Reducibility

Purushothama B R

Department of Computer Science and Engineering
National Institute of Technology Goa ,INDIA

November 21, 2017

I want to Begin With... Michael Jordan's Quote

I've missed more than 9000 shots in my career. I've lost almost 300 games. 26 times, I've been trusted to take the game winning shot and missed. I've failed over and over and over again in my life. And that is why I succeed.

Emptiness Problem

Emptiness Problem

- Let,

$$E_{TM} = \{< M > | M \text{ is a TM and } L(M) = \emptyset\}.$$

Emptiness Problem

- Let,

$$E_{TM} = \{< M > | M \text{ is a TM and } L(M) = \emptyset\}.$$

Theorem 5.2

E_{TM} is undecidable.

Proof Idea

Proof Idea

- This proof is by contradiction.

Proof Idea

- This proof is by contradiction.
- We assume that E_{TM} is decidable

Proof Idea

- This proof is by contradiction.
- We assume that E_{TM} is decidable
- And use that assumption to show that A_{TM} is decidable.

Proof Idea

- This proof is by contradiction.
- We assume that E_{TM} is decidable
- And use that assumption to show that A_{TM} is decidable.
- Contradicting the assumption that A_{TM} is undecidable (**We know it!!!**) .

Proof Idea

- This proof is by contradiction.
- We assume that E_{TM} is decidable
- And use that assumption to show that A_{TM} is decidable.
- Contradicting the assumption that A_{TM} is undecidable (**We know it!!!**) .
- The key idea is to show that A_{TM} is reducible to E_{TM} .

Proof

Proof

- Lets assume that we have a **TM R** that decides E_{TM}

Proof

- Lets assume that we have a **TM R** that decides E_{TM}
- Then, we use R to construct **S**

Proof

- Lets assume that we have a **TM R** that decides E_{TM}
- Then, we use R to construct **S** (TM that decides A_{TM}) .

Proof

- Lets assume that we have a **TM R** that decides E_{TM}
- Then, we use R to construct **S** (TM that decides A_{TM}) .
- How will S work when it receives input $< M, w >$?

Proof

- Lets assume that we have a **TM R** that decides E_{TM}
- Then, we use R to construct **S** (TM that decides A_{TM}) .
- How will S work when it receives input $< M, w >$?
- One Idea

Proof

- Lets assume that we have a **TM R** that decides E_{TM}
- Then, we use R to construct **S** (TM that decides A_{TM}) .
- How will S work when it receives input $< M, w >$?
- One Idea is to

Proof

- Lets assume that we have a **TM R** that decides E_{TM}
- Then, we use R to construct **S** (TM that decides A_{TM}) .
- How will S work when it receives input $< M, w >$?
- One Idea is to
  - Run R on input $< M >$ and see whether it accepts.

Proof

- Lets assume that we have a **TM R** that decides E_{TM}
- Then, we use R to construct **S** (TM that decides A_{TM}) .
- How will S work when it receives input $< M, w >$?
- One Idea is to
  - Run R on input $< M >$ and see whether it accepts.
  - If it does, we know that $L(M)$ is empty.

Proof

- Lets assume that we have a **TM R** that decides E_{TM}
- Then, we use R to construct **S** (TM that decides A_{TM}) .
- How will S work when it receives input $< M, w >$?
- One Idea is to
  - Run R on input $< M >$ and see whether it accepts.
  - If it does, we know that $L(M)$ is empty.
  - And therefore that M does not accept w.

Proof

- Lets assume that we have a **TM R** that decides E_{TM}
- Then, we use R to construct **S** (TM that decides A_{TM}) .
- How will S work when it receives input $< M, w >$?
- One Idea is to
  - Run R on input $< M >$ and see whether it accepts.
  - If it does, we know that $L(M)$ is empty.
  - And therefore that M does not accept w.
  - But if R rejects $< M >$
    - All we know is that $L(M)$ is not empty

Proof

- Lets assume that we have a **TM R** that decides E_{TM}
- Then, we use R to construct **S** (TM that decides A_{TM}) .
- How will S work when it receives input $< M, w >$?
- One Idea is to
  - Run R on input $< M >$ and see whether it accepts.
  - If it does, we know that $L(M)$ is empty.
  - And therefore that M does not accept w.
  - But if R rejects $< M >$
    - All we know is that $L(M)$ is not empty and therefore that M accepts some string.
    - But we still does not know

Proof

- Lets assume that we have a **TM R** that decides E_{TM}
- Then, we use R to construct **S** (TM that decides A_{TM}) .
- How will S work when it receives input $< M, w >$?
- One Idea is to
  - Run R on input $< M >$ and see whether it accepts.
  - If it does, we know that $L(M)$ is empty.
  - And therefore that M does not accept w.
  - But if R rejects $< M >$
    - All we know is that $L(M)$ is not empty and therefore that M accepts some string.
    - But we still does not know whether M accepts the particular string w.
- So, we need a different Idea.

Proof

Proof

- Instead of running R on $< M >$,

Proof

- Instead of running R on $< M >$, we run R on a modification of $< M >$.

Proof

- Instead of running R on $< M >$, we run R on a modification of $< M >$.
- We modify $< M >$ to guarantee that M rejects all strings except w,

Proof

- Instead of running R on $< M >$, we run R on a modification of $< M >$.
- We modify $< M >$ to guarantee that M rejects all strings except w,
- But on input w, it works as usual.

Proof

- Instead of running R on $< M >$, we run R on a modification of $< M >$.
- We modify $< M >$ to guarantee that M rejects all strings except w,
- But on input w, it works as usual.
- Then, we use R to determine whether the modified machine recognizes the empty language.

Proof

- Instead of running R on $< M >$, we run R on a modification of $< M >$.
- We modify $< M >$ to guarantee that M rejects all strings except w,
- But on input w, it works as usual.
- Then, we use R to determine whether the modified machine recognizes the empty language.
- The only string the machine can accept now is w.

Proof

- Instead of running R on $< M >$, we run R on a modification of $< M >$.
- We modify $< M >$ to guarantee that M rejects all strings except w,
- But on input w, it works as usual.
- Then, we use R to determine whether the modified machine recognizes the empty language.
- The only string the machine can accept now is w.
- So, its language will be non-empty iff it accepts w.

Proof

- Instead of running R on $< M >$, we run R on a modification of $< M >$.
- We modify $< M >$ to guarantee that M rejects all strings except w,
- But on input w, it works as usual.
- Then, we use R to determine whether the modified machine recognizes the empty language.
- The only string the machine can accept now is w.
- So, its language will be non-empty iff it accepts w.
- If R accepts when it is fed a description of the modified machine
    - We know that the modified machine does not accept anything,
    - And, M does not accept w.

Lets Complete it!!!

- Let M_1 be the modified machine.

M_1 = "On input x:

1. If $x \neq w$, *reject*.
2. If $x = w$, run M on input w and *accept* if M does."

Putting all together!!!

Putting all together!!!

- We assume that TM R decides E_{TM}

Putting all together!!!

- We assume that $TM\ \ R$ decides E_{TM} and construct $TM\ S$ that decides A_{TM} as below.

Putting all together!!!

- We assume that $TM \ R$ decides E_{TM} and construct $TM \ S$ that decides A_{TM} as below.
- We construct TM S to decide A_{TM}. S operates as below.

S = "On input $\langle M, w\rangle$, an encoding of a TM M and a string w:

1. Use the description of M and w to construct the TM M_1 just described.
2. Run R on input $\langle M_1\rangle$.
3. If R accepts, *reject*; if R rejects, *accept*."

Putting all together!!!

- We assume that $TM \ \ R$ decides E_{TM} and construct $TM\ S$ that decides A_{TM} as below.
- We construct TM S to decide A_{TM}. S operates as below.

S = "On input $\langle M, w\rangle$, an encoding of a TM M and a string w:

1. Use the description of M and w to construct the TM M_1 just described.
2. Run R on input $\langle M_1\rangle$.
3. If R accepts, *reject*; if R rejects, *accept*."

- Note that S must actually be able to compute description of M_1 from a description of M and w.

Putting all together!!!

- We assume that $TM\ \ R$ decides E_{TM} and construct $TM\ S$ that decides A_{TM} as below.
- We construct TM S to decide A_{TM}. S operates as below.

S = "On input $\langle M, w\rangle$, an encoding of a TM M and a string w:

1. Use the description of M and w to construct the TM M_1 just described.
2. Run R on input $\langle M_1\rangle$.
3. If R accepts, *reject*; if R rejects, *accept*."

- Note that S must actually be able to compute description of M_1 from a description of M and w.
- It is able to do so, because it needs to only add extra stated to M that perform the $x = w$ test.

Putting all together!!!

- We assume that $TM \; R$ decides E_{TM} and construct $TM \; S$ that decides A_{TM} as below.
- We construct TM S to decide A_{TM}. S operates as below.

S = "On input $\langle M, w \rangle$, an encoding of a TM M and a string w:

1. Use the description of M and w to construct the TM M_1 just described.
2. Run R on input $\langle M_1 \rangle$.
3. If R accepts, *reject*; if R rejects, *accept*."

- Note that S must actually be able to compute description of M_1 from a description of M and w.
- It is able to do so, because it needs to only add extra stated to M that perform the $x = w$ test.
- If R were a decider for E_{TM}, S would be decider for A_{TM}.

Another interesting problem

Another interesting problem

- Let,

$$REGULAR_{TM} = \{< M > | M \text{ is a TM and } L(M) \text{ is a regular language}\}.$$

Another interesting problem

- Let,

  $REGULAR_{TM} = \{< M > | M \text{ is a TM and } L(M) \text{ is a regular language}\}$.

Theorem 5.2

$REGULAR_{TM}$ is undecidable.

Proof

Proof

- This proof is also a reduction from A_{TM}.

Proof

- This proof is also a reduction from A_{TM}.
- Assume that $REGULAR_{TM}$ is decidable by a TM R.

Proof

- This proof is also a reduction from A_{TM}.
- Assume that $REGULAR_{TM}$ is decidable by a TM R.
- Use this assumption to construct a TM S that decides A_{TM}.

Idea

Idea

- Let S take its input $< M, w >$.

Idea

- Let S take its input $< M, w >$.
- Modify M so that the resulting TM recognizes a regular language

Idea

- Let S take its input $< M, w >$.
- Modify M so that the resulting TM recognizes a regular language if and only if M accepts w.

Idea

- Let S take its input $< M, w >$.
- Modify M so that the resulting TM recognizes a regular language if and only if M accepts w.
- Call the modified machine M_2.

Idea

- Let S take its input $< M, w >$.
- Modify M so that the resulting TM recognizes a regular language if and only if M accepts w.
- Call the modified machine M_2.
- Design M_2 to recognize the non-regular language $\{0^n1^n | n \geq 0\}$

Idea

- Let S take its input $< M, w >$.
- Modify M so that the resulting TM recognizes a regular language if and only if M accepts w.
- Call the modified machine M_2.
- Design M_2 to recognize the non-regular language $\{0^n1^n | n \geq 0\}$ if M does not accept w.

Idea

- Let S take its input $< M, w >$.
- Modify M so that the resulting TM recognizes a regular language if and only if M accepts w.
- Call the modified machine M_2.
- Design M_2 to recognize the non-regular language $\{0^n1^n | n \geq 0\}$ if M does not accept w.
- And, to recognize the regular language Σ^* if M accepts w.

Idea

- Let S take its input $< M, w >$.
- Modify M so that the resulting TM recognizes a regular language if and only if M accepts w.
- Call the modified machine M_2.
- Design M_2 to recognize the non-regular language $\{0^n1^n | n \geq 0\}$ if M does not accept w.
- And, to recognize the regular language Σ^* if M accepts w.
- We must specify how S can construct such an M_2 from M and w.

Idea

- Let S take its input $< M, w >$.
- Modify M so that the resulting TM recognizes a regular language if and only if M accepts w.
- Call the modified machine M_2.
- Design M_2 to recognize the non-regular language $\{0^n1^n | n \geq 0\}$ if M does not accept w.
- And, to recognize the regular language Σ^* if M accepts w.
- We must specify how S can construct such an M_2 from M and w.
- Here, M_2 works by automatically accepting all strings in $\{0^n1^n | n \geq 0\}$

Idea

- Let S take its input $< M, w >$.
- Modify M so that the resulting TM recognizes a regular language if and only if M accepts w.
- Call the modified machine M_2.
- Design M_2 to recognize the non-regular language $\{0^n 1^n | n \geq 0\}$ if M does not accept w.
- And, to recognize the regular language Σ^* if M accepts w.
- We must specify how S can construct such an M_2 from M and w.
- Here, M_2 works by automatically accepting all strings in $\{0^n 1^n | n \geq 0\}$
- In addition if M accepts w, M_2 accepts all other strings.

- We assume that $TM \;\; R$ decides $REGULAR_{TM}$

- We assume that $TM\ \ R$ decides $REGULAR_{TM}$ and construct $TM\ S$ that decides A_{TM} as below.

$S =$ "On input $\langle M, w\rangle$, where M is a TM and w is a string:

1. Construct the following TM M_2.

$M_2 =$ "On input x:

1. If x has the form 0^n1^n, *accept*.

2. If x does not have this form, run M on input w and *accept* if M accepts w."

2. Run R on input $\langle M_2\rangle$.

3. If R accepts, *accept*; if R rejects, *reject*."

THANK YOU