MA/CSSE 474 Day 13 Summary

1. **Recap: For Every DFSM, there is an equivalent regular expression:**
   a. Number the states $q_1, \ldots, q_n$.
   b. Define $\mathbf{R_{ijk}}$ to be the set of all strings $x \in \Sigma^*$ such that
      $(q_i,x) \mid\!\!-_M^* (q_j, \varepsilon)$, and
      if $(q_i,y) \mid\!\!-_M^* (q_\ell, \varepsilon)$, for any prefix y of x (except y=ε and y=x), then $\ell \le k$
   c. That is, R_{ijk} is the set of all strings that take us from q_i to q_j without passing through any intermediate states numbered higher than k.
      i. In this case, "passing through" means both entering and leaving.
      ii. Note that either i or j (or both) may be greater than k.
2. **Formulas for $\mathbf{R_{ijk}}$:**
   a. Base cases (k = 0):
      i. If $i \neq j$, $R_{ij0} = \{a \in \Sigma : \delta(q_i, a) = q_j\}$
      ii. If $i = j$, $R_{ii0} = \{a \in \Sigma : \delta(q_i, a) = q_i\} \cup \{\varepsilon\}$
   b. Recursive case (k > 0):
      R_{ijk} is $R_{ij(k-1)} \cup R_{ik(k-1)}(R_{kk(k-1)})^*R_{kj(k-1)}$

3. In the **DFSMtoRegExp** example machine M from the slides, show how to get

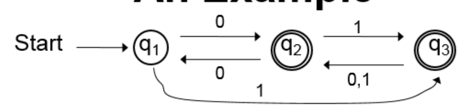
r_{221}

r_{132}

r_{123}

r_{133}

An Example



	k=0	k=1	k=2
r_{11k}	ε	ε	(00)*
r_{12k}	0	0	0(00)*
r_{13k}	1	1	0*1
r_{21k}	0	0	0(00)*
r_{22k}	ε	ε ∪ 00	(00)*
r_{23k}	1	1 ∪ 01	0*1
r_{31k}	∅	∅	(0 ∪ 1)(00)*0
r_{32k}	0 ∪ 1	0 ∪ 1	(0 ∪ 1)(00)*
r_{33k}	ε	ε	ε ∪(0 ∪ 1)0*1

A regular expression r such that L(R) = L(M)

4. Many programming languages include regular-expression processing. Examples we will look at are from Perl. Most other programming languages have very similar notation.
   a. The number of operators is extended significantly.
   b. The addition of variables enables the recognition of some languages that are *not* regular.
   c. Examples:
      i. `\b[A-Za-z0-9_%-]+@[A-Za-z0-9_%-]+(\.[A-Za-z]+){1,4}\b`
      ii. `^([ab]*)\1$`
      iii. `\b([A-Za-z]+)\s+\1\b`
      iv. `$text =~ s/\b([A-Za-z]+)\s+\1\b/\1/g;`
5. *Rich* pages 150-151 has a list of rules for simplifying REs, but you can also use "what language is this?".
6. The number of languages over any nonempty alphabet Σ is uncountable.
   a. Suppose we could enumerate all languages over Σ={a} as $L_0, L_1, \ldots$
   b. Consider $L_d = \{a^i : i \ge 0 \text{ and } a^i \notin L_i\}$. **Is there a j such that $L_d = L_j$?**

7. The number of regular languages over any nonempty alphabet Σ is countable. How do we know?

8. Are there more regular languages or non-regular ones?

9. Finite languages are regular, but not necessarily tractable.

10. What are the approaches that we have seen so far for showing that a language is/isn't regular?
    is: **is not:**

    Closure properties of regular languages: closed under union, concatenation, Kleene star, complement, difference, reverse, letter substitution (a function that maps each letter in Σ to a string in Σ').

11. In the homework you showed closure under intersection by construction. We can also do it using some other closure properties and DeMorgan's laws:

    Using closure backwards: If $L_1 \cap L_2$ is regular, what can we say about L_1 and L_2?

    If L_1 L_2 is regular, what can we say about L_1 and L_2?

12. **Showing that a language is *not* regular.**
    a. All non-regular languages are infinite.
    b. Where does "infinitenesss" come from in a regular language?
        i. Regular expression approach:
        ii. DFSM approach:
    c. If a DFSM has k states and a string whose length is at least k …

    d. This is the basis for the "pumping theorem" (also known as the "pumping lemma") for regular languages.

13. Pumping theorem: **Informally**: If *L* is regular, then every long string in *L* is "infinitely pumpable (in and out)".

Formally, if L is regular, then
$\exists k \geq 1$ such that
  ($\forall$ strings $w \in L$,
    ($|w| \geq k$ →
      ($\exists$ x, y, z ($w = xyz$,
        $|xy| \leq k$,
        $y \neq \varepsilon$, and
        $\forall q \geq 0$ (xy^qz is in L)))))

The contrapositive form:
(

)
)
)
)
)
→ L is not regular

14. Example: $\{a^nb^n: n \geq 0\}$