Today’s lecture

- Review of diagonalization
- Examples of undecidable languages
- Examples of decidable languages
- Reading: Finish Sipser Chapter 3 and 4
A language not in RE

- We want a language $L$ that is
  - different from $L(M_1)$ at $<M_1>$
  - different from $L(M_2)$ at $<M_2>$
  - different from $L(M_3)$ at $<M_3>$
  - ....
  - different from $L(M_k)$ at $<M_k>$
  - ....

- $\text{Diag} = \{<M> \mid M \text{ is a TM s.t. } <M> \text{ is not in } L(M)\}$
Diag is in coRE

• Diag = \{ <M> \mid M \text{ is a TM s.t. } <M> \text{ is not in } L(M) \} 

• Here is a recognizer for Diag

   \[ M_{\text{diag}}(w) = \]

   1) Check if \( w = <M> \) for some TM \( M \). If not, accept.
   2) Parse \( w \) as \( <M> \) for some TM \( M \)
   3) Run \( M \) on input \( w \)
   4) If \( M(w) \) accepts, then accept, else reject.
Diag is in coRE

- Diag = \{ <M> \mid M \text{ is a TM s.t. } <M> \notin L(M) \}
- Here is a recognizer for Diag:

  \[ M_{\text{diag}}(w) = \]
  1) Check if w = <M> for some TM M. If not, accept.
  2) Parse w as <M> for some TM M
  3) Run M on input w
  4) If M(w) accepts, then accept, else reject.

Is \( M_{\text{diag}} \) a decider?

A) Yes, because \( L(M_{\text{diag}}) = \text{Diag} \)
B) No, because it can loop at step 1
C) No, because it can loop at step 2
D) No, because it can loop at step 3
E) I don’t know
Diag is in coRE

• Diag = \{ \langle M \rangle \mid M \text{ is a TM s.t. } \langle M \rangle \notin L(M) \}

• Here is a recognizer for Diag:

  \[ M_{\text{diag}}(w) = \]
  1) Check if \( w = \langle M \rangle \) for some TM \( M \). If not, accept.
  2) Parse \( w \) as \( \langle M \rangle \) for some TM \( M \)
  3) Run \( M \) on input \( w \)
  4) If \( M(w) \) accepts, then accept, else reject.

What can you say about Diag

A) Diag is decidable
B) Diag is in RE, but not coRE
C) Diag is in coRE, but not RE
D) Diag is neither in RE nor coRE
E) I don’t know
Summary

- Diag is undecidable
- **Diag** is in coRE, but not RE
- Can we find more interesting examples of undecidable languages?
  - What about $\text{HALT}_{TM} = \{<M,w> \mid M(w) \text{ terminates}\}$?
  - What about $\text{A}_{TM} = \{<M,w> \mid M(w) \text{ accepts}\}$?
Warm up

- Diag = \{ <M> | \text{M is a TM s.t. } <M> \notin L(M) \} 
- DD = \{ <M,<M>> | \text{M is a TM s.t. } <M> \notin L(M) \} 

Question: What can you say about DD?

A) DD = Diag
B) DD is undecidable
C) DD is in RE, but not in coRE
D) DD is neither in RE nor in coRE
Warm up

- Diag=\{ <M> | M is a TM s.t. <M> \notin L(M) \} 
- DD= \{ <M, <M>>| M is a TM s.t. <M> \notin L(M) \} 
- Claim: DD is undecidable. *But how can we prove it?*

**Proof: We give a proof for contradiction!**
- Assume for contradiction DD is decidable.
- Let P be a decider such that L(P)=DD.
- Use P to build a decider P’ for Diag.
- Contradiction, because Diag is undecidable!
Proof details

- **Diag**: \( \{ <M> \mid M \text{ is a TM s.t. } <M> \notin L(M) \} \)
- **DD**: \( \{ <M,<M>> \mid M \text{ is a TM s.t. } <M> \notin L(M) \} \)
- Assume for contradiction P is a decider for DD
- Define the program \( P'(w) \):
  1) Parse \( w \) as \( <M> \). If parse fails, then reject.
  2) Build the string \( w'=<M,<M>> \)
  3) Run \( P(w') \). If \( P \) accepts, then accept, else reject.
Proof details

- \text{Diag}=\{<M> \mid M \text{ is a TM s.t. } M \not\in L(M)\}
- \text{DD}= \{<M,<M>> \mid M \text{ is a TM s.t. } M \not\in L(M)\}

- Assume for contradiction \text{P} is a decider

- Define the program \text{P}'(w):

  1) Parse \text{w} as <M>. If parse fails, then reject.

  2) Build the string \text{w'}=<M,<M>>

  3) Run \text{P}(w'). If \text{P} accepts, then accept, else reject.

Is \text{P}' a decider?

A) No, because Diag is undecidable
B) No, because DD is undecidable
C) No, because it can loop in step 3
D) Yes, because it always terminate
Proof details

- Diag = \{ <M> | M is a TM s.t. <M> \not\in L(M) \}
- DD = \{ <M,<M>> | M is a TM \}
- Assume for contradiction P is a decider for DD
- Define the program P'(w):
  1) Parse w as <M>. If parse fails, then reject.
  2) Build the string w' = <M,<M>>
  3) Run P(w'). If P accepts, then accept, else reject.

What is the language of P'?  
A) L(P') = {}  
B) L(P') = Diag  
C) L(P') = DD  
D) L(P') = Diag U DD  
E) I am completely lost, help!
Proof details

- **Diag**=\{<M>| M is a TM s.t. <M> ∉ L(M)\}
- **DD**= \{<M,<M>>| M is a TM s.t. <M> ∉ L(M)\}
- Assume for contradiction P decides DD
- Define the program P’(w):
  1) Parse w as <M>. If parse fails, then reject.
  2) Build the string w’=<M,<M>>
  3) Run P(w’). If P accepts, then accept, else reject.
- So, P’ is a decider and L(P’)=Diag. Contradiction!

What did we prove?

A) Diag is decidable
B) DD is decidable
C) Diag is undecidable
D) DD is undecidable
E) DD = Diag
$A_{TM}$ is undecidable

- Assume for contradiction $A_{TM}$ is decidable
- $MM = \{<M,<M>> \mid M \text{ is a TM}\}$ is decidable
- $MM - A_{TM}$ is decidable because the class of decidable languages is closed under set difference.
- Notice: $MM - A_{TM}$
  
  $= \{<M,<M>> \mid M \text{ is a TM}\} - \{<M,w> \mid w \in L(M)\}$
  
  $= \{<M,<M>> \mid M \text{ is a TM and } <M> \notin L(M)\} = DD$

- So, $DD$ is decidable. Contradiction!
Some example languages

- **Acceptance problem:**
  - \( A_{\text{DFA}} = \{<M,w> | M \text{ is a DFA and } M(w) \text{ accepts} \} \)
  - \( A_{\text{TM}} = \{<M,w> | M \text{ is a TM and } M(w) \text{ accepts} \} \)

- **Emptyness problem:**
  - \( E_{\text{DFA}} = \{<M> | M \text{ is a DFA and } L(M) \text{ is the empty set} \} \)
  - \( E_{\text{TM}} = \{<M> | M \text{ is a TM and } L(M) \text{ is the empty set} \} \)

- **Equivalence problem:**
  - \( EQ_{\text{DFA}} = \{<M,M'> | M \text{ and } M' \text{ are DFAs and } L(M) = L(M') \} \)
Some example languages

• Acceptance problem:
  - $A_{\text{DFA}} = \{ <M,w> \mid M \text{ is a DFA and } M(w) \text{ accepts} \}$ DECIDABLE
  - $A_{\text{TM}} = \{ <M,w> \mid M \text{ is a TM and } M(w) \text{ accepts} \}$ UNDECIDABLE

• Emptyness problem:
  - $E_{\text{DFA}} = \{ <M> \mid M \text{ is a DFA and } L(M) \text{ is the empty set} \}$ DECIDABLE
  - $E_{\text{TM}} = \{ <M> \mid M \text{ is a TM and } L(M) \text{ is the empty set} \}$ UNDECIDABLE

• Equivalence problem:
  - $EQ_{\text{DFA}} = \{ <M,M'> \mid M \text{ and } M' \text{ are DFAs and } L(M) = L(M') \}$ DECIDABLE
E_{DFA} is decidable

- \( P_{EDFA}(w) = \)
  - Parse input \( w \) as \( \langle Q, \Sigma, \delta, s, F \rangle \). If parse fails, then reject.
  - Let \( X = \{s\} \)
  - For all \( q \) in \( X \) and \( a \) in \( \Sigma \)
    - Let \( q' = \delta(q, a) \)
    - If \( q' \) is not in \( X \), then \( X \leftarrow X \cup \{q'\} \) and restart the loop at 3)
  - If \( X \) intersects \( F \), then accept, else reject.
**EQ\textsubscript{DFA} is decidable**

- $P_{\text{EQDFA}}(<M,M'>)$
  1) Check if both $M$ and $M'$ are DFA. If not, then reject.
  2) Use closure properties of regular languages to build a DFA $M''$ for
     \[ L(M'') = (L(M) - L(M')) \cup (L(M') - L(M)) \]
     - Run $P_{\text{EDFA}}(<M''>)$. If $P_{\text{EDFA}}$ accepts, then accept, else reject.

- Notice $L(M)=L(M')$ if and only if $L(M'')$ is empty.
For next Time

- Try to prove that $\text{HALT}^\text{TM}$ is undecidable
- Reading: Sipser *Chapters 3 and 4*
- Midterm review: Tonight 8pm
- Haskell 4 due Nov 21
- Exam 3: Nov 18