose that P = NP. Then, for every $L \in NP$ and a verifier TM M for L, there is a polytime TM B that on input $x \in L$ outputs a certificate for x w.r.t L and M, if $x \in L$.
- Create a polytime TM B that on input ϕ , finds a satisfying assignment for ϕ if it exists.



Proof: Let L = SAT and A be a polytime TM that decides SAT. $B(\phi)$:

if (ϕ is not satisfiable) return NULL

- **Theorem:** Suppose that P = NP. Then, for every $L \in NP$ and a verifier TM M for L, there is a polytime TM B that on input $x \in L$ outputs a certificate for x w.r.t L and M, if $x \in L$.
- Create a polytime TM B that on input ϕ , finds a satisfying assignment for ϕ if it exists.



Proof: Let L = SAT and A be a polytime TM that decides SAT. $B(\phi)$: if (ϕ is not satisfiable) return NULL

for i = 1 to n

- **Theorem:** Suppose that P = NP. Then, for every $L \in NP$ and a verifier TM M for L, there is a polytime TM B that on input $x \in L$ outputs a certificate for x w.r.t L and M, if $x \in L$.
- Create a polytime TM B that on input ϕ , finds a satisfying assignment for ϕ if it exists.



Proof: Let L = SAT and A be a polytime TM that decides SAT. $B(\phi)$:

if (ϕ is not satisfiable) return NULL

for i = 1 to n // n = # of variables of ϕ .

- **Theorem:** Suppose that P = NP. Then, for every $L \in NP$ and a verifier TM M for L, there is a polytime TM B that on input $x \in L$ outputs a certificate for x w.r.t L and M, if $x \in L$.
- Create a polytime TM B that on input ϕ , finds a satisfying assignment for ϕ if it exists.



Proof: Let L = SAT and A be a polytime TM that decides SAT. $B(\phi)$:

if (ϕ is not satisfiable) return NULL

for i = 1 to n // n = # of variables of ϕ .

if $(\phi_{u=0} \text{ is satisfiable})$

- **Theorem:** Suppose that P = NP. Then, for every $L \in NP$ and a verifier TM M for L, there is a polytime TM B that on input $x \in L$ outputs a certificate for x w.r.t L and M, if $x \in L$.
- Create a polytime TM B that on input ϕ , finds a satisfying assignment for ϕ if it exists.



Proof: Let L = SAT and A be a polytime TM that decides SAT. $B(\phi)$:

if (ϕ is not satisfiable) return NULL

for i = 1 to n // n = # of variables of ϕ .

if $(\phi_{u=0} \text{ is satisfiable})$ $u_i = 0, \, \phi = \phi_{u_i = 0}$

- **Theorem:** Suppose that P = NP. Then, for every $L \in NP$ and a verifier TM M for L, there is a polytime TM B that on input $x \in L$ outputs a certificate for x w.r.t L and M, if $x \in L$.
- Create a polytime TM B that on input ϕ , finds a satisfying assignment for ϕ if it exists.



Proof: Let L = SAT and A be a polytime TM that decides SAT. $B(\phi)$:

if (ϕ is not satisfiable) return NULL

for i = 1 to n // n = # of variables of ϕ .

if $(\phi_{u=0} \text{ is satisfiable})$ $u_i = 0, \, \phi = \phi_{u_i = 0}$ else if ($\phi_{u_i=1}$ is satisfiable)

- **Theorem:** Suppose that P = NP. Then, for every $L \in NP$ and a verifier TM M for L, there is a polytime TM B that on input $x \in L$ outputs a certificate for x w.r.t L and M, if $x \in L$.
- Create a polytime TM B that on input ϕ , finds a satisfying assignment for ϕ if it exists.



Proof: Let L = SAT and A be a polytime TM that decides SAT. $B(\phi)$:

- if (ϕ is not satisfiable) return NULL
- for i = 1 to n // n = # of variables of ϕ .

if $(\phi_{u=0} \text{ is satisfiable})$ $u_i = 0, \, \phi = \phi_{u_i = 0}$ else if ($\phi_{u_i=1}$ is satisfiable) $u_i = 1, \, \phi = \phi_{u_i = 1}$

- **Theorem:** Suppose that P = NP. Then, for every $L \in NP$ and a verifier TM M for L, there is a polytime TM B that on input $x \in L$ outputs a certificate for x w.r.t L and M, if $x \in L$.
- Create a polytime TM B that on input ϕ , finds a satisfying assignment for ϕ if it exists.



Theorem: Suppose that P = NP. Then, for every $L \in NP$ and a verifier TM M for L, there is a polytime TM B that on input $x \in L$ outputs a certificate for x w.r.t L and M, if $x \in L$. **Proof:** Let L = SAT and A be a polytime TM that decides SAT. Create a polytime TM B that on input ϕ , finds a satisfying assignment for ϕ if it exists. $B(\phi)$:

- if (ϕ is not satisfiable) return NULL
- for i = 1 to n // n = # of variables of ϕ .

if ($\phi_{u_i=0}$ is satisfiable) $u_i = 0, \ \phi = \phi_{u_i = 0}$ else if ($\phi_{u_i=1}$ is satisfiable) $u_i = 1, \, \phi = \phi_{u_i = 1}$

Runtime of B if ϕ has n variables:



Theorem: Suppose that P = NP. Then, for every $L \in NP$ and a verifier TM M for L, there is a polytime TM B that on input $x \in L$ outputs a certificate for x w.r.t L and M, if $x \in L$. **Proof:** Let L = SAT and A be a polytime TM that decides SAT. Create a polytime TM B that on input ϕ , finds a satisfying assignment for ϕ if it exists. $B(\phi)$:

- if (ϕ is not satisfiable) return NULL
- for i = 1 to n // n = # of variables of ϕ .

if $(\phi_{u_i=0} \text{ is satisfiable})$ $u_i = 0, \ \phi = \phi_{u_i = 0}$ else if ($\phi_{u_i=1}$ is satisfiable) $u_i = 1$, $\phi = \phi_{u_i = 1}$

Runtime of B if ϕ has n variables: 2n + 1 calls to A



Theorem: Suppose that P = NP. Then, for every $L \in NP$ and a verifier TM M for L, there is a polytime TM B that on input $x \in L$ outputs a certificate for x w.r.t L and M, if $x \in L$. **Proof:** Let L = SAT and A be a polytime TM that decides SAT. Create a polytime TM B that on input ϕ , finds a satisfying assignment for ϕ if it exists. $B(\phi)$:

- if (ϕ is not satisfiable) return NULL
- for i = 1 to n // n = # of variables of ϕ .

if ($\phi_{u_i=0}$ is satisfiable) $u_i = 0, \ \phi = \phi_{u_i = 0}$ else if ($\phi_{u_i=1}$ is satisfiable) $u_i = 1, \, \phi = \phi_{u_i = 1}$

Runtime of B if ϕ has n variables: 2n + 1 calls to A and some polytime work



- $x \in L \iff \exists y \in \{0,1\}^{|x|+p(|x|)}$ and $ID = (ID_1, ID_2, \dots, ID_{p'(|x|)})$, where $|\mathcal{S}_i| = c$, such that: 1) First |x| bits of y = x. (Linear size ϕ_1 . If x = 101, then $\phi_1 = (Y_1) \land (\neg Y_2) \land (Y_3)$) 2) $ID_1 = (q_{start}, \triangleright, \triangleright)$. (Constant size ϕ_2)

 - 3) $\forall i \in [2, p'(|x|)], F_i(ID_{i-1}, y_{inputpos(i)}, ID_{prev(i)}, ID_i) = 1.$ ((3c + 1).2^(3c+1) size ϕ_3) 4) $ID_{p'(|x|)} = (q_{halt}, -, 1)$. (Constant size ϕ_4)

Recall: For $L \in NP$:

- $x \in L \iff \exists y \in \{0,1\}^{|x|+p(|x|)}$ and $ID = (ID_1, ID_2, \dots, ID_{p'(|x|)})$, where $|\mathcal{S}_i| = c$, such that: 1) First |x| bits of y = x. (Linear size ϕ_1 . If x = 101, then $\phi_1 = (Y_1) \land (\neg Y_2) \land (Y_3)$) 2) $ID_1 = (q_{start}, \triangleright, \triangleright)$. (Constant size ϕ_2)

 - 3) $\forall i \in [2, p'(|x|)], F_i(ID_{i-1}, y_{inputpos(i)}, ID_{prev(i)}, ID_i) = 1.$ ((3c + 1).2^(3c+1) size ϕ_3)
 - 4) $ID_{p'(|x|)} = (q_{halt}, -, 1)$. (Constant size ϕ_4)

Observation: Satisfying assignment for $\phi_x = \phi_1 \wedge \phi_2 \wedge \phi_3 \wedge \phi_4$ contains certificate *u* for *x*.



Recall: For $L \in NP$:

- $x \in L \iff \exists y \in \{0,1\}^{|x|+p(|x|)}$ and $ID = (ID_1, ID_2, \dots, ID_{p'(|x|)})$, where $|\mathcal{S}_i| = c$, such that: 1) First |x| bits of y = x. (Linear size ϕ_1 . If x = 101, then $\phi_1 = (Y_1) \land (\neg Y_2) \land (Y_3)$) 2) $ID_1 = (q_{start}, \triangleright, \triangleright)$. (Constant size ϕ_2)

 - 3) $\forall i \in [2, p'(|x|)], F_i(ID_{i-1}, y_{inputpos(i)}, ID_{prev(i)}, ID_i) = 1.$ ((3c + 1).2^(3c+1) size ϕ_3)
 - 4) $ID_{p'(|x|)} = (q_{halt}, -, 1)$. (Constant size ϕ_4)

Observation: Satisfying assignment for $\phi_x = \phi_1 \wedge \phi_2 \wedge \phi_3 \wedge \phi_4$ contains certificate u for x.

Let $L \in \mathbf{NP}$ and M be its verifier.



- $x \in L \iff \exists y \in \{0,1\}^{|x|+p(|x|)}$ and $ID = (ID_1, ID_2, \dots, ID_{p'(|x|)})$, where $|\mathcal{S}_i| = c$, such that: 1) First |x| bits of y = x. (Linear size ϕ_1 . If x = 101, then $\phi_1 = (Y_1) \land (\neg Y_2) \land (Y_3)$) 2) $ID_1 = (q_{start}, \triangleright, \triangleright)$. (Constant size ϕ_2)

 - 3) $\forall i \in [2, p'(|x|)], F_i(ID_{i-1}, y_{inputpos(i)}, ID_{prev(i)}, ID_i) = 1.$ ((3c + 1).2^(3c+1) size ϕ_3) 4) $ID_{p'(|x|)} = (q_{halt}, -, 1)$. (Constant size ϕ_4)

- **Observation:** Satisfying assignment for $\phi_x = \phi_1 \wedge \phi_2 \wedge \phi_3 \wedge \phi_4$ contains certificate u for x.
- Let $L \in NP$ and M be its verifier. We can find the certificate of $x \in L$ in the following way:



- $x \in L \iff \exists y \in \{0,1\}^{|x|+p(|x|)}$ and $ID = (ID_1, ID_2, \dots, ID_{p'(|x|)})$, where $|\mathcal{S}_i| = c$, such that: 1) First |x| bits of y = x. (Linear size ϕ_1 . If x = 101, then $\phi_1 = (Y_1) \land (\neg Y_2) \land (Y_3)$) 2) $ID_1 = (q_{start}, \triangleright, \triangleright)$. (Constant size ϕ_2)

 - 3) $\forall i \in [2, p'(|x|)], F_i(ID_{i-1}, y_{inputpos(i)}, ID_{prev(i)}, ID_i) = 1.$ ((3c + 1).2^(3c+1) size ϕ_3)
 - 4) $ID_{p'(|x|)} = (q_{halt}, -, 1)$. (Constant size ϕ_4)
- **Observation:** Satisfying assignment for $\phi_x = \phi_1 \wedge \phi_2 \wedge \phi_3 \wedge \phi_4$ contains certificate u for x.
- Let $L \in NP$ and M be its verifier. We can find the certificate of $x \in L$ in the following way:
- Map x to ϕ_x .



- $x \in L \iff \exists y \in \{0,1\}^{|x|+p(|x|)}$ and $ID = (ID_1, ID_2, \dots, ID_{p'(|x|)})$, where $|\mathcal{S}_i| = c$, such that: 1) First |x| bits of y = x. (Linear size ϕ_1 . If x = 101, then $\phi_1 = (Y_1) \land (\neg Y_2) \land (Y_3)$) 2) $ID_1 = (q_{start}, \triangleright, \triangleright)$. (Constant size ϕ_2)

 - 3) $\forall i \in [2, p'(|x|)], F_i(ID_{i-1}, y_{inputpos(i)}, ID_{prev(i)}, ID_i) = 1.$ ((3c + 1).2^(3c+1) size ϕ_3) 4) $ID_{p'(|x|)} = (q_{halt}, -, 1)$. (Constant size ϕ_4)

- Map x to ϕ_x .
- Find a satisfying assignment for ϕ_x and get certificate of x from it.

- **Observation:** Satisfying assignment for $\phi_x = \phi_1 \wedge \phi_2 \wedge \phi_3 \wedge \phi_4$ contains certificate u for x.
- Let $L \in NP$ and M be its verifier. We can find the certificate of $x \in L$ in the following way:



- $x \in L \iff \exists y \in \{0,1\}^{|x|+p(|x|)}$ and $ID = (ID_1, ID_2, \dots, ID_{p'(|x|)})$, where $|\mathcal{S}_i| = c$, such that: 1) First |x| bits of y = x. (Linear size ϕ_1 . If x = 101, then $\phi_1 = (Y_1) \land (\neg Y_2) \land (Y_3)$) 2) $ID_1 = (q_{start}, \triangleright, \triangleright)$. (Constant size ϕ_2)

 - 3) $\forall i \in [2, p'(|x|)], F_i(ID_{i-1}, y_{inputpos(i)}, ID_{prev(i)}, ID_i) = 1.$ ((3c + 1).2^(3c+1) size ϕ_3) 4) $ID_{p'(|x|)} = (q_{halt}, -, 1)$. (Constant size ϕ_4)

- Map x to ϕ_x .
- Find a satisfying assignment for ϕ_x and get certificate of x from it.

- **Observation:** Satisfying assignment for $\phi_x = \phi_1 \wedge \phi_2 \wedge \phi_3 \wedge \phi_4$ contains certificate u for x.
- Let $L \in NP$ and M be its verifier. We can find the certificate of $x \in L$ in the following way:

