Lecture 8

Last time:
- Pumping lemma for CFLs
- Proving that a language is not CF

Today:
- Equivalence of CFGs and PDAs
- Turing Machines
Read on your own

1. Ambiguous CFGs.
2. Chomsky normal form for CFGs
3. (Skipping Chapter 2.4 in Sipser).
The language of $P$ is the set of strings it accepts. PDAs are nondeterministic.
Exercise

When PDA takes a nondeterministic step, what happens to the stack?

A. First 0 is pushed onto the stack, then 1.
B. First 1 is pushed onto the stack, then 0.
C. Now PDA has access to two stacks and it pushes 0 onto one and 1 onto the other.
D. Two different computational branches are available to the PDA: it pushes exactly one symbol (0 or 1) onto the stack on each branch.
E. None of the above
A language is generated by a CFG

$\iff$

It is recognized by a PDA
Suppose $L$ is generated by a CFG $G = (V, \Sigma, R, S)$. Construct a PDA $P = (Q, \Sigma, \Gamma, \delta, q, F)$ that recognizes $L$.

**Idea:** $P$ will guess steps of a derivation of its input $w$ and use its stack to derive it.
Algorithmic description of PDA

(1) Place the marker symbol $ and the start variable on the stack.

(2) Repeat forever:
   (a) If a variable A is on top of the stack, and (A → s) ∈ R, pop A and push string s on the stack in reverse order.
   (b) If a terminal is on top of the stack, pop it and match it with input.

(3) On (ε,$), accept.
Designing states of PDA

(q_{start}) Push S$ and go to q_{loop}

(q_{loop}) Repeat the following steps forever:
   (a) On (\varepsilon, A) where (A \rightarrow s) \in R, push s^R and go to q_{loop}
   (b) On (\sigma, \sigma), pop \sigma and go to q_{loop}
   (c) On (\varepsilon, $) go to q_{accept}

Otherwise, the PDA will get stuck!
Designing PDA

\[ \varepsilon, \varepsilon \rightarrow S\$ \]

\[ \varepsilon, A \rightarrow s^R \text{ for each rule } A \rightarrow s \]

\[ \sigma, \sigma \rightarrow \varepsilon \text{ for each terminal } \sigma \]

\[ \varepsilon, \$ \rightarrow \varepsilon \]
S → aTb
T → Ta | ε

ε,ε → S
ε,ε → $ 
ε,ε → ε
ε,ε → T
ε,T → a
b,b → ε
a,a → ε
ε,ε → a
ε,ε → T
ε,ε → T
A language is generated by a CFG

⇒

It is recognized by a PDA
Converting a PDA to a CFG

Given PDA \( P = (Q, \Sigma, \Gamma, \delta, q, F) \)
Construct a CFG \( G = (V, \Sigma, R, S) \) with \( L(G) = L(P) \)

First, simplify \( P \) so that:

1. It has a single accept state, \( q_{\text{accept}} \)
2. It empties the stack before accepting
3. Each transition does exactly one of:
   • pushes a symbol;
   • pops a symbol.
SIMPLIFY

q₀

ε, ε → $ → q₁

0, ε → 0

ε, ε → θ → q₃

ε, $ → ε → q₂

1, 0 → ε

ε, ε → θ → q₄

ε, 0 → ε → q₅
For each pair of states $p$ and $q$ in $P$, add a variable $A_{pq}$ to CFG that generates all strings that can take $P$ from $p$ to $q$ without changing the stack*

$$V = \{A_{pq} \mid p, q \in Q\}$$

$$S = A_{q0q_{accept}}$$

*Starting from any stack $S$ in $p$, including empty stack, $P$ has stack $S$ at $q$. 
What strings does \( A_{q_0q_1} \) generate? none

What strings does \( A_{q_1q_2} \) generate? \( \{0^n1^n \mid n > 0\} \)

What strings does \( A_{q_1q_3} \) generate? none
A_{pq} generates all strings that take P from p to q without changing the stack

Let x be such a string

- P’s first move on x must be a push
- P’s last move on x must be a pop

Consider the stack while reading x. Either:

1. New portion of the stack first empties only at the end of x
2. New portion empties before the end of x
1. New portion of the stack first empties only at the end of x

\[ A_{pq} \rightarrow aA_{rs}b \]
2. New portion empties **before the end of x**

\[
A_{pq} \rightarrow A_{pr}A_{rq}
\]
\[ V = \{ A_{pq} \mid p, q \in Q \} \]

\[ S = A_{q_0 q_{\text{accept}}} \]

For each \( p, q, r, s \in Q, t \in \Gamma \) and \( a, b \in \Sigma_{\varepsilon} \)

If \((r, t) \in \delta(p, a, \varepsilon) \text{ and } (q, \varepsilon) \in \delta(s, b, t)\)

Then add the rule \( A_{pq} \rightarrow aA_{rs}b \)

For each \( p, q, r \in Q \),

add the rule \( A_{pq} \rightarrow A_{pr}A_{rq} \)

For each \( p \in Q \),

add the rule \( A_{pp} \rightarrow \varepsilon \)
What strings does $A_{q_0q_1}$ generate? none
What strings does $A_{q_1q_2}$ generate? $\{0^n1^n \mid n > 0\}$
What strings does $A_{q_1q_3}$ generate? none
A language is generated by a CFG

$\iff$

It is recognized by a PDA
TURING MACHINE (TM)

\[ q_1 \]

INPUT

UNBOUNDED (on the right) TAPE
A TM can loop forever
TM can both write to and read from the tape

The head can move left and right

The input does not have to be read entirely

Accept and Reject take immediate effect

Infinite tape on the right, stick on the left

TM is deterministic (will consider NTMs later)
Testing membership in $B = \{ w#w \mid w \in \{0,1\}^* \}$

- **STATE**
  - $q_0$, FIND $\#$
  - $q_1$, FIND $\#$, $q_0$, FIND $\square$, $q_1$, FIND $\square$
  - GO LEFT

- **Transition Table**:
  - 0: X X 1 # X 1 1
  - X: X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X X
A **TM** is a 7-tuple $T = (Q, \Sigma, \Gamma, \delta, q_0, q_{\text{accept}}, q_{\text{reject}})$:

- **Q** is a finite set of states
- **$\Sigma$** is the input alphabet, where $\square \notin \Sigma$
- **$\Gamma$** is the tape alphabet, where $\square \in \Gamma$ and $\Sigma \subseteq \Gamma$
- $\delta : Q \times \Gamma \rightarrow Q \times \Gamma \times \{L,R\}$
- $q_0 \in Q$ is the start state
- $q_{\text{accept}} \in Q$ is the accept state
- $q_{\text{reject}} \in Q$ is the reject state, and $q_{\text{reject}} \neq q_{\text{accept}}$
CONFIGURATIONS

1101000110

q_7

111010000110
\{ 0^{2^n} \mid n \geq 0 \}
\{ 0^{2^n} \mid n \geq 0 \}
Accepting and rejecting

A TM on input string $w$ may

either halt (enter $q_{\text{accept}}$ or $q_{\text{reject}}$)
or never halt (loop)

A TM is a decider if it halts on every input.
A TM **recognizes** a language L if it accepts all strings in L and no other strings.

- A language is called **recognizable** (or enumerable) if some TM recognizes it.

A TM **decides** a language L if it accepts all strings in L and rejects all strings not in L.

- A language is called **decidable** (or recursive) if some TM decides it.