MA/CSSE 474 Day 05 Summary

1. Constructing one machine based on another machine. Consider the multiplication language:

   INTEGERPROD = {*w* of the form: ***<int*₁*>x<int*₂*>=<int*₃*>***, where each <*int*$_n$> is an encoding (decimal in this case) of an integer, and ***int*₃ == *int*₁ * *int*₂**}

   Given a multiplication procedure for integers, we can build a procedure that recognizes the INTEGERPROD language:

   This is easy; we did it on Friday.
   Suppose we have a machine M(x,y) that multiplies two integers.
   Given a string w, if it is not in the form <int1>*<int2>=<int3>, reject. If it is in that form,
   X = convertToInt(<int1>)
   Y = convertToInt(<int2>)
   Z = convertToInt(<int3>)
   If z = M(x,y) then accept. Else reject.

Given function R(w) that recognizes INTEGERPROD, build function Mult(m,n) that computes the product of two integers:

2. A *configuration* of a DFSM M is an element of $K \times \Sigma^*$.
   Contains all info needed to complete the computation.
   Initial configuration of M: (s_M, w), Where s_M is the start state of M.

3. The *yields-in-one-step* relation: $|\!-_M$:
   $(q, w)\ |\!-_M\ (q', w')$ iff
   - $w = a\, w'$ for some symbol $a \in \Sigma$, and
   - $\delta(q, a) = q'$

4. The *yields-in-zero-or-more-steps* relation: $|\!-_M{}^*$
   $|\!-_M{}^*$ is the reflexive, transitive closure of $|\!-_M$.

5. A ***computation*** by *M* is a finite sequence of configurations $C_0, C_1, \ldots, C_n$ for some $n \geq 0$ such that:
   - C_0 is an initial configuration,
   - C_n is of the form (q, ε), for some state $q \in K_M$,
   - $\forall i \in \{0, 1, \ldots, n\text{-}1\}\ (C_i\ |\!-_M\ C_{i+1})$

> **Recap - Definition of a DFSM**
>
> $M = (K, \Sigma, \delta, s, A)$, where:
>
> The D is for Deterministic
>
> K is a finite set of ***states***
> Σ is a (finite) ***alphabet***
> $s \in K$ is the ***initial state*** (a.k.a. start state)
> $A \subseteq K$ is the set of ***accepting states***
> $\delta: (K \times \Sigma) \rightarrow K$ is the ***transition function***
>
> Sometimes we will put an M subscript on K, Σ, δ, s, or A (for example, s_M), to indicate that this component is part of machine M.

6. A DFSM *M* ***accepts*** a string *w* iff $(s_M, w)\ |\!-_M{}^*\ (q, \varepsilon)$, for some $q \in A_M$
   rejects *w* iff $(s_M, w)\ |\!-_M{}^*\ (q, \varepsilon)$, for some $q \notin A_M$. The ***language accepted by*** *M*, denoted *L*(*M*), is the set of all strings accepted by *M*. A language is ***regular*** if it is L(M) for dome DFSM M.

7. ***Theorem:*** Every DFSM *M*, in configuration (q, w), halts after $|w|$ steps.

DFSM exercises:

8. $L = \{w \in \{0, 1\}^* : w$ has odd parity}. I.e. an odd number of 1's.

9. $L = \{w \in \{a, b\}^*$: no two consecutive characters in w are the same}.

10. $L = \{w \in \{a, b\}^* : \#_a(w) >= \#_b(w) \}$.

11. $L = \{w \in \{a, b\}^* : \forall x,y \in \{a, b\}^* \ (w=xy \rightarrow |\#_a(x) - \#_b(x)| <= 2) \}$ Vertical bars mean "absolute value" here.

12. DFSM programming techniques
    a. States remember info relevant to the goal of the machine (e.g., odd/even).
    b. Feel free to label states by "everything from Σ except …"
    c. Can make a DFSM for the negation of the desired condition, then ______________________________.
    d. A DFSM for the "missing letter language" is difficult to construct! Try it.

13. Nondeterminism. Machine may have "transition choices".
    a. If one choice leads to acceptance, accept
    b. Else if all choices lead to halting and rejecting, reject
    c. Else run forever

14. Why is nondeterminism necessary for any PDA that accepts PalEven?