CS310 : Automata Theory 2019

Lecture 31: Rice's theorem and other undecidable problems

Instructor: S. Akshay

IITB, India

25-03-2019

Recap

Turing machines and computability

1. Definition of Turing machines: high level and low-level descriptions
2. Variants of Turing machines
3. Decidable and Turing recognizable languages
4. Church-Turing Hypothesis
5. Undecidability and a proof technique by diagonalization
   - A universal TM lang $L^A_{TM} = \{\langle M, w\rangle \mid M \text{ is a TM and } M \text{ accepts } w\}$
6. Reductions: a powerful way to show undecidability.
7. Rice's theorem

Rice's Theorem

Let P be a non-trivial property of r.e. languages. Then $\mathcal{L}_P = \{\langle M \rangle \mid L(M) \in P\}$ is undecidable.

Rice's Theorem

Let P be a non-trivial property of r.e. languages. Then $\mathcal{L}_P = \{\langle M \rangle \mid L(M) \in P\}$ is undecidable.

For the following, is Rice's theorem applicable?

1. $\{\langle M \rangle \mid M$ runs for 5 steps on word $010\}$. No. Property of TMs.
2. $\{\langle M \rangle \mid M$ has at most 25 states.$\}$. No. Property of TMs.
3. $\{\langle M \rangle \mid L(M)$ is recognized by a TM with at least 25 states.$\}$. No. Trivial property.
4. $\{\langle M \rangle \mid L(M)$ is recognized by a TM with at most 25 states and tape alphabet at most 10.$\}$. Yes.
5. $\{\langle M \rangle \mid L(M)$ is infinite.$\}$. Yes.
6. $\{\langle M \rangle \mid M$ with alphabet $\{0, 1, \sqcup\}$ ever prints three consecutive $1's$ on the tape$\}$. No. Property of TMs, but undecidable!

Rice's Theorem

Let P be a non-trivial property of r.e. languages. Then $\mathcal{L}_P = \{\langle M \rangle \mid L(M) \in P\}$ is undecidable.

For the following, is Rice's theorem applicable?

1. $\{\langle M \rangle \mid M$ runs for 5 steps on word $010\}$. No. Property of TMs.
2. $\{\langle M \rangle \mid M$ has at most 25 states.$\}$. No. Property of TMs.
3. $\{\langle M \rangle \mid L(M)$ is recognized by a TM with at least 25 states.$\}$. No. Trivial property.
4. $\{\langle M \rangle \mid L(M)$ is recognized by a TM with at most 25 states and tape alphabet at most 10.$\}$. Yes.
5. $\{\langle M \rangle \mid L(M)$ is infinite.$\}$. Yes.
6. $\{\langle M \rangle \mid M$ with alphabet $\{0, 1, \sqcup\}$ ever prints three consecutive $1's$ on the tape$\}$. No. Property of TMs, but undecidable!

- For No answers, language can still be decidable or undecidable.
- If Rice's theorem does not apply, fall back on reductions!

Proof idea

Rice's Theorem
Let P be a non-trivial property of r.e. languages. Then $\mathcal{L}_P = \{\langle M \rangle \mid L(M) \in P\}$ is undecidable.

- Let P be a non-trivial property of r.e, such that $\emptyset \notin P$.

Proof idea

Rice's Theorem
Let P be a non-trivial property of r.e. languages. Then
$\mathcal{L}_P = \{\langle M \rangle \mid L(M) \in P\}$ is undecidable.

- Let P be a non-trivial property of r.e, such that $\emptyset \notin P$.
- Since P is non-trivial, $\exists L, M_L$ with property P.

Proof idea

Rice's Theorem
Let P be a non-trivial property of r.e. languages. Then $\mathcal{L}_P = \{\langle M \rangle \mid L(M) \in P\}$ is undecidable.

- Let P be a non-trivial property of r.e, such that $\emptyset \notin P$.
- Since P is non-trivial, $\exists L, M_L$ with property P.
- If P is decidable, there exists an algo M_P for deciding P

Proof idea

Rice's Theorem
Let P be a non-trivial property of r.e. languages. Then $\mathcal{L}_P = \{\langle M \rangle \mid L(M) \in P\}$ is undecidable.

- Let P be a non-trivial property of r.e, such that $\emptyset \notin P$.
- Since P is non-trivial, $\exists L, M_L$ with property P.
- If P is decidable, there exists an algo M_P for deciding P
- We combine M_L and M_P to get algo for A_{TM}.

Proof idea

Rice's Theorem
Let P be a non-trivial property of r.e. languages. Then
$\mathcal{L}_P = \{\langle M \rangle \mid L(M) \in P\}$ is undecidable.

- Let P be a non-trivial property of r.e, such that $\emptyset \notin P$.
- Since P is non-trivial, $\exists L, M_L$ with property P.
- If P is decidable, there exists an algo M_P for deciding P
- We combine M_L and M_P to get algo for A_{TM}.
  - For $\langle M, w \rangle$ i/p, design $\langle M' \rangle$, s.t $L(M') \in P$ iff M acc w.

Proof idea

Rice's Theorem
Let P be a non-trivial property of r.e. languages. Then
$\mathcal{L}_P = \{\langle M \rangle \mid L(M) \in P\}$ is undecidable.

- Let P be a non-trivial property of r.e, such that $\emptyset \notin P$.
- Since P is non-trivial, $\exists L, M_L$ with property P.
- If P is decidable, there exists an algo M_P for deciding P
- We combine M_L and M_P to get algo for A_{TM}.
  - For $\langle M, w \rangle$ i/p, design $\langle M' \rangle$, s.t $L(M') \in P$ iff M acc w.
  - M' is the foll:
    1. ignore i/p x , simulate M on w. if reject, then rejects x.
    2. if acc, then simulate M_L on x, acc iff M_L acc x.

Proof idea

Rice's Theorem
Let P be a non-trivial property of r.e. languages. Then $\mathcal{L}_P = \{\langle M \rangle \mid L(M) \in P\}$ is undecidable.

- Let P be a non-trivial property of r.e, such that $\emptyset \notin P$.
- Since P is non-trivial, $\exists L, M_L$ with property P.
- If P is decidable, there exists an algo M_P for deciding P
- We combine M_L and M_P to get algo for A_{TM}.
  - For $\langle M, w \rangle$ i/p, design $\langle M' \rangle$, s.t $L(M') \in P$ iff M acc w.
  - M' is the foll:
    1. ignore i/p x , simulate M on w. if reject, then rejects x.
    2. if acc, then simulate M_L on x, acc iff M_L acc x.
  - Thus M' either acc $\emptyset$ or L depending on if M acc w.

Proof idea

Rice's Theorem

Let P be a non-trivial property of r.e. languages. Then $\mathcal{L}_P = \{\langle M \rangle \mid L(M) \in P\}$ is undecidable.

- Let P be a non-trivial property of r.e, such that $\emptyset \notin P$.
- Since P is non-trivial, $\exists L, M_L$ with property P.
- If P is decidable, there exists an algo M_P for deciding P
- We combine M_L and M_P to get algo for A_{TM}.
  - For $\langle M, w \rangle$ i/p, design $\langle M' \rangle$, s.t $L(M') \in P$ iff M acc w.
  - M' is the foll:
    1. ignore i/p x , simulate M on w. if reject, then rejects x.
    2. if acc, then simulate M_L on x, acc iff M_L acc x.
  - Thus M' either acc $\emptyset$ or L depending on if M acc w.
- Thus $\langle M' \rangle \in \mathcal{L}_P$ iff $L(M') \in P$ iff $L(M') = L$ iff M acc w.
- This gives an algo for A_{TM} so contradiction!

Proof idea

- But what if P has $\emptyset$ in it!?

Proof idea

- But what if P has $\emptyset$ in it!?
- Take $\overline{P}$.
- Now $\emptyset \notin \overline{P}$.

Proof idea

- But what if P has $\emptyset$ in it!?
- Take $\overline{P}$.
- Now $\emptyset \notin \overline{P}$.
- Apply proof to get undecidability of $\mathcal{L}_{\overline{P}}$.
- Conclude undecidability of $\mathcal{L}_P$.

More “useful” undecidability?

Are only problems about Turing machines undecidable?

More "useful" undecidability?

Are only problems about Turing machines undecidable?

- Computers, C-programs, counter machines

More “useful” undecidability?

Are only problems about Turing machines undecidable?

- Computers, C-programs, counter machines But these are just Turing machines?

More "useful" undecidability?

Are only problems about Turing machines undecidable?

- Computers, C-programs, counter machines
- Problems on CFLs: Given CFG G, is $L(G) = \Sigma^*$?
- Problems on Tiling

More "useful" undecidability?

Are only problems about Turing machines undecidable?

- Computers, C-programs, counter machines
- Problems on CFLs: Given CFG G, is $L(G) = \Sigma^*$?
- Problems on Tiling
- Problems on String Matching

A simple programming exercise

A string matching problem

Given two lists $A = \{s_1, \ldots s_n\}$ and $B = \{t_1, \ldots, t_n\}$, over the same alphabet, is there a sequence of combining elements that produces the same string in both lists?

A simple programming exercise

A string matching problem

Given two lists $A = \{s_1, \ldots s_n\}$ and $B = \{t_1, \ldots, t_n\}$, over the same alphabet, is there a sequence of combining elements that produces the same string in both lists?

- Does there exist a finite sequence $1 \leq i_1, \ldots, i_m \leq n$ such that

$$s_{i_1} \ldots s_{i_m} = t_{i_1} \ldots t_{i_m}$$

A simple programming exercise

A string matching problem

Given two lists $A = \{s_1, \ldots s_n\}$ and $B = \{t_1, \ldots, t_n\}$, over the same alphabet, is there a sequence of combining elements that produces the same string in both lists?

- Does there exist a finite sequence $1 \leq i_1, \ldots, i_m \leq n$ such that

$$s_{i_1} \ldots s_{i_m} = t_{i_1} \ldots t_{i_m}$$

Consider the following lists

- $A = \{110, 0011, 0110\}$ and $B = \{110110, 00, 110\}$

A simple programming exercise

A string matching problem

Given two lists $A = \{s_1, \ldots s_n\}$ and $B = \{t_1, \ldots, t_n\}$, over the same alphabet, is there a sequence of combining elements that produces the same string in both lists?

- Does there exist a finite sequence $1 \leq i_1, \ldots, i_m \leq n$ such that

$$s_{i_1} \ldots s_{i_m} = t_{i_1} \ldots t_{i_m}$$

Consider the following lists

- $A = \{110, 0011, 0110\}$ and $B = \{110110, 00, 110\}$ 2, 3, 1!

A simple programming exercise

A string matching problem

Given two lists $A = \{s_1, \ldots s_n\}$ and $B = \{t_1, \ldots, t_n\}$, over the same alphabet, is there a sequence of combining elements that produces the same string in both lists?

- Does there exist a finite sequence $1 \leq i_1, \ldots, i_m \leq n$ such that

$$s_{i_1} \ldots s_{i_m} = t_{i_1} \ldots t_{i_m}$$

Consider the following lists

- $A = \{110, 0011, 0110\}$ and $B = \{110110, 00, 110\}$ 2, 3, 1!
- $A = \{0011, 11, 1101\}$ and $B = \{101, 011, 110\}$

A simple programming exercise

A string matching problem

Given two lists $A = \{s_1, \dots s_n\}$ and $B = \{t_1, \dots, t_n\}$, over the same alphabet, is there a sequence of combining elements that produces the same string in both lists?

- Does there exist a finite sequence $1 \leq i_1, \dots, i_m \leq n$ such that

$$s_{i_1} \dots s_{i_m} = t_{i_1} \dots t_{i_m}$$

Consider the following lists

- $A = \{110, 0011, 0110\}$ and $B = \{110110, 00, 110\}$ 2, 3, 1!
- $A = \{0011, 11, 1101\}$ and $B = \{101, 011, 110\}$
- $A = \{100, 0, 1\}$ and $B = \{1, 100, 0\}$

A simple programming exercise

A string matching problem

Given two lists $A = \{s_1, \ldots s_n\}$ and $B = \{t_1, \ldots, t_n\}$, over the same alphabet, is there a sequence of combining elements that produces the same string in both lists?

- Does there exist a finite sequence $1 \leq i_1, \ldots, i_m \leq n$ such that

$$s_{i_1} \ldots s_{i_m} = t_{i_1} \ldots t_{i_m}$$

Consider the following lists

- $A = \{110, 0011, 0110\}$ and $B = \{110110, 00, 110\}$ 2, 3, 1!
- $A = \{0011, 11, 1101\}$ and $B = \{101, 011, 110\}$
- $A = \{100, 0, 1\}$ and $B = \{1, 100, 0\}$

Can you write an algorithm for solving this?