Week 5 at a glance

Textbook reading: Section 2.2, 2.1.
Before Monday, read Theorem 2.20.

Before Wednesday, read Example 2.18 (page 114).

Before Friday, read Figure 3.1.

For Week 6 Monday: Page 165-166 Introduction to Section 3.1.

We will be learning and practicing to:
- Clearly and unambiguously communicate computational ideas using appropriate formalism. Translate across levels of abstraction.
    - Describe and use models of computation that don't involve state machines.
        * **Use context-free grammars and relate them to languages and pushdown automata.**
    - Use precise notation to formally define the state diagram of a Turing machine
    - Use clear English to describe computations of Turing machines informally.
        * **Design a PDA that recognizes a given language.**
    - Give examples of sets that are context-free (and prove that they are).
        * **State the definition of the class of context-free languages**
        * **Explain the limits of the class of context-free languages**
        * **Identify some context-free sets and some non-context-free sets**
- Know, select and apply appropriate computing knowledge and problem-solving techniques. Reason about computation and systems.
    - Describe and closure properties of classes of languages under certain operations.
        * **Apply a general construction to create a new PDA or CFG from an example one.**
        * **Formalize a general construction from an informal description of it.**
        * **Use general constructions to prove closure properties of the class of context-free langugages.**
        * **Use counterexamples to prove non-closure properties of the class of context-free langugages.**

TODO:

Schedule your Test 1 Attempt 1, Test 2 Attempt 1, Test 1 Attempt 2, and Test 2 Attempt 2 times at PrairieTest (http://us.prairietest.com)

Review Quiz 5 on PrairieLearn (http://us.prairielearn.com), complete by Sunday 11/4/2024

$$G = (V, \Sigma, R, S) \qquad L(G) = \{w \in \Sigma^* \mid S \Rightarrow^* w\}$$

start variable

Monday: Context-free languages

Warmup: Design a CFG to generate the language $\{a^i b^j \mid j \geq i \geq 0\}$ over $\{a, b\}$
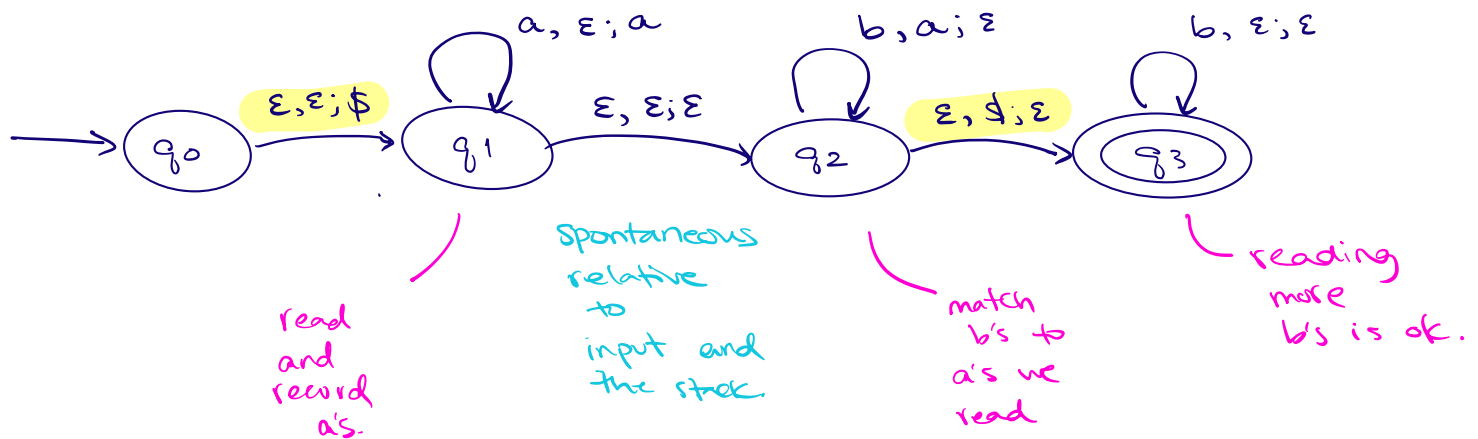
$$G = (\{S\}, \{a, b\}, R, S)$$

where R is given by

$$S \rightarrow \underset{①}{\varepsilon} \mid \underset{②}{a S b} \mid \underset{③}{S b}$$

Remember: to generate $\{a^n b^n \mid n \geq 0\}$, can use

$$(\{S\}, \{a, b\}, \{S \rightarrow a S b \mid \varepsilon\}, S)$$

Sample derivation:

$$S \underset{①}{\Rightarrow} \varepsilon \qquad \text{good b/c} \qquad \varepsilon = a^0 b^0 \quad i = j = 0$$

$$S \underset{③}{\Rightarrow} S b \underset{③}{\Rightarrow} S b b \underset{②}{\Rightarrow} a S b b b \underset{①}{\Rightarrow} a b b b$$

good by $abbb = a^1 b^3 \quad i = 1 \quad j = 3$

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

$a, \varepsilon ; a$ $b, a ; \varepsilon$ $b, \varepsilon ; \varepsilon$

$\varepsilon, \varepsilon ; \$$ $\varepsilon, \varepsilon ; \varepsilon$ $\varepsilon, \$; \varepsilon$

$q_0 \quad q_1 \quad q_2 \quad q_3$

read and record a's.

Spontaneous relative to input and the stack

match b's to a's we read

reading more b's is ok.

Trace ab

$\underset{q_0}{\llcorner} \underset{q_1}{\overset{\$}{\llcorner}} \underset{q_1}{\overset{\boxed{a}}{\underset{\$}{\llcorner}}} \underset{q_2}{\overset{\boxed{a}}{\underset{\$}{\llcorner}}} \underset{q_2}{\overset{\$}{\llcorner}} \underset{q_3}{\llcorner}$

ε

$\underset{q_0}{\llcorner} \underset{q_1}{\overset{\$}{\llcorner}} \underset{q_2}{\overset{\$}{\llcorner}} \underset{q_3}{\llcorner}$
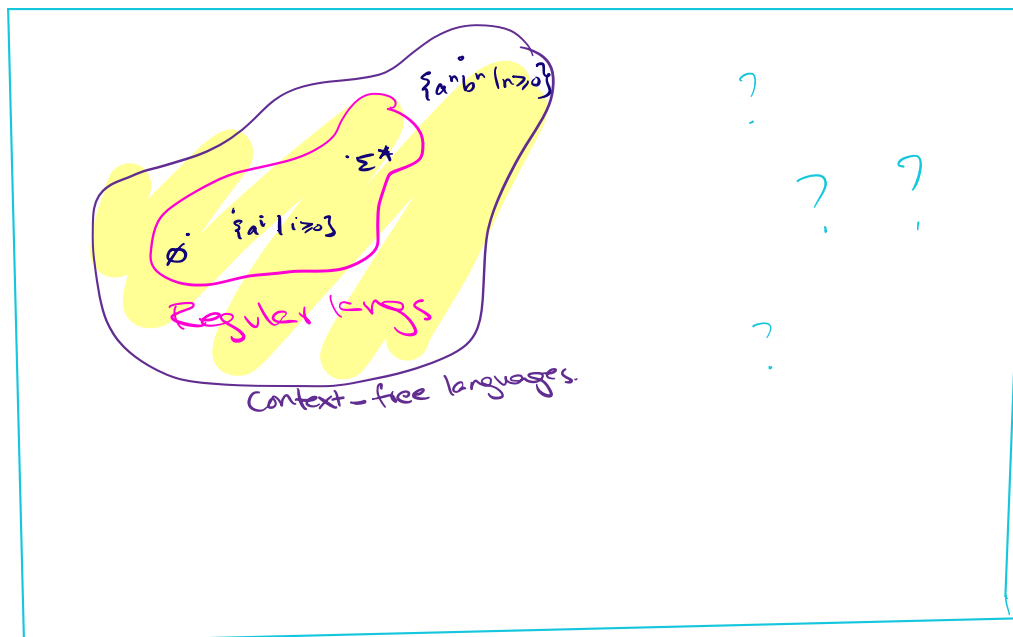
abb

aba

Version October 24, 2024 (2)

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 Σ is called **CFL**.

Consequences:

start w/ a NFA recognizing a language. Define PDA simulates it, recognizes the same language.

- Quick proof that every regular language is context free

  start w/ DFA recognizing a language. Define CFG generates language

- 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.
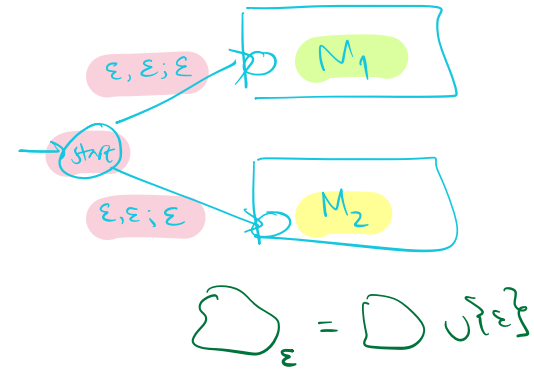

$\mathcal{P}(\Sigma^*)$ the set of all languages over Σ

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 $M = (\ Q, \Sigma, \Gamma, \delta, q_{new}, F_1 \cup F_2\ )$

Assume $Q_1 \cap Q_2 = \emptyset$
and $q_{new} \notin Q_1 \cup Q_2$

Define $\quad Q = Q_1 \cup Q_2 \cup \{q_{new}\}$

$\Gamma = \Gamma_1 \cup \Gamma_2$

$\delta: \underline{Q \times \Sigma_\varepsilon \times \Gamma_\varepsilon} \to \mathcal{P}(Q \times \Gamma_\varepsilon)$

power set

$\delta((q, a, b)) = \begin{cases} \underline{\{(q_1, \varepsilon), (q_2, \varepsilon)\}} & q = q_{new}, a = \varepsilon, b = \varepsilon \\ \delta_1((q, a, b)) & q \in Q_1, a \in \Sigma_\varepsilon, b \in \Gamma_{1\varepsilon} \\ \delta_2((q, a, b)) & q \in Q_2, a \in \Sigma_\varepsilon, b \in \Gamma_{2\varepsilon} \\ \emptyset & \text{otherwise} \end{cases}$

$\varepsilon, \varepsilon; \varepsilon$ → M_1

start

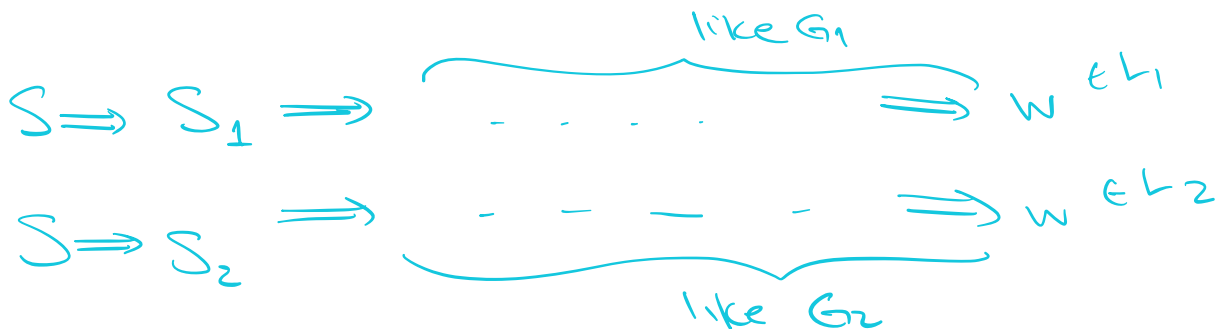$\varepsilon, \varepsilon; \varepsilon$ → M_2

$\bigcirc_\varepsilon = \bigcirc \cup \{\varepsilon\}$

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 $G = (\ V_1 \cup V_2 \cup \{S\},\ \Sigma,\ R_1 \cup R_2 \cup \{S \to S_1, S \to S_2\}, S\ )$

Assume $V_1 \cap V_2 = \emptyset$
and $S \notin V_1 \cup V_2$

like G_1

$S \Rightarrow S_1 \Rightarrow \quad - - - - - \Rightarrow w \quad \in L_1$

$S \Rightarrow S_2 \quad \Rightarrow \quad - - - - - \Rightarrow w \quad \in L_2$

like G_2

Suppose L_1 and L_2 are context-free languages over Σ. **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 $G =$

Version October 24, 2024 (5)

Wednesday: Context-free and non-context-free languages

Summary

Over a fixed alphabet Σ, 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 Σ, 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

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

(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 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 $i \ge 0$, $uv^i xy^i z \in A$, (2) $|uv| > 0$, (3) $|vxy| \le p$. *We will not go into the details of the proof or application of Pumping Lemma for CFLs this quarter.*

Recall: 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 Σ is closed under complementation.
	The class of regular languages over Σ is closed under union.
	The class of regular languages over Σ is closed under intersection.
	The class of regular languages over Σ is closed under concatenation.
	The class of regular languages over Σ is closed under Kleene star.
	The class of context-free languages over Σ is closed under complementation.
	The class of context-free languages over Σ is closed under union.
	The class of context-free languages over Σ is closed under intersection.
	The class of context-free languages over Σ is closed under concatenation.
	The class of context-free languages over Σ is closed under Kleene star.

Friday: Turing machines

We are ready to introduce a formal model that will capture a notion of general purpose computation.

- *Similar to DFA, NFA, PDA*: input will be an arbitrary string over a fixed alphabet.

- *Different from NFA, PDA*: machine is deterministic.

- *Different from DFA, NFA, PDA*: read-write head can move both to the left and to the right, and can extend to the right past the original input.

- *Similar to DFA, NFA, PDA*: transition function drives computation one step at a time by moving within a finite set of states, always starting at designated start state.

- *Different from DFA, NFA, PDA*: the special states for rejecting and accepting take effect immediately.

(See more details: Sipser p. 166)

Formally: a Turing machine is $M = (Q, \Sigma, \Gamma, \delta, q_0, q_{accept}, q_{reject})$ where δ is the **transition function**

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

The **computation** of M on a string w over Σ 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 Γ with $\sqcup \in \Gamma$ and $\Sigma \subseteq \Gamma$. The blank symbol $\sqcup \notin \Sigma$.
- 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).
- 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 $L(M) = \{w \in \Sigma^* \mid w \text{ is accepted by } M\}$, which is defined as
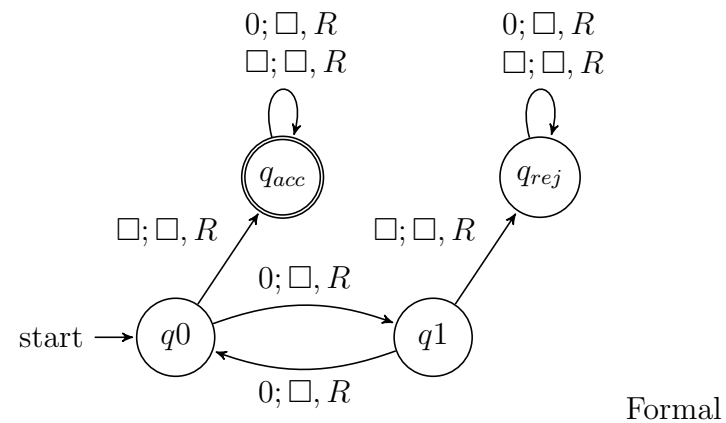
$$\{w \in \Sigma^* \mid \text{computation of } M \text{ on } w \text{ halts after entering the accept state}\}$$

definition:

Sample computation:

| $q0\downarrow$						
0	0	0	␣	␣	␣	␣

Formal

The language recognized by this machine is . . .

Describing Turing machines (Sipser p. 185) To define a Turing machine, we could give a

- **Formal definition**: the 7-tuple of parameters including set of states, input alphabet, tape alphabet, transition function, start state, accept state, and reject state; or,

- **Implementation-level definition**: English prose that describes the Turing machine head movements relative to contents of tape, and conditions for accepting / rejecting based on those contents.

- **High-level description**: description of algorithm (precise sequence of instructions), without implementation details of machine. As part of this description, can "call" and run another TM as a subroutine.

Fix $\Sigma = \{0, 1\}$, $\Gamma = \{0, 1, \llcorner\}$ for the Turing machines with the following state diagrams:

$$\square; \square, R$$
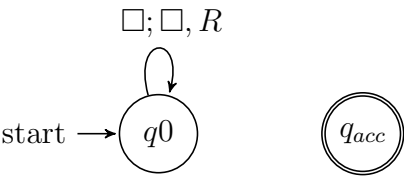
start $\longrightarrow$ ($q0$)   ((q_{acc}))

Example of string accepted:
Example of string rejected:

Implementation-level description

High-level description

start $\longrightarrow$ (q_{rej})   ((q_{acc}))

Example of string accepted:
Example of string rejected:

Implementation-level description

High-level description

Version October 24, 2024 (10)

Example of string accepted:
Example of string rejected:
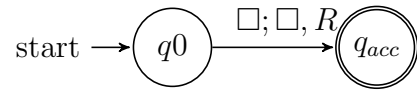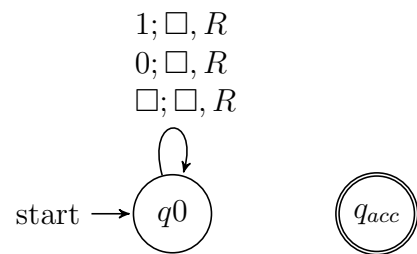

Implementation-level description

High-level description

Example of string accepted:
Example of string rejected:

Implementation-level description

High-level description