Regular Expressions

Def: A language (i.e., a set of strings) is regular if it is possible to construct a DFA that accepts the language (and rejects everything not in the language).

Note: 1) We can prove a language is regular by constructing a DFA (or NFA) for it.

2) We don't yet (in the class) know ways to prove a language is not regular— an example of such a language is

$$\{a^n b^n | n \geq 0\} = \{\lambda, ab, aabb, aaabbb, aaaaabbbb, \ldots\}$$
A regular expression is a way of describing a regular language by a single string.

Today and Friday:

1) Tell you what a regular expression is, and what language a reg. expr. stands for.
2) How we can construct a NFA that accepts the lang. described by a reg. exp.
3) How we can construct a reg. exp. given an NFA.
A regular expression on an alphabet \( \Sigma \) is a string on \( \Sigma \cup \{\emptyset, +, (, ), \ast \} \) that can be formed using the following rules.

1) a) Any character in \( \Sigma \) is a reg. expr.
   b) \( \lambda \) is a reg. expr.
   c) \( \emptyset \) is a reg. expr.

2) Given any two reg. expr. \( u, v \),
   a) \( u + v \) is a reg. expr.
   b) \( (u) \) is a reg. expr.
   c) \( uv \) is a reg. expr.
   d) \( u^* \) is a reg. expr.

E.g. \( a^*(b+ab)^*a+(ba)^*a \)
     is a reg. expr.
The language described by a regular expression is as follows:

1) a) $E = \varepsilon$, $L(E) = \{ \varepsilon \}$.
   b) $E = \lambda$, $L(E) = \{ \lambda \}$.
   c) $E = \emptyset$, $L(E) = \emptyset$.

2) a) $u + v$: $L(u + v) = L(u) \cup L(v)$.
   (i.e. A string $w$ matches $u + v$ if $w$ matches $u$ or $w$ matches $v$)
   b) $(u)$: $L((u)) = L(u)$
   c) $uv$: $L(\cdot uv) = L(u)L(v)$
   (A string $w$ matches $uv$ if there are strings $w', w''$ where $w = w'w''$, and $w'$ matches $u$, $w''$ matches $v$).
d) \( u^* \ L(u^*) = L(u)^* \)

A string \( w \) matches \( u^* \) if either \( w = \lambda \), or 
\[ w = w_1 \cdots w_k \] 
(for some \( k \)), where \( w_i \) matches \( u \) for all \( i \).

E.g. \( a^* \) means any number of \( a \)'s (i.e. \( a, aa, aaa, \ldots \) all match \( a^* \))

E.g. \( (ba)^* \) means any number of \( ba \)'s (i.e. \( ba, baba, babba, \ldots \) all match \( ba^* \))

E.g. \( (b+ab)^* \) is matched by
\[ \lambda, b, ab, bb, bbb, bababbab \]

Strings matching
\[ a^* (b+ab)^* a + (ba)^* a \]

babaqa, aaababbbbaba

Any string w \( \lambda \) no consec a's except at beginning?
Next goal: given a regular expr., build an NFA that accepts the strings matching the reg. exp. (and no others)

We do this by going through the possible methods for building a reg expr and see how to build a corresp. NFA.

1) a) \( E = l \)  
   b) \( E = \lambda \)  
   c) \( E = \emptyset \)

2) a) \( u + v \)
b) (u)

Machine for u

Machine for v

Connect all the final states of NFA for u to the initial state of NFA for v.

The final states of the new machine are final states in NFA for v (but final states in NFA for u are not final).
(Make a single new state, which is final, with a $\lambda$-transition to the init state of $u$ and $\lambda$-transitions from all the final states of $u$)
E.g. \( a*(b+ab)^*a+(ba)^*a \) has the \(\text{NFA}\)