Week 5 at a glance

Textbook reading: Chapter 2

Before Monday, read Introduction to Section 2.1 (pages 101-102).

Before Wednesday, read Section 2.1

Before Friday, read Theorem 2.20.

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.
 - * Identify the components of a formal definition of a context-free grammar (CFG)
 - * Derive strings in the language of a given CFG
 - * Determine the language of a given CFG
 - * Design a CFG generating a given language
 - * 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 prove 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 languages.
 - * Use counterexamples to prove non-closure properties of the class of context-free languages.

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) . The first Test 1 sessions are next week!

Review Quiz 4 on PrairieLearn (http://us.prairielearn.com), due 2/5/2025

Homework 3 submitted via Gradescope (https://www.gradescope.com/), due 2/6/2025

Review Quiz 5 on PrairieLearn (http://us.prairielearn.com), due 2/12/2025

Monday: More Pushdown Automata

Definition A **pushdown automaton** (PDA) is specified by a 6-tuple $(Q, \Sigma, \Gamma, \delta, q_0, F)$ where Q is the finite set of states, Σ is the input alphabet, Γ is the stack alphabet,

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

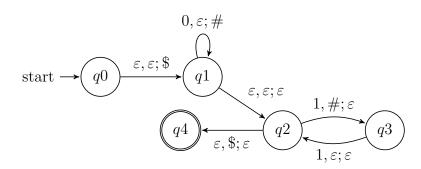
is the transition function, $q_0 \in Q$ is the start state, $F \subseteq Q$ is the set of accept states.

For the PDA state diagrams below, $\Sigma = \{0, 1\}$.

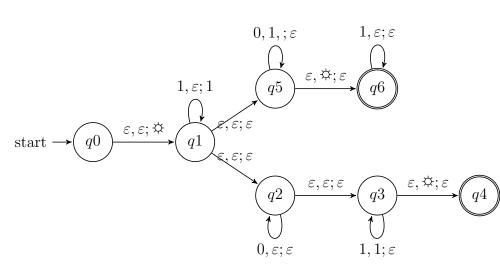
Mathematical description of language

State diagram of PDA recognizing language

$$\Gamma = \{\$, \#\}$$

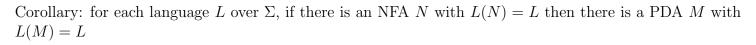


$$\Gamma = \{ \circlearrowleft, 1 \}$$



$$\{0^i 1^j 0^k \mid i,j,k \geq 0\}$$

Note: alternate notation is to replace; with \rightarrow on arrow labels.



Proof idea: Declare stack alphabet to be $\Gamma = \Sigma$ and then don't use stack at all.

Big picture: PDAs are motivated by wanting to add some memory of unbounded size to NFA. How do we accomplish a similar enhancement of regular expressions to get a syntactic model that is more expressive?

DFA, NFA, PDA: Machines process one input string at a time; the computation of a machine on its input string reads the input from left to right.

Regular expressions: Syntactic descriptions of all strings that match a particular pattern; the language described by a regular expression is built up recursively according to the expression's syntax

Context-free grammars: Rules to produce one string at a time, adding characters from the middle, beginning, or end of the final string as the derivation proceeds.

Wednesday: Context-free Grammars and Languages

Definitions below are on pages 101-102.

Term	Typical symbol	Meaning
	or Notation	
Context-free grammar (CFG)	G	$G = (V, \Sigma, R, S)$
The set of variables	V	Finite set of symbols that represent phases in pro-
		duction pattern
The set of terminals	Σ	Alphabet of symbols of strings generated by CFG $V \cap \Sigma = \emptyset$
The set of rules	R	Each rule is $A \to u$ with $A \in V$ and $u \in (V \cup \Sigma)^*$
The start variable	S	Usually on left-hand-side of first/ topmost rule
Derivation	$S \Rightarrow \cdots \Rightarrow w$	Sequence of substitutions in a CFG (also written $S \Rightarrow^* w$). At each step, we can apply one rule to one occurrence of a variable in the current string by substituting that occurrence of the variable with the right-hand-side of the rule. The derivation must end when the current string has only terminals (no
Language generated by the context-free grammar G	L(G)	variables) because then there are no instances of variables to apply a rule to. The set of strings for which there is a derivation in G . Symbolically: $\{w \in \Sigma^* \mid S \Rightarrow^* w\}$ i.e.
Context-free language		$\{w \in \Sigma^* \mid \text{there is derivation in } G \text{ that ends in } w\}$ A language that is the language generated by some context-free grammar

Examples of context-free grammars, derivations in those grammars, and the languages generated by those grammars

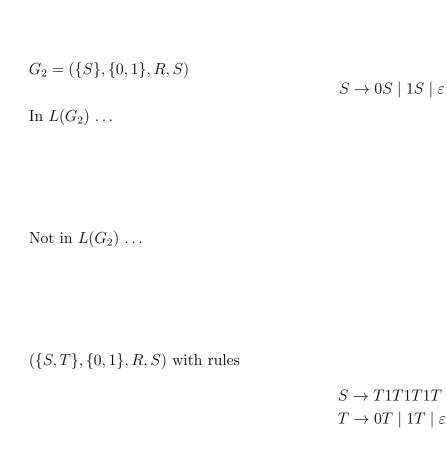
$$G_1 = (\{S\}, \{0\}, R, S)$$
 with rules

$$S \to 0S$$

$$S \to 0$$

In $L(G_1)$...

Not in $L(G_1)$...



In $L(G_3)$...

Not in $L(G_3)$...

 $G_4 = (\{A, B\}, \{0, 1\}, R, A)$ with rules

 $A \to 0A0 \mid 0A1 \mid 1A0 \mid 1A1 \mid 1$

In $L(G_4)$...

Not in $L(G_4)$...



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

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:

- 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.

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

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 infinitely many context-free languages.

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

Examples of non-context-free languages

$$\begin{aligned} &\{a^nb^nc^n\mid 0\leq n, n\in\mathbb{Z}\}\\ &\{a^ib^jc^k\mid 0\leq i\leq j\leq k, i\in\mathbb{Z}, j\in\mathbb{Z}, k\in\mathbb{Z}\}\\ &\{ww\mid w\in\{0,1\}^*\} \end{aligned}$$

(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 $i \ge 0$, $uv^ixy^iz \in A$, (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.		