Theory of Computation
Decidable Languages

Arjun Chandrasekhar

State Machines as Input and Output

- A Turing Machine can take a DFA as input
  - This is what you did on assignment 1!
- A Turing Machine can create a DFA
  - This is what you did on assignment 2!
- A Turing machine can take a TM as input
  - Think about how the java compiler takes your source code as it its input
- A Turing machine can create other TMs
  - Think about how the compiler uses your source code to create executable files

Decidable DFA Properties

Given an arbitrary DFA D, can we write a program to determine certain properties about it?

- Does D accept a particular string w?
- Does D accept *anything*? Does it reject everything?
- Does D accept every string that contains 111?

The language $\mathrm{A}_{\mathrm{DFA}}$

Consider the following language

$$\mathrm{A}_{\mathrm{DFA}} = \{\langle D, w\rangle | D \text{ is a DFA description}, w \in L(D)\}$$

What is the input to the decision problem associated with this language?

A) A description of a DFA

B) A regular language

C) A DFA description, and an input string

D) A DFA that is designed to recognize the string *w*

The language $\mathrm{A_{DFA}}$

Consider the following language

$$\mathrm{A_{DFA}} = \{\langle D, w\rangle | D \text{ is a DFA description}, w \in L(D)\}$$

What is the input to the decision problem associated with this language?

A) A description of a DFA

C) A DFA description, and an input string ✓

B) A regular language

D) A DFA that is designed to recognize the string w

Decidability of $\mathrm{A}_{\mathrm{DFA}}$

Consider the following language

$$\mathrm{A}_{\mathrm{DFA}} = \{\langle D, w\rangle | D \text{ is a DFA description}, w \in L(D)\}$$

What should we do to show that L is decidable?

A) Design a DFA that accepts the string w

B) Design a TM that accepts the string w

C) Design a TM that designs a DFA D that accepts w

D) Design a TM that checks if D accepts w

Decidability of $\mathrm{A}_{\mathrm{DFA}}$

Consider the following language

$$\mathrm{A}_{\mathrm{DFA}} = \{\langle D, w\rangle | D \text{ is a DFA description}, w \in L(D)\}$$

What should we do to show that L is decidable?

A) Design a DFA that accepts the string w

B) Design a TM that accepts the string w

C) Design a TM that designs a DFA D that accepts w

D) Design a TM that checks if D accepts w ✓

Decidability of $\mathrm{A}_{\mathrm{DFA}}$

Let's prove that the following language is decidable

$\mathrm{A}_{\mathrm{DFA}} = \{\langle D, w\rangle | D \text{ is a DFA description}, w \in L(D)\}$

- You did this on Program 1!
- Create a machine M that decides $\mathrm{A}_{\mathrm{DFA}}$
  1. M takes $\langle D, w\rangle$ as input
  2. Construct the state graph for D
  3. Simulate D on w
  4. If D accepts w, then M accepts $\langle D, w\rangle$
  5. Otherwise, M rejects $\langle D, w\rangle$

Decidable DFA Properties

Prove that the following language is decidable

$$\mathrm{A}_{\mathrm{NFA}} = \{\langle N, w\rangle | N \text{ is an NFA description}, w \in L(N)\}$$

- You did this on Program 2!
- Create a machine M that decides $\mathrm{A}_{\mathrm{NFA}}$
  1. M takes $\langle N, w\rangle$ as input
  2. Convert N to a a DFA D
  3. Check if $\langle D, w\rangle \in \mathrm{A}_{\mathrm{DFA}}$
     - 3.1 If $\langle D, w\rangle \in \mathrm{A}_{\mathrm{DFA}}$, M accepts $\langle N, w\rangle$
     - 3.2 Otherwise, M rejects $\langle N, w\rangle$

This is an example of "reducing" or "converting" one problem ($\mathrm{A}_{\mathrm{NFA}}$) to another one ($\mathrm{A}_{\mathrm{DFA}}$) that we already know how to solve

Decidability of $\mathrm{A}_{\mathrm{REG}}$

Prove that the following language is decidable

$$\mathrm{A}_{\mathrm{REG}} = \{\langle R, w\rangle | R \text{ is a regex}, w \in L(R)\}$$

- You will do this on Program 3!
- Create a machine M that decides $\mathrm{A}_{\mathrm{REG}}$
  1. M takes $\langle R, w\rangle$ as input
  2. Convert R to an equivalent NFA N
  3. Check if $\langle N, w\rangle \in \mathrm{A}_{\mathrm{NFA}}$
     - 3.1 If $\langle N, w\rangle \in \mathrm{A}_{\mathrm{NFA}}$, then M accepts $\langle R, w\rangle$
     - 3.2 Otherwise, M rejects $\langle R, w\rangle$

Decidable DFA Properties

Recap

- We can write a program to test if an arbitrary DFA accepts an arbitrary string
- We can do the same with NFAs, regexes, and anything equivalent

The language $\mathrm{E}_{\mathrm{DFA}}$

Consider the following language

$$\mathrm{E}_{\mathrm{DFA}} = \{\langle D\rangle | D \text{ is a DFA}, L(D) = \emptyset\}$$

What should we do to show that L is decidable?

A) Check if a DFA recognizes the empty set

B) Design a TM that can check if a DFA recognizes the empty set

C) Design a DFA that recognizes the empty set

D) Design a TM that can design a DFA to recognize the empty set

The language $\mathrm{E}_{\mathrm{DFA}}$

Consider the following language

$$\mathrm{E}_{\mathrm{DFA}} = \{\langle D \rangle | D \text{ is a DFA}, L(D) = \emptyset\}$$

What should we do to show that L is decidable?

A) Check if a DFA recognizes the empty set

C) Design a DFA that recognizes the empty set

B) Design a TM that can check if a DFA recognizes the empty set ✓

D) Design a TM that can design a DFA to recognize the empty set

Decidability of $\mathrm{E}_{\mathrm{DFA}}$

Let's prove that the following language is decidable

$$\mathrm{E}_{\mathrm{DFA}} = \{\langle D \rangle | D \text{ is a DFA}, L(D) = \emptyset\}$$

- We'll design a machine M that decides L
  1. M takes $\langle D \rangle$ as input
  2. Go through every string $w_1, w_2, \cdots \in \Sigma^*$
  3. If D ever accepts any w_i, then M rejects $\langle D \rangle$
  4. Otherwise M accepts $\langle M \rangle$

Decidability of $\mathrm{E}_{\mathrm{DFA}}$

Let's prove that the following language is decidable

$$\mathrm{E}_{\mathrm{DFA}} = \{\langle D \rangle | D \text{ is a DFA}, L(D) = \emptyset\}$$

- We'll design a machine M that decides L
  1. M takes $\langle D \rangle$ as input
  2. Go through every string $w_1, w_2, \cdots \in \Sigma^*$
  3. If D ever accepts any w_i, then M rejects $\langle D \rangle$
  4. Otherwise M accepts $\langle M \rangle$

Will this work?

Decidability of $\mathrm{E}_{\mathrm{DFA}}$

Let's prove that the following language is decidable

$$\mathrm{E}_{\mathrm{DFA}} = \{\langle D \rangle | D \text{ is a DFA}, L(D) = \emptyset\}$$

- We'll design a machine M that decides L
  1. M takes $\langle D \rangle$ as input
  2. Go through every string $w_1, w_2, \cdots \in \Sigma^*$
  3. If D ever accepts any w_i, then M rejects $\langle D \rangle$
  4. Otherwise M accepts $\langle M \rangle$

Will this work?

No! This will run forever if $\langle D \rangle \in \mathrm{E}_{\mathrm{DFA}}$!

Decidability of $\mathrm{E}_{\mathrm{DFA}}$

Let's prove that the following language is decidable

$$\mathrm{E}_{\mathrm{DFA}} = \{\langle D \rangle | D \text{ is a DFA}, L(D) = \emptyset\}$$

- We'll design a machine M that decides L
  1. M takes $\langle D \rangle$ as input
  2. Construct the state graph for D
  3. Check if there exists a path from the start state to each accept state
     - 3.1 If there is no path to *any* accept state, M accepts $\langle D \rangle$.
     - 3.2 Otherwise, M rejects $\langle D \rangle$

Decidable DFA Properties

Consider the following language:

$$L = \{\langle D \rangle | D \text{ is a DFA}, L(D) \neq \emptyset\}$$

What should we do to show that L is decidable?

A) Check if a DFA accepts at least one string

B) Design a DFA that accepts at least one string

C) Design a TM that can design a DFA to accept at least one string

D) Design a TM that can check if a given DFA accepts at least one string

Decidable DFA Properties

Consider the following language:

$$L = \{\langle D \rangle | D \text{ is a DFA}, L(D) \neq \emptyset\}$$

What should we do to show that L is decidable?

A) Check if a DFA accepts at least one string

B) Design a DFA that accepts at least one string

C) Design a TM that can design a DFA to accept at least one string

D) Design a TM that can check if a given DFA accepts at least one string ✓

Decidable DFA Properties

Let's prove that the following language is decidable

$$L = \{\langle D\rangle | D \text{ is a DFA}, L(D) \neq \emptyset\}$$

- **Approach 1**: Design a machine M that decides L
  1. M takes $\langle D\rangle$ as input
  2. Construct the state graph for D
  3. Check if there is a path from the start state to *any* accept state
     3.1 If there is a path to an accept state, M acceepts $\langle D\rangle$
     3.2 Otherwise M rejects $\langle D\rangle$

Decidable DFA Properties

Let's prove that the following language is decidable

$$L = \{\langle D \rangle | D \text{ is a DFA}, L(D) \neq \emptyset\}$$

- **Approach 2**: Appeal to TM closure properties
- We showed that $\mathrm{E}_{\mathrm{DFA}} = \{\langle D \rangle | L(D) = \emptyset\}$ is decidable
- Note that $L = (\mathrm{E}_{\mathrm{DFA}})^c$
- Decidable languages are closed under complement
- So L is decidable

Decidable DFA Properties

Recap

- We can design a TM to test if an arbitrary DFA accepts anything at all
- We can design a TM to test if an arbitrary DFA rejects everything

Decidable DFA Properties

Consider the following language:

$$L = \{\langle D \rangle | x111y \in L(D) \text{ for some } x, y \in \Sigma^*\}$$

What should we do to show that L is decidable?

A) Design a TM to check if a given DFA accepts the string 111

C) Design a TM to check if a given DFA accepts any string containing 111

B) Design a TM to check if a given DFA accepts the string $x111y$

D) Design a TM to check if a given DFA only accepts strings containing 111

Decidable DFA Properties

Consider the following language:

$$L = \{\langle D \rangle | x111y \in L(D) \text{ for some } x, y \in \Sigma^*\}$$

What should we do to show that L is decidable?

A) Design a TM to check if a given DFA accepts the string 111

C) Design a TM to check if a given DFA accepts any string containing 111 ✓

B) Design a TM to check if a given DFA accepts the string $x111y$

D) Design a TM to check if a given DFA only accepts strings containing 111

Decidable DFA Properties

Let's prove that the following language is decidable

$$L = \{\langle D \rangle | x111y \in L(D) \text{ for some } x, y \in \Sigma^*\}$$

- Hint 1: $\Sigma^*111\Sigma^*$ can be described by a DFA
- Hint 2: Regular languages are closed under intersection
- Hint 3: We can check if a DFA is non-empty

Decidable DFA Properties

Let's prove that the following language is decidable

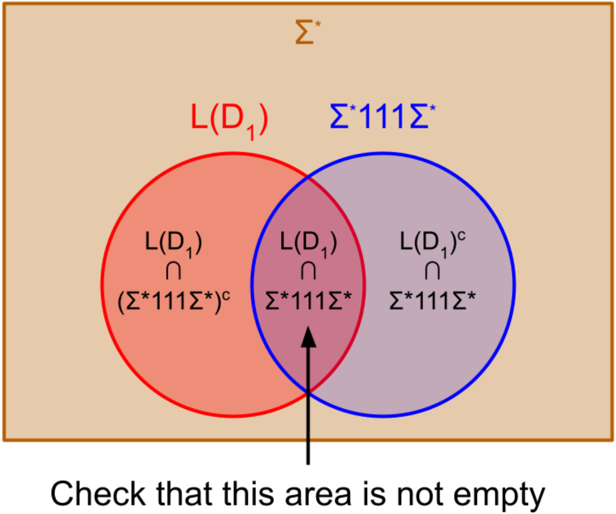
$$L = \{\langle D \rangle | x111y \in L(D) \text{ for some } x, y \in \Sigma^*\}$$

Proof Idea: Check that the set of $x111y$ strings that are also accepted by D is not empty

- Suppose D accepts $x111y$ for some $x, y \in \Sigma^*$
- Then $L(D) \cap (\Sigma^* 111 \Sigma^*) \neq \emptyset$
  - $x111y$ is accepted by both D and by the regex $\Sigma^* 111 \Sigma^*$

Decidable DFA Properties

$L = \{<D_1> \mid D_1 \text{ is a DFA, } x111y \in L(D_1) \text{ for some } x, y\}$



Decidable DFA Properties

Let's prove that the following language is decidable

$$L = \{\langle D \rangle | x111y \in L(D) \text{ for some } x, y \in \Sigma^*\}$$

- Let M_E decide $\mathrm{E}_{\mathrm{DFA}}$
- Design a machine M that decides L
  1. M takes $\langle D \rangle$ as input
  2. Create a DFA D_2 that recognizes $\Sigma^*111\Sigma^*$
  3. Create a DFA D_3 that recognizes $L(D) \cap L(D_2)$
  4. Check if $\langle D_3 \rangle \in \mathrm{E}_{\mathrm{DFA}}$
     - 4.1 If M_E accepts $\langle D_3 \rangle$, then M rejects $\langle D \rangle$
     - 4.2 Otherwise, M accepts $\langle D \rangle$

This is an example of a machine/program that creates *another* machine/program *at runtime*

Decidable DFA Properties

Prove that the following language is decidable

$$L = \{\langle D\rangle | x111y \in L(D) \text{ for } \underline{\text{all}}\ x, y \in \Sigma^*\}$$

- Hint 1: $\Sigma^*111\Sigma^*$ can be recognized by a DFA
- Hint 2: Regular languages are closed under intersection *and complement*
- Hint 3: we can check if a DFA is empty ...

Decidable DFA Properties

Prove that the following language is decidable

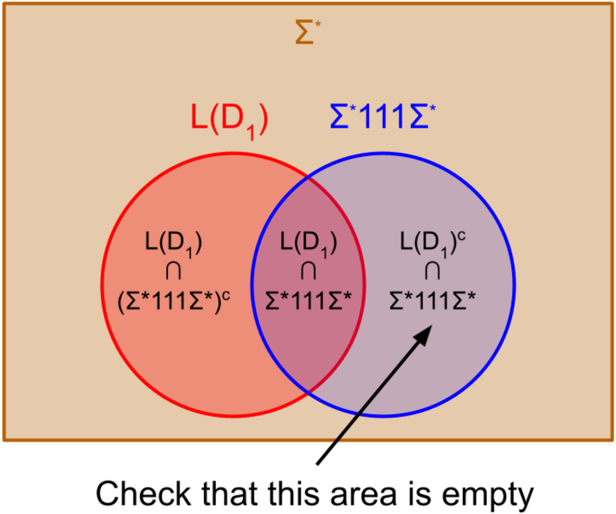
$$L = \{\langle D \rangle | x111y \in L(D) \text{ for } \underline{\text{all}}\ x, y \in \Sigma^*\}$$

Proof Idea: We check that the set of $x111y$ string that are rejected by D is empty

- We check that $\Sigma^*111\Sigma^*$ has nothing in common with the complement of $L(D)$
- Suppose $\Sigma^*111\Sigma^* \subseteq L(D)$
- Then $L(D)^c \cap (\Sigma^*111\Sigma^*) = \emptyset$

Decidable DFA Properties

L = {<D_1> | D_1 is a DFA, x111y ∈ L(D_1) for all x, y}



Decidable DFA Properties

Prove that the following language is decidable

$$L = \{\langle D\rangle | x111y \in L(D) \text{ for } \underline{\text{all}}\ x, y \in \Sigma^*\}$$

- Let M_E decide $\mathrm{E}_{\mathrm{DFA}}$
- Design a machine M that decides L
  1. M takes $\langle D\rangle$ as input
  2. Create a DFA D_2 that recognizes $(\Sigma^* 111 \Sigma^*)$
  3. Create a DFA D_3 that recognizes $L(D_1)^c \cap L(D_2)$
  4. Use M_E to check if $\langle D_3\rangle \in \mathrm{E}_{\mathrm{DFA}}$
     - 4.1 If M_E accepts $\langle D_3\rangle$, then M accepts $\langle D\rangle$
     - 4.2 Otherwise, M rejects $\langle D\rangle$

Decidability of $\mathrm{EQ}_{\mathrm{DFA}}$

Theorem: The following language is decidable

$$\mathrm{EQ}_{\mathrm{DFA}} = \{\langle D_1, D_2\rangle | L(D_1) = L(D_2)\}$$

- Hint: regular languages are closed under complement, and union
- Hint: try to figure out if there is a string that is accepted by one machine but not the other

Decidability of $\mathrm{EQ}_{\mathrm{DFA}}$

Theorem: The following language is decidable

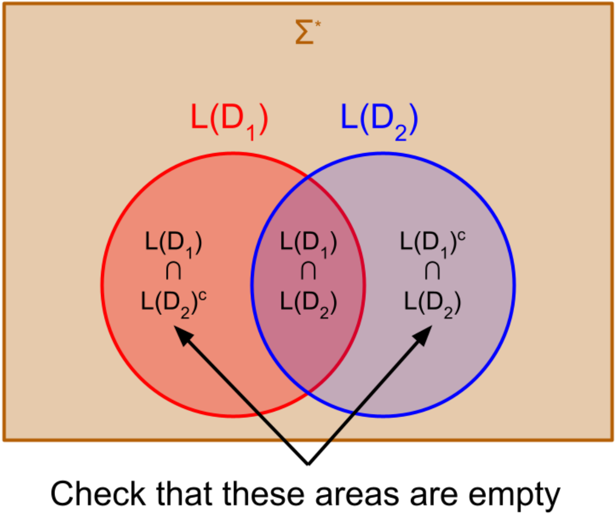
$$\mathrm{EQ}_{\mathrm{DFA}} = \{\langle D_1, D_2\rangle | L(D_1) = L(D_2)\}$$

Proof Idea: check that the set of strings that are accepted by one DFA and rejected by the other is empty

- Let $L_1 = L(D_1)$, and let $L_2 = L(D_2)$
- Suppose $L_1 = L_2$
- Then $L_1 \cap L_2^c = \emptyset$
    - No string is in L_1 but not L_2
- $L_1^c \cap L_2 = \emptyset$
    - No string is in L_2 but not L_1
- $(L_1 \cap L_2^c) \cup (L_1^c \cap L_2) = \emptyset$
    - No string is in one language but not the other

Decidability of $\mathrm{EQ}_{\mathrm{DFA}}$

$L = \{\langle D_1, D_2 \rangle \mid D_1, D_2 \text{ are DFAs}, L(D_1) = L(D_2)\}$



Decidability of $\mathrm{EQ}_{\mathrm{DFA}}$

Theorem: The following language is decidable

$$\mathrm{EQ}_{\mathrm{DFA}} = \{\langle D_1, D_2 \rangle | L(D_1) = L(D_2)\}$$

- Let M_E decide $\mathrm{E}_{\mathrm{DFA}}$
- Design a machine M that decides L
  1. M takes $\langle D_1, D_2 \rangle$ as input
  2. Create a DFA D_3 that recognizes $L(D_1) \cap L(D_2)^c$
  3. Create a DFA D_4 that recognizes $L(D_1)^c \cap L(D_2)$
  4. Create a DFA D_5 that recognizes $L(D_3) \cup L(D_4)$
  5. Use M_E to check if $\langle D_5 \rangle \in \mathrm{E}_{\mathrm{DFA}}$
     - 5.1 If M_E accepts $\langle D_5 \rangle$, then M accepts $\langle D_1, D_2 \rangle$
     - 5.2 Otherwise, M rejects $\langle D_1, D_2 \rangle$

Decidable DFA Properties

Recap

- We can test if an arbitrary DFA intersects with or contains a certain collections of strings
- We can test if two DFAs are equivalent

Decidable Turing Machine Properties

Consider the following language

$L = \{\langle M, w\rangle | M$ is a TM that halts on w in 100 steps$\}$

What is input to the decision problem associated with this language?

A) The description of a TM

B) The description of a TM, as well as an input string

C) The description of a TM that runs for 100 steps

D) The description of a TM, as well as a string that it finishes processing in 100 steps

Decidable Turing Machine Properties

Consider the following language

$L = \{\langle M, w\rangle | M$ is a TM that halts on w in 100 steps$\}$

What is input to the decision problem associated with this language?

A) The description of a TM

B) The description of a TM, as well as an input string ✓

C) The description of a TM that runs for 100 steps

D) The description of a TM, as well as a string that it finishes processing in 100 steps

Decidable Turing Machine Properties

Consider the following language

$L = \{\langle M, w\rangle | M$ is a TM that halts on w in 100 steps$\}$

What should we do to show that L is decidable?

A) Design a machine that can check if a given TM halts on a given string in 100 steps

B) Design a machine that can create a machine that runs for 100 steps

C) Design a machine that can accept w within 100 steps

D) Design a machine that can check if a given TM always runs for 100 steps

Decidable Turing Machine Properties

Consider the following language

$L = \{\langle M, w \rangle | M \text{ is a TM that halts on w in 100 steps}\}$

What should we do to show that L is decidable?

A) Design a machine that can check if a given TM halts on a given string in 100 steps ✓

B) Design a machine that can create a machine that runs for 100 steps

C) Design a machine that can accept w within 100 steps

D) Design a machine that can check if a given TM always runs for 100 steps

Decidable Turing Machine Properties

Let's prove that the following language is decidable

$L = \{\langle M, w \rangle | M$ is a TM that halts on w in 100 steps$\}$

- Design a machine M_L that decides L
  1. M_L takes $\langle M, w \rangle$ as input
  2. Run M on w
  3. If M accepts or rejects within the first 100 steps, then M_L accepts $\langle M, w \rangle$
  4. If M runs for 100 steps without accepting or rejecting, M_L rejects $\langle M, w \rangle$

Decidability of HALT

Prove that the following language is decidable

$$\mathrm{HALT} = \{\langle M, w \rangle | M \text{ halts on } w\}$$

Check if an arbitrary program M finishes processing an arbitrary input w in *any* number of steps

- Design a machine M_H that decides HALT
  1. M_H takes $\langle M, w \rangle$ as input
  2. Run M on w
  3. If M ever accepts or rejects w, M_L accepts $\langle M, w \rangle$
  4. If M loops on w, M_H...loops on $\langle M, w \rangle$

Decidability of HALT

Undecidablity of HALT

It turns out this is NOT a decidable language

$$\mathrm{HALT} = \{\langle M, w\rangle | M \text{ halts on } w\}$$

- No matter how smart you are, you will NEVER be able to write a program to decide this language
- The java compiler cannot be trained to detect infinite loops before you run the code
- How do we prove such a statement?