Lecture 7

CSE 431
Intro to Theory of Computation

Last time:
Church-Turing Thesis
- This precisely captures our intuitive notion of algorithm

High-level view of input encoding

\[
\begin{align*}
\langle \text{M} \rangle & \quad \langle \text{M}, \omega \rangle \\
L(\text{M}) & = \emptyset \quad \text{w+L(} \text{M} \text{)}?
\end{align*}
\]

E DFA, ENFA, A DFA, ANFA, A REG, EU DFA

Context-Free Grammar

Given by

- Set of variables
- alphabet (terminals)
- Set of rules of form

\[
A \rightarrow w \quad A \in V \\
\text{w+}(V \cup \Sigma)^* \\
S \in V \quad \text{start symbol}
\]

A \Rightarrow^* w \quad \text{repeatedly apply rules for } A \text{ to set } w
\[ (16) = 3 \omega \in \mathbb{R}^3 \quad \text{A} \Rightarrow \mathbb{A}_c \mathbb{A}_w \]

CSE 111

Examples:
- Set of binary palindromes:
  
  \[
  \begin{align*}
  S & \rightarrow \epsilon \\
  S & \rightarrow 0S0 \\
  S & \rightarrow 1S1 \\
  S & \rightarrow \epsilon 
  \end{align*}
  \]

  Shorthand:
  
  \[ S \Rightarrow \epsilon | 0 | 1 | 0 | 1 | 0 | 1 | 1 | 0 | 1 | 0 | 1 | 0 | 1 \]

- Strings of balanced parentheses:
  
  \[ S \Rightarrow \epsilon | (S) | SS \]

- Strings with equal numbers of 0's and 1's:
  
  \[ S \Rightarrow \epsilon | 0S1 | 1S0 | 1S1 | 1S0 | 1S1 | 0S1 | 1S0 | 1S1 | 0S1 | 1S0 | 1S1 | 0S1 | 1S0 | 1S1 | \]

  Obvious that every string produced has an equal number of 0's and 1's.

  To see that all such strings are produced, see that if string begins and ends in same symbol (say 0),

  Graph starts at 1 and at n/2 it is -1, must cross the axis so \[ S \Rightarrow SS \] will generate.
\[
\text{ACFG} = \{ <G, w> | G \text{ is a CFG and } w \in L(G) \}
\]

**Theorem:** \text{ACFG} is decidable.

**Proof:** "Obvious" thing to try.

(BFS) search through the tree of possible derived strings by applying the rules in all possible ways.

- A parse tree $S^{#}$ if $w$ is found then accept.
- A finite # of possibilities: $0S0S \rightarrow S0S0$.

**Problem:** Can't tell when to reject.
- No bound on # of steps to derive $w$ since rules may repeatedly add and remove characters.

**Solution:** Can convert any CFG to an equivalent CFG that is in a special form for which this algorithm works.

**Chomsky Normal Form**

All rules of form $A \rightarrow BC$, $A \rightarrow a$, $S \rightarrow \varepsilon$.  

$B, C \in V \setminus \{S\}$
There is an algorithm that converts any CFG \( G \) to a new CFG \( G' \) in Chomsky Normal Form with \( L(G') = L(G) \).

**Proof:** Later in the quarter.

**Note:** Each rule

- \( A \rightarrow BC \) increases length by 1
- \( A \rightarrow a \) gets rid of a variable

To produce \( \varepsilon \), only rule \( S \rightarrow \varepsilon \) (if it exists)

To produce \( w \) with \( |w| = n > 1 \)

Need precisely \( 2n-1 \) steps

- \( n-1 \) steps to increase length from 1 to \( n \)
- \( n \) steps to replace variables by terminals

**Algorithm for AFCFG:**

On input \( <G,w> \)

1. Convert \( G \) to equivalent Chomsky Normal Form \( G' \)

2. Run "obvious" BFL alg for \( G' \) and \( w \)
   - accept if \( w \) found
   - stop after depth \( 2n-1 \) and reject
$ECFG = \{ <G> | G \text{ is a CFG and } L(G) = \emptyset \}$

Thus $ECFG$ is decidable.

Example

$S \rightarrow O S O | 1 S I$
\[\text{empty}\]

$S \rightarrow O S A | 1 S I$
\[A \rightarrow 0\]
\[\text{empty}\]

$S \rightarrow O B A | 1 S I$
\[A \rightarrow 0, \ B \rightarrow 1\]
\[\text{not empty}\]

Proof Idea: keep track of all symbols that can produce strings of terminals.

Call a variable $A \in V$ "productive" if $A \Rightarrow^* w$ for some $w \in \Sigma^*$

Let $P$ be the set of productive variables.

Can compute $P$ as follows:

- Put all variable $A$ with rule $A \Rightarrow w$ for $w \in \Sigma^*$ into $P$.
- Repeatedly:
  - Add $A \in V$ to $P$ if there is a rule $A \Rightarrow w$ with $w \in (\Sigma \cup P)^*$.
This algorithm to compute $P$ will stop.

Algorithm for $E_{\text{CFG}}$:

1. On input $\langle G \rangle$
2. Compute $P$.
3. If $S \in P$ reject; if $S \notin P$ accept.

Now to $TM$: 

$A_{TM} = \{ \langle M, \omega \rangle \mid M \text{ is a TM that accepts } \omega \}$

Then $A_{TM}$ is Turing recognizable.

This is a weaker statement than decidable.

If not, can either reject or run forever.

Proof (Turing's idea): 

Algorithm: Universal Turing Machine $U$

$U$: On input $\langle M, \omega \rangle$

Perform a step-by-step simulation of $M$ on input $\omega$. 

\[ \text{TM description/diagram} \]

perform a step-by-step simulation of $M$ on input $\omega$. 

\[ \text{TM description/diagram} \]
Details of $U$:

- Start by putting $<w>$ on work tape and $<q_0>$ on state tape.
- Maintain encoding of current state (and head position) on work tape and current state on state tape.
- Use $\delta$ table on input to figure out how to update worktape and state tape.
- Accept if $M$ does.
- Reject if $M$ does not.

**Theorem:** $A_{TM}$ is not decidable.

**Proof idea:** diagonalization.

Recall Cantor's ideas.

-**Rationals are Countable**

  - Can list in increasing order of all $\sqrt{a/b}$.
Thin (Central) $\mathbb{R}^{(0,1)}$ are not countable

some messy details related
+ representations with reals $0.5 < 0.9$

so instead we review

Thin (Central) The set $\mathcal{P}(\mathbb{N})$ of all subsets of $\mathbb{N}$ is not countable

Proof

Suppose that $\mathcal{P}(\mathbb{N})$ is countable

with a listing

$S_0, S_1, S_2, S_3, \ldots$

Table to represent this

\[
\begin{array}{cccccc}
0 & 1 & 2 & 3 & 4 & 5 & 6 \\
S_0 & 1 & 0 & 0 & 1 & 0 & 0 \\
S_1 & 0 & 1 & 1 & 0 & 1 & 0 & 0 \\
S_2 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\
S_3 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\
\end{array}
\]

flipped diagonal set

\[
\begin{bmatrix}
\text{Define a set } D \in \mathcal{P}(\mathbb{N}) \\
\text{by } i \in D \text{ if } i \notin S_i
\end{bmatrix}
\]

Observe that $D \neq S_j$ for any $j$

since they disagree on whether
they include $j$

i.e., contraction since $\mathcal{P}(\mathbb{N})$ was supposed

\[\boxed{\text{to be complete}}\]