Lecture 11

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

Constructing the ϕ_x

Constructing the ϕ_x

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

Constructing the ϕ_x

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

Constructing the ϕ_x

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

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

Constructing the ϕ_x

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

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

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

Constructing the ϕ_x

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

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

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

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

Constructing the ϕ_x

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

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

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

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

Constructing the ϕ_x

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

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

Constructing the ϕ_x

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

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

Constructing ϕ_x from x:

Constructing the ϕ_x

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

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

Constructing ϕ_x from x:

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

Constructing the ϕ_x

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

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

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 the ϕ_x

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

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

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 the ϕ_x

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

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

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

Constructing the ϕ_x

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

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

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

Constructing the ϕ_x

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

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

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

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

Constructing the ϕ_x

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

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

2) $ID_1 = (q_{start}, \triangleright, \triangleright)$. (Constant size ϕ_2)

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

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

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

Constructing the ϕ_x

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

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

2) $ID_1 = (q_{start}, \triangleright, \triangleright)$. (Constant size ϕ_2)

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

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

Constructing the ϕ_x

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

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

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_i})

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

Constructing the ϕ_x

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

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

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_i})

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

∎

3SAT **is** NP-Complete

3SAT is NP-Complete

Idea:

3SAT is NP-Complete

Idea: Reduce *SAT* to *3SAT* by repeatedly breaking down clauses of $k > 3$ literals into

3SAT is NP-Complete

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

3SAT is NP-Complete

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

$$\phi =$$

3SAT is NP-Complete

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

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

3SAT is NP-Complete

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

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

3SAT is NP-Complete

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

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

3SAT is NP-Complete

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

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

$$\phi' = \qquad \wedge$$

3SAT is NP-Complete

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

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

$$\phi' = \quad (u_1 \vee u_2 \ldots \vee u_{k/2} \vee u) \quad \wedge$$

3SAT is NP-Complete

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

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

$$\phi' = \quad (u_1 \vee u_2 \ldots \vee u_{k/2} \vee u) \quad \wedge \quad (u_{k/2+1} \vee u_{k/2+2} \ldots \vee u_k \vee \neg u)$$

3SAT is NP-Complete

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

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

$$\phi' = \quad (u_1 \vee u_2 \ldots \vee u_{k/2} \vee u) \quad \wedge \quad (u_{k/2+1} \vee u_{k/2+2} \ldots \vee u_k \vee \neg u)$$

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

3SAT is NP-Complete

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

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

$$\phi' = \quad (u_1 \vee u_2 \ldots \vee u_{k/2} \vee u) \quad \wedge \quad (u_{k/2+1} \vee u_{k/2+2} \ldots \vee u_k \vee \neg u)$$

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

- $T(k) =$

3SAT is NP-Complete

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

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

$$\phi' = \quad (u_1 \vee u_2 \ldots \vee u_{k/2} \vee u) \quad \wedge \quad (u_{k/2+1} \vee u_{k/2+2} \ldots \vee u_k \vee \neg u)$$

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

- $T(k) =$
- $T(3) = c$

3SAT is NP-Complete

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

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

$$\phi' = (u_1 \vee u_2 \ldots \vee u_{k/2} \vee u) \quad \wedge \quad (u_{k/2+1} \vee u_{k/2+2} \ldots \vee u_k \vee \neg u)$$

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 2*SAT* also NP-Complete?

Isn't 2*SAT* also NP-Complete?

$$(u_1 \lor u_2 \lor u_3)$$

Isn't 2*SAT* also NP-Complete?

$$\begin{tikzcd} & (u_1 \lor u_2 \lor u_3) \arrow[dl] \arrow[dr] & \\ (u_1 \lor u_2 \lor u) & & (u_3 \lor \neg u) \end{tikzcd}$$

Isn't 2*SAT* also NP-Complete?

$$\begin{tikzcd} & (u_1 \lor u_2 \lor u_3) \arrow[dl] \arrow[dr] & \\ (u_1 \lor u_2 \lor u) & & (u_3 \lor \neg u) \end{tikzcd}$$

Further breakdown isn't possible.

Search vs Decision

Search vs Decision

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

Search vs Decision

Theorem: Suppose that $\mathsf{P} = \mathsf{NP}$. Then, for every $L \in \mathsf{NP}$ 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 $x \in L$.

Search vs Decision

Theorem: Suppose that $\mathbf{P} = \mathbf{NP}$. Then, for every $L \in \mathbf{NP}$ 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 $L = \mathit{SAT}$ and A be a polytime TM that decides *SAT*.

Search vs Decision

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.

Search vs Decision

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.

Idea:

Search vs Decision

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.

Idea: Let $\phi = (u_1 \vee u_2) \wedge (\neg u_2 \vee \neg u_3) \wedge (\neg u_1 \vee u_3)$

Search vs Decision

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.

Idea: Let $\phi = (u_1 \vee u_2) \wedge (\neg u_2 \vee \neg u_3) \wedge (\neg u_1 \vee u_3)$

$$\phi_{u_1=0} = (0 \vee u_2) \wedge (\neg u_2 \vee \neg u_3) \wedge (1 \vee u_3)$$

Search vs Decision

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.

Idea: Let $\phi = (u_1 \vee u_2) \wedge (\neg u_2 \vee \neg u_3) \wedge (\neg u_1 \vee u_3)$

$$\phi_{u_1=0} = (0 \vee u_2) \wedge (\neg u_2 \vee \neg u_3) \wedge (1 \vee u_3)$$

$$\phi_{u_1=1} = (1 \vee u_2) \wedge (\neg u_2 \vee \neg u_3) \wedge (0 \vee u_3)$$

Search vs Decision

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.

Idea: Let $\phi = (u_1 \vee u_2) \wedge (\neg u_2 \vee \neg u_3) \wedge (\neg u_1 \vee u_3)$

$$\phi_{u_1=0} = (0 \vee u_2) \wedge (\neg u_2 \vee \neg u_3) \wedge (1 \vee u_3) = (u_2) \wedge (\neg u_2 \vee \neg u_3)$$

$$\phi_{u_1=1} = (1 \vee u_2) \wedge (\neg u_2 \vee \neg u_3) \wedge (0 \vee u_3)$$

Search vs Decision

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.

Idea: Let $\phi = (u_1 \vee u_2) \wedge (\neg u_2 \vee \neg u_3) \wedge (\neg u_1 \vee u_3)$

$$\phi_{u_1=0} = (0 \vee u_2) \wedge (\neg u_2 \vee \neg u_3) \wedge (1 \vee u_3) = (u_2) \wedge (\neg u_2 \vee \neg u_3)$$

$$\phi_{u_1=1} = (1 \vee u_2) \wedge (\neg u_2 \vee \neg u_3) \wedge (0 \vee u_3) = (\neg u_2 \vee \neg u_3) \wedge (u_2)$$

Search vs Decision

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.

Idea: Let $\phi = (u_1 \vee u_2) \wedge (\neg u_2 \vee \neg u_3) \wedge (\neg u_1 \vee u_3)$

$$\phi_{u_1=0} = (0 \vee u_2) \wedge (\neg u_2 \vee \neg u_3) \wedge (1 \vee u_3) = (u_2) \wedge (\neg u_2 \vee \neg u_3)$$

$$\phi_{u_1=1} = (1 \vee u_2) \wedge (\neg u_2 \vee \neg u_3) \wedge (0 \vee u_3) = (\neg u_2 \vee \neg u_3) \wedge (u_2)$$

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

Search vs Decision

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.

Search vs Decision

Theorem: Suppose that $\mathbf{P} = \mathbf{NP}$. Then, for every $L \in \mathbf{NP}$ 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 $L = \textit{SAT}$ 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.

$B(\phi)$**:**

Search vs Decision

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

Search vs Decision

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

Search vs Decision

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.

Search vs Decision

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)

Search vs Decision

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)

$u_i = 0$, $\phi = \phi_{u_i=0}$

Search vs Decision

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)

Search vs Decision

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)

$u_i = 1,\ \phi = \phi_{u_i=1}$

Search vs Decision

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:

Search vs Decision

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

Search vs Decision

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

Search vs Decision

Search vs Decision

Recall: For $L \in \mathsf{NP}$:

Search vs Decision

Recall: For $L \in$ NP:

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

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

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_i})

4) $ID_{p'(|x|)} = (q_{halt}, \_, 1)$. (Constant size ϕ_4)

Search vs Decision

Recall: For $L \in$ NP:

$x \in L \iff \exists y \in \{0,1\}^{|x|+p(|x|)}$ and $ID = (ID_1, ID_2, \ldots, ID_{p'(|x|)})$, 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_i})

4) $ID_{p'(|x|)} = (q_{halt}, \_, 1)$. (Constant size ϕ_4)

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

Search vs Decision

Recall: For $L \in$ NP:

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

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

Search vs Decision

Recall: For $L \in$ **NP**:

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

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

Search vs Decision

Recall: For $L \in \mathbf{NP}$:

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

1) First $|x|$ bits of $y = x$. (Linear size ϕ_1. If $x = 101$, then $\phi_1 = (Y_1) \wedge (\neg Y_2) \wedge (Y_3)$)
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_i})
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. We can find the certificate of $x \in L$ in the following way:

- Map x to ϕ_x.

Search vs Decision

Recall: For $L \in$ **NP**:

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

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

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_i})

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

Search vs Decision

Recall: For $L \in$ **NP**:

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

1) First $|x|$ bits of $y = x$. (Linear size ϕ_1. If $x = 101$, then $\phi_1 = (Y_1) \wedge (\neg Y_2) \wedge (Y_3)$)
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_i})
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.
- Find a satisfying assignment for ϕ_x and get certificate of x from it.

$\blacksquare$