The Pumping Lemma

If you reach the end of a section early, please do not work on the next section - instead work on the questions at the end ("Side Questions" section).

1. Infinite regular languages

a. Given a DFA, what features of the machine would you look for to determine if it accepts infinitely many strings or only finitely many? That is, fill in the blank with something that refers only to the structure of the machine (e.g. features of the diagram like paths and cycles): A DFA M accepts infinitely many strings if and only if ____________________.

b. Given a DFA which recognizes an infinite language L, create a procedure which outputs a description of some infinite subset of L. (For example, if the machine actually recognizes $1 \cup (00)^*$, some valid outputs for your procedure include $(00)^*$ and $(0000)^*$. Describe your procedure in high-level steps/pseudocode - for example, you could have the step "First, find some path in the diagram from one accept state to another.")

The Pumping Lemma

[Theorem 1.70 in Sipser] If A is a regular language, then there is a number p (the pumping length) where if s is any string in A of length at least p, then s may be divided into three pieces, $s = xyz$, satisfying the following conditions:

1. for each $i \geq 0, xy^iz \in A$,
2. $|y| > 0$, and
3. $|xy| \leq p$.

2. Pumping Lemma + Regular Languages

a. Find the smallest possible pumping length for $11111 \cup (01)^*$.

b. Does the pumping lemma also apply to *finite* regular languages? For example, is there a valid pumping length for the language $\{001, 11010\}$?

c. Is there a way to use the pumping lemma as a tool to prove a language is regular? Is there a way to use it to prove a language is *not* regular?

3. Pumping Lemma + Nonregular Languages

a. Let $L = \{ww^{\mathcal{R}} \mid w \in \{0,1\}^*\}$. *(Recall $w^{\mathcal{R}}$ is the string w reversed.)* Prove that L is nonregular using the pumping lemma by filling in this template:

Proof: Fix p an arbitrary positive integer. We will show that p is not a pumping length for L by picking a string $s \in L$ with length greater than or equal to p, and showing s cannot be pumped. Let s be ____________________. *(Some values for s will be much easier to work with than others, so list a few first, try to pick an easy one, and come back and change it if it doesnt' work.)*

Suppose $s = xyz$ with $|xy| \leq p$ and $|y| > 0$. Then the following must be true about x and/or y:

But then when $i =$ ____________________, $xy^iz =$ ____________________, which is not in L because ____________________. So s cannot be pumped, which means p is not a pumping length for L. And since p was arbitrary, we've shown that there is *no* pumping length for L. So by the (contrapositive of the) pumping lemma, L is not regular.

b. Prove $L = \{0^j 1^k \mid j \geq k \geq 0\}$ is nonregular using the pumping lemma.

c. Prove $L = \overline{\{0^j 1^k \mid j \geq k \geq 0\}}$ is nonregular. (Hint: the pumping lemma is not the best tool for this.)

d. Prove $L = \{0^n 1^m 0^n \mid m, n \geq 0\}$ is nonregular.

4. Side Questions

a. In this class we are only considering models of computation which output a single bit: accept or reject. Why is this acceptable - don't our computers have to be able to give more interesting outputs? To answer this, imagine we want to build an algorithm (at a high level - don't worry about specific models like DFAs; you can imagine doing this in your favorite programming language instead) which computes some more general function $\Sigma^* \to \{0,1\}^*$ instead of just $\Sigma^* \to \{accept, reject\}$, but where almost all the work has to be done by one or two subroutines that can only output one bit. What would you want the inputs and outputs of those subroutines to be, and then how would you use them to build the main algorithm? Very informally, how much longer does your algorithm take to run than an algorithm which did not have to use this 1-bit-subroutine restriction?

b. Extra challenge question: We're going to need a model of computation which is more powerful than DFAs, i.e. a model that can handle some nonregular languages. Brainstorm a few models that seem a little more powerful than DFAs (for example, by coming up with "extra powers" like nondeterminism we could give them). Find at least one model which is *actually* more powerful than DFAs, as well as at least one model which *looks* like it might be more powerful (i.e. you gave it extra powers) but which is actually still the same power as DFAs. (Justify your answers.)