Lecture 5: **NP** and beyond

Arijit Bishnu

09.02.2010
Outline

1. Another Class: coNP
2. The Classes EXP and NEXP
3. Cook Levin’s Theorem
Outline

1. Another Class: coNP
2. The Classes EXP and NEXP
3. Cook Levin’s Theorem
Understanding coNP

Recall Definition of class NP

A language $L \subseteq \{0, 1\}^*$ is in NP if there exists a polynomial $p : \mathbb{N} \rightarrow \mathbb{N}$ and a polynomial-time TM $M$ such that for every $x \in \{0, 1\}^*$,

$$x \in L \iff \exists u \in \{0, 1\}^{p(|x|)} \text{ such that } M(x, u) = 1$$
Another Class: coNP

The Classes EXP and NEXP

Cook Levin’s Theorem

Understanding coNP

Recall Definition of class NP

A language $L \subseteq \{0, 1\}^*$ is in NP if there exists a polynomial $p : \mathbb{N} \rightarrow \mathbb{N}$ and a polynomial-time TM $M$ such that for every $x \in \{0, 1\}^*$,

$$x \in L \iff \exists u \in \{0, 1\}^{p(|x|)} \text{ such that } M(x, u) = 1$$

Simply Speaking

An input string $x$ is a YES instance iff $\exists$ a short $u$ such that $M(x, u) = 1$.
Understanding coNP

Recall Definition of class NP
A language $L \subseteq \{0, 1\}^*$ is in NP if there exists a polynomial $p : \mathbb{N} \rightarrow \mathbb{N}$ and a polynomial-time TM $M$ such that for every $x \in \{0, 1\}^*$,

$$x \in L \iff \exists u \in \{0, 1\}^{p(|x|)} \text{ such that } M(x, u) = 1$$

Simply Speaking
An input string $x$ is a YES instance iff $\exists$ a short $u$ such that $M(x, u) = 1$.

Negate the above
An input string $x$ is a NO instance iff $\forall$ short $u$, it is the case that $M(x, u) = 0$. 
Another Definition of coNP

For every $L \subseteq \{0, 1\}^*$, we say that $L \in \text{coNP}$ if there exists a polynomial $p : \mathbb{N} \rightarrow \mathbb{N}$ and a polynomial-time TM $M$ such that for every $x \in \{0, 1\}^*$,

$$x \in L \iff \forall u \in \{0, 1\}^{p(|x|)} \text{ such that } M(x, u) = 0$$

[Note the use of $\forall$ in coNP definition instead of $\exists$ in NP definition]
An Alternate Definition of coNP

For every $L \subseteq \{0, 1\}^*$, we say that $L \in \text{coNP}$ if there exists a polynomial $p : \mathbb{N} \rightarrow \mathbb{N}$ and a polynomial-time TM $M$ such that for every $x \in \{0, 1\}^*$,

$$x \in L \iff \forall u \in \{0, 1\}^{p(|x|)} \text{ such that } M(x, u) = 0$$

[Note the use of $\forall$ in coNP definition instead of $\exists$ in NP definition]

Definition: coNP-complete

A language is coNP-complete if it $\in \text{coNP}$ and every coNP language is poly-time reducible to it.
Exploring Relations between P, NP and coNP

Lemma

$P \subseteq NP \cap coNP$
Exploring Relations between P, NP and coNP

Lemma

\[ P \subseteq \text{NP} \cap \text{coNP} \]

Proof

\[ \text{coP} (= P) \subseteq \text{coNP} \]. So, the result follows.
Exploring Relations between P, NP and coNP

Lemma

\[ P \subseteq NP \cap coNP \]

Proof

\[ coP (= P) \subseteq coNP. \] So, the result follows.

Good characterizations and \( NP \cap coNP \)

If \( L \in NP \cap coNP \), then \( L \) has the following property:
Exploring Relations between P, NP and coNP

**Lemma**

\[ P \subseteq NP \cap coNP \]

**Proof**

coP (= P) \(\subseteq\) coNP. So, the result follows.

**Good characterizations and NP \(\cap\) coNP**

If \( L \in NP \cap coNP \), then \( L \) has the following property:

- For a **YES** answer, there is a **short proof**.
Exploring Relations between P, NP and coNP

Lemma

\[ P \subseteq NP \cap coNP \]

Proof

\[ coP \ ( = P ) \subseteq coNP. \] So, the result follows.

Good characterizations and \( NP \cap coNP \)

If \( L \in NP \cap coNP \), then \( L \) has the following property:

- For a **YES** answer, there is a short proof.
- For a **NO** answer, there is also a short proof.
Exploring Relations between P, NP and coNP

Lemma

\[ P \subseteq NP \cap coNP \]

Proof

\[ coP (= P) \subseteq coNP. \] So, the result follows.

Good characterizations and NP \( \cap \) coNP

If \( L \in NP \cap coNP \), then \( L \) has the following property:

- For a **YES** answer, there is a short proof.
- For a **NO** answer, there is also a short proof.
- Look at the decision version of **Max-Flow** problem.
Exploring Relations between P, NP and coNP

Lemma

\[ P \subseteq NP \cap coNP \]

Proof

\[ coP (= P) \subseteq coNP. \text{ So, the result follows.} \]

Good characterizations and NP \( \cap \) coNP

If \( L \in NP \cap coNP \), then \( L \) has the following property:

- For a *YES* answer, there is a short proof.
- For a *NO* answer, there is also a short proof.
- Look at the decision version of *Max-Flow* problem.
- There is a short proof of the *YES* answer via Max-Flow algorithm and there is also a short proof of the *NO* answer via exhibiting a *cut*. 
Exploring Relations between P, NP and coNP

Is $P = NP \cap \text{coNP}$?

So, is $P = NP \cap \text{coNP}$? No one knows till now. Neither there is any strong opinion on this.
Is $P = NP \cap coNP$?

So, is $P = NP \cap coNP$? No one knows till now. Neither there is any strong opinion on this.

What about the relation between $NP$ and $coNP$?
Exploring Relations between $P$, $NP$ and $coNP$

**Is $P = NP \cap coNP?$**

So, is $P = NP \cap coNP$? No one knows till now. Neither there is any strong opinion on this.

**What about the relation between $NP$ and $coNP?$**

- People believe $NP \neq coNP$ just like the belief of $P \neq NP$. 
Exploring Relations between P, NP and coNP

Is $P = NP \cap \text{coNP}$?

So, is $P = NP \cap \text{coNP}$? No one knows till now. Neither there is any strong opinion on this.

What about the relation between NP and coNP?

- People believe $NP \neq \text{coNP}$ just like the belief of $P \neq \text{NP}$.
- The reason is: It is difficult to believe that as there exists short proofs of YES instances, there will also exist short proofs of the NO instances.
Exploring Relations between P, NP and coNP

Is NP \neq \text{coNP}? Proving this would be a bigger step than proving P \neq NP. The next theorem shows that.

**Theorem**

If NP \neq \text{coNP}, then P \neq NP.
Exploring Relations between P, NP and coNP

Is NP $\neq$ coNP? Proving this would be a bigger step than proving $P \neq NP$. The next theorem shows that.

**Theorem**
If NP $\neq$ coNP, then P $\neq$ NP.

**Proof (via the contrapositive, i.e. $P = NP \implies NP = coNP$)**
Exploring Relations between P, NP and coNP

Is \( NP \neq coNP \)? Proving this would be a bigger step than proving \( P \neq NP \). The next theorem shows that.

**Theorem**

If \( NP \neq coNP \), then \( P \neq NP \).

**Proof (via the contraposition, i.e. \( P = NP \implies NP = coNP \))**

\[
\begin{align*}
L \in NP & \implies L \in P \\
L \in coNP & \implies \overline{L} \in P \\
\overline{L} \in NP & \implies \overline{L} \in P \\
\therefore L \in coNP.
\end{align*}
\]
Exploring Relations between P, NP and coNP

Is NP ≠ coNP? Proving this would be a bigger step than proving P ≠ NP. The next theorem shows that.

**Theorem**

If NP ≠ coNP, then P ≠ NP.

**Proof (via the contrapositive, i.e. P = NP ⇒ NP = coNP)**

\[ L \in \text{NP} \implies L \in \text{P} \implies \overline{L} \in \text{P} \implies \overline{L} \in \text{NP} \implies L \in \text{coNP}. \]

\[ L \in \text{coNP} \implies \overline{L} \in \text{NP} \implies \overline{L} \in \text{P} \implies L \in \text{P} \implies L \in \text{NP}. \]
Outline

1. Another Class: coNP
2. The Classes EXP and NEXP
3. Cook Levin’s Theorem
Exponential Analogue of P and NP

**Definition: The Class EXP**

\[ \text{EXP} = \bigcup_{c \geq 0} \text{DTIME}(2^{n^c}) \]
Definition: The Class EXP

\[ \text{EXP} = \bigcup_{c \geq 0} \text{DTIME}(2^{n^c}) \]

Definition: The Class NEXP

\[ \text{NEXP} = \bigcup_{c \geq 0} \text{NTIME}(2^{n^c}) \]
Exponential Analogue of P and NP

**Definition: The Class EXP**

\[ \text{EXP} = \bigcup_{c \geq 0} \text{DTIME}(2^{n^c}) \]

**Definition: The Class NEXP**

\[ \text{NEXP} = \bigcup_{c \geq 0} \text{NTIME}(2^{n^c}) \]

**Lemma**

\[ P \subseteq \text{NP} \subseteq \text{EXP} \subseteq \text{NEXP} \]

Because every problem in NP can be solved in exponential time by a brute force search for the certificate.
Exponential Analogue of P and NP

Definition: The Class EXP

\[ \text{EXP} = \bigcup_{c \geq 0} \text{DTIME}(2^{n^c}) \]

Definition: The Class NEXP

\[ \text{NEXP} = \bigcup_{c \geq 0} \text{NTIME}(2^{n^c}) \]

Lemma

\[ P \subseteq \text{NP} \subseteq \text{EXP} \subseteq \text{NEXP} \]

Proof

Because every problem in NP can be solved in exponential time by a brute force search for the certificate.
Interplay of EXP, NEXP and P, NP

Theorem
If EXP $\neq$ NEXP, then P $\neq$ NP
Interplay of EXP, NEXP and P, NP

Theorem
If \( \text{EXP} \neq \text{NEXP} \), then \( \text{P} \neq \text{NP} \)

Proof (via the contraposition, i.e. \( \text{P} = \text{NP} \implies \text{EXP} = \text{NEXP} \))
### Theorem

If $\text{EXP} \neq \text{NEXP}$, then $\text{P} \neq \text{NP}$

### Proof (via the contrapositive, i.e. $\text{P} = \text{NP} \implies \text{EXP} = \text{NEXP}$)

1. Suppose, $L \in \text{NTIME}(2^{nc})$ and NDTM $M$ decides it.
Interplay of EXP, NEXP and P, NP

**Theorem**
If \( \text{EXP} \neq \text{NEXP} \), then \( \text{P} \neq \text{NP} \)

**Proof (via the contrapositive, i.e. \( \text{P} = \text{NP} \implies \text{EXP} = \text{NEXP} \))**

- Suppose, \( L \in \text{NTIME}(2^{n^c}) \) and NDTM \( M \) decides it.
- We use a **padding technique** where every string in the language is **padded** with a string of useless symbols.
Interplay of EXP, NEXP and P, NP

Theorem
If \( \text{EXP} \neq \text{NEXP} \), then \( \text{P} \neq \text{NP} \)

Proof (via the contrapositive, i.e. \( \text{P} = \text{NP} \implies \text{EXP} = \text{NEXP} \))

- Suppose, \( L \in \text{NTIME}(2^{n^c}) \) and NDTM \( M \) decides it.
- We use a padding technique where every string in the language is padded with a string of useless symbols.
- Let \( L_{\text{pad}} = \{ < x, 1^{2|x|^c} > \mid x \in L \} \).
Another Class: $\text{coNP}$

The Classes $\text{EXP}$ and $\text{NEXP}$

Cook Levin's Theorem

**Lemma**

$L_{\text{pad}} \in \text{NP}.$
Lemma

$L_{pad} \in \text{NP}$.

Proof (Designing a NDTM for $L_{pad}$)

Given $y$, check if $\exists$ a string $z$ s.t. $y = <z, 1^{|x|c}>$. If not, output REJECT. Else, run $M$ on $z$ for $2^{|z|c}$ steps and output the answer. Running time is polynomial in $|y|$. So, $L_{pad} \in \text{NP}$. As we assumed NP = P, so, $L_{pad} \in \text{P}$.

Now, if $L_{pad} \in \text{P}$, $L \in \text{EXP}$. (WHY?) To determine if $x \in L$, we just pad the input and decide whether it is in $L_{pad}$ using the poly-time NDTM for $L_{pad}$.
Lemma

$L_{pad} \in \text{NP}$.

Proof (Designing a NDTM for $L_{pad}$)

- Given $y$, check if $\exists$ a string $z$ s.t. $y = < z, 1^{2|x|^c} >$. 
Lemma

$L_{pad} \in \text{NP}$.

Proof (Designing a NDTM for $L_{pad}$)

- Given $y$, check if $\exists$ a string $z$ s.t. $y = \langle z, 1^{2|x|c} \rangle$.
- If not, output REJECT. Else, run $M$ on $z$ for $2|z|^c$ steps and output the answer. Running time is polynomial in $|y|$.
Lemma

$L_{pad} \in \text{NP}$.

Proof (Designing a NDTM for $L_{pad}$)

- Given $y$, check if $\exists$ a string $z$ s.t. $y = \langle z, 1^{2|x|^c} \rangle$.
- If not, output REJECT. Else, run $M$ on $z$ for $2^{|z|^c}$ steps and output the answer. Running time is polynomial in $|y|$.
- So, $L_{pad} \in \text{NP}$. As we assumed $\text{NP} = \text{P}$, so, $L_{pad} \in \text{P}$.
Lemma

$L_{pad} \in \text{NP}.$

Proof (Designing a NDTM for $L_{pad}$)

- Given $y$, check if $\exists$ a string $z$ s.t. $y = \langle z, 1^{2|x|^c} \rangle$.
- If not, output REJECT. Else, run $M$ on $z$ for $2|z|^c$ steps and output the answer. Running time is polynomial in $|y|$.
- So, $L_{pad} \in \text{NP}$. As we assumed $\text{NP} = \text{P}$, so, $L_{pad} \in \text{P}$.
- Now, if $L_{pad} \in \text{P}$, $L \in \text{EXP}$. (WHY?)
Lemma

$L_{pad} \in \text{NP}$. 

Proof (Designing a NDTM for $L_{pad}$)

- Given $y$, check if $\exists$ a string $z$ s.t. $y = \langle z, 1^{2|x|^c} \rangle$.
- If not, output REJECT. Else, run $M$ on $z$ for $2|z|^c$ steps and output the answer. Running time is polynomial in $|y|$.
- So, $L_{pad} \in \text{NP}$. As we assumed $\text{NP} = \text{P}$, so, $L_{pad} \in \text{P}$.
- Now, if $L_{pad} \in \text{P}$, $L \in \text{EXP}$. (WHY?)
- To determine if $x \in L$, we just pad the input and decide whether it is in $L_{pad}$ using the poly-time NDTM for $L_{pad}$.
Outline

1. Another Class: coNP
2. The Classes EXP and NEXP
3. Cook Levin’s Theorem
Warm Up

What is needed to be proved?

- We need to find a poly-time reduction that turns any $x \in \{0, 1\}^*$ into a CNF formula $\varphi_x$ such that $x \in L$ iff $\varphi_x$ is satisfiable.
Warm Up

What is needed to be proved?

- We need to find a poly-time reduction that turns any \( x \in \{0, 1\}^* \) into a CNF formula \( \varphi_x \) such that \( x \in L \) iff \( \varphi_x \) is satisfiable.

- We only know that \( L \in \text{NP} \). The reduction has to rely only on the definition of computation.
What is needed to be proved?

- We need to find a poly-time reduction that turns any \( x \in \{0, 1\}^* \) into a CNF formula \( \varphi_x \) such that \( x \in L \) iff \( \varphi_x \) is satisfiable.

- We only know that \( L \in \text{NP} \). The reduction has to rely only on the definition of computation.

- We take help of the fact that any algorithm that takes a fixed number \( |x| \) of bits as input and produces a yes/no answer can be represented by a circuit equivalent to a CNF formula.
What is needed to be proved?

- We need to find a poly-time reduction that turns any $x \in \{0, 1\}^*$ into a CNF formula $\varphi_x$ such that $x \in L$ iff $\varphi_x$ is satisfiable.
- We only know that $L \in \text{NP}$. The reduction has to rely only on the definition of computation.
- We take help of the fact that any algorithm that takes a fixed number $|x|$ of bits as input and produces a yes/no answer can be represented by a circuit equivalent to a CNF formula.
- The circuit has to be equivalent to the algorithm, i.e. its output is 1 precisely on those inputs for which the algorithm outputs yes.
## Warm Up

<table>
<thead>
<tr>
<th>What is needed to be proved?</th>
</tr>
</thead>
<tbody>
<tr>
<td>We need to find a poly-time reduction that turns any $x \in {0, 1}^*$ into a CNF formula $\varphi_x$ such that $x \in L$ iff $\varphi_x$ is satisfiable.</td>
</tr>
<tr>
<td>We only know that $L \in \text{NP}$. The reduction has to rely only on the definition of computation.</td>
</tr>
<tr>
<td>We take help of the fact that any algorithm that takes a fixed number $</td>
</tr>
<tr>
<td>The circuit has to be equivalent to the algorithm, i.e. its output is 1 precisely on those inputs for which the algorithm outputs yes.</td>
</tr>
<tr>
<td>If the algorithm takes number of steps that is polynomial in $</td>
</tr>
</tbody>
</table>
Warm Up

What is needed to be proved?

- We are trying to show that $L \leq_P$ SAT.
Another Class: coNP

The Classes EXP and NEXP

Cook Levin’s Theorem

Warm Up

What is needed to be proved?

- We are trying to show that \( L \leq_P \text{SAT} \).
- So, given an input \( x \), we want to decide whether \( x \in L \) using a black box that can solve instances of SAT.
Warm Up

What is needed to be proved?

- We are trying to show that $L \leq_p \text{SAT}$.
- So, given an input $x$, we want to decide whether $x \in L$ using a black box that can solve instances of SAT.
- We know that $L \in \text{NP}$, i.e. $L$ has an efficient certifier $M(\cdot , \cdot)$. 
Warm Up

What is needed to be proved?

- We are trying to show that $L \leq_P \text{SAT}$.
- So, given an input $x$, we want to decide whether $x \in L$ using a black box that can solve instances of SAT.
- We know that $L \in \text{NP}$, i.e. $L$ has an efficient certifier $M(\cdot, \cdot)$.
- So, to determine whether $x \in L$, for some specific input of length $|x|$, we need to answer: Is there a $u$, $|u| = p(|x|)$, such that $M(x, u) = 1$?
Warm Up

Proof Idea

- We need the answer only for a specific input $x$. 
Warm Up

Proof Idea

- We need the answer only for a specific input $x$.
- We view $M(\cdot, \cdot)$ as an algorithm on $|x| + |u|$ bits.
Warm Up

Proof Idea

- We need the answer only for a specific input $x$.
- We view $M(\cdot, \cdot)$ as an algorithm on $|x| + |u|$ bits.
- Convert $M$ to a poly-size circuit $K$ with $|x| + |u|$ sources.
Warm Up

Proof Idea

- We need the answer only for a specific input $x$.
- We view $M(\cdot, \cdot)$ as an algorithm on $|x| + |u|$ bits.
- Convert $M$ to a poly-size circuit $K$ with $|x| + |u|$ sources.
- The first $|x|$ sources will be hard-coded with the values of the bits in $x$. 
Proof Idea

- We need the answer only for a specific input $x$.
- We view $M(\cdot, \cdot)$ as an algorithm on $|x| + |u|$ bits.
- Convert $M$ to a poly-size circuit $K$ with $|x| + |u|$ sources.
- The first $|x|$ sources will be hard-coded with the values of the bits in $x$.
- The remaining $|u|$ sources will be labeled with variables representing the bits of $u$; these will be inputs to the circuit $K$. 

Observe that $x \in L$ iff there is a way to set the input bits to $K$ so that $K$ produces an output of 1.
Warm Up

Proof Idea

- We need the answer only for a specific input $x$.
- We view $M(\cdot, \cdot)$ as an algorithm on $|x| + |u|$ bits.
- Convert $M$ to a poly-size circuit $K$ with $|x| + |u|$ sources.
- The first $|x|$ sources will be hard-coded with the values of the bits in $x$.
- The remaining $|u|$ sources will be labeled with variables representing the bits of $u$; these will be inputs to the circuit $K$.
- Observe that $x \in L$ iff there is a way to set the input bits to $K$ so that $K$ produces an output of 1.
An Example

Example

Given a graph $G$ does it contain a 2 node independent set? The problem is in NP. How an instance of this problem can be solved by constructing an equivalent SAT?
Expressiveness of Boolean Formula

Boolean Formula can represent anything!! (An Example)

Let $\langle x_1, x_2, \ldots, x_n \rangle$ and $\langle y_1, y_2, \ldots, y_n \rangle$ be two strings. How do we check for $x = y$?

$(x_1 \lor y_1) \land (x_2 \lor y_2) \land \cdots \land (x_n \lor y_n)$ is TRUE iff the strings $x$ and $y$ are equal to one another.

Claim: For every boolean function $f: \{0, 1\}^\ell \to \{0, 1\}$ there is an $\ell$-variable CNF formula $\phi$ of size $\ell^2$ s.t. $\phi(u) = f(u)$ for every $u \in \{0, 1\}^\ell$, where the size of a CNF formula is defined to be the number of $\lor$/$\land$ symbols it contains.
Expresiveness of Boolean Formula

**Boolean Formula can represent anything!! (An Example)**

- Let \( x = < x_1, x_2, \ldots, x_n > \) and \( y = < y_1, y_2, \ldots, y_n > \) be two strings. How do we check for \( x = y \)?

- \((x_1 \lor \overline{y_1}) \land (x_2 \lor \overline{y_2}) \land \cdots \land (x_n \lor \overline{y_n})\) is TRUE iff the strings \( x \) and \( y \) are equal to one another.
Expresiveness of Boolean Formula

Boolean Formula can represent anything!! (An Example)

- Let \( x = \langle x_1, x_2, \ldots, x_n \rangle \) and \( y = \langle y_1, y_2, \ldots, y_n \rangle \) be two strings. How do we check for \( x = y \)?
- \((x_1 \lor \overline{y_1}) \land (x_2 \lor \overline{y_2}) \land \cdots \land (x_n \lor \overline{y_n})\) is TRUE iff the strings \( x \) and \( y \) are equal to one another.

Claim

For every boolean function \( f : \{0, 1\}^\ell \to \{0, 1\} \) there is an \( \ell \)-variable CNF formula \( \varphi \) of size \( \ell 2^\ell \) s.t. \( \varphi(u) = f(u) \) for every \( u \in \{0, 1\}^\ell \), where the size of a CNF formula is defined to be the number of \( \lor/\land \) symbols it contains.
Proof of Claim

Proof

For every $v \in \{0, 1\}^\ell$, $\exists$ a clause $C_v$ s.t. $C_v(v) = 0$ and $C_v(u) = 1$ for every $u \neq v$. For example, if $v = \langle 1, 0, 1, 0 \rangle$, then the corr. clause is $\overline{u_1} \lor u_2 \lor \overline{u_3} \lor u_4$. 
## Proof of Claim

**Proof**

- For every $\nu \in \{0, 1\}^\ell$, $\exists$ a clause $C_\nu$ s.t. $C_\nu(\nu) = 0$ and $C_\nu(u) = 1$ for every $u \neq \nu$. For example, if $\nu = <1, 0, 1, 0>$, then the corr. clause is $\overline{u_1} \lor u_2 \lor \overline{u_3} \lor u_4$.
- Let $\varphi = \bigvee C_\nu$ for $\nu$ s.t. $f(\nu) = 0$. $|\varphi| = \ell 2^\ell$
Proof of Claim

Proof

- For every $v \in \{0, 1\}^\ell$, $\exists$ a clause $C_v$ s.t. $C_v(v) = 0$ and $C_v(u) = 1$ for every $u \neq v$. For example, if $v = <1, 0, 1, 0>$, then the corr. clause is $\overline{u_1} \lor u_2 \lor \overline{u_3} \lor u_4$.

- Let $\varphi = \bigvee C_v$ for $v$ s.t. $f(v) = 0$. $|\varphi| = \ell 2^\ell$

- Then for every $u$ s.t. $f(u) = 0$ it holds that $C_u(u) = 0$ and hence, $\varphi(u) = 0$. 
Proof of Claim

**Proof**

- For every $v \in \{0, 1\}^\ell$, $\exists$ a clause $C_v$ s.t. $C_v(v) = 0$ and $C_v(u) = 1$ for every $u \neq v$. For example, if $v = <1, 0, 1, 0>$, then the corr. clause is $\overline{u_1} \lor u_2 \lor \overline{u_3} \lor u_4$.

- Let $\varphi = \bigvee C_v$ for $v$ s.t. $f(v) = 0$. $|\varphi| = \ell 2^\ell$

- Then for every $u$ s.t. $f(u) = 0$ it holds that $C_u(u) = 0$ and hence, $\varphi(u) = 0$.

- On the other hand, if $f(u) = 1$ then $C_v(u) = 1$ for every $v$ s.t. $f(v) = 0$ and hence, $\varphi(u) = 1$. 
Proof of Claim

**Proof**

- For every $v \in \{0, 1\}^\ell$, $\exists$ a clause $C_v$ s.t. $C_v(v) = 0$ and $C_v(u) = 1$ for every $u \neq v$. For example, if $v = \langle 1, 0, 1, 0 \rangle$, then the corr. clause is $\overline{u_1} \lor u_2 \lor \overline{u_3} \lor u_4$.

- Let $\varphi = \bigvee C_v$ for $v$ s.t. $f(v) = 0$. $|\varphi| = \ell 2^\ell$

- Then for every $u$ s.t. $f(u) = 0$ it holds that $C_u(u) = 0$ and hence, $\varphi(u) = 0$.

- On the other hand, if $f(u) = 1$ then $C_v(u) = 1$ for every $v$ s.t. $f(v) = 0$ and hence, $\varphi(u) = 1$.

- So, we get that for every $u$, $\varphi(u) = f(u)$.