Agenda

• **Yesterday**
  - Prove equivalence of deterministic and nondeterministic Turing machines

• **Today**
  - Enumerators
  - Definition of algorithm
Announcements

• **Quiz tomorrow**
  - High-level description of TM
  - Trace through a TM’s operation
    • Tape notation & configuration notation
  - High-level description of equivalences

• **Reminder: tutorial sessions are back**
  - Office hours return to normal
    • Tuesday 3:00 – 4:00
    • Wednesday 3:00 – 4:00
Enumerators

- Instead of reading an input and processing it, enumerators start with an empty tape and print out strings in $\Sigma^*$

\[ M \rightarrow \text{Printer} \]

Printer:

- aaba
- bbcc
- aaaaabb

Output:

1 1 2 1 3 ~ ~ ~
Machine equivalence

Theorem: A language is Turing-recognizable if and only of some enumerator enumerates it.

Proof technique: Construction in each direction
TM accept enumerator language

• TM = “On input w:
  - Run enumerator E. Every time E prints a string, compare it to w.
  - If w appears in the output, accept.”
Enumerator accepts TM language

• Let $s_1, s_2, s_3, \ldots$ be all the strings in $\Sigma^*$

• $E = \text{"Ignore the input.} \quad$
  - For $i = 1, 2, 3, \ldots$
    • Run $M$ for $i$ steps on each input $s_1, s_2, \ldots, s_i$
    • Whenever $M$ accepts a string, print it
What is an algorithm?

• Intuitively, an algorithm is anything that can be simulated by a Turing machine
  - Many algorithms can be simulated by Turing machines
  - Inputs can be represented as strings
    • Graphs
    • Polynomials
    • Automata
    • Etc.
Example algorithm

• Depth-first walk-through of binary tree

• Which nodes do you visit, and in what order, when doing a depth-first search?
  - Visit each leaf node from left to right
  - Recursive algorithm
Depth-first walk-through

• Start at root
• Process left subtree (if one exists)
• Process right subtree (if one exists)
• Process how?
  - Print the node name
  - If there is a left subtree then
    • Process the left subtree
    • Print the node name again
  - If there is a right subtree then
    • Process the right subtree
    • Print the node name again
Example

1

2

4

8

4

2

5

9

5

2

1

3

6

10

6

3

7
Can a Turing machine do this?

• Input must be a string (not a tree)
  - Can we represent a tree with a string?
  - Yes.
String representation of a tree

```
1  2  3  4  5  6  7 8  #  #  9  10 #  # #  ~
```
Can a Turing machine do this?

• Input must be a string (not a tree)
  - Can we represent a tree with a string?
  - Yes

• How do we know which node(s) are children of current node
  - If node is \( k^{th} \) node at depth \( d \), it’s position in the string is \( 2^d+k-1 \) and its children are at position \( 2^{d+1}+2(k-1) \) and \( 2^{d+1}+2k-1 \)
What about the output?

- Need to write out nodes in a particular order
  - Can we do this with a TM?
  - Yes. Add output tape
  - A TM can move left and right on the input tape writing to the output tape whenever appropriate
Decidability

• A language is decidable if some Turing machine decides it
• Not all languages are decidable
  – We will see examples of both decidable and undecidable languages
DFA acceptance problem

- Consider the language
  \[ A_{\text{DFA}} = \{ \langle B, w \rangle \mid B \text{ is a DFA that accepts the string } w \} \]

Theorem: \( A_{\text{DFA}} \) is a decidable language

Proof: Consider the following TM, \( M \)

\( M = \) “On input string \( \langle B, w \rangle \)

1. Simulate \( B \) on input \( w \)
2. If simulation ends in accept state, accept. Otherwise, reject.”
NFA acceptance problem

- Consider the language
  \[ A_{\text{NFA}} = \{ \langle B, w \rangle \mid B \text{ is a NFA that accepts the string } w \} \]

Theorem: \( A_{\text{NFA}} \) is a decidable language

Proof: Consider the following TM, \( N \)

\( N = \) “On input string \( \langle B, w \rangle \)

1. Convert \( B \) to a DFA \( C \)

2. Run TM M from previous slide on \( \langle C, w \rangle \)

3. If M accepts, accept. Otherwise, reject.”
RE acceptance problem

Consider the language

\[ A_{\text{REX}} = \{ \langle R, w \rangle \mid R \text{ is an RE that generates the string } w \} \]

**Theorem:** \( A_{\text{REX}} \) is a decidable language

**Proof:** Consider the following TM, \( P \)

\( P = \) “On input string \( \langle R, w \rangle \)

1. **Convert \( R \) to a DFA \( C \) using algorithm discussed in class and in text**
2. **Run TM \( M \) from earlier slide on \( \langle C, w \rangle \)
3. **If \( M \) accepts, accept. Otherwise, reject.”
Emptiness testing problem

- Consider the language

\[ E_{\text{DFA}} = \{ \langle A \rangle \mid A \text{ is a DFA and } L(A) = \emptyset \} \]

Theorem: \( E_{\text{DFA}} \) is a decidable language

Proof: Consider the following TM, \( T \)

\( T = "\text{On input string } \langle A \rangle \)

1. Mark the start state
2. Repeat until no new states get marked
   - Mark any state that has a transition coming into it from any state already marked
3. If no accept states are marked, accept. Otherwise, reject."