Lecture 15: Boolean Circuits II

Arijit Bishnu

20.04.2010

Outline

Outline

NP and $P_{/poly}$: Basic Definitions again

Certificate definition of NP

A language $L \subseteq \{0,1\}^*$ is in NP if $\exists$ a polynomial $p : \mathbb{N} \to \mathbb{N}$ and a poly-time TM M s.t. $\forall x \in \{0,1\}^*$,

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

NP and P$_{/poly}$: Basic Definitions again

Certificate definition of NP

A language $L \subseteq \{0,1\}^*$ is in NP if $\exists$ a polynomial $p : \mathbb{N} \to \mathbb{N}$ and a poly-time TM M s.t. $\forall x \in \{0,1\}^*$,

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

Advice TM definition of P$_{/poly}$

A language $L \subseteq \{0,1\}^*$ is in P$_{/poly}$ if $\exists$ a polynomial $p : \mathbb{N} \to \mathbb{N}$ and a poly-time TM M and a sequence $\{a_n\}$ of advice strings where $|a_n| \leq p(n), \forall n \in \mathbb{N}$, s.t. $\forall n$ and $\forall x \in \{0,1\}^*$,

$$x \in L \iff M(x, a_n) = 1$$

NP and $P_{/poly}$: Basic Definitions again

Certificate definition of NP

A language $L \subseteq \{0,1\}^*$ is in NP if $\exists$ a polynomial $p : \mathbb{N} \to \mathbb{N}$ and a poly-time TM M s.t. $\forall x \in \{0,1\}^*$,

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

Advice TM definition of $P_{/poly}$

A language $L \subseteq \{0,1\}^*$ is in $P_{/poly}$ if $\exists$ a polynomial $p : \mathbb{N} \to \mathbb{N}$ and a poly-time TM M and a sequence $\{a_n\}$ of advice strings where $|a_n| \leq p(n), \forall n \in \mathbb{N}$, s.t. $\forall n$ and $\forall x \in \{0,1\}^*$,

$$x \in L \iff M(x, a_n) = 1$$

- These two classes are different. We already know $P \subset P_{/poly}$. If we can show $NP \not\subset P_{/poly}$, then we are done!

Karp-Lipton Theorem

Theorem

If $NP \subseteq P_{/poly}$, then $PH = \Sigma_2^p$

Karp-Lipton Theorem

Theorem

If NP $\subseteq$ P$_{/poly}$, then PH $= \Sigma_2^p$

Proof Idea

Karp-Lipton Theorem

Theorem

If $NP \subseteq P_{/poly}$, then $PH = \Sigma_2^p$

Proof Idea

- We already know that if $\Sigma_2^p = \Sigma_3^p$, then the polynomial hierarchy collapses, i.e. $PH = \Sigma_2^p$.

Karp-Lipton Theorem

Theorem

If NP $\subseteq$ P$_{/poly}$, then PH $= \Sigma_2^p$

Proof Idea

- We already know that if $\Sigma_2^p = \Sigma_3^p$, then the polynomial hierarchy collapses, i.e. PH $= \Sigma_2^p$.
- So, it is sufficient to show that $\Sigma_3^p \subseteq \Sigma_2^p$ as that would imply $\Sigma_2^p = \Sigma_3^p$.

Karp-Lipton Theorem

Theorem

If $NP \subseteq P_{/poly}$, then $PH = \Sigma_2^p$

Proof Idea

- We already know that if $\Sigma_2^p = \Sigma_3^p$, then the polynomial hierarchy collapses, i.e. $PH = \Sigma_2^p$.
- So, it is sufficient to show that $\Sigma_3^p \subseteq \Sigma_2^p$ as that would imply $\Sigma_2^p = \Sigma_3^p$.
- So, we want to show $(NP \subseteq P_{/poly}) \Rightarrow (\Sigma_3^p \subseteq \Sigma_2^p) \Rightarrow (PH = \Sigma_2^p)$.

The Proof

- Fix a language $L \in \Sigma_3^p$. So, $\exists$ a poly-time TM M and a polynomial q s.t.

$$x \in L \iff \exists u_1 \in \{0,1\}^{q(|x|)} \forall u_2 \in \{0,1\}^{q(|x|)} \exists u_3 \in \{0,1\}^{q(|x|)} M(x, u_1, u_2, u_3) = 1$$

The Proof

- Fix a language $L \in \Sigma_3^p$. So, $\exists$ a poly-time TM M and a polynomial q s.t.

$$x \in L \iff \exists u_1 \in \{0,1\}^{q(|x|)} \forall u_2 \in \{0,1\}^{q(|x|)} \exists u_3 \in \{0,1\}^{q(|x|)} M(x, u_1, u_2, u_3) = 1$$

- Focus on the highlighted part and define a new language L' s.t.

$$< x, u_1, u_2 > \in L' \iff \exists u_3 \in \{0,1\}^{q(|x|)} M(x, u_1, u_2, u_3) = 1$$

The Proof

- Fix a language $L \in \Sigma_3^p$. So, $\exists$ a poly-time TM M and a polynomial q s.t.

$$x \in L \iff \exists u_1 \in \{0,1\}^{q(|x|)} \forall u_2 \in \{0,1\}^{q(|x|)} \exists u_3 \in \{0,1\}^{q(|x|)} M(x, u_1, u_2, u_3) = 1$$

- Focus on the highlighted part and define a new language L' s.t.

$$< x, u_1, u_2 > \in L' \iff \exists u_3 \in \{0,1\}^{q(|x|)} M(x, u_1, u_2, u_3) = 1$$

- $L' \in$ NP and as 3SAT is NP-complete, $L' \leq_P$ 3SAT.

The Proof

- Fix a language $L \in \Sigma_3^p$. So, $\exists$ a poly-time TM M and a polynomial q s.t.

$$x \in L \iff \exists u_1 \in \{0,1\}^{q(|x|)} \forall u_2 \in \{0,1\}^{q(|x|)} \exists u_3 \in \{0,1\}^{q(|x|)} M(x, u_1, u_2, u_3) = 1$$

- Focus on the highlighted part and define a new language L' s.t.

$$< x, u_1, u_2 > \in L' \iff \exists u_3 \in \{0,1\}^{q(|x|)} M(x, u_1, u_2, u_3) = 1$$

- $L' \in$ NP and as 3SAT is NP-complete, $L' \leq_P$ 3SAT.
- So, every input of the form $< x, u_1, u_2 >$ can be polynomially (Cook) reduced to a 3CNF expression $\varphi(x, u_1, u_2)$ and

$$< x, u_1, u_2 > \in L' \iff \varphi(x, u_1, u_2) \in \text{3SAT}$$

The Proof

- So, an NDTM M' decides L'.

The Proof

- So, an NDTM M' decides L'.
- Plug in M' in the original expression

$$x \in L \iff \exists u_1 \in \{0,1\}^{q(|x|)} \forall u_2 \in \{0,1\}^{q(|x|)} \exists u_3 \in \{0,1\}^{q(|x|)} M(x, u_1, u_2, u_3) = 1$$

to get

$$x \in L \iff \exists u_1 \in \{0,1\}^{q(|x|)} \forall u_2 \in \{0,1\}^{q(|x|)} M'(x, u_1, u_2) = 1$$

or

$$x \in L \iff \exists u_1 \in \{0,1\}^{q(|x|)} \forall u_2 \in \{0,1\}^{q(|x|)} \varphi(x, u_1, u_2) \in 3\text{SAT}$$

The Proof

- So, an NDTM M' decides L'.
- Plug in M' in the original expression

$$x \in L \iff \exists u_1 \in \{0,1\}^{q(|x|)} \forall u_2 \in \{0,1\}^{q(|x|)} \exists u_3 \in \{0,1\}^{q(|x|)} M(x, u_1, u_2, u_3) = 1$$

to get

$$x \in L \iff \exists u_1 \in \{0,1\}^{q(|x|)} \forall u_2 \in \{0,1\}^{q(|x|)} M'(x, u_1, u_2) = 1$$

or

$$x \in L \iff \exists u_1 \in \{0,1\}^{q(|x|)} \forall u_2 \in \{0,1\}^{q(|x|)} \varphi(x, u_1, u_2) \in 3\text{SAT}$$

The Proof

- So, an NDTM M' decides L'.
- Plug in M' in the original expression

$$x \in L \iff \exists u_1 \in \{0,1\}^{q(|x|)} \forall u_2 \in \{0,1\}^{q(|x|)} \exists u_3 \in \{0,1\}^{q(|x|)} M(x, u_1, u_2, u_3) = 1$$

to get

$$x \in L \iff \exists u_1 \in \{0,1\}^{q(|x|)} \forall u_2 \in \{0,1\}^{q(|x|)} M'(x, u_1, u_2) = 1$$

or

$$x \in L \iff \exists u_1 \in \{0,1\}^{q(|x|)} \forall u_2 \in \{0,1\}^{q(|x|)} \varphi(x, u_1, u_2) \in 3\text{SAT}$$

The Proof

- Now, 3SAT $\in$ NP and as NP $\subseteq$ P$_{/\text{poly}}$, $\exists$ a polynomial-sized circuit C for 3SAT, i.e. a poly-sized circuit C that decides $\varphi(x, u_1, u_2) \in$ 3SAT.

The Proof

- Now, 3SAT $\in$ NP and as NP $\subseteq$ P$_{/poly}$, $\exists$ a polynomial-sized circuit C for 3SAT, i.e. a poly-sized circuit C that decides $\varphi(x, u_1, u_2) \in$ 3SAT.
- So, we can use the circuit C to check the highlighted part of the following:

$$x \in L \iff \exists u_1 \in \{0,1\}^{q(|x|)} \forall u_2 \in \{0,1\}^{q(|x|)} \; \varphi(x, u_1, u_2) \in \text{3SAT}$$

The Proof

- Now, 3SAT $\in$ NP and as NP $\subseteq$ P$_{/poly}$, $\exists$ a polynomial-sized circuit C for 3SAT, i.e. a poly-sized circuit C that decides $\varphi(x, u_1, u_2) \in$ 3SAT.
- So, we can use the circuit C to check the highlighted part of the following:

$$x \in L \iff \exists u_1 \in \{0,1\}^{q(|x|)} \forall u_2 \in \{0,1\}^{q(|x|)} \; \varphi(x, u_1, u_2) \in \text{3SAT}$$

- How to find the circuit C?

Finding the Circuit

Lemma

$\exists$ a poly-time TM M_C that takes a 3-CNF formula φ, $|varphi| = n$ and a circuit C with n inputs such that

Finding the Circuit

Lemma

$\exists$ a poly-time TM M_C that takes a 3-CNF formula φ, $|varphi| = n$ and a circuit C with n inputs such that

- If C recognizes 3SAT for inputs of length n and φ is satisfiable, then M_C accepts.

Finding the Circuit

Lemma

$\exists$ a poly-time TM M_C that takes a 3-CNF formula φ, $|varphi| = n$ and a circuit C with n inputs such that

- If C recognizes 3SAT for inputs of length n and φ is satisfiable, then M_C accepts.
- If φ is not satisfiable, then M_C rejects.

The Proof

- Repeatedly use C and extract a satisfying assignment for φ. The input to M_C is C and φ.

The Proof

- Repeatedly use C and extract a satisfying assignment for φ. The input to M_C is C and φ.
- Let the variables for φ be $x_1, \ldots, x_m$.

The Proof

- Repeatedly use C and extract a satisfying assignment for φ. The input to M_C is C and φ.
- Let the variables for φ be $x_1, \ldots, x_m$.
- If $C(\varphi) = 0$, then M_C REJECTS.

The Proof

- Repeatedly use C and extract a satisfying assignment for φ. The input to M_C is C and φ.
- Let the variables for φ be $x_1, \ldots, x_m$.
- If $C(\varphi) = 0$, then M_C REJECTS.
- Fix $x_i = 0$ in φ and check if $C(\varphi|_{x_i=0}) = 1$, then let $a_i = 0$, else $a_i = 1$. Make $\varphi = \varphi[x_i = a_i]$.

The Proof

- Repeatedly use C and extract a satisfying assignment for φ. The input to M_C is C and φ.
- Let the variables for φ be $x_1, \ldots, x_m$.
- If $C(\varphi) = 0$, then M_C REJECTS.
- Fix $x_i = 0$ in φ and check if $C(\varphi|_{x_i=0}) = 1$, then let $a_i = 0$, else $a_i = 1$. Make $\varphi = \varphi[x_i = a_i]$.
- Do this in a loop for $i = 1$ to m.

The Proof

- Repeatedly use C and extract a satisfying assignment for φ. The input to M_C is C and φ.
- Let the variables for φ be $x_1, \ldots, x_m$.
- If $C(\varphi) = 0$, then M_C REJECTS.
- Fix $x_i = 0$ in φ and check if $C(\varphi|_{x_i=0}) = 1$, then let $a_i = 0$, else $a_i = 1$. Make $\varphi = \varphi[x_i = a_i]$.
- Do this in a loop for $i = 1$ to m.
- If $\varphi[x_i = a_i, \forall i = 1(1)m] = 1$, then ACCEPT; else REJECT.

Going Back to the Original Proof

- Existentially check the circuit. So, the earlier condition

$$x \in L \iff \exists u_1 \in \{0,1\}^{q(|x|)} \forall u_2 \in \{0,1\}^{q(|x|)} \varphi(x, u_1, u_2) \in \text{3SAT}$$

becomes

$$x \in L \iff \exists C \text{ of poly-size } \exists u_1 \in \{0,1\}^{q(|x|)} \forall u_2 \in \{0,1\}^{q(|x|)} M_C(C, < x, u_1, u_2 >) = 1$$

Going Back to the Original Proof

- Existentially check the circuit. So, the earlier condition

$$x \in L \iff \exists u_1 \in \{0,1\}^{q(|x|)} \forall u_2 \in \{0,1\}^{q(|x|)} \varphi(x, u_1, u_2) \in \text{3SAT}$$

becomes

$$x \in L \iff \exists C \text{ of poly-size } \exists u_1 \in \{0,1\}^{q(|x|)} \forall u_2 \in \{0,1\}^{q(|x|)} M_C(C, < x, u_1, u_2 >) = 1$$

- So, $L \in \Sigma_2^p$.

$P_{/poly}$ and EXP

- $P_{/poly}$ is unlikely to contain EXP.

$P_{/poly}$ and EXP

- $P_{/poly}$ is unlikely to contain EXP.

Theorem

If $EXP \subseteq P_{/poly}$, then $EXP = \Sigma_2^p$.

$P_{/poly}$ and EXP

- $P_{/poly}$ is unlikely to contain EXP.

Theorem

If $EXP \subseteq P_{/poly}$, then $EXP = \Sigma_2^p$.

Proof

Left as an exercise. Use the concept of snapshots of oblivious TM.

Outline

Circuit Lower Bounds

- Since, $P \subset P_{/poly}$, if we prove $NP \not\subseteq P_{/poly}$, then $P \neq NP$.

Circuit Lower Bounds

- Since, $P \subset P_{/poly}$, if we prove $NP \not\subseteq P_{/poly}$, then $P \neq NP$.
- The earlier theorem gives hope in this direction.

Circuit Lower Bounds

- Since, $P \subset P_{/poly}$, if we prove $NP \not\subseteq P_{/poly}$, then $P \neq NP$.
- The earlier theorem gives hope in this direction.
- Using Circuit lower bounds (what it is?), one can try to show $NP \not\subseteq P_{/poly}$.

Circuit Lower Bounds

- Since, $P \subset P_{/poly}$, if we prove $NP \not\subseteq P_{/poly}$, then $P \neq NP$.
- The earlier theorem gives hope in this direction.
- Using Circuit lower bounds (what it is?), one can try to show $NP \not\subseteq P_{/poly}$.
- Circuits have their advantages over TMs structurally.

Circuit Lower Bounds

- Since, $P \subset P_{/poly}$, if we prove $NP \not\subseteq P_{/poly}$, then $P \neq NP$.
- The earlier theorem gives hope in this direction.
- Using Circuit lower bounds (what it is?), one can try to show $NP \not\subseteq P_{/poly}$.
- Circuits have their advantages over TMs structurally.
- One can show that some functions do require large circuits to compute.

A Theorem on Circuits

Theorem

For every $n > 1$, $\exists$ a function $f : \{0,1\}^n \rightarrow \{0,1\}$ that cannot be computed by a circuit C of size $\frac{2^n}{cn}$ where c is a suitable constant.

A Theorem on Circuits

Theorem

For every $n > 1$, $\exists$ a function $f : \{0,1\}^n \rightarrow \{0,1\}$ that cannot be computed by a circuit C of size $\frac{2^n}{cn}$ where c is a suitable constant.

Proof

A Theorem on Circuits

Theorem

For every $n > 1$, $\exists$ a function $f : \{0,1\}^n \rightarrow \{0,1\}$ that cannot be computed by a circuit C of size $\frac{2^n}{cn}$ where c is a suitable constant.

Proof

- $|f : \{0,1\}^n \rightarrow \{0,1\}| = 2^{2^n}$.

A Theorem on Circuits

Theorem

For every $n > 1$, $\exists$ a function $f : \{0,1\}^n \rightarrow \{0,1\}$ that cannot be computed by a circuit C of size $\frac{2^n}{cn}$ where c is a suitable constant.

Proof

- $|f : \{0,1\}^n \rightarrow \{0,1\}| = 2^{2^n}$.
- Every circuit of size at most S can be represented as a string of $c_1 S \log S$ bits (why?).

A Theorem on Circuits

Theorem

For every $n > 1$, $\exists$ a function $f : \{0,1\}^n \rightarrow \{0,1\}$ that cannot be computed by a circuit C of size $\frac{2^n}{cn}$ where c is a suitable constant.

Proof

- $|f : \{0,1\}^n \rightarrow \{0,1\}| = 2^{2^n}$.
- Every circuit of size at most S can be represented as a string of $c_1 S \log S$ bits (why?).
- The number of such circuits that are representable by a string of length at most $c_1 S \log S$ is at most $2^{c_1 S \log S}$.

A Theorem on Circuits

Theorem

For every $n > 1$, $\exists$ a function $f : \{0,1\}^n \rightarrow \{0,1\}$ that cannot be computed by a circuit C of size $\frac{2^n}{cn}$ where c is a suitable constant.

Proof

- $|f : \{0,1\}^n \rightarrow \{0,1\}| = 2^{2^n}$.
- Every circuit of size at most S can be represented as a string of $c_1 S \log S$ bits (why?).
- The number of such circuits that are representable by a string of length at most $c_1 S \log S$ is at most $2^{c_1 S \log S}$.
- Set $S = \frac{2^n}{cn}$ where $c > c_1$.

A Theorem on Circuits

Theorem

For every $n > 1$, $\exists$ a function $f : \{0,1\}^n \rightarrow \{0,1\}$ that cannot be computed by a circuit C of size $\frac{2^n}{cn}$ where c is a suitable constant.

Proof

- $|f : \{0,1\}^n \rightarrow \{0,1\}| = 2^{2^n}$.
- Every circuit of size at most S can be represented as a string of $c_1 S \log S$ bits (why?).
- The number of such circuits that are representable by a string of length at most $c_1 S \log S$ is at most $2^{c_1 S \log S}$.
- Set $S = \frac{2^n}{cn}$ where $c > c_1$.
- The number of circuits of size S is at most $2^{c_1 S \log S} < 2^{2^n}$.

A Theorem on Circuits

Theorem

For every $n > 1$, $\exists$ a function $f : \{0,1\}^n \rightarrow \{0,1\}$ that cannot be computed by a circuit C of size $\frac{2^n}{cn}$ where c is a suitable constant.

Proof

- $|f : \{0,1\}^n \rightarrow \{0,1\}| = 2^{2^n}$.
- Every circuit of size at most S can be represented as a string of $c_1 S \log S$ bits (why?).
- The number of such circuits that are representable by a string of length at most $c_1 S \log S$ is at most $2^{c_1 S \log S}$.
- Set $S = \frac{2^n}{cn}$ where $c > c_1$.
- The number of circuits of size S is at most $2^{c_1 S \log S} < 2^{2^n}$.
- This implies $\exists$ a function that is not computed by that sized circuit.

Outline

NC and AC

- We look at some subclasses of $P_{/poly}$.

NC and AC

- We look at some subclasses of $P_{/poly}$.
- The subclasses has relation to parallel computation.

NC and AC

- We look at some subclasses of $P_{/poly}$.
- The subclasses has relation to parallel computation.

Definition: The Class NC

For every d, a language $L \in NC^d$ if L can be decided by a family of circuits $\{C_n\}$ where C_n has size polynomial in n and depth $O(\log^d n)$. The class $NC = \bigcup_{i \geq 1} NC^i$.

NC and AC

- We look at some subclasses of $P_{/poly}$.
- The subclasses has relation to parallel computation.

Definition: The Class NC

For every d, a language $L \in NC^d$ if L can be decided by a family of circuits $\{C_n\}$ where C_n has size polynomial in n and depth $O(\log^d n)$. The class $NC = \bigcup_{i \geq 1} NC^i$.

Definition: The Class AC

For every d, a language $L \in AC^d$ if L can be decided by a family of circuits $\{C_n\}$ where C_n has size polynomial in n and depth $O(\log^d n)$ but gates are allowed to have unbounded fan-in. The class $AC = \bigcup_{i \geq 0} AC^i$.

Relation between NC^i and AC^i

Relation between NC^i and AC^i

$NC^i \subseteq AC^i \subseteq NC^{i+1}$

Relation between NC^i and AC^i

Relation between NC^i and AC^i

$NC^i \subseteq AC^i \subseteq NC^{i+1}$

Examples

Relation between NC^i and AC^i

Relation between NC^i and AC^i

$NC^i \subseteq AC^i \subseteq NC^{i+1}$

Examples

- What about the language PARITY $= \{x \mid x \text{ has an odd number of 1s}\}$?

Relation between NC^i and AC^i

Relation between NC^i and AC^i

$NC^i \subseteq AC^i \subseteq NC^{i+1}$

Examples

- What about the language PARITY $= \{x \mid x \text{ has an odd number of 1s}\}$?
- One can show using logarithmic depth recursion that PARITY $\in NC^1$.

Relation between NC^i and AC^i

Relation between NC^i and AC^i

$NC^i \subseteq AC^i \subseteq NC^{i+1}$

Examples

- What about the language PARITY $= \{x \mid x \text{ has an odd number of 1s}\}$?
- One can show using logarithmic depth recursion that PARITY $\in NC^1$.

Theorem

A language has efficient parallel algorithms iff it is in NC.

Relation between NC^i and AC^i

Relation between NC^i and AC^i

$NC^i \subseteq AC^i \subseteq NC^{i+1}$

Examples

- What about the language PARITY $= \{x \mid x \text{ has an odd number of 1s}\}$?
- One can show using logarithmic depth recursion that PARITY $\in NC^1$.

Theorem

A language has efficient parallel algorithms iff it is in NC.

Proof

Left as an exercise.

P-completeness

Definition: P-completeness

A language L is P-complete if $L \in \mathrm{P}$ and for every language $L' \in \mathrm{P}$, $L' \leq_{\log} L$ where log stands for log-space reducibility.

P-completeness

Definition: P-completeness

A language L is P-complete if $L \in \mathrm{P}$ and for every language $L' \in \mathrm{P}$, $L' \leq_{\log} L$ where log stands for log-space reducibility.

Theorem

If a language L is P-complete, then

P-completeness

Definition: P-completeness

A language L is P-complete if $L \in \mathrm{P}$ and for every language $L' \in \mathrm{P}$, $L' \leq_{\log} L$ where log stands for log-space reducibility.

Theorem

If a language L is P-complete, then

- $L \in \mathrm{NC}$ iff $\mathrm{P} = \mathrm{NC}$.

P-completeness

Definition: P-completeness

A language L is P-complete if $L \in \mathrm{P}$ and for every language $L' \in \mathrm{P}$, $L' \leq_{\log} L$ where log stands for log-space reducibility.

Theorem

If a language L is P-complete, then

- $L \in \mathrm{NC}$ iff $\mathrm{P} = \mathrm{NC}$.
- $L \in \mathrm{L}$ iff $\mathrm{P} = \mathrm{L}$.

P-completeness

Definition: P-completeness

A language L is P-complete if $L \in \mathrm{P}$ and for every language $L' \in \mathrm{P}$, $L' \leq_{\log} L$ where log stands for log-space reducibility.

Theorem

If a language L is P-complete, then

- $L \in \mathrm{NC}$ iff $\mathrm{P} = \mathrm{NC}$.
- $L \in \mathrm{L}$ iff $\mathrm{P} = \mathrm{L}$.

Proof

Proof omitted.