

# Monday

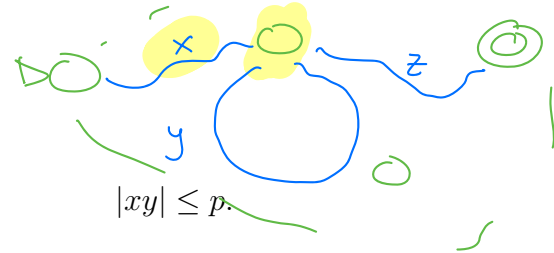
Recap so far: In DFA, the only memory available is in the states. Automata can only “remember” finitely far in the past and finitely much information, because they can have only finitely many states. If a computation path of a DFA visits the same state more than once, the machine can’t tell the difference between the first time and future times it visits this state. Thus, if a DFA accepts one long string, then it must accept (infinitely) many similar strings.

**Definition** A positive integer  $p$  is a **pumping length** of a language  $L$  over  $\Sigma$  means that, for each string  $s \in \Sigma^*$ , if  $|s| \geq p$  and  $s \in L$ , then there are strings  $x, y, z$  such that

$$s = xyz$$

and

$$|y| > 0, \quad \text{for each } i \geq 0, xy^iz \in L, \quad \text{and} \quad |xy| \leq p.$$



**Negation:** A positive integer  $p$  is **not a pumping length** of a language  $L$  over  $\Sigma$  iff

$$\exists s ( |s| \geq p \wedge s \in L \wedge \forall x \forall y \forall z ( (s = xyz \wedge |y| > 0 \wedge |xy| \leq p) \rightarrow \exists i (i \geq 0 \wedge xy^iz \notin L) ) )$$

Informally:

kicked out of  $L$  for some  $i$ .

Restating **Pumping Lemma**: If  $L$  is a regular language, then it has a pumping length.

\* **Contrapositive:** If  $L$  has no pumping length, then it is nonregular.

To prove  $L$  is not regular, we can show that it has no pumping length.

The Pumping Lemma *cannot* be used to prove that a language is regular.

The Pumping Lemma **can** be used to prove that a language *is not* regular.

Extra practice: Exercise 1.49 in the book.

## ROADMAP

**Proof strategy:** To prove that a language  $L$  is **not** regular,

- Consider an arbitrary positive integer  $p$
- Prove that  $p$  is not a pumping length for  $L$
- Conclude that  $L$  does not have *any* pumping length, and therefore it is not regular.

Note:  $\Sigma \in L$

Candidate counterexamples "strings that cannot be pumped"

Example:  $\Sigma = \{0, 1\}$ ,  $L = \{0^n 1^n \mid n \geq 0\}$ .

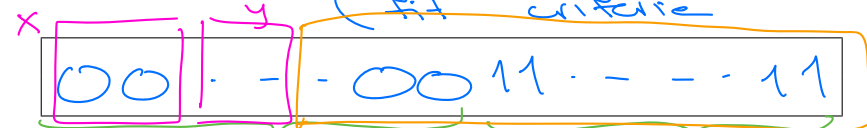
We will prove that  $L$  has no pumping length. Want to show  $L$  is nonregular.  
Fix  $p$  an arbitrary positive integer. List strings that are in  $L$  and have length greater than or equal to  $p$ :  
(fixed but unknown) WTS  $p$  is not a pumping length for  $L$ .

Pick  $s = 0^p 1^p$   
(depends on  $p$ )

Notice  $s \in L$  and we will show not pumpable relative to  $L$  and  $p$ .

Suppose  $s = xyz$  with  $|xy| \leq p$  and  $|y| > 0$ .

(arbitrary  $x, y, z$  so long as they fit criteria)

$S$  

$x = 0^k$   
 $y = 0^m$   
 $z = 0^r 1^p$

Then when  $i = 0$ ,  $xy^0z = x y^0 z = 0^k 0^r 1^p = 0^{p-m} 1^p \notin L$

because  $m > 0$  so  $p-m < p$ .

Since we've found a string  $s = 0^p 1^p$  that should be pumpable but isn't, we've shown  $p$  is not a pumping length for  $L$ . But  $p$  was arbitrary, so there is no pumping length for  $L$ . Hence,  $L$  is nonregular.

where  
 $k \geq 0$   
 $m > 0$   
 $k+m \leq p$   
 $r = p - (k+m)$

Working out example.

$$x y^0 z = \underbrace{0^k}_x \underbrace{0^r}_z 1^p = 0^{p-m} 1^p$$

$$x y^2 z = \underbrace{0^k}_x \underbrace{0^m}_y \underbrace{0^m}_y \underbrace{0^r}_z 1^p = 0^{p+m} 1^p$$

## Summary:

To prove that a language  $L$  is nonregular

① Consider arbitrary pos int  $p$ .

② Define a specific string  $s$  (parametrized by  $p$ )  
-  $s \in L$   
-  $|s| \geq p$ .

③ show  $s$  is not pumpable relative to  $p, L$

- Consider arbitrary  $x, y, z$  satisfying  $s = xyz, |y| > 0, |xy| \leq p$

- Prove  $\exists i (xy^i z \notin L)$  NEVER PICK  $i=1$ .

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

$0^p 0^p, 1^p 1^p, 0^p 11 0^p$

Fix  $p$  an arbitrary positive integer. List strings that are in  $L$  and have length greater than or equal to  $p$ :

Pick  $s = 0^p 11 0^p$

$(01)^p (10)^p$

Suppose  $s = xyz$  with  $|xy| \leq p$  and  $|y| > 0$ . There are  $k \geq 0, m > 0$ , let  $m = p - k - r$   
 then  $x = 0^k$   $y = 0^m$   $z = 0^r 11 0^p$

Then when  $i = 0$ ,  $xy^i z = 0^k 0^r 11 0^p = 0^{p-m} 11 0^p \notin L$   
 because either  $zz$  is odd length (when  $m$  is odd) and so not an even length palindrome hence not in  $L$ , or  $zz$  is even length and first half has two ones while second half has no ones, so can't be a palindrome.

**Example:**  $\Sigma = \{0, 1\}$ ,  $L = \{0^j 1^k \mid j \geq k \geq 0\}$ .

Fix  $p$  an arbitrary positive integer. List strings that are in  $L$  and have length greater than or equal to  $p$ :

Pick  $s =$

Suppose  $s = xyz$  with  $|xy| \leq p$  and  $|y| > 0$ .

Then when  $i =$  ,  $xy^i z =$

**Example:**  $\Sigma = \{0, 1\}$ ,  $L = \{0^n 1^m 0^n \mid m, n \geq 0\}$ .

Fix  $p$  an arbitrary positive integer. List strings that are in  $L$  and have length greater than or equal to  $p$ :

Pick  $s =$

Suppose  $s = xyz$  with  $|xy| \leq p$  and  $|y| > 0$ .

Then when  $i =$  ,  $xy^i z =$

*Extra practice:*

Language	$s \in L$	$s \notin L$	Is the language regular or nonregular?
$\{a^n b^n \mid 0 \leq n \leq 5\}$			
$\{b^n a^n \mid n \geq 2\}$			
$\{a^m b^n \mid 0 \leq m \leq n\}$			
$\{a^m b^n \mid m \geq n + 3, n \geq 0\}$			
$\{b^m a^n \mid m \geq 1, n \geq 3\}$			
$\{w \in \{a, b\}^* \mid w = w^R\}$			
$\{ww^R \mid w \in \{a, b\}^*\}$			

## Review: Week 4 Monday

Recall: Review quizzes based on class material are assigned each day. These quizzes will help you track and confirm your understanding of the concepts and examples we work in class. Quizzes can be submitted on Gradescope as many times (with no penalty) as you like until the quiz deadline: the three quizzes each week are all due on Friday (with no penalty late submission open until Sunday).

Please complete the review quiz questions on [Gradescope](#) about pumping lemma and nonregular sets.

**Pre class reading for next time:** Page 112

Wednesday

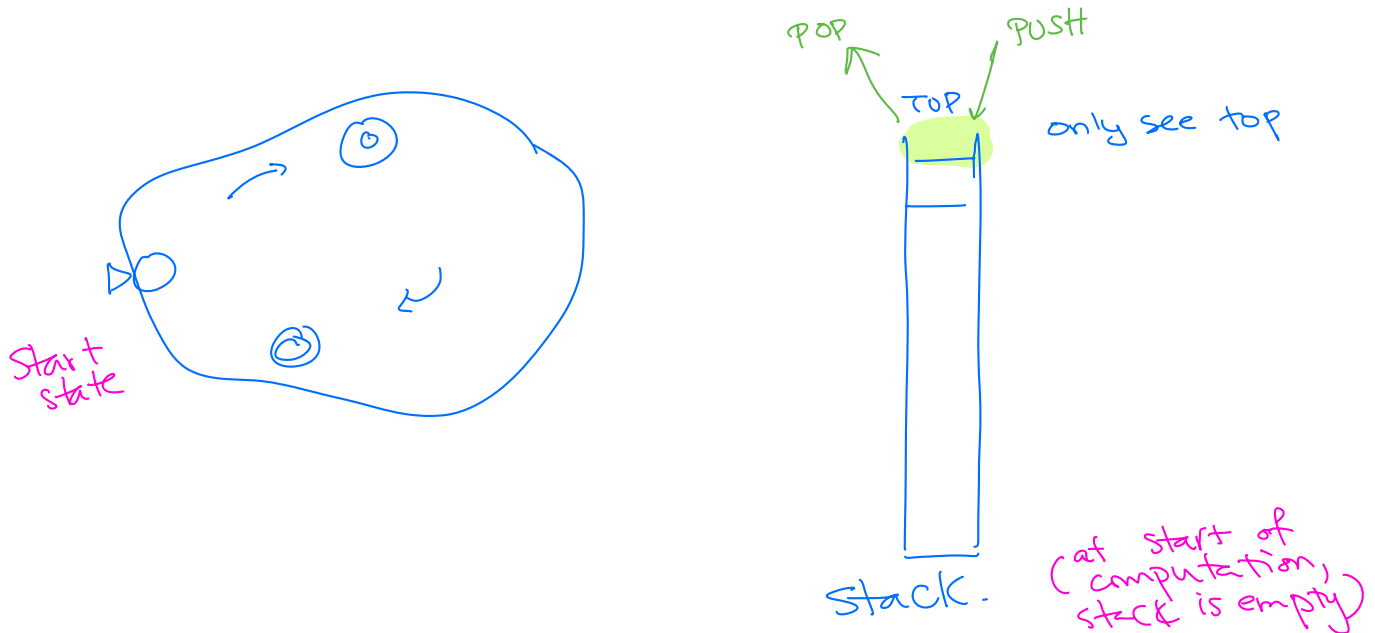
In the textbook: moving on to Chapter 2.

Regular sets are not the end of the story

- Many nice / simple / important sets are not regular
- Limitation of the finite-state automaton model: Can't "count", Can only remember finitely far into the past, Can't backtrack, Must make decisions in "real-time"
- We know actual computers are more powerful than this model...

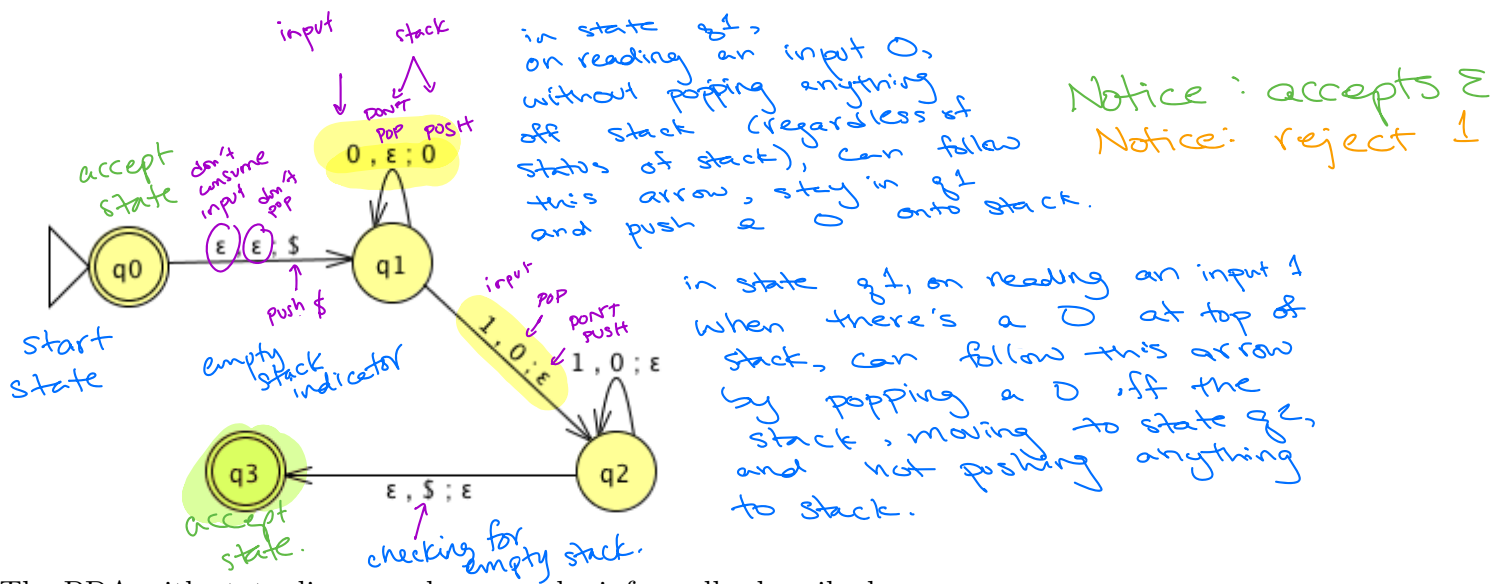
The **next** model of computation. Idea: allow some memory of unbounded size. How?

- sec 2.1
- To generalize regular expressions: **context-free grammars**
  - To generalize NFA: **Pushdown automata**, which is like an NFA with access to a stack: Number of states is fixed, number of entries in stack is unbounded. At each step (1) Transition to new state based on current state, letter read, and top letter of stack, then (2) (Possibly) push or pop a letter to (or from) top of stack. Accept a string iff there is some sequence of states and some sequence of stack contents which helps the PDA process the entire input string and ends in an accepting state.
- sec 2.2



Is there a PDA that recognizes the nonregular language  $\{0^n 1^n \mid n \geq 0\}$ ?

Idea: remember the # of 0s using the stack, match against the # of 1s.



The PDA with state diagram above can be informally described as:

$q_1$   
 $q_1 \rightarrow q_2$   
 $q_2 \rightarrow q_3$   
Read symbols from the input. As each 0 is read, push it onto the stack. As soon as 1s are seen, pop a 0 off the stack for each 1 read. If the stack becomes empty and we are at the end of the input string, accept the input. If the stack becomes empty and there are 1s left to read, or if 1s are finished while the stack still contains 0s, or if any 0s appear in the string following 1s, reject the input.

accepting

Trace the computation of this PDA on the input string 01.

start:  $q_0$   $\lfloor$  then  $q_1$   $\lfloor \$$  then  $q_1$   $\lfloor \$$  then  $q_2$   $\lfloor \$$  then  $q_3$   $\lfloor$   
Need to read 01 then Need to read 01 then Need to read 1 then don't have any more input to read then don't have any more input to read.

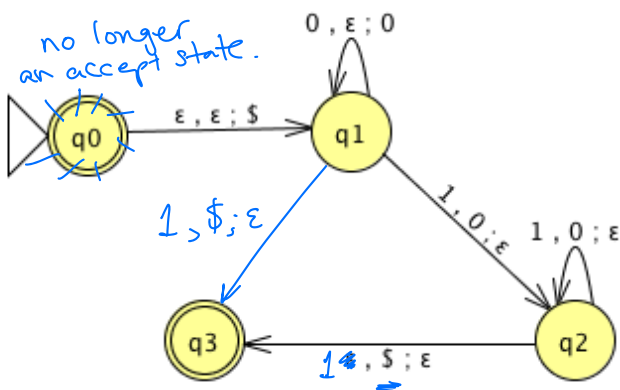
Trace the computation of this PDA on the input string 011.

one computation: gets stuck at  $q_0$   
another computation: similar to above for 01 but then gets stuck at  $q_3$  and can't read last 1 so not accepting.

A PDA recognizing the set  $\{0^n 1^{n+1} \mid n \geq 0\}$  can be informally described as:

★ Read symbols from the input. As each 0 is read, push it onto the stack. As soon as 1s are seen, pop a 0 off the stack for each 1 read. If the stack becomes empty and there is exactly one 1 left to read, read that 1 and accept the input. If the stack becomes empty and there are either zero or more than one 1s left to read, or if the 1s are finished while the stack still contains 0s, or if any 0s appear in the input following 1s, reject the input. ★

Modify the state diagram below to get a PDA that implements this description:



$n=0$  1  
 $n>0$  0...01-11

stack alphabet

**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,  $\Sigma$  is the input alphabet,  $\Gamma$  is the stack alphabet,

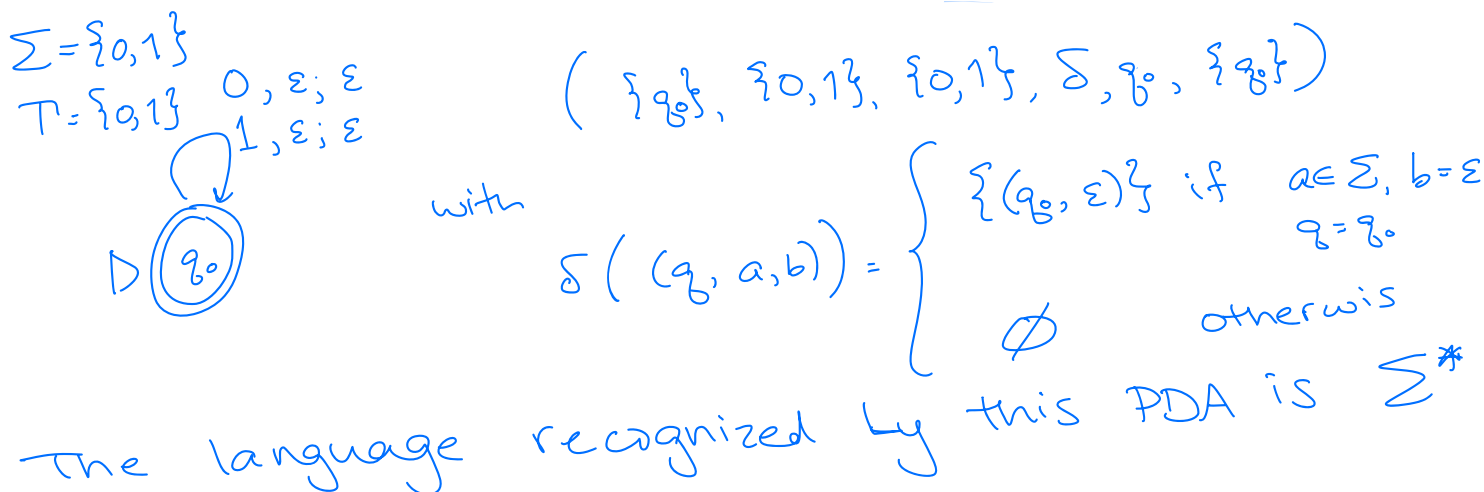
$$\delta : Q \times \Sigma_{\epsilon} \times \Gamma_{\epsilon} \rightarrow \mathcal{P}(Q \times \Gamma_{\epsilon})$$

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

$$\delta \left( \begin{matrix} \text{state} \\ (q, x, y) \end{matrix} \right) = \left\{ \begin{matrix} \text{state} & \text{stack} \\ (q', z) & \text{action} \end{matrix} \right\}$$

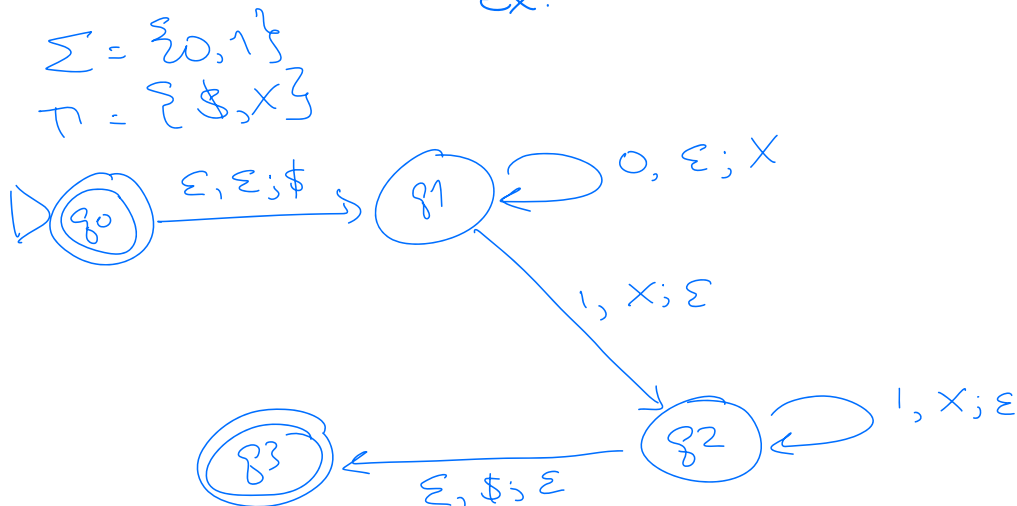
for each  $(q', z) \in \delta(q, x, y)$

Draw the state diagram and give the formal definition of a PDA with  $\Sigma = \Gamma$ .



Draw the state diagram and give the formal definition of a PDA with  $\Sigma \cap \Gamma = \emptyset$ .

ex.



*Extra practice:* Consider the state diagram of a PDA with input alphabet  $\Sigma$  and stack alphabet  $\Gamma$ .

Label	means
$a, b; c$ when $a \in \Sigma, b \in \Gamma, c \in \Gamma$	
$a, \varepsilon; c$ when $a \in \Sigma, c \in \Gamma$	
$a, b; \varepsilon$ when $a \in \Sigma, b \in \Gamma$	
$a, \varepsilon; \varepsilon$ when $a \in \Sigma$	

How does the meaning change if  $a$  is replaced by  $\varepsilon$ ?

*Note: alternate notation is to replace ; with  $\rightarrow$*

## Review: Week 4 Wednesday

Please complete the review quiz questions on [Gradescope](#) about PDA definitions.

**Pre class reading for next time:** Page 102



# Friday

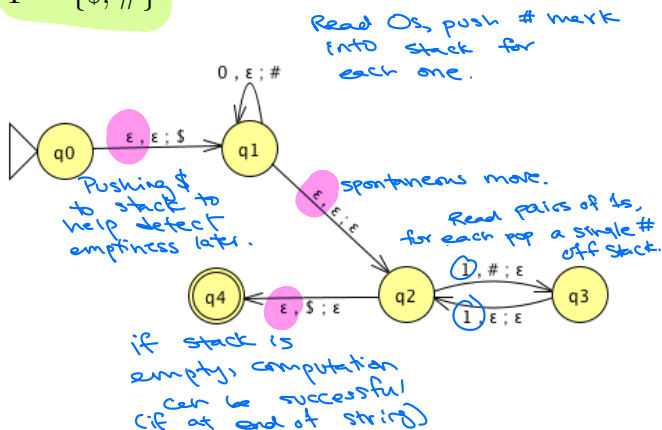
For the PDA state diagrams below,  $\Sigma = \{0, 1\}$ .

Mathematical description of language

State diagram of PDA recognizing language

$\{ \underbrace{0^n 1^{2n}}_{\text{string}} \mid n \geq 0 \}$   
 0s first, then 1s.

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



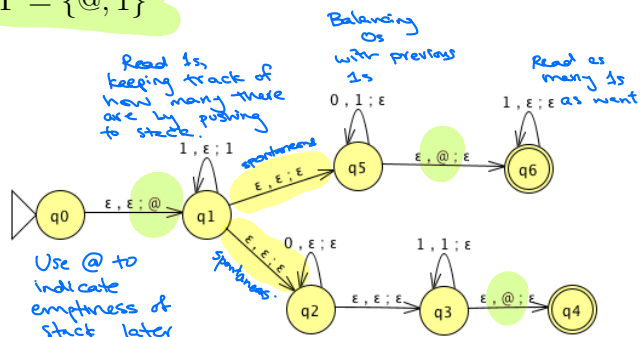
TOP PATH

$\{ \underbrace{1^n 0^n 1^m}_{\text{string}} \mid n \geq 0, m \geq 0 \}$

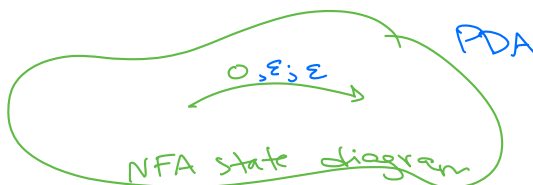
BOTTOM PATH.

$\{ 1^n 0^m 1^n \mid n \geq 0, m \geq 0 \}$ .

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



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



Big picture: represent DFA, NFAs, Regular expressions "simplest" languages Regular Lang.

PDA's are NFAs + stack represent regular languages and more...

*Big picture:* PDAs were 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.

Term	Typical symbol	Definition
<b>Context-free grammar</b> (CFG)	$G$	$G = (V, \Sigma, R, S)$
<b>Variables</b>	$V$	Finite set of symbols that represent phases in production pattern
<b>Terminals</b>	$\Sigma$	Alphabet of symbols of strings generated by CFG $V \cap \Sigma = \emptyset$
<b>Rules</b>	$R$	Each rule is $A \rightarrow u$ with $A \in V$ and $u \in (V \cup \Sigma)^*$
Start variable	$S$	Usually on LHS of first / topmost rule
<b>Derivation</b>	$S \Rightarrow \dots \Rightarrow w$	Sequence of substitutions in a CFG Start with <u>start variable</u> , apply <u>one rule</u> to <u>one occurrence</u> of a <u>variable</u> at a time
<b>Language generated by the CFG <math>G</math></b>	$L(G)$	$\{w \in \Sigma^* \mid \text{there is derivation in } G \text{ that ends in } w\} = \{w \in \Sigma^* \mid S \Rightarrow^* w\}$
<b>Context-free language</b>		A language that is the language generated by some CFG
Sipser pages 102-103		

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

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

$V \quad \Sigma$    
  $\uparrow$  start variable

- ①  $S \rightarrow 0S$
- ②  $S \rightarrow 0$

can replace an instance of  $S$  with  $0S$   
can replace an instance of  $S$  with  $0$

$L(G_1) = L(00^*)$  regular!

In  $L(G_1) \dots$

$S \xRightarrow{②} 0$  witnesses that  $0 \in L(G_1)$

$S \xRightarrow{①} 0S \xRightarrow{②} 00$  witnesses that  $00 \in L(G_1)$

Not in  $L(G_1) \dots$

$\epsilon$

because each application of a rule in this grammar adds at least one character so can't have zero characters

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

$$S \rightarrow 0S \mid 1S \mid \varepsilon$$

In  $L(G_2) \dots$

any string over  $\{0, 1\}$

idea: build strings one character at a time from left to right

~~Not in  $L(G_2) \dots$~~

$$L(G_2) = \{0, 1\}^* \text{ regular!}$$

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

$$\textcircled{1} S \rightarrow T1T1T1T$$

$$\textcircled{2} T \rightarrow 0T \mid 1T \mid \varepsilon$$

ensure at least three 1s

looks like  $G_2$   
i.e. can replace each  $T$  with arbitrary string.

In  $L(G_3) \dots$

1101 as witnessed by

$$S \xRightarrow{\textcircled{1}} T1T1T1T \xRightarrow{\textcircled{2}} 1T1T1T \xRightarrow{\textcircled{2}} 11T1T \xRightarrow{\textcircled{2}} 110T1T \xRightarrow{\textcircled{2}} 1101T \xRightarrow{\textcircled{2}} 1101$$

Not in  $L(G_3) \dots$

11 Since first production in each derivation in  $G_3$  adds three 1s and no characters can be erased.

$$L(G_3) = L(\Sigma^*1\Sigma^*1\Sigma^*1\Sigma^*) \text{ regular!}$$

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

*start variable* (pointing to  $A$ )

*redundant* (pointing to  $B$ )

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

(1) (2) (3) (4) (5)

In  $L(G_4) \dots$

$L(G_4)$  is the set of strings over  $\{0, 1\}$  that have odd length whose middle char is 1. *Not regular!*

Not in  $L(G_4) \dots$  any even length string

also  $\epsilon$  not in  $L(G_4)$ .

*Extra practice:* Is there a CFG  $G$  with  $L(G) = \emptyset$ ?

## Review: Week 4 Friday

Please complete the review quiz questions on [Gradescope](#) about PDA construction.