CS310 Automata Theory – 2016-2017

Nutan Limaye

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

Lecture 32: Effective computation
April 04, 2017

Summary of Module III

Introduction to Turing machines

Summary of Module III

Introduction to Turing machines

Equivalent models

Summary of Module III

Introduction to Turing machines

Equivalent models

- multi-tape TM,
- non-deterministic TM

Summary of Module III

Introduction to Turing machines

Equivalent models

- multi-tape TM,
- non-deterministic TM

Turing decidable languages

Summary of Module III

Introduction to Turing machines

Equivalent models

- multi-tape TM,
- non-deterministic TM

Turing decidable languages

Turing recognizable languages

Summary of Module III

Introduction to Turing machines

Equivalent models
- multi-tape TM,
- non-deterministic TM

Turing decidable languages

Turing recognizable languages

Diagonalization in automata theory

Summary of Module III

Introduction to Turing machines

Equivalent models

- multi-tape TM,
- non-deterministic TM

Turing decidable languages

Turing recognizable languages

Diagonalization in automata theory

Proving undecidability

- $A_{TM} = \{(M, w) \mid M \text{ accepts } w\}$,
- $\text{Halt} = \{(M, w) \mid M \text{ hants on } w\}$

Summary of Module III

Introduction to Turing machines

Equivalent models

multi-tape TM,

non-deterministic TM

Turing decidable languages

Turing recognizable languages

Diagonalization in automata theory

Proving undecidability

$A_{TM} = \{(M, w) \mid M \text{ accepts } w\}$,

$\text{Halt} = \{(M, w) \mid M \text{ hants on } w\}$,

$E_{TM} = \{\langle M \rangle \mid L(M) = \varnothing\}$

Summary of Module III

Introduction to Turing machines

Equivalent models

- multi-tape TM,
- non-deterministic TM

Turing decidable languages

Turing recognizable languages

Diagonalization in automata theory

Proving undecidability

$A_{TM} = \{(M, w) \mid M \text{ accepts } w\}$, $\text{Halt} = \{(M, w) \mid M \text{ hants on } w\}$, $E_{TM} = \{\langle M \rangle \mid L(M) = \varnothing\}$, $EQ_{TM} = \{(M_1, M_2) \mid L(M_1) = L(M_2)\}$

Summary of Module III

Introduction to Turing machines

Equivalent models

multi-tape TM,

non-deterministic TM

Turing decidable languages

Turing recognizable languages

Diagonalization in automata theory

Proving undecidability

$A_{TM} = \{(M, w) \mid M \text{ accepts } w\}$,

$\text{Halt} = \{(M, w) \mid M \text{ hants on } w\}$,

$E_{TM} = \{\langle M \rangle \mid L(M) = \varnothing\}$, $EQ_{TM} = \{(M_1, M_2) \mid L(M_1) = L(M_2)\}$,

$REG_{TM} = \{\langle M \rangle \mid L(M) \text{ is regular}\}$

Summary of Module III

Introduction to Turing machines

Equivalent models

- multi-tape TM,
- non-deterministic TM

Turing decidable languages

Turing recognizable languages

Diagonalization in automata theory

Proving undecidability

$A_{TM} = \{(M, w) \mid M \text{ accepts } w\}$,
Halt $= \{(M, w) \mid M \text{ hants on } w\}$,
$E_{TM} = \{\langle M\rangle \mid L(M) = \varnothing\}$, $EQ_{TM} = \{(M_1, M_2) \mid L(M_1) = L(M_2)\}$,
$REG_{TM} = \{\langle M\rangle \mid L(M) \text{ is regular}\}$,
$\mathrm{ALL}_{CFL} = \{\langle M\rangle \mid M \text{ is a PDA and } L(M) = \Sigma^*\}$.

Summary of Module III

Introduction to Turing machines

Equivalent models

- multi-tape TM,
- non-deterministic TM

Turing decidable languages

Turing recognizable languages

Diagonalization in automata theory

Proving undecidability

- $A_{TM} = \{(M, w) \mid M \text{ accepts } w\}$,
- $\text{Halt} = \{(M, w) \mid M \text{ hants on } w\}$,
- $E_{TM} = \{\langle M \rangle \mid L(M) = \varnothing\}$, $EQ_{TM} = \{(M_1, M_2) \mid L(M_1) = L(M_2)\}$,
- $REG_{TM} = \{\langle M \rangle \mid L(M) \text{ is regular}\}$,
- $\text{ALL}_{CFL} = \{\langle M \rangle \mid M \text{ is a PDA and } L(M) = \Sigma^*\}$.

Rice's theorem

Summary of Module III

Introduction to Turing machines

Equivalent models

- multi-tape TM,
- non-deterministic TM

Turing decidable languages

Turing recognizable languages

Diagonalization in automata theory

Proving undecidability

- $A_{TM} = \{(M, w) \mid M \text{ accepts } w\}$, Halt $= \{(M, w) \mid M \text{ hants on } w\}$, $E_{TM} = \{\langle M\rangle \mid L(M) = \varnothing\}$, $EQ_{TM} = \{(M_1, M_2) \mid L(M_1) = L(M_2)\}$, $REG_{TM} = \{\langle M\rangle \mid L(M) \text{ is regular}\}$, $\text{ALL}_{CFL} = \{\langle M\rangle \mid M \text{ is a PDA and } L(M) = \Sigma^*\}$.

Rice's theorem

MPCP problem

Summary of Module III

Introduction to Turing machines

Equivalent models

- multi-tape TM,
- non-deterministic TM

Turing decidable languages

Turing recognizable languages

Diagonalization in automata theory

Proving undecidability

- $A_{TM} = \{(M, w) \mid M \text{ accepts } w\}$,
- Halt $= \{(M, w) \mid M \text{ hants on } w\}$,
- $E_{TM} = \{\langle M \rangle \mid L(M) = \varnothing\}$, $EQ_{TM} = \{(M_1, M_2) \mid L(M_1) = L(M_2)\}$,
- $REG_{TM} = \{\langle M \rangle \mid L(M) \text{ is regular}\}$,
- $\text{ALL}_{CFL} = \{\langle M \rangle \mid M \text{ is a PDA and } L(M) = \Sigma^*\}$.

Rice's theorem

MPCP problem (Tutorial 10)

Summary of Module III

Introduction to Turing machines

Equivalent models

- multi-tape TM,
- non-deterministic TM

Turing decidable languages

Turing recognizable languages

Diagonalization in automata theory

Proving undecidability

$A_{TM} = \{(M, w) \mid M \text{ accepts } w\}$,
Halt $= \{(M, w) \mid M \text{ hants on } w\}$,
$E_{TM} = \{\langle M \rangle \mid L(M) = \emptyset\}$, $EQ_{TM} = \{(M_1, M_2) \mid L(M_1) = L(M_2)\}$,
$REG_{TM} = \{\langle M \rangle \mid L(M) \text{ is regular}\}$,
$\text{ALL}_{CFL} = \{\langle M \rangle \mid M \text{ is a PDA and } L(M) = \Sigma^*\}$.

Rice's theorem

MPCP problem (Tutorial 10)

Notion of reduction

Summary of Module III

Introduction to Turing machines

Equivalent models

- multi-tape TM,
- non-deterministic TM

Turing decidable languages

Turing recognizable languages

Diagonalization in automata theory

Proving undecidability

- $A_{TM} = \{(M, w) \mid M \text{ accepts } w\}$,
- Halt $= \{(M, w) \mid M \text{ hants on } w\}$,
- $E_{TM} = \{\langle M\rangle \mid L(M) = \emptyset\}$, $EQ_{TM} = \{(M_1, M_2) \mid L(M_1) = L(M_2)\}$,
- $REG_{TM} = \{\langle M\rangle \mid L(M) \text{ is regular}\}$,
- $\text{ALL}_{CFL} = \{\langle M\rangle \mid M \text{ is a PDA and } L(M) = \Sigma^*\}$.

Rice's theorem

MPCP problem (Tutorial 10)

Notion of reduction (Tutorial 9)

At the end of last class

Undecidability of the following languages:

$A_{TM} = \{(M, w) \mid M \text{ accepts } w\}.$

$\text{Halt} = \{(M, w) \mid M \text{ hants on } w\}.$

$E_{TM} = \{\langle M \rangle \mid L(M) = \varnothing\}.$

$EQ_{TM} = \{(M_1, M_2) \mid L(M_1) = L(M_2)\}.$

$REG_{TM} = \{\langle M \rangle \mid L(M) \text{ is regular}\}.$

$\text{ALL}_{CFL} = \{\langle M \rangle \mid M \text{ is a PDA and } L(M) = \Sigma^*\}.$

Note that undecidability of REG_{TM} and E_{TM} can be proved using Rice's theorem.

Module IV: Effective computation

Turing machines with resource constraints.

Module IV: Effective computation

Turing machines with resource constraints.

Resources for computation.

Module IV: Effective computation

Turing machines with resource constraints.

- Resources for computation.
  - Time

Module IV: Effective computation

Turing machines with resource constraints.

- Resources for computation.
  - Time: the number steps for which the TM runs

Module IV: Effective computation

Turing machines with resource constraints.

- Resources for computation.
  - Time: the number steps for which the TM runs
  - Space

Module IV: Effective computation

Turing machines with resource constraints.

- Resources for computation.
  - Time: the number steps for which the TM runs
  - Space: the number of different cells on which the TM writes

Module IV: Effective computation

Turing machines with resource constraints.

- Resources for computation.
  - Time: the number steps for which the TM runs
  - Space: the number of different cells on which the TM writes
  - The number of times an input bit can be read

Module IV: Effective computation

Turing machines with resource constraints.

- Resources for computation.
  - Time: the number steps for which the TM runs
  - Space: the number of different cells on which the TM writes
  - The number of times an input bit can be read
  - The amount of energy used

Module IV: Effective computation

Turing machines with resource constraints.

- Resources for computation.
  - Time: the number steps for which the TM runs
  - Space: the number of different cells on which the TM writes
  - The number of times an input bit can be read
  - The amount of energy used
  - ⋮

Module IV: Effective computation

Turing machines with resource constraints.

- Resources for computation.
  - Time: the number steps for which the TM runs
  - Space: the number of different cells on which the TM writes
  - The number of times an input bit can be read
  - The amount of energy used
  - ⋮
- Why bound resources?
  - Viewing TM as algorithms.

Module IV: Effective computation

Turing machines with resource constraints.

- Resources for computation.
  - Time: the number steps for which the TM runs
  - Space: the number of different cells on which the TM writes
  - The number of times an input bit can be read
  - The amount of energy used
  - ⋮
- Why bound resorces?
  - Viewing TM as algorithms.
  - TM to help in computation of important problems.

Module IV: Effective computation

Turing machines with resource constraints.

- Resources for computation.
  - Time: the number steps for which the TM runs
  - Space: the number of different cells on which the TM writes
  - The number of times an input bit can be read
  - The amount of energy used
  - ⋮
- Why bound resources?
  - Viewing TM as algorithms.
  - TM to help in computation of important problems.
  - Finer study of decidable languages.

Module IV: Effective computation

Turing machines with resource constraints.

- Resources for computation.
  - Time: the number steps for which the TM runs
  - Space: the number of different cells on which the TM writes
  - The number of times an input bit can be read
  - The amount of energy used
  - ⋮
- Why bound resorces?
  - Viewing TM as algorithms.
  - TM to help in computation of important problems.
  - Finer study of decidable languages.
- How should we bound the resources?

Module IV: Effective computation

Turing machines with resource constraints.

- Resources for computation.
  - Time: the number steps for which the TM runs
  - Space: the number of different cells on which the TM writes
  - The number of times an input bit can be read
  - The amount of energy used
  - ⋮
- Why bound resorces?
  - Viewing TM as algorithms.
  - TM to help in computation of important problems.
  - Finer study of decidable languages.
- How should we bound the resources?
  - Many different ways exist. . . .

Time complexity and complexity classes

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

Time complexity and complexity classes

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

Definition

A language $L \subseteq \Sigma^*$ is said to be in class $\mathrm{TIME}(t(n))$

Time complexity and complexity classes

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

Definition

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

Time complexity and complexity classes

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

Definition

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

M halts on x in time $O(t(|x|))$, where $|x|$ indicates the length of x.

Time complexity and complexity classes

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

Definition

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

- M halts on x in time $O(t(|x|))$, where $|x|$ indicates the length of x.
- if $x \in L$ then M accepts x.

Time complexity and complexity classes

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

Definition

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

- M halts on x in time $O(t(|x|))$, where $|x|$ indicates the length of x.
- if $x \in L$ then M accepts x.
- if $x \notin L$ then M rejects x.

Time complexity and complexity classes

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

Definition

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

- M halts on x in time $O(t(|x|))$, where $|x|$ indicates the length of x.
- if $x \in L$ then M accepts x.
- if $x \notin L$ then M rejects x.

$$P = \bigcup_k \mathrm{TIME}(n^k)$$

Time complexity and complexity classes

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

Definition

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

M halts on x in time $O(t(|x|))$, where $|x|$ indicates the length of x.

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

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

$$P = \bigcup_k \text{TIME}(n^k)$$

$$EXP = \bigcup_k \text{TIME}(2^{n^k})$$

Time complexity and complexity classes

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

Time complexity and complexity classes

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

Definition

A language $L \subseteq \Sigma^*$ is said to be in class $\mathrm{NTIME}(t(n))$

Time complexity and complexity classes

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

Definition

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

Time complexity and complexity classes

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

Definition

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

> each run of M halts on x in time $O(t(|x|))$, where $|x|$ indicates the length of x.

Time complexity and complexity classes

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

Definition

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

each run of M halts on x in time $O(t(|x|))$, where $|x|$ indicates the length of x.

if $x \in L$ then

Time complexity and complexity classes

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

Definition

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

- each run of M halts on x in time $O(t(|x|))$, where $|x|$ indicates the length of x.
- if $x \in L$ then M accepts x on at least one run.

Time complexity and complexity classes

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

Definition

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

- each run of M halts on x in time $O(t(|x|))$, where $|x|$ indicates the length of x.
- if $x \in L$ then M accepts x on at least one run.
- if $x \notin L$ then

Time complexity and complexity classes

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

Definition

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

- each run of M halts on x in time $O(t(|x|))$, where $|x|$ indicates the length of x.
- if $x \in L$ then M accepts x on at least one run.
- if $x \notin L$ then M rejects x on all runs.

Time complexity and complexity classes

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

Definition

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

- each run of M halts on x in time $O(t(|x|))$, where $|x|$ indicates the length of x.
- if $x \in L$ then M accepts x on at least one run.
- if $x \notin L$ then M rejects x on all runs.

$$NP = \bigcup_k \mathrm{NTIME}(n^k)$$

Time complexity and complexity classes

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

Definition

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

- each run of M halts on x in time $O(t(|x|))$, where $|x|$ indicates the length of x.
- if $x \in L$ then M accepts x on at least one run.
- if $x \notin L$ then M rejects x on all runs.

$$NP = \bigcup_k \mathrm{NTIME}(n^k)$$

$$NEXP = \bigcup_k \mathrm{NTIME}(2^{n^k})$$

Relationships between models

Lemma

Let $t(n) > n$. Let L be a language decided by a multitape TM in time $t(n)$.

Relationships between models

Lemma

Let $t(n) > n$. Let L be a language decided by a multitape TM in time $t(n)$. Then there is a single tape TM that decides L in time

Relationships between models

Lemma

Let $t(n) > n$*. Let* L *be a language decided by a multitape TM in time* $t(n)$*. Then there is a single tape TM that decides* L *in time* $O((t(n))^2)$*.*

Relationships between models

Lemma

Let $t(n) > n$. Let L be a language decided by a multitape TM in time $t(n)$. Then there is a single tape TM that decides L in time $O((t(n))^2)$.

Proof idea:

Relationships between models

Lemma

Let $t(n) > n$. Let L be a language decided by a multitape TM in time $t(n)$. Then there is a single tape TM that decides L in time $O((t(n))^2)$.

Proof idea:

East step of multitape machine can be executed on a single tape machine in time $O\left(t(n)\right)$.

Relationships between models

Lemma

Let $t(n) > n$. Let L be a language decided by a multitape TM in time $t(n)$. Then there is a single tape TM that decides L in time $O((t(n))^2)$.

Proof idea:

> East step of multitape machine can be executed on a single tape machine in time $O(t(n))$.

Lemma

Let $t(n) > n$. Let L be a language decided by a non-deterministic TM in time $t(n)$.

Relationships between models

Lemma

Let $t(n) > n$. Let L be a language decided by a multitape TM in time $t(n)$. Then there is a single tape TM that decides L in time $O((t(n))^2)$.

Proof idea:

> East step of multitape machine can be executed on a single tape machine in time $O(t(n))$.

Lemma

Let $t(n) > n$. Let L be a language decided by a non-deterministic TM in time $t(n)$. Then there is a deterministic TM that decides L in time

Relationships between models

Lemma

Let $t(n) > n$. Let L be a language decided by a multitape TM in time $t(n)$. Then there is a single tape TM that decides L in time $O((t(n))^2)$.

Proof idea:

> East step of multitape machine can be executed on a single tape machine in time $O(t(n))$.

Lemma

Let $t(n) > n$. Let L be a language decided by a non-deterministic TM in time $t(n)$. Then there is a deterministic TM that decides L in time $2^{O(t(n))}$.

Relationships between models

Lemma

Let $t(n) > n$. Let L be a language decided by a multitape TM in time $t(n)$. Then there is a single tape TM that decides L in time $O((t(n))^2)$.

Proof idea:

> East step of multitape machine can be executed on a single tape machine in time $O(t(n))$.

Lemma

Let $t(n) > n$. Let L be a language decided by a non-deterministic TM in time $t(n)$. Then there is a deterministic TM that decides L in time $2^{O(t(n))}$.

Proof idea:

Relationships between models

Lemma

Let $t(n) > n$. Let L be a language decided by a multitape TM in time $t(n)$. Then there is a single tape TM that decides L in time $O((t(n))^2)$.

Proof idea:

> East step of multitape machine can be executed on a single tape machine in time $O(t(n))$.

Lemma

Let $t(n) > n$. Let L be a language decided by a non-deterministic TM in time $t(n)$. Then there is a deterministic TM that decides L in time $2^{O(t(n))}$.

Proof idea: DIY