CS310 : Automata Theory 2019

Lecture 26: Turing Machines and Decidability

Instructor: S. Akshay

IITB, India

11-03-2019

Recap: Turing machines and their variants

Turing machines

- Finite state automata with infinite tape

Recap: Turing machines and their variants

Turing machines

- Finite state automata with infinite tape
- Formal definition as a 7 tuple.

Recap: Turing machines and their variants

Turing machines

- Finite state automata with infinite tape
- Formal definition as a 7 tuple.
- Configurations and computations of a TM

Recap: Turing machines and their variants

Turing machines

- Finite state automata with infinite tape
- Formal definition as a 7 tuple.
- Configurations and computations of a TM
- Turing recognizable and decidable languages

Recap: Turing machines and their variants

Turing machines

- Finite state automata with infinite tape
- Formal definition as a 7 tuple.
- Configurations and computations of a TM
- Turing recognizable and decidable languages

Variants of a Turing machine

- Single vs Multiple heads
- Single vs Multiple tapes
- Deterministic vs Non-deterministic TM

Decidable languages

- A TM *accepts* language L if it has an *accepting run* on each word in L.
- A TM *decides* language L if it accepts L and halts on all inputs.

Decidable and Turing recognizable languages

- A language L is *decidable (recursive)* if there exists a Turing machine M which decides L (i.e., M halts on all inputs and M accepts L).
- A language L is *Turing recognizable (recursively enumerable)* if there exists a Turing machine M which accepts L.

Algorithms and Decidability

Algorithms $\Longleftrightarrow$ Decidable (i.e, TM decides it)

- A decision problem P is said to be *decidable (i.e., have an algorithm)* if the language L of all *yes* instances to P is decidable.
- A decision problem P is said to be *semi-decidable (i.e., have a semi-algorithm)* if the language L of all *yes* instances to P is r.e.
- A decision problem P is said to be *undecidable* if the language L of all *yes* instances to P is not decidable.

Examples of Decidable languages and problems

- (Acceptance problem for DFA) Given a DFA does it accept a given word?

Examples of Decidable languages and problems

- (Acceptance problem for DFA) Given a DFA does it accept a given word?
  - $L^A_{DFA} = \{< B, w >|\ B \text{ is a DFA that accepts input word } w\}$

Examples of Decidable languages and problems

- (Acceptance problem for DFA) Given a DFA does it accept a given word?
  - $L^A_{DFA} = \{< B, w >|\ B$ is a DFA that accepts input word $w\}$

But how do we encode $< B >$? In general a TM?

Encoding Turing machines as strings

Every Turing machine can be encoded as a string in $\{0, 1\}^*$

Encoding Turing machines as strings

Every Turing machine can be encoded as a string in $\{0,1\}^*$

Just encode the description of the machine (in binary)!

Encoding Turing machines as strings

Every Turing machine can be encoded as a string in $\{0, 1\}^*$

Just encode the description of the machine (in binary)!

Every string in $\{0, 1\}^*$ is a TM

Encoding Turing machines as strings

Every Turing machine can be encoded as a string in $\{0,1\}^*$

Just encode the description of the machine (in binary)!

Every string in $\{0,1\}^*$ is a TM

Of course, some strings don't represent a valid encoding, then we can map them to a TM that does nothing.

Encoding Turing machines as strings

Every Turing machine can be encoded as a string in $\{0,1\}^*$

Just encode the description of the machine (in binary)!

Every string in $\{0,1\}^*$ is a TM

Of course, some strings don't represent a valid encoding, then we can map them to a TM that does nothing.

Notation

- $M \rightarrow < M >$, a string representation of M.
- $\alpha \rightarrow M_\alpha$, a TM representation of a string.

How many Turing machines are there?

Countable set

A set is countable if there is an injective map from the set to $\mathbb{N}$.

How many Turing machines are there?

Countable set

A set is countable if there is an injective map from the set to $\mathbb{N}$.

The set $\{0, 1\}^*$ is countable

Examples of Decidable languages and problems

- (Acceptance problem for DFA) Given a DFA does it accept a given word?
- (Emptiness problem for DFA) Given a DFA does it accept any word?
- (Equivalence problem for DFA) Given two DFAs, do they accept the same language?

Examples of Decidable languages and problems

- (Acceptance problem for DFA) Given a DFA does it accept a given word?
  - $L^A_{DFA} = \{< B, w >|\ B \text{ is a DFA that accepts input word } w\}$
- (Emptiness problem for DFA) Given a DFA does it accept any word?
- (Equivalence problem for DFA) Given two DFAs, do they accept the same language?

Examples of Decidable languages and problems

- (Acceptance problem for DFA) Given a DFA does it accept a given word?
  - $L^{A}_{DFA} = \{< B, w >| B \text{ is a DFA that accepts input word } w\}$
- (Emptiness problem for DFA) Given a DFA does it accept any word?
  - $L^{\emptyset}_{DFA} = \{< B >| B \text{ is a DFA, } L(B) = \emptyset\}$
- (Equivalence problem for DFA) Given two DFAs, do they accept the same language?

Examples of Decidable languages and problems

- (Acceptance problem for DFA) Given a DFA does it accept a given word?
  - $L^{A}_{DFA} = \{< B, w >\mid B \text{ is a DFA that accepts input word } w\}$
- (Emptiness problem for DFA) Given a DFA does it accept any word?
  - $L^{\emptyset}_{DFA} = \{< B >\mid B \text{ is a DFA, } L(B) = \emptyset\}$
- (Equivalence problem for DFA) Given two DFAs, do they accept the same language?
  - $L^{EQ}_{DFA} = \{< A, B >\mid A, B \text{ are DFAs, } L(A) = L(B)\}$

Examples of Decidable languages and problems

- (Acceptance problem for DFA) Given a DFA does it accept a given word?
  - $L^{A}_{DFA} = \{< B, w >| B$ is a DFA that accepts input word $w\}$
- (Emptiness problem for DFA) Given a DFA does it accept any word?
  - $L^{\emptyset}_{DFA} = \{< B >| B$ is a DFA, $L(B) = \emptyset\}$
- (Equivalence problem for DFA) Given two DFAs, do they accept the same language?
  - $L^{EQ}_{DFA} = \{< A, B >| A, B$ are DFAs, $L(A) = L(B)\}$
- What about NFAs, regular expressions

Relationship among languages

$$\text{Regular} \subsetneq \text{Decidable} \underset{?}{\subseteq} \text{Turing recognizable} \underset{?}{\subseteq} \text{All languages}$$

Relationship among languages

$$\text{Regular} \subsetneq \text{Decidable} \underset{?}{\subsetneq} \text{Turing recognizable} \underset{?}{\subsetneq} \text{All languages}$$

$$\text{DFA/NFA} < \text{Algorithms/Halting TM} \underset{?}{\leq} \text{Semi-algorithms/TM}$$

Relationship among languages

$$\text{Regular} \subsetneq \text{Decidable} \underset{?}{\subseteq} \text{Turing recognizable} \underset{?}{\subseteq} \text{All languages}$$

$$\text{DFA/NFA} < \text{Algorithms/Halting TM} \underset{?}{\leq} \text{Semi-algorithms/TM}$$

Languages outside R.E.

Thm: There exist languages that are not R.E

- Number of R.E languages is countable. Why?

Languages outside R.E.

Thm: There exist languages that are not R.E

- Number of R.E languages is countable. Why?
- Set S of all words over a finite alphabet Σ is countably infinite.

Languages outside R.E.

Thm: There exist languages that are not R.E

- Number of R.E languages is countable. Why?
- Set S of all words over a finite alphabet Σ is countably infinite.
- Set of all languages over Σ is the set of subsets of S and is therefore uncountable Why?

Languages outside R.E.

Thm: There exist languages that are not R.E

- Number of R.E languages is countable. Why?
- Set S of all words over a finite alphabet Σ is countably infinite.
- Set of all languages over Σ is the set of subsets of S and is therefore uncountable Why? - recall Cantor from Discrete Structure's course.

Languages outside R.E.

Thm: There exist languages that are not R.E

- Number of R.E languages is countable. Why?
- Set S of all words over a finite alphabet Σ is countably infinite.
- Set of all languages over Σ is the set of subsets of S and is therefore uncountable Why? - recall Cantor from Discrete Structure's course.
- So for some such language, there must be no accepting TM.

Diagonalization