Lecture 11

Homework #11: 2.4.2, 2.4.3 (not b), 2.4.4, 2.4.5 a (can follow hint to use intersection, but not necessary), 2.4.7, 2.4.8

Hand in: 2.4.3 a, c, f
2.4.5 a
2.4.8 a, b, c

We now have a number of tools that allow us to show that a language is regular:

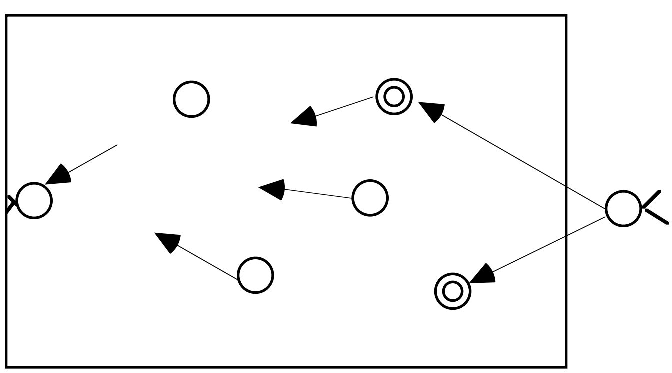
(1) We can do a direct construction of a DFA, NFA, or regular expression.
(2) We can construct one of the above out of simpler versions.
(3) We can refer to the Closure Theorem.

Some nice examples in the text. Here is another:

Show that L^R is regular, where $L^R = \{x: x^R \in L\}$.

Let $L = L(M)$, where $M = (K, \Sigma, \delta, s, F)$
[Note that I have chosen M to be deterministic]

Pictorially, let's think about what we can do:



Let's (1) reverse all the transitions.
(2) create a new start state with e-transitions to the final states of M.
(3) make the start state of M a final state of the new FA.

Define $M^R = (K \cup \{s^R\}, \Sigma, \Delta, s^R, \{s\})$,

$\Delta = \{(s^R,e,q): q \in F\} \cup \{(q,a,p): \delta(p,a) = q\}$

Now . . . how to show that a language is **not** regular

First, let's consider what makes a language regular:

(1) generally, simple periodicity

ab^*a

i.e., simple repetition of a pattern

(2) very limited memory

the classic example: $\{a^n b^n: n > 0\}$
is not regular - no way to remember the number of a's while you're counting b's.

Thm. **(Pumping Thm)** Let L be an **infinite** regular language. Then there are strings x, y, z such that $y \neq e$ and $x\, y^n\, z \in L$ for each $n \geq 0$.

Note:

The pumping theorem refers only to infinite languages. Remember that every finite language is regular.

The theorem will be useful for showing that languages are not regular - by showing that the strings x, y, z don't exist such that . . .

Proof. If L is a regular language, then it is accepted by some DFA M.

Suppose M has n states.

L is infinite, so it has some string w, such that $|w| > n$.

Let $|w| = m$ and $w = \sigma_1\sigma_2\sigma_2...\sigma_m$

Now consider the computation of M on w:

$(q0,\sigma_1\sigma_2\sigma_2\ldots\sigma_m) \ |\!\!— (q1,\sigma_2\sigma_2\ldots\sigma_m) \ . \ . \ . \ (q(m-1),\sigma_m)$
$|\!\!—(qm,e)$

$q0$ = the start state
$qm \in F$

Since $m \geq n$ and M has n states, there must be some
$qi = qj \qquad 0 \leq i < j \leq m$
by the Pigeonhole Principle.

This means that the string $\sigma_{i+1}\ldots\sigma_j$ starts at state qi and loops back to state qi.

But then you could remove the string and the resulting string would still be accepted

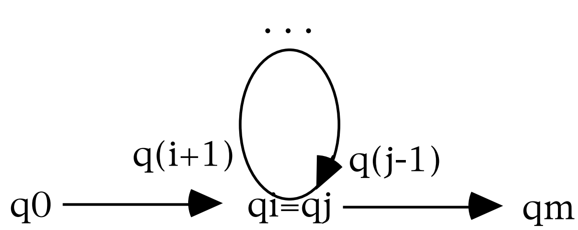
or

you could follow the cycle any number of times.

That is, M accepts

$$\sigma_1\ldots\sigma_i(\sigma_{i+1}\ldots\sigma_j)^n\sigma_{j+1}\ldots\sigma_m \quad n \geq 0$$

The following picture might help to visualize what's happening:



Then $\quad x = \sigma_1\ldots\sigma_i$
$y = \sigma_{i+1}\ldots\sigma_j$
$z = \sigma_{j+1}\ldots\sigma_m$

Note that y (the string being repeated) must be pretty close to the beginning of the string - or you would have started to re-use states earlier.

So we have a **stronger version of the Pumping Thm**:

Thm. **(Pumping Thm)** Let L be an **infinite** regular language. There is an integer $n \geq 1$ such that any string $w \in L$ with $|w| \geq n$ can be rewritten as $w = xyz$ such that $y \neq e$, $|xy| \leq n$, and $x y^i z \in L$ for each
$i \geq 0$.

Let $M = (K, \Sigma, \delta, s, F)$ be a DFA, and let w be any string in L(M) such that $|w| \geq |K| = n$.

Let $|w| = m$.

$w = \sigma_1\sigma_2 \ldots \sigma_m$, where each $\sigma_i \in \Sigma$.

Now consider the computation of M on w:

$(s,w) = (s,\sigma_1\sigma_2 \ldots \sigma_m) \vdash (q_1,\sigma_2 \ldots \sigma_m) \ldots (q_{m-1},\sigma_m) \vdash (q_m,e)$, $q_m \in F$.

And, in particular, let's focus on the first n steps of the computation:

$(s,\sigma_1\sigma_2 \ldots \sigma_n\sigma_{n+1} \ldots \sigma_m) \vdash (q_1,\sigma_2 \ldots \sigma_n\sigma_{n+1} \ldots \sigma_m) \ldots$
$(q_{n-1},\sigma_n\sigma_{n+1} \ldots \sigma_m) \vdash (q_n, \sigma_{n+1} \ldots \sigma_m)$

In order to process the first n symbols of w, n steps are required, and n+1 configurations are represented in the computation. Since $n+1 > |K|$, there must be some $q_i = q_j$ in the first n steps of the computation. ($i < j$). [By the Pigeonhole Principle]

Let $x = \sigma_1\sigma_2 \ldots \sigma_i$ (or x=e if q_i=s)
Let $y = \sigma_{i+1} \ldots \sigma_j$
Let $z = \sigma_{j+1} \ldots \sigma_m$ (z=e if j=m)

Then the above computation can be written:

$(s,w) = (s,xyz) \mid\!\!—^* (q,yz) \ldots (q,z) \mid\!\!—^* (q_m,e)$, $q=q_i=q_j$.

But $(q,yz) \mid\!\!—^* (q,z)$ iff $(q,y) \mid\!\!—^* (q,e)$
iff $(q,y^k) \mid\!\!—^* (q,e)$, $k \geq 0$
iff $(q,y^kz) \mid\!\!—^* (q,z)$

Also $(s,xyz) \mid\!\!—^* (q,yz)$ iff $(s,x) \mid\!\!—^* (q,e)$
iff $(s,xy^kz) \mid\!\!—^* (q,y^kz)$

So $(s,xy^kz) \mid\!\!—^* (q,y^kz) \mid\!\!—^* (q,z) \mid\!\!—^* (q_m,e)$, $k \geq 0$,
$q_m \in F$.

Now, back to the classic **example**.

Show that $L = \{a^nb^n: n \geq 0\}$ is not regular.

Assume the contrary. If it is regular, then there are strings x, y, z such that $y \neq e$ and $x\, y^n\, z \in L$.

Let's look at the different possibilities for what y might be, and show that $x\, y^n\, z \notin L$ for all the possibilities.

I. y is all a's.

$x = a^p$
$y = a^q$
$z = a^rb^s$ $\quad s = p+q+r$

but then $x\, y^n\, z = a^pa^{q(n)}a^rb^s$, which is clearly not in L.

II. y is all b's.

similar argument.

III. y is a^rb^s

but then y^n will have alternating a's and b's, so the resulting string will not be in L.

The argument is even easier to make by the stronger version of the Pumping Lemma.

Now what about $\{ww \mid w \in \{a,b\}^*\}$? Is it regular? Why or why not?