Week4 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 Σ 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

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

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

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

Informally:

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.

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.

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.

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

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 $i =$, $xy^i z =$

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

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 $i =$, $xy^iz =$

Example: $\Sigma = \{0, 1\}$, $L = \{0^j1^k \mid j \geq k \geq 0\}$.

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 $i =$, $xy^iz =$

Example: $\Sigma = \{0, 1\}$, $L = \{0^n1^m0^n \mid m, n \geq 0\}$.

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 $i =$, $xy^iz =$

Extra practice:

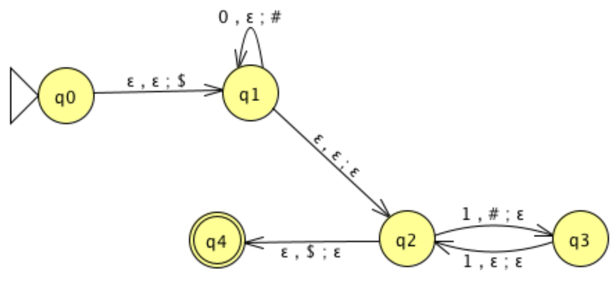
Language	$s \in L$	$s \notin L$	Is the language regular or nonregular?
$\{a^n b^n \mid 0 \leq n \leq 5\}$			
$\{b^n a^n \mid n \geq 2\}$			
$\{a^m b^n \mid 0 \leq m \leq n\}$			
$\{a^m b^n \mid m \geq n+3, n \geq 0\}$			
$\{b^m a^n \mid m \geq 1, n \geq 3\}$			
$\{w \in \{a,b\}^* \mid w = w^{\mathcal{R}}\}$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

Week4 friday

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

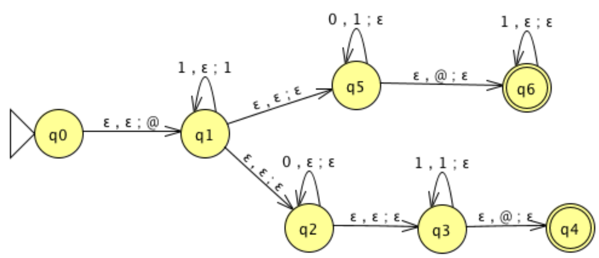
<table>
<tr><th>Mathematical description of language</th><th>State diagram of PDA recognizing language</th></tr>
<tr><td></td><td>

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



</td></tr>
<tr><td></td><td>

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



</td></tr>
<tr><td>

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

</td><td></td></tr>
</table>

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
Context-free grammar (CFG)	G	$G = (V, \Sigma, R, S)$
Variables	V	Finite set of symbols that represent phases in production pattern
Terminals	Σ	Alphabet of symbols of strings generated by CFG $V \cap \Sigma = \emptyset$
Rules	R	Each rule is $A \to u$ with $A \in V$ and $u \in (V \cup \Sigma)^*$
Start variable	S	Usually on LHS of first / topmost rule
Derivation		Sequence of substitutions in a CFG
	$S \implies \cdots \implies w$	Start with start variable, apply one rule to one occurrence of a variable at a time
Language generated by the CFG G	$L(G)$	$\{w \in \Sigma^* \mid$ there is derivation in G that ends in $w\} = \{w \in \Sigma^* \mid S \implies^* w\}$
Context-free language		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

$$S \to 0S$$
$$S \to 0$$

In $L(G_1)$ ...

Not in $L(G_1)$ ...

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

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

In $L(G_2)$ ...

Not in $L(G_2)$ ...

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

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

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 $L(G_4)$ ...

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

Week3 friday

Theorem: For an alphabet Σ, For each language L over Σ,

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

If (any, hence all) these conditions apply, L is called **regular**.

Prove or Disprove: There is some alphabet Σ for which there is some language recognized by an NFA but not by any DFA.

Prove or Disprove: There is some alphabet Σ for which there is some finite language not described by any regular expression over Σ.

Prove or Disprove: If a language is recognized by an NFA then the complement of this language is not recognized by any DFA.

Set	Cardinality
$\{0,1\}$	
$\{0,1\}^*$	
$\mathcal{P}(\{0,1\})$	
The set of all languages over $\{0,1\}$	
The set of all regular expressions over $\{0,1\}$	
The set of all regular languages over $\{0,1\}$	

Pumping Lemma (Sipser Theorem 1.70): If A is a regular language, then there is a number p (a *pumping length*) where, if s is any string in A of length at least p, then s may be divided into three pieces, $s = xyz$ such that

- $|y| > 0$
- for each $i \geq 0$, $xy^iz \in A$
- $|xy| \leq p$.

True or False: A pumping length for $A = \{0, 1\}^*$ is $p = 5$.

True or False: A pumping length for $A = \{1, 01, 001, 0001, 00001\}$ is $p = 4$.

True or False: A pumping length for $A = \{0^j1 \mid j \geq 0\}$ is $p = 3$.

True or False: For any language A, if p is a pumping length for A and $p' > p$, then p' is also a pumping length for A.