Monday: Context-free grammars

These definitions are on pages 101-102.

Term	**Typical symbol** or **Notation**	**Meaning**
Context-free grammar (CFG)	G	$G = (V, \Sigma, R, S)$
The set of **variables**	V	Finite set of symbols that represent phases in production 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 Var → RHS	$S \Rightarrow \cdots \Rightarrow w$	Sequence of substitutions in a CFG (also written $S \Rightarrow^* w$). At each step, we can $\boxed{\text{apply}}$ one rule to one $\boxed{\text{occurrence}}$ of a variable in the current string by substituting that occurrence of the variable with the ~~left~~ right-hand-side of the rule. The derivation must end when the current string has only terminals (no variables) because then there are no instances of variables to apply a rule to.
Language **generated** by the context-free grammar G	$L(G)$	The set of strings for which there is a derivation in G. Symbolically: $\{w \in \Sigma^* \mid S \Rightarrow^* w\}$ i.e. $\{w \in \Sigma^* \mid$ there is derivation in G that ends in $w\}$
Context-free language		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

set of variables

set of terminals

start variables

LHS RHS

$$R = \{ \; ① \; S \to 0S, \quad ② \; S \to 0 \; \}.$$

In $L(G_1)$ ...

$S \overset{①}{\Longrightarrow} 0S \overset{①}{\Longrightarrow} 00S \overset{②}{\Longrightarrow} 000$ so $000 \in L(G_1)$

$S \overset{②}{\Longrightarrow} 0$ so $0 \in L(G_1)$

Not in $L(G_1)$ ... ε b/c all rules introduce (at least one) 0. $\varepsilon \notin L(G_1)$.

$R = \{S \to 0S, S \to 1S, S \to \varepsilon\}$

$G_2 = (\{S\}, \{0,1\}, R, S)$

$$S \to 0S \mid 1S \mid \varepsilon$$

notation abbreviating multiple rules with same variable on LHS.

In $L(G_2)$ ...

$S \Rightarrow 0S \Rightarrow 01S \Rightarrow 01\varepsilon = 01$

start variable

Not in $L(G_2)$ ... There is no such string!

G_2 lets us build up each string over $\{0,1\}$ by introducing characters from left to right. $L(G_2) = \{0,1\}^*$.

BUILD AT ANY PT IN THE STRING

$(\{S,T\}, \{0,1\}, R, S)$ with rules

2 variables!

$$S \to T1T1T1T$$
$$T \to 0T \mid 1T \mid \varepsilon$$

plug in some string over $\{0,1\}$

In $L(G_3)$ ...

$S \Rightarrow T1T1T1T \Rightarrow 1T1T1T \Rightarrow 1T1T1 \Rightarrow 11T1 \Rightarrow 111$

$S \Rightarrow T1T1T1T \Rightarrow T11T1T \Rightarrow T111T \Rightarrow T111 \Rightarrow 111$

$S \Rightarrow T1T1T1T \Rightarrow 0T1T1T1T \Rightarrow 01T1T1T \Rightarrow \cdots \Rightarrow 0111$

Not in $L(G_3)$ ...

Regular

$L(G_3) = \{w \in \{0,1\}^* \mid w \text{ has at least three 1s}\}$.

Note: rules can have any string over $V \cup \Sigma$ on RHS

both have variables on RHS

$\tilde{G} = (\{S,T\}, \{0,1\}, \{S \to 0S, T \to 1T\}, S)$

$S \Rightarrow 0S \cdots ?$ $\quad S \Rightarrow 0S \Rightarrow 00S \cdots ?$ $\quad S \Rightarrow 0S \Rightarrow 00S \cdots ?$

$L(\tilde{G}) = \emptyset$.

What about a grammar that generates a nonregular language?

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

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

In $L(G_4)$ ... 1 $\quad A \Rightarrow 1$

010

011

$A \Rightarrow 0A0 \Rightarrow 01A10 \Rightarrow 010A110 \Rightarrow 0101110$

Not in $L(G_4)$ ... ε

$L(G_4) = \{ w \in \{0,1\}^* \mid$ w is odd length and middle character is 1 $\}$

Claim: $L(G_4)$ is nonregular.

(extra practice: prove this with pumping lemma)

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

$$G_5 = (\underbrace{\{S\}}_{V}, \underbrace{\{a, b\}}_{\Sigma}, \underbrace{\{S \to \varepsilon \mid aSb\}}_{R}, \underbrace{S}_{S})$$

Sample derivation:

$$S \xRightarrow{1} \varepsilon$$

$$S \xRightarrow{2} aSb \xRightarrow{1} ab$$

$$S \xRightarrow{2} aSb \xRightarrow{2} aaSbb \xRightarrow{1} aabb$$

"grow out string from center"

Wednesday: Context-free languages

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

$$G_1 = (\underbrace{\{S, T\}}_{\text{variables}}, \underbrace{\{a, b\}}_{\text{terminal}}, \underbrace{\{S \to Sb \mid T, \ T \to aTb \mid \varepsilon\}}_{\text{rules}}, \underbrace{S}_{\text{start var}})$$

$$G_2 = (\{S\}, \{a, b\}, \{S \to Sb \mid aSb \mid \varepsilon\}, S)$$

$a \; b$ ← at least as many b's as a's.

$\underbrace{a^i b^i}_{T} \; b^{j-i}$

$L(G_1) = L(G_2) = L.$

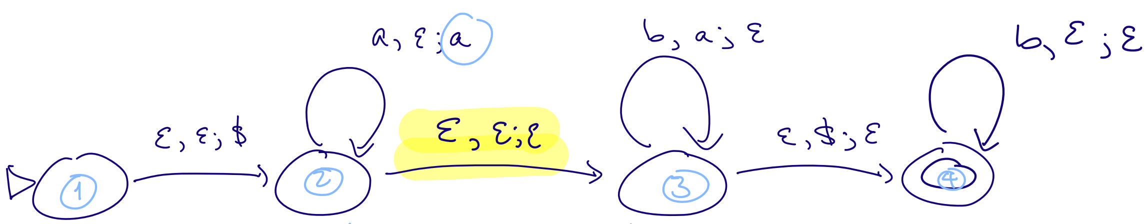
Sample derivation:

in G_1: $S \Rightarrow Sb \Rightarrow Sbb \Rightarrow Sbbb \Rightarrow Tbbb \Rightarrow bbb$

in G_1: $S \Rightarrow Sb \Rightarrow Sbb \Rightarrow Sbbb \Rightarrow Tbbb \Rightarrow aTbbbb \Rightarrow abbbb$

in G_2: $S \Rightarrow Sb \Rightarrow aSbb \Rightarrow aaSbbb \Rightarrow aaSbbbb \Rightarrow aabbbb.$

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



read a's, push them on stack.

read b's, popping one a from ~~stack~~ for each b.

special symbol to help detect empty stack.

On bbb start at 1 move to 2 move to 3 move to 4
read b's in 4, accept

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:

DFA NFA regexp PDA CFG.

- Quick proof that every regular language is context free (HW3).
- 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 $M = \left(\underbrace{Q_1 \cup Q_2 \cup \{q_{new}\}}_{Q}, \Sigma, \underbrace{\Gamma_1 \cup \Gamma_2}_{\Gamma}, \delta, q_{new}, F_1 \cup F_2 \right)$

ASSUME

$Q_1 \cap Q_2 = \emptyset$

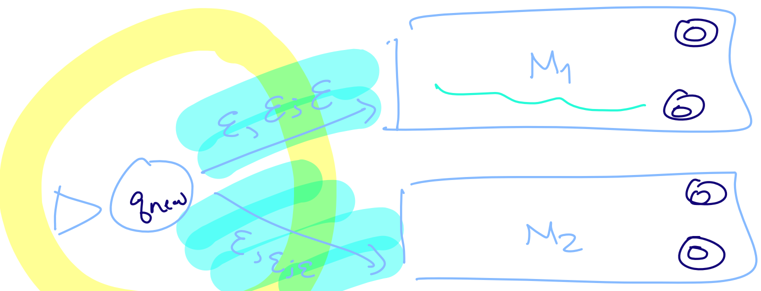
$q_{new} \notin Q_1 \cup Q_2$.

where

curr state, input label, top of stack label

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

$\delta : Q \times \Sigma_\varepsilon \times \Gamma_\varepsilon$



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 $G = \left(V_1 \cup V_2 \cup \{S_{new}\}, \Sigma, R_1 \cup R_2 \cup \{S_{new} \to S_1 \mid S_2\}, S_{new} \right)$

ASSUME

$V_1 \cap V_2 = \emptyset$

$S_{new} \notin V_1 \cup V_2$

$S_{new} \Rightarrow S_1 \Rightarrow$ ______________

$S_{new} \Rightarrow S_2 \Rightarrow$ ______________

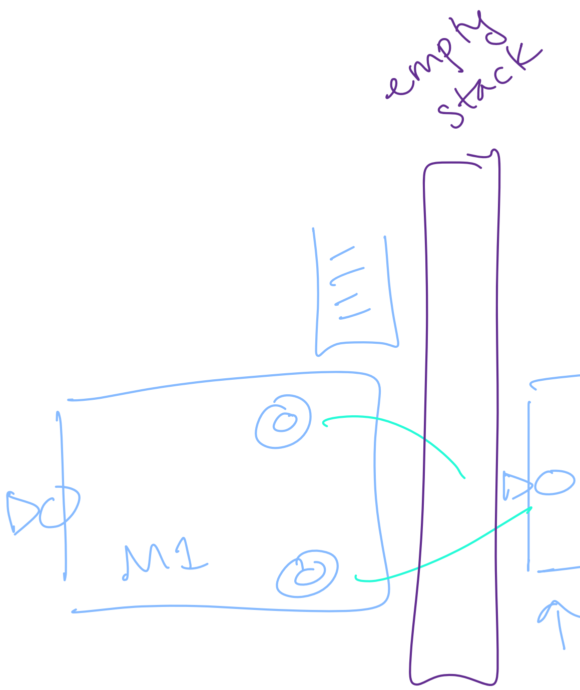
Suppose L_1 and L_2 are context-free languages over Σ. **Goal**: $L_1 \circ L_2$ is also context-free.

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 $M =$

Idea:



HW3

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

ASSUME

$V_1 \cap V_2 = \emptyset$

$S_{new} \notin V_1 \cup V_2$

$$S \Rightarrow \underbrace{S_1}_{\text{first part of string derived from } S_1} \underbrace{S_2}_{\text{second part of string } \cancel{\text{derived}} \text{ from } S_2}$$

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 (Monday)

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

(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.*

Friday: Review

HW2 Q4b

(b) (*Graded for completeness*) Consider the following attempted "proof" that the set

$$X = \{uw \mid u \text{ and } w \text{ are strings over } \{0,1\} \text{ and have the same length}\}$$

is nonregular. $X \neq \{0,1\}^* \circ \{0,1\}^*$ b/c $0 \in \{0,1\}^* \circ \{0,1\}^*$

"Proof" that X is not regular using the Pumping Lemma: Let p be an arbitrary positive integer. We will show that p is not a pumping length for X.
Choose s to be the string 1^p0^p, which is in X because we can choose $u = 1^p$ and $w = 0^p$ which each have length p. Since s is in X and has length greater than or equal to p, if p were to be a pumping length for X, s ought to be pump'able. That is, there should be a way of dividing s into parts x, y, z where $s = xyz$, $|y| > 0$, $|xy| \leq p$, and for each $i \geq 0$, $xy^iz \in X$. Suppose x, y, z are such that $s = xyz$, $|y| > 0$ and $|xy| \leq p$. Since the first p letters of s are all 1 and $|xy| \leq p$, we know that x and y are made up of all 1s. If we let $i = 2$, we get a string xy^iz that is not in X because repeating y twice adds 1s to u but not to w, and strings in X are required to have u and w be the same length. Thus, s is not pumpable (even though it should have been if p were to be a pumping length) and so p is not a pumping length for X. Since p was arbitrary, we have demonstrated that X has no pumping length. By the Pumping Lemma, this implies that X is nonregular.

This language is regular. How do we prove?

- Designing DFA M $L(M) = X$, or design NFA N $L(N) = X$. or designing a regular expression R $L(R) = X$
- Showing that X is the result of applying operations like complementation, setwise concatenation, union, intersection, SUBSTRING(.) to language(s) that are known to be regular.

$$X = \{ s \in \{0,1\}^* \mid s \text{ has even length} \}.$$

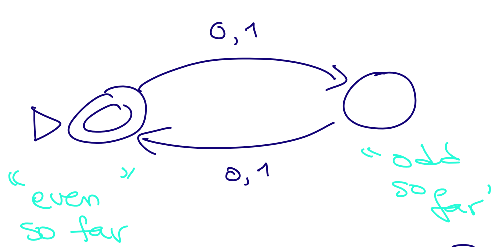
Reg Exp.

ε $\emptyset$ 0 1

$\circ$ $\cup$ $*$ ( )

$$((0 \cup 1) \circ (0 \cup 1))^*$$

DFA or NFA



formal definition — DFA δ / NFA δ

Note: $\varepsilon \in L^*$? Yes.

$L^* = \{ w_1 \cdots w_k \mid k \geq 0 \text{ each } w_i \in L \}$.

Even when $L = \emptyset$ $\emptyset^* \neq \emptyset$, $\varepsilon \in \emptyset^*$.

Related languages $X = \{ uw \mid u,w \in \{0,1\}^*,\ |u| = |w| \}$ regular.

$\{ ww \mid w \in \{0,1\}^* \}$ nonregular but not context free.

Claim: $\{ ww^R \mid w \in \{0,1\}^* \}$ nonregular, but context free

Pf: Let p be arb positive int.

WTS p not pumping length for

$L = \{ ww^R \mid w \in \{0,1\}^* \}$.

We need string $\boxed{s = 0^p 1 1 0^p}$

that is in this language, with length $\geq p$,

not pumpable relative to this language and p.

Confirming, $s \in L$ b/c when $w = 0^p 1$ $s = ww^R$.

Also $|s| = \underset{\text{beginning}}{\#0s} + 1 + 1 + \underset{\text{end}}{\#0s} = 2p+2 \geq p$

WTS s is not pumpable wrt L, p.

WTS $\forall x,y,z\ (\underbrace{s = xyz \wedge |xy| \leq p \wedge |y| > 0}_{\text{HYPs.}} \to \exists i\ xy^iz \notin L)$

Consider any choice of x,y,z that makes HYPs true

There must be nonneg ints m, r where

$x = 0^m$ $y = 0^r$ $z = 0^{p-m-r} 1 1 0^p$

$m + r \leq p$, $r > 0$.

Consider $i = 0$ $xy^iz = xz$ ← can show not in L

$= 0^m 0^{p-m-r} 110^p = 0^{p-r} 110^p$

~~Consider $i = 1$ $xy^iz = xyz = z \notin L$.~~

Consider $i = 2$ $xy^iz = xyyz$. ← can show not in L

$= 0^m 0^r 0^r 0^{p-m-r} 110^p = 0^{p+r} 110^p$

⋮

//

Week 5 at a glance

Textbook reading: Chapter 2

For Monday: Introduction to Section 2.1 (page 102)

For Wednesday: Figure 3.1 (Pages 165-167)

For Friday: Test 1 is Friday Feb 9 in discussion section 4pm-4:50pm WLH 2001. The test covers material in Weeks 1 through 4 and Monday of Week 5. To study for the exam, we recommend reviewing class notes (e.g. annotations linked on the class website, podcast, supplementary video from the class website), reviewing homework (and its posted sample solutions), and in particular *working examples* (extra examples in lecture notes, textbook examples listed in hw, review quizzes – PDFs now available on the class website, discussion examples) and getting feedback (office hours and Piazza).

Make sure you can:

- Classify the computational complexity of a set of strings by determining whether it is regular
  - Determine whether a language is recognizable by a (D or N) FA and/or a PDA
- Use context-free grammars and relate them to languages and pushdown automata
  - Identify the components of a formal definition of a context-free grammar (CFG)
  - Use context-free grammars and relate them to languages and pushdown automata.
  - Derive strings in the language of a given CFG
  - Determine the language of a given CFG
  - Design a CFG generating a given language

TODO:

Review quizzes based on class material each day.

Test this Friday in Discussion section.

Homework assignment 3 due next Thursday.