# Week5 monday

Three different CFGs that each generate the language $\{abba\}$

$$(\{S, T, V, W\}, \{a, b\}, \{S \to aT, T \to bV, V \to bW, W \to a\}, S)$$

$$(\{Q\}, \{a, b\}, \{Q \to abba\}, Q)$$

$$(\{X, Y\}, \{a, b\}, \{X \to aYa, Y \to bb\}, X)$$

Design a CFG to generate the language $\{a^n b^n \mid n \geq 0\}$

*Sample derivation:*

Design a CFG to generate the language $\{a^i b^j \mid j \geq i \geq 0\}$

*Sample derivation:*

**Theorem 2.20**: A language is generated by some context-free grammar if and only if it is recognized by some push-down automaton.

Definition: a language is called **context-free** if it is the language generated by a context-free grammar. The class of all context-free language over a given alphabet $\Sigma$ is called **CFL**.

Consequences:

- Quick proof that every regular language is context free
- To prove closure of the class of context-free languages under a given operation, we can choose either of two modes of proof (via CFGs or PDAs) depending on which is easier
- To fully specify a PDA we could give its 6-tuple formal definition or we could give its input alphabet, stack alphabet, and state diagram. An informal description of a PDA is a step-by-step description of how its computations would process input strings; the reader should be able to reconstruct the state diagram or formal definition precisely from such a descripton. The informal description of a PDA can refer to some common modules or subroutines that are computable by PDAs:
  - PDAs can "test for emptiness of stack" without providing details. *How?* We can always push a special end-of-stack symbol, \$, at the start, before processing any input, and then use this symbol as a flag.
  - PDAs can "test for end of input" without providing details. *How?* We can transform a PDA to one where accepting states are only those reachable when there are no more input symbols.

Over $\Sigma = \{a, b\}$, let $L = \{a^n b^m \mid n \neq m\}$. **Goal**: Prove $L$ is context-free.

Suppose $L_1$ and $L_2$ are context-free languages over $\Sigma$. **Goal**: $L_1 \cup L_2$ is also context-free.

*Approach 1: with PDAs*

Let $M_1 = (Q_1, \Sigma, \Gamma_1, \delta_1, q_1, F_1)$ and $M_2 = (Q_2, \Sigma, \Gamma_2, \delta_2, q_2, F_2)$ be PDAs with $L(M_1) = L_1$ and $L(M_2) = L_2$.

Define $M =$

*Approach 2: with CFGs*

Let $G_1 = (V_1, \Sigma, R_1, S_1)$ and $G_2 = (V_2, \Sigma, R_2, S_2)$ be CFGs with $L(G_1) = L_1$ and $L(G_2) = L_2$.

Define $G =$

Suppose $L_1$ and $L_2$ are context-free languages over $\Sigma$. **Goal**: $L_1 \circ L_2$ is also context-free.

*Approach 1: with PDAs*

Let $M_1 = (Q_1, \Sigma, \Gamma_1, \delta_1, q_1, F_1)$ and $M_2 = (Q_2, \Sigma, \Gamma_2, \delta_2, q_2, F_2)$ be PDAs with $L(M_1) = L_1$ and $L(M_2) = L_2$.

Define $M =$

*Approach 2: with CFGs*

Let $G_1 = (V_1, \Sigma, R_1, S_1)$ and $G_2 = (V_2, \Sigma, R_2, S_2)$ be CFGs with $L(G_1) = L_1$ and $L(G_2) = L_2$.

Define $G =$

*Summary*

Over a fixed alphabet $\Sigma$, a language $L$ is **regular**

iff it is described by some regular expression
iff it is recognized by some DFA
iff it is recognized by some NFA

Over a fixed alphabet $\Sigma$, a language $L$ is **context-free**

iff it is generated by some CFG
iff it is recognized by some PDA

**Fact**: Every regular language is a context-free language.

**Fact**: There are context-free languages that are not nonregular.

**Fact**: There are countably many regular languages.

**Fact**: There are countably inifnitely many context-free languages.

*Consequence*: Most languages are **not** context-free!

**Examples of non-context-free languages**

$$\{a^n b^n c^n \mid 0 \leq n, n \in \mathbb{Z}\}$$
$$\{a^i b^j c^k \mid 0 \leq i \leq j \leq k, i \in \mathbb{Z}, j \in \mathbb{Z}, k \in \mathbb{Z}\}$$
$$\{ww \mid w \in \{0,1\}^*\}$$

(Sipser Ex 2.36, Ex 2.37, 2.38)

There is a Pumping Lemma for CFL that can be used to prove a specific language is non-context-free: If $A$ is a context-free language, there there is a number $p$ where, if $s$ is any string in $A$ of length at least $p$, then $s$ may be divided into five pieces $s = uvxyz$ where (1) for each $i \geq 0$, $uv^ixy^iz \in A$, (2) $|uv| > 0$, (3) $|vxy| \leq p$. *We will not go into the details of the proof or application of Pumping Lemma for CFLs this quarter.*

# Week5 wednesday

A set $X$ is said to be **closed** under an operation $OP$ if, for any elements in $X$, applying $OP$ to them gives an element in $X$.

| True/False | Closure claim |
|---|---|
| True | The set of integers is closed under multiplication. $\forall x \forall y \, (\, (x \in \mathbb{Z} \land y \in \mathbb{Z}) \to xy \in \mathbb{Z} \,)$ |
| True | For each set $A$, the power set of $A$ is closed under intersection. $\forall A_1 \forall A_2 \, (\, (A_1 \in \mathcal{P}(A) \land A_2 \in \mathcal{P}(A) \in \mathbb{Z}) \to A_1 \cap A_2 \in \mathcal{P}(A) \,)$ |
| | The class of regular languages over $\Sigma$ is closed under complementation. |
| | The class of regular languages over $\Sigma$ is closed under union. |
| | The class of regular languages over $\Sigma$ is closed under intersection. |
| | The class of regular languages over $\Sigma$ is closed under concatenation. |
| | The class of regular languages over $\Sigma$ is closed under Kleene star. |
| | The class of context-free languages over $\Sigma$ is closed under complementation. |
| | The class of context-free languages over $\Sigma$ is closed under union. |
| | The class of context-free languages over $\Sigma$ is closed under intersection. |
| | The class of context-free languages over $\Sigma$ is closed under concatenation. |
| | The class of context-free languages over $\Sigma$ is closed under Kleene star. |

Assume $\Sigma = \{0, 1, \#\}$

| | | | | | |
|---|---|---|---|---|---|
| $\Sigma^*$ | Regular | / | nonregular and context-free | / | not context-free |
| $\{0^i \# 1^j \mid i \geq 0, j \geq 0\}$ | Regular | / | nonregular and context-free | / | not context-free |
| $\{0^i 1^j \# 1^j 0^i \mid i \geq 0, j \geq 0\}$ | Regular | / | nonregular and context-free | / | not context-free |
| $\{0^i 1^j \# 0^i 1^j \mid i \geq 0, j \geq 0\}$ | Regular | / | nonregular and context-free | / | not context-free |

**Turing machines**: unlimited read + write memory, unlimited time (computation can proceed without "consuming" input and can re-read symbols of input)

- Division betweeen program (CPU, state diagram) and data
- Unbounded memory gives theoretical limit to what modern computation (including PCs, supercomputers, quantum computers) can achieve
- State diagram formulation is simple enough to reason about (and diagonalize against) while expressive enough to capture modern computation

For Turing machine $M = (Q, \Sigma, \Gamma, \delta, q_0, q_{accept}, q_{reject})$ the **computation** of $M$ on a string $w$ over $\Sigma$ is:

- Read/write head starts at leftmost position on tape.
- Input string is written on $|w|$-many leftmost cells of tape, rest of the tape cells have the blank symbol. **Tape alphabet** is $\Gamma$ with $\llcorner\!\lrcorner \in \Gamma$ and $\Sigma \subseteq \Gamma$. The blank symbol $\llcorner\!\lrcorner \notin \Sigma$.
- Given current state of machine and current symbol being read at the tape head, the machine transitions to next state, writes a symbol to the current position of the tape head (overwriting existing symbol), and moves the tape head L or R (if possible). Formally, **transition function** is

$$\delta : Q \times \Gamma \to Q \times \Gamma \times \{L, R\}$$

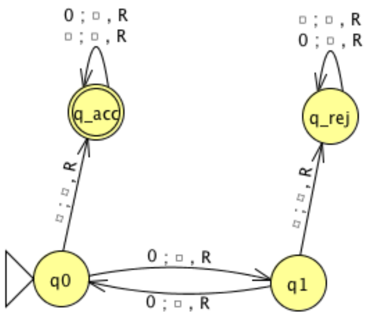
- Computation ends if and when machine enters either the accept or the reject state. This is called **halting**. Note: $q_{accept} \neq q_{reject}$.

The **language recognized by the Turing machine** $M$, is

$\{w \in \Sigma^* \mid \text{computation of } M \text{ on } w \text{ halts after entering the accept state}\} = \{w \in \Sigma^* \mid w \text{ is accepted by } M\}$

An example Turing machine: $\Sigma =$ $\qquad\qquad$ $, \Gamma =$

$$\delta((q0, 0)) =$$



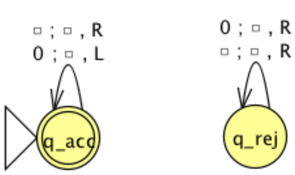

Formal definition:

Sample computation:

| | | | | | | |
|---|---|---|---|---|---|---|
| $q0 \downarrow$ | | | | | | |
| 0 | 0 | 0 | ␣ | ␣ | ␣ | ␣ |
| | | | | | | |
| | | | | | | |
| | | | | | | |
| | | | | | | |
| | | | | | | |
| | | | | | | |
| | | | | | | |
| | | | | | | |

The language recognized by this machine is . . .

*Extra practice:*




Formal definition:

Sample computation: