CS310 Automata Theory – 2016-2017

Nutan Limaye

Indian Institute of Technology, Bombay
nutan@cse.iitb.ac.in

Lecture 35: Effective computation
April 10, 2017

Relationships between complexity classes

How are P, NP, EXP, and NEXP related?

Relationships between complexity classes

How are P, NP, EXP, and NEXP related?

P $\subseteq$ NP by definition.

Relationships between complexity classes

How are P, NP, EXP, and NEXP related?

- P $\subseteq$ NP by definition.
- P $\subseteq$ EXP again by definition.

Relationships between complexity classes

How are P, NP, EXP, and NEXP related?

- $\mathrm{P} \subseteq \mathrm{NP}$ by definition.
- $\mathrm{P} \subseteq \mathrm{EXP}$ again by definition.
- Similarly, $\mathrm{NP} \subseteq \mathrm{NEXP}$ by definition.

Relationships between complexity classes

How are P, NP, EXP, and NEXP related?

- P $\subseteq$ NP by definition.
- P $\subseteq$ EXP again by definition.
- Similarly, NP $\subseteq$ NEXP by definition.
- Finally, NP $\subseteq$ EXP due to the previous lemma.

Relationships between complexity classes

How are P, NP, EXP, and NEXP related?

- $\mathsf{P} \subseteq \mathsf{NP}$ by definition.
- $\mathsf{P} \subseteq \mathsf{EXP}$ again by definition.
- Similarly, $\mathsf{NP} \subseteq \mathsf{NEXP}$ by definition.
- Finally, $\mathsf{NP} \subseteq \mathsf{EXP}$ due to the previous lemma.

$$\begin{tikzcd} \mathsf{P} \arrow[r] & \mathsf{NP} \\ \mathsf{EXP} & \mathsf{NEXP} \end{tikzcd}$$

Relationships between complexity classes

How are P, NP, EXP, and NEXP related?

- P $\subseteq$ NP by definition.
- P $\subseteq$ EXP again by definition.
- Similarly, NP $\subseteq$ NEXP by definition.
- Finally, NP $\subseteq$ EXP due to the previous lemma.

$$\begin{tikzcd}
\mathsf{P} \arrow[r] \arrow[d] & \mathsf{NP} \\
\mathsf{EXP} & \mathsf{NEXP}
\end{tikzcd}$$

Relationships between complexity classes

How are P, NP, EXP, and NEXP related?

- P $\subseteq$ NP by definition.
- P $\subseteq$ EXP again by definition.
- Similarly, NP $\subseteq$ NEXP by definition.
- Finally, NP $\subseteq$ EXP due to the previous lemma.

$$\begin{tikzcd} \mathrm{P} \arrow[r] \arrow[d] & \mathrm{NP} \\ \mathrm{EXP} \arrow[r] & \mathrm{NEXP} \end{tikzcd}$$

Relationships between complexity classes

How are P, NP, EXP, and NEXP related?

P $\subseteq$ NP by definition.

P $\subseteq$ EXP again by definition.

Similarly, NP $\subseteq$ NEXP by definition.

Finally, NP $\subseteq$ EXP due to the previous lemma.

$$\begin{tikzcd}
\mathrm{P} \arrow[r] \arrow[d] & \mathrm{NP} \arrow[d] \\
\mathrm{EXP} \arrow[r] & \mathrm{NEXP}
\end{tikzcd}$$

Relationships between complexity classes

How are P, NP, EXP, and NEXP related?

- P $\subseteq$ NP by definition.
- P $\subseteq$ EXP again by definition.
- Similarly, NP $\subseteq$ NEXP by definition.
- Finally, NP $\subseteq$ EXP due to the previous lemma.

$$\begin{tikzcd}
\mathrm{P} \arrow[r] \arrow[d] & \mathrm{NP} \arrow[d] \arrow[dl, red] \\
\mathrm{EXP} \arrow[r] & \mathrm{NEXP}
\end{tikzcd}$$

Relationships between complexity classes

How are P, NP, EXP, and NEXP related?

P $\subseteq$ NP by definition.

P $\subseteq$ EXP again by definition.

Similarly, NP $\subseteq$ NEXP by definition.

Finally, NP $\subseteq$ EXP due to the previous lemma.

$$
\begin{tikzcd}
\mathrm{P} \arrow[r] \arrow[d] & \mathrm{NP} \arrow[d] \arrow[dl, red] \\
\mathrm{EXP} \arrow[r] & \mathrm{NEXP}
\end{tikzcd}
$$

P vs. NP

P vs. NP

P the class of languages where membership can be decided quickly.

P vs. NP

P — the class of languages where membership can be decided quickly.

NP — the class of languages where membership can be verified quickly.

Time heirarchy theorem

How do we separate NP from P?

Time heirarchy theorem

How do we separate NP from P?

To prove	Method used

Time heirarchy theorem

How do we separate NP from P?

To prove	Method used
not regular	

Time heirarchy theorem

How do we separate NP from P?

To prove	Method used
not regular	pumping lemma for REG

Time heirarchy theorem

How do we separate NP from P?

To prove	Method used
not regular	pumping lemma for REG
non-context-free	

Time heirarchy theorem

How do we separate NP from P?

To prove	Method used
not regular	pumping lemma for REG
non-context-free	pumping lemma or CFLs

Time heirarchy theorem

How do we separate NP from P?

To prove	Method used
not regular	pumping lemma for REG
non-context-free	pumping lemma or CFLs
not recognizable	

Time heirarchy theorem

How do we separate NP from P?

To prove	Method used
not regular	pumping lemma for REG
non-context-free	pumping lemma or CFLs
not recognizable	diagonalization

Time heirarchy theorem

How do we separate NP from P?

To prove	Method used
not regular	pumping lemma for REG
non-context-free	pumping lemma or CFLs
not recognizable	diagonalization
not decidable	

Time heirarchy theorem

How do we separate NP from P?

To prove	Method used
not regular	pumping lemma for REG
non-context-free	pumping lemma or CFLs
not recognizable	diagonalization
not decidable	Rice's theorem

Time heirarchy theorem

How do we separate NP from P?

To prove	Method used
not regular	pumping lemma for REG
non-context-free	pumping lemma or CFLs
not recognizable	diagonalization
not decidable	Rice's theorem or diagonalization and reductions

Time heirarchy theorem

How do we separate NP from P?

To prove	Method used
not regular	pumping lemma for REG
non-context-free	pumping lemma or CFLs
not recognizable	diagonalization
not decidable	Rice's theorem or diagonalization and reductions
not in P	

Time heirarchy theorem

How do we separate NP from P?

To prove	Method used
not regular	pumping lemma for REG
non-context-free	pumping lemma or CFLs
not recognizable	diagonalization
not decidable	Rice's theorem or diagonalization and reductions
not in P	???

Finer structure inside P

Definition

A function $t : \mathbb{N} \to \mathbb{N}$ is said to be time constructible

Finer structure inside P

Definition

A function $t : \mathbb{N} \to \mathbb{N}$ is said to be time constructible if the there exists a TM that on input 1^n, it outputs $t(n)$ in time $O(t(n))$.

Finer structure inside P

Definition

A function $t : \mathbb{N} \to \mathbb{N}$ is said to be time constructible if the there exists a TM that on input 1^n, it outputs $t(n)$ in time $O(t(n))$.

Examples

Finer structure inside P

Definition

A function $t : \mathbb{N} \to \mathbb{N}$ is said to be time constructible if the there exists a TM that on input 1^n, it outputs $t(n)$ in time $O(t(n))$.

Examples

n^2

Finer structure inside P

Definition

A function $t : \mathbb{N} \to \mathbb{N}$ is said to be time constructible if the there exists a TM that on input 1^n, it outputs $t(n)$ in time $O(t(n))$.

Examples

n^2, $n \log n$.

Finer structure inside P

Definition

A function $t : \mathbb{N} \to \mathbb{N}$ is said to be time constructible if the there exists a TM that on input 1^n, it outputs $t(n)$ in time $O(t(n))$.

Examples

n^2, $n \log n$.

Theorem

Let $t : \mathbb{N} \to \mathbb{N}$ be a time constructible function. There exists a language L such that $L \in TIME(t(n)^2)$, but $L \notin TIME(o(t(n)))$.

Finer structure inside P

Definition

A function $t : \mathbb{N} \to \mathbb{N}$ is said to be time constructible if the there exists a TM that on input 1^n, it outputs $t(n)$ in time $O(t(n))$.

Examples

n^2, $n \log n$.

Theorem

Let $t : \mathbb{N} \to \mathbb{N}$ be a time constructible function. There exists a language L such that $L \in TIME(t(n)^2)$, but $L \notin TIME(o(t(n)))$.

As a result of the theorem we have

Finer structure inside P

Definition

A function $t : \mathbb{N} \to \mathbb{N}$ is said to be time constructible if the there exists a TM that on input 1^n, it outputs $t(n)$ in time $O(t(n))$.

Examples

n^2, $n \log n$.

Theorem

Let $t : \mathbb{N} \to \mathbb{N}$ be a time constructible function. There exists a language L such that $L \in TIME(t(n)^2)$, but $L \notin TIME(o(t(n)))$.

As a result of the theorem we have

For any $i \geq 2$ and $1 > \epsilon > 0$, $TIME(n^i) \subsetneq TIME(n^{2i+\epsilon})$.

Finer structure inside P

Definition

A function $t : \mathbb{N} \to \mathbb{N}$ is said to be time constructible if the there exists a TM that on input 1^n, it outputs $t(n)$ in time $O(t(n))$.

Examples

$$n^2, \; n \log n.$$

Theorem

Let $t : \mathbb{N} \to \mathbb{N}$ be a time constructible function. There exists a language L such that $L \in TIME(t(n)^2)$, but $L \notin TIME(o(t(n)))$.

As a result of the theorem we have

For any $i \geq 2$ and $1 > \epsilon > 0$, $TIME(n^i) \subsetneq TIME(n^{2i+\epsilon})$.

Polynomial time reductions and NP-hardness

Definition

A function $f : \Sigma^* \to \Sigma^*$ is polynomial time computable if there is a polynomial time Turing machine TM, say M, such that on any input $w \in \Sigma^*$, M stops with only $f(w)$ on its tape.

Polynomial time reductions and NP-hardness

Definition

A function $f : \Sigma^* \to \Sigma^*$ is polynomial time computable if there is a polynomial time Turing machine TM, say M, such that on any input $w \in \Sigma^*$, M stops with only $f(w)$ on its tape.

Polynomial time reductions and NP-hardness

Definition

A language L_1 is said to be polynomial time reducible to another language L_2

Polynomial time reductions and NP-hardness

Definition

A language L_1 is said to be polynomial time reducible to another language L_2, denoted as $L_1 \leq_m L_2$

Polynomial time reductions and NP-hardness

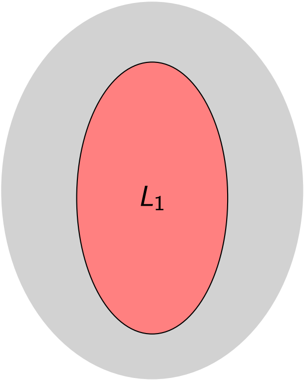
Definition

A language L_1 is said to be polynomial time reducible to another language L_2, denoted as $L_1 \leq_m L_2$, if there exists a polynomial time computable function f such that for all $w \in \Sigma^*$, $w \in L_1 \Leftrightarrow f(w) \in L_2$.

Polynomial time reductions and NP-hardness

Definition

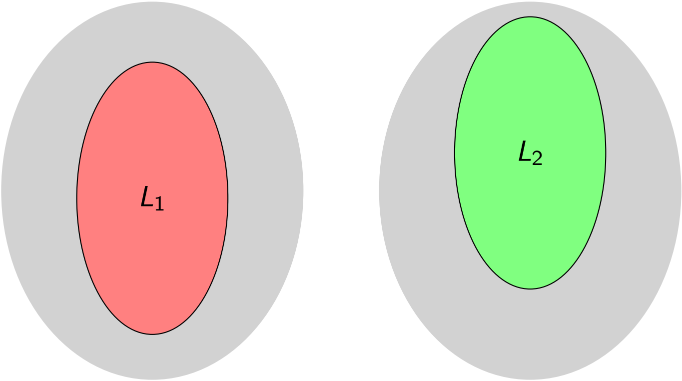
A language L_1 is said to be polynomial time reducible to another language L_2, denoted as $L_1 \leq_m L_2$, if there exists a polynomial time computable function f such that for all $w \in \Sigma^*$, $w \in L_1 \Leftrightarrow f(w) \in L_2$.



Polynomial time reductions and NP-hardness

Definition

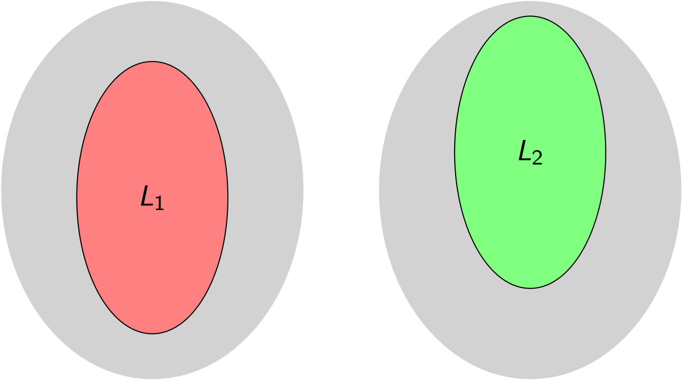
A language L_1 is said to be polynomial time reducible to another language L_2, denoted as $L_1 \leq_m L_2$, if there exists a polynomial time computable function f such that for all $w \in \Sigma^*$, $w \in L_1 \Leftrightarrow f(w) \in L_2$.



Polynomial time reductions and NP-hardness

Definition

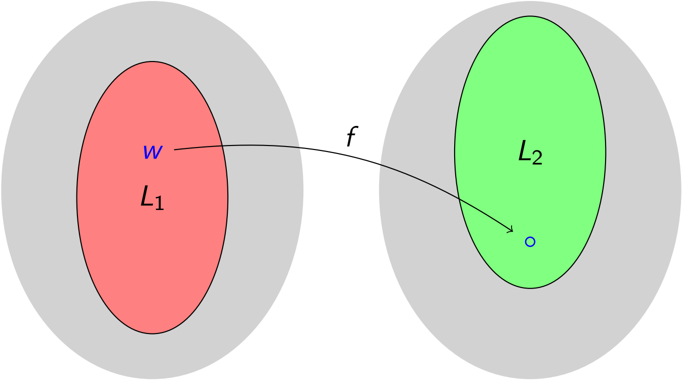
A language L_1 is said to be polynomial time reducible to another language L_2, denoted as $L_1 \leq_m L_2$, if there exists a polynomial time computable function f such that for all $w \in \Sigma^*$, $w \in L_1 \Leftrightarrow f(w) \in L_2$.



Polynomial time reductions and NP-hardness

Definition

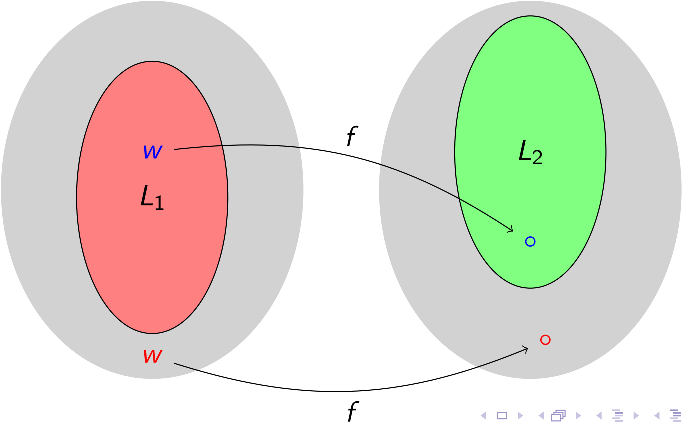
A language L_1 is said to be polynomial time reducible to another language L_2, denoted as $L_1 \leq_m L_2$, if there exists a polynomial time computable function f such that for all $w \in \Sigma^*$, $w \in L_1 \Leftrightarrow f(w) \in L_2$.



Polynomial time reductions and NP-hardness

Definition

A language L_1 is said to be polynomial time reducible to another language L_2, denoted as $L_1 \leq_m L_2$, if there exists a polynomial time computable function f such that for all $w \in \Sigma^*$, $w \in L_1 \Leftrightarrow f(w) \in L_2$.



Polynomial time reductions and NP-hardness

Definition

A language L is said to be NP-hard if for every language $L' \in \text{NP}$, there is a polynomial time reduction such that $L' \leq_m L$.

Polynomial time reductions and NP-hardness

Definition

A language L is said to be NP-hard if for every language $L' \in$ NP, there is a polynomial time reduction such that $L' \leq_m L$.

Definition

A language L is said to be NP-complete if the following two conditions hold:

Polynomial time reductions and NP-hardness

Definition

A language L is said to be NP-hard if for every language $L' \in$ NP, there is a polynomial time reduction such that $L' \leq_m L$.

Definition

A language L is said to be NP-complete if the following two conditions hold:

L is in NP.

Polynomial time reductions and NP-hardness

Definition

A language L is said to be NP-hard if for every language $L' \in$ NP, there is a polynomial time reduction such that $L' \leq_m L$.

Definition

A language L is said to be NP-complete if the following two conditions hold:

- L is in NP.
- L is NP-hard.

Theorem ([Cook-Levin, 1970])

SAT is NP-complete.

Space bounded Turing Machines

The Turing Machine model with space bounds

Space bounded Turing Machines

The Turing Machine model with space bounds

The input tape is assumed to be read-only.

Space bounded Turing Machines

The Turing Machine model with space bounds

- The input tape is assumed to be read-only.
- The space required to write down the input is not counted towards the space of the machine.

Space bounded Turing Machines

The Turing Machine model with space bounds

- The input tape is assumed to be read-only.
- The space required to write down the input is not counted towards the space of the machine.
- The output tape assumed to be write-only.

Space bounded Turing Machines

The Turing Machine model with space bounds

- The input tape is assumed to be read-only.

- The space required to write down the input is not counted towards the space of the machine.

- The output tape assumed to be write-only.

- The space required to write down the output is not counted towards the space of the machine.

Space complexity and complexity classes

Let $s : \mathbb{N} \to \mathbb{N}$.

Space complexity and complexity classes

Let $s : \mathbb{N} \to \mathbb{N}$.

Definition

A language $L \subseteq \Sigma^*$ is said to be in class SPACE($s(n)$)

Space complexity and complexity classes

Let $s : \mathbb{N} \rightarrow \mathbb{N}$.

Definition

A language $L \subseteq \Sigma^*$ is said to be in class SPACE$(s(n))$ if there exists a deterministic Turing machine M such that $\forall x \in \Sigma^*$,

Space complexity and complexity classes

Let $s : \mathbb{N} \to \mathbb{N}$.

Definition

A language $L \subseteq \Sigma^*$ is said to be in class SPACE$(s(n))$ if there exists a deterministic Turing machine M such that $\forall x \in \Sigma^*$,

M halts on x using at most space $O(s(|x|))$

Space complexity and complexity classes

Let $s : \mathbb{N} \to \mathbb{N}$.

Definition

A language $L \subseteq \Sigma^*$ is said to be in class SPACE($s(n)$) if there exists a deterministic Turing machine M such that $\forall x \in \Sigma^*$,

M halts on x using at most space $O(s(|x|))$,

where $|x|$ indicates the length of x.

Space complexity and complexity classes

Let $s : \mathbb{N} \to \mathbb{N}$.

Definition

A language $L \subseteq \Sigma^*$ is said to be in class SPACE($s(n)$) if there exists a deterministic Turing machine M such that $\forall x \in \Sigma^*$,

M halts on x using at most space $O(s(|x|))$,

where $|x|$ indicates the length of x.

if $x \in L$ then M accepts x.

Space complexity and complexity classes

Let $s : \mathbb{N} \to \mathbb{N}$.

Definition

A language $L \subseteq \Sigma^*$ is said to be in class SPACE($s(n)$) if there exists a deterministic Turing machine M such that $\forall x \in \Sigma^*$,

M halts on x using at most space $O(s(|x|))$,

where $|x|$ indicates the length of x.

if $x \in L$ then M accepts x.

if $x \notin L$ then M rejects x.

Space complexity and complexity classes

Let $s : \mathbb{N} \to \mathbb{N}$.

Definition

A language $L \subseteq \Sigma^*$ is said to be in class $\text{SPACE}(s(n))$ if there exists a deterministic Turing machine M such that $\forall x \in \Sigma^*$,

- M halts on x using at most space $O(s(|x|))$,
- where $|x|$ indicates the length of x.
- if $x \in L$ then M accepts x.
- if $x \notin L$ then M rejects x.

$$L = \text{SPACE}(\log n)$$

Space complexity and complexity classes

Let $s : \mathbb{N} \to \mathbb{N}$.

Definition

A language $L \subseteq \Sigma^*$ is said to be in class SPACE($s(n)$) if there exists a deterministic Turing machine M such that $\forall x \in \Sigma^*$,

- M halts on x using at most space $O(s(|x|))$,
- where $|x|$ indicates the length of x.
- if $x \in L$ then M accepts x.
- if $x \notin L$ then M rejects x.

$$L = \mathrm{SPACE}(\log n)$$

$$PSPACE = \bigcup_k \mathrm{SPACE}(n^k)$$

Examples of languages in Log

Min = $\{(w_1, w_2, \ldots, w_n, i) \mid w_i$ is the minimum among $w_1 \ldots w_n\}$.

Examples of languages in Log

Min $= \{(w_1, w_2, \ldots, w_n, i) \mid w_i$ is the minimum among $w_1 \ldots w_n\}$.

Deg $= \{(G = (V, E), d, i) \mid v_i$ has degree $d\}$.

Examples of languages in Log

Min = $\{(w_1, w_2, \ldots, w_n, i) \mid w_i$ is the minimum among $w_1 \ldots w_n\}$.

Deg = $\{(G = (V, E), d, i) \mid v_i$ has degree $d\}$.

ADD = $\{(u, v, i) \mid i$th bit of $u + v$ is $1\}$.

Examples of languages in Log

Min $= \{(w_1, w_2, \ldots, w_n, i) \mid w_i$ is the minimum among $w_1 \ldots w_n\}$.

Deg $= \{(G = (V, E), d, i) \mid v_i$ has degree $d\}$.

ADD $= \{(u, v, i) \mid i$th bit of $u + v$ is $1\}$.

Verify-SAT

$= \{(\phi, a) \mid a = a_1, a_2, \ldots, a_n$ is an assignment satisfying $\phi\}$.

Examples of languages in Log

Min $= \{(w_1, w_2, \ldots, w_n, i) \mid w_i$ is the minimum among $w_1 \ldots w_n\}$.

Deg $= \{(G = (V, E), d, i) \mid v_i$ has degree $d\}$.

ADD $= \{(u, v, i) \mid i$th bit of $u + v$ is $1\}$.

Verify-SAT

$$= \{(\phi, a) \mid a = a_1, a_2, \ldots, a_n \text{ is an assignment satisfying } \phi\}.$$

Space complexity and complexity classes

Let $s : \mathbb{N} \to \mathbb{N}$.

Space complexity and complexity classes

Let $s : \mathbb{N} \to \mathbb{N}$.

Definition

A language $L \subseteq \Sigma^*$ is said to be in class NSPACE$(s(n))$

Space complexity and complexity classes

Let $s : \mathbb{N} \to \mathbb{N}$.

Definition

A language $L \subseteq \Sigma^*$ is said to be in class NSPACE($s(n)$) if there exists a non-deterministic Turing machine M such that $\forall x \in \Sigma^*$,

Space complexity and complexity classes

Let $s : \mathbb{N} \to \mathbb{N}$.

Definition

A language $L \subseteq \Sigma^*$ is said to be in class NSPACE$(s(n))$ if there exists a non-deterministic Turing machine M such that $\forall x \in \Sigma^*$,

> M halts on x using at most space $O(s(|x|))$ on any run of the machine

Space complexity and complexity classes

Let $s : \mathbb{N} \to \mathbb{N}$.

Definition

A language $L \subseteq \Sigma^*$ is said to be in class NSPACE($s(n)$) if there exists a non-deterministic Turing machine M such that $\forall x \in \Sigma^*$,

> M halts on x using at most space $O(s(|x|))$ on any run of the machine,
>
> where $|x|$ indicates the length of x.

Space complexity and complexity classes

Let $s : \mathbb{N} \to \mathbb{N}$.

Definition

A language $L \subseteq \Sigma^*$ is said to be in class NSPACE($s(n)$) if there exists a non-deterministic Turing machine M such that $\forall x \in \Sigma^*$,

> M halts on x using at most space $O(s(|x|))$ on any run of the machine,
>
> where $|x|$ indicates the length of x.
>
> if $x \in L$ then there exists an accepting run of M on x.

Space complexity and complexity classes

Let $s : \mathbb{N} \to \mathbb{N}$.

Definition

A language $L \subseteq \Sigma^*$ is said to be in class NSPACE($s(n)$) if there exists a non-deterministic Turing machine M such that $\forall x \in \Sigma^*$,

> M halts on x using at most space $O(s(|x|))$ on any run of the machine,
>
> where $|x|$ indicates the length of x.
>
> if $x \in L$ then there exists an accepting run of M on x.
>
> if $x \notin L$ then M rejects x on all the runs.

Space complexity and complexity classes

Let $s : \mathbb{N} \to \mathbb{N}$.

Definition

A language $L \subseteq \Sigma^*$ is said to be in class NSPACE($s(n)$) if there exists a non-deterministic Turing machine M such that $\forall x \in \Sigma^*$,

- M halts on x using at most space $O(s(|x|))$ on any run of the machine,
- where $|x|$ indicates the length of x.
- if $x \in L$ then there exists an accepting run of M on x.
- if $x \notin L$ then M rejects x on all the runs.

$$\mathrm{NL} = \mathrm{NSPACE}(\log n)$$

Space complexity and complexity classes

Let $s : \mathbb{N} \to \mathbb{N}$.

Definition

A language $L \subseteq \Sigma^*$ is said to be in class NSPACE$(s(n))$ if there exists a non-deterministic Turing machine M such that $\forall x \in \Sigma^*$,

- M halts on x using at most space $O(s(|x|))$ on any run of the machine,
- where $|x|$ indicates the length of x.
- if $x \in L$ then there exists an accepting run of M on x.
- if $x \notin L$ then M rejects x on all the runs.

$$\mathrm{NL} = \mathrm{NSPACE}(\log n)$$

$$\mathrm{NPSPACE} = \bigcup_k \mathrm{NSPACE}(n^k)$$

Example of a language in NL

$$\text{Reach} = \{(G = (V, E), s, t) \mid \text{there is a path in } G \text{ from } s \text{ to } t\}$$

Example of a language in NL

$$\text{Reach} = \{(G = (V, E), s, t) \mid \text{there is a path in } G \text{ from } s \text{ to } t\}$$