CPS 220 – Theory of Computation
Non-regular Languages

Warm up Problem

Problem #1.48 (p.90)

Let $\Sigma=\{0,1\}$ and let
D={w|w contains an equal number of occurrences of the substrings 01 and 10}.

Thus 101 $\in$ D because 101 contains a single 01 and a single 10, but 1010 $\notin$ D because 1010 contains two 10s and only one 01. Show that D is a regular language.

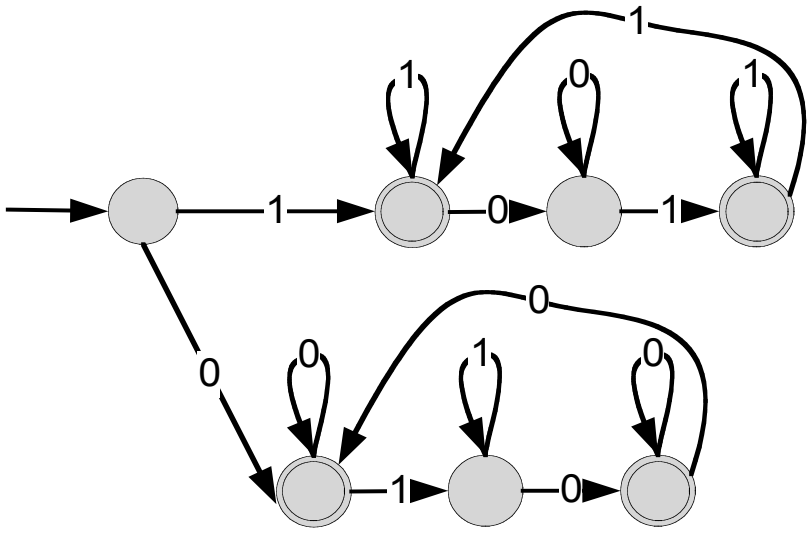
Solution:

This language is regular because it can be described by a regular expression and a FA (NFA):

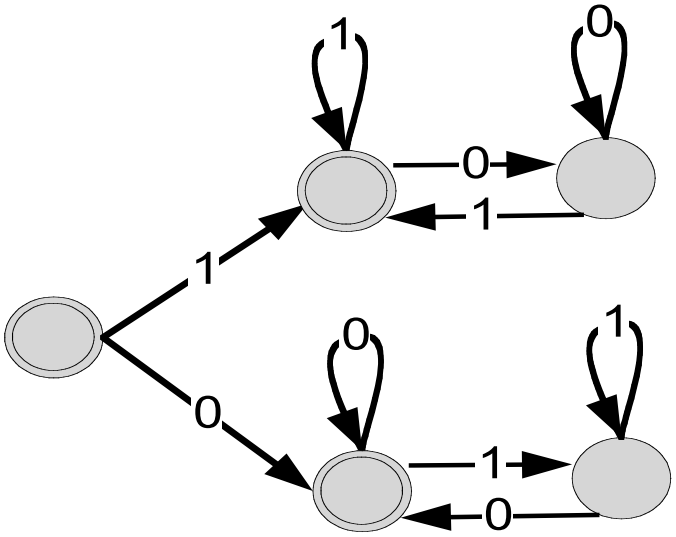
Regular Expression

$$\left(1^{+}0^{*}1^{+}\right)^{*}+\left(0^{+}1^{*}0^{+}\right)^{*}$$

NFA



DFA



Proving that a language is not regular—the Pumping Lemma

Consider the language B = { 0^n1^n | n ≥ 0 }. Is B regular?

If B is regular, then there is a DFA M recognizing B. That means, M accepts the string $0^{2003}1^{2003}$,
but rejects the string $0^{2003}1^{1999}$. How can M achieve that? As it reads the input, it has to remember
how many 0s it encountered so far. Then, when it starts reading 1s, it has to count the 1s and match
them with the number of 0s. However, a DFA by definition is finite, i.e., it has limited memory, which
is just the current state in which it is. To count it would require enough bits of memory to store a number…

How can we prove that a language is not regular? We will now demonstrate a method of doing that.

++

Theorem. The Pumping lemma. 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. $|y| > 0$
3. $|xy| \leq p$

Note: either x or z may be ε

++

So if **s** is long enough, there is a nonempty string y within s, which can be "pumped".

==========

In other words: If a language **A** is accepted by a DFA **M** with **q** states, then every string **s** in A

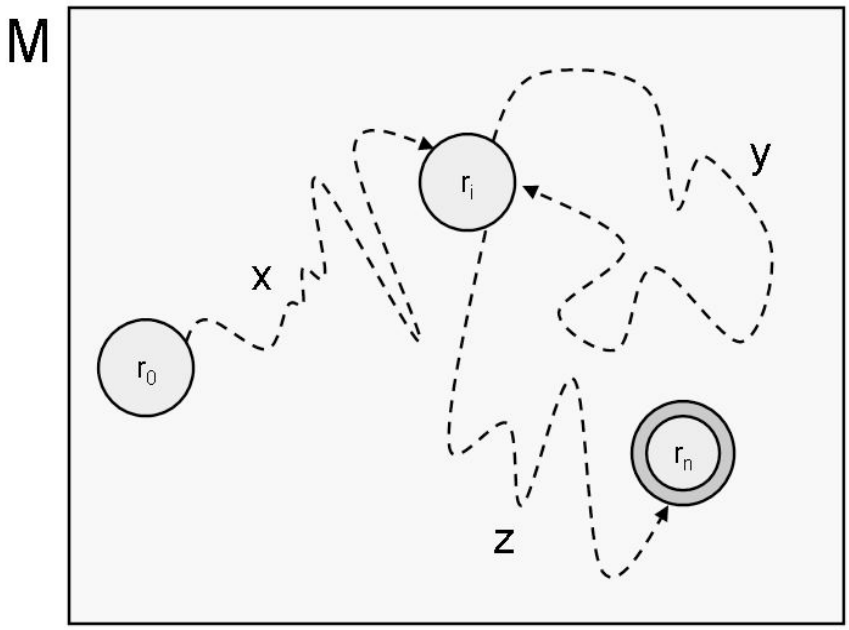
with |s|>=q can be written as **s = xyz** such that $y \neq \varepsilon \quad and \quad xy^*z \subseteq A$

==========

Proof. If A is regular, then there is a DFA M that accepts A.

Say that M has p states.
Let s be a string of length $n \geq p$. Let $r_1 \ldots r_{n+1}$ be the sequence of states of M, when processing s.
Since $n+1 > p$, at least one of M's states appears twice: $r_i = r_j$ for $j \neq i$.
Let $x = s_1 \ldots s_i$, $y = s_{i+1} \ldots s_j$, $z = s_{j+1} \ldots s_n$. Then, $xyz = s_1 \ldots s_n = s$, and $|y| > 0$.

Here is why xy^iz is accepted by M:



How can we maintain this condition - $|xy| \leq p$? We can make sure that this is true, by selecting the *smallest* i and j such that $r_i = r_j$.
The first p+1 states in the sequence must contain a repetition - therefore $|xy| \leq p$.

Pigeonhole principle - if **p** pigeons are placed into fewer than **p** holes, some hole has to have more than one pigeon in it**.**

Example of using the Pumping Lemma

Theorem. $B = \{ 0^n1^n \mid n \geq 0 \}$ is not regular.

Proof. Proof by contradiction.

Assume that language B is regular. [If B is regular, then there exists a constant p (the pumping length) such that the conditions of the pumping lemma hold.] Let p be the pumping length
and let's choose a string s that fits the conditions for our pumping lemma. Let $s = 0^p1^p$.
Because $s \in B$
and $|s| \geq p$ - then s can be broken into xyz where for any $i \geq 0$ the string $xy^iz \in B$.

Must consider 3 cases:
Case 1:

The substring y consists of only 0s. In this case the string xyyz has more 0s than 1s and therefore

xyyz $\notin$ B. This case is a contradiction.

Case 2:

The string y consists of only 1s. For the same basic reason, this case is a contradiction.

Case 3:

The string y consists of both 0s and 1s. In this case the string xyyz may have the same number of
1s and 0s - however it violates the basic structure of the language - where all 0 come before
all 1s. This case is a contradiction

Thus a contradiction is unavoidable.

Definition. The complement of a language A, $\overline{A} = \Sigma^* - A$, is all strings except those in A.

Theorem. Regular languages are closed under complementation.

Proof. Let A be a regular language. We will show that $\overline{A}$ is regular.

Since A is regular, a DFA M accepts it.

By turning all the accept states of M into non-accept, and all non-accept states into accept, every input in A ends up in a non-accept state, and every input not in A ends up in an accept state. Therefore, the modified M machine accepts $\overline{A}$.

Theorem. Regular languages are closed under intersection, ∩.

Proof. Recall that if A and B are two sets, A ∩ B is the set of all the common elements.

A simple logic fact that you can prove as an exercise is that:

$$A \cap B = \overline{\overline{A} \cup \overline{B}}$$

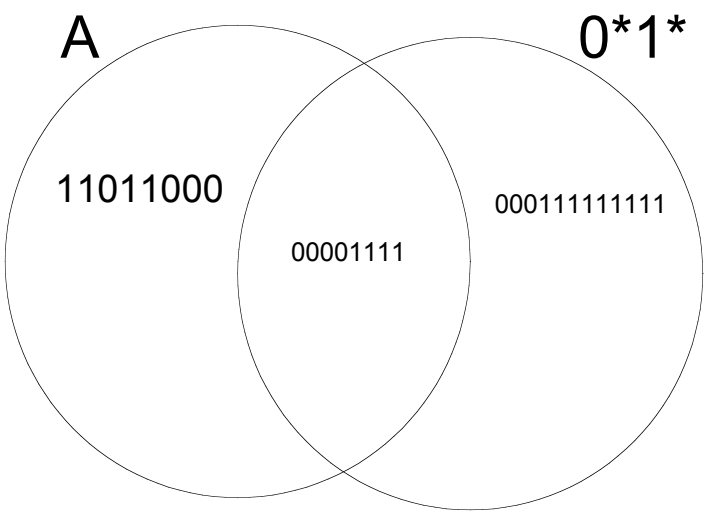
Then, the result follows by closure of regular languages under complement and union.

Theorem. A = { w | w has an equal number of 0s and 1s } is not regular.

Proof. Assume that A is regular, in order to arrive at a contradiction. (Very cool proof)

We know that 0^*1^* is regular.
Therefore A ∩ (0^*1^*) is regular, by closure of regular languages under ∩.

However, A ∩ (0^*1^*) = { 0^n1^n | n ≥ 0 }, which we proved not to be regular.
Therefore A cannot be regular.



Another pumping lemma proof:

Theorem. A = { 0^p | p is a prime} is not a regular language.

Proof. Assume that A is regular, in order to arrive at a contradiction.
Then A is accepted by a DFA M. Let s be the number of states in M. Consider a prime number p > s.
Note that $0^p \in A$ and $|0^p|=p > s$. Therefore, by the pumping lemma, 0^p can be written as $0^p = xyz$ such that $|y| > 0$ and $xy^iz \in A$

Let i = |x|+|z| and j = |y|. Then, the condition $xy^*z \in A$ means that, for any k>=0, i+kj is a prime.
In particular , when k = 0 it means that i is a prime. So i >=2. When k = i, this means that i(1 + j) is a prime.
However, since |y|>0 (not the empty string), we have j = |y|>=1 and so i(1+j) is not prime. This is a contradiction.

References:

Introduction to the Theory of Computation (2nd ed.) Michael Sipser
Problem Solving in Automata, Languages, and Complexity Ding-Zhu Du and Ker-I Ko