Problem 4.15

Let $E = \{ \langle M \rangle \mid M$ is a DFA that accepts some string of the form $ww^R$ for $w \in \{0,1\}^* \}$. Show that $E$ is decidable.

- Consider the DFA $M$ with start state $q_0$ and set of accept states $F$. Let $q_1$ be the state that $M$ transitions to on input 1, and let $F'$ be the set of states that lead to $F$ on an input of 1. Clearly, $M$ accepts a palindrome beginning and ending with 1 iff the similar DFA (call it $M'$), with $q_1$ as the start state and $F'$ as the set of accept states, itself accepts a palindrome. This observation leads to a nice recursive algorithm.

- Palindrom-checker = “On input $\langle M \rangle$, where $M$ is a DFA:
  
  1. If $M$’s start state is included in its accept states, then $M$ generates a palindrome. Otherwise...
  2. Run the palindrom-checker on $M'$ (which is just $M$ with a different start state and set of accept states) to see whether it generates a palindrome beginning and ending with 1.
  3. If it doesn’t, run palindrom-checker on $M''$ (I hope you can guess what that is) to see if it generates a palindrome beginning and ending with 0.

- There’s a hitch in the above algorithm. It’s essentially performing a depth-first search, but there’s no guarantee that it will terminate. It may go into an infinite loop. Fortunately, there is a way around this. Consider the input to palindrom-checker. It differs on each call in the start state and the set of accepts states. Since there are $q$ possible start states, and $2^q$ possible sets of accept states, the total number of inputs to palindrom-checker is $q2^q$. As the algorithm proceeds, it simply keeps track of whether or not palindrom-checker has been called with those inputs before. This way the palindrom-checker subroutine should never be called more than $q2^q$ times, and the algorithm will terminate (ok, so I don’t promise efficiency or anything like that...).
Problem 5.17

Show that PCP is decidable over a unary alphabet.

- If the alphabet is unary, the dominoes only differ in the number of 1s that
each has on the top and bottom. This specific case of PCP is solved easily
by the following algorithm:

- M= “Given a collection of dominoes

  1. If some dominoe has the same number of 1s on top and bottom, there
     is a trivial match, so accept.

  2. If all the dominoes have more 1’s on top than on bottom, there is no
     possibility of a match, so reject. Likewise, if all the dominoes have
     less 1’s on top than on bottom, reject.

  3. Find one dominoe with more 1’s on top than on bottom (say a differ-
     ence of a 1s), and one dominoe with more 1s on bottom than on
     top (say a difference of b 1s). Choosing b of the first dominoe and a
     of the second should make an equal number of 1s on both top and
     bottom, and hence a match.

Problem 5.20

a. Let $A_{2DFA} = \{ \langle M, x \rangle | M \text{ is a 2DFA and } M \text{ accepts } x \}$. Show that $A_{2DFA}$
is decidable.

  - As with a Linear Bounded Automata, there is only a finite number of
    configurations that a 2DFA can be in on a given input. Specifically, consider a
    2DFA with q states on an input of size n. Each head can be in n different
    spots, so the total number of configurations for the machine is $g_n^2$. Thus,
    the algorithm that decides $A_{2DFA}$ is as follows:

  L= “On input $\langle M, w \rangle$ where $M$ is a 2DFA and $w$ is a string

    1. Simulate $M$ on $w$ for $g_n^2$ steps or until it halts.

    2. If $M$ has halted, accept if it has accepted, otherwise reject.

b. Let $E_{2DFA} = \{ \langle M, x \rangle | M \text{ is a 2DFA and } L(M) = \emptyset \}$. Show that $E_{2DFA}$ is
not decidable.

  - Again, as with Linear Bounded Automata, we prove this via a reduction
to $A_{TM}$ via computation histories. The idea is that we show how to solve
$A_{TM}$ by translating any instance of an $A_{TM}$ problem into an $E_{2DFA}$
problem. Given a machine $\langle M \rangle$ and an input $w$, we construct a 2DFA
that accepts all accepting computation histories for $\langle M \rangle$ on $w$ and rejects
everything else. Then we can decide whether $M$ accepts $w$ by deciding
whether the language of the 2DFA is empty. The proof is almost exactly
like Theorem 5.9, so I will omit some details.
• Presume, as in Theorem 5.9, that the computation history is presented as a single string, with configurations separated by # marks. Either one of the 2DFA’s heads can easily check that $C_1$ is indeed the start configuration, and that $C_l$ is an accepting configuration (these are within the realm of a regular DFA. They only require a linear scan, and no writing on the tape). The slightly more difficult part is checking whether each $C_{i+1}$ legally follows from $C_i$. The Linear Bounded Automota does so by zigzagging between corresponding positions of $C_i$ and $C_{i+1}$, and marking the current position on the tape to keep track of the head position. The 2DFA cannot write like the LBA, but it can utilize the fact that it has two heads.

• After checking $C_1$ and $C_l$ for validity, the first head moves back to the leftmost symbol of $C_1$, and the second head moves to the leftmost symbol of $C_2$. Together, they move right simultaneously, checking that the symbols they read are equivalent, except to the immediate right or left of the state symbol. For these few areas, validity checks are built-in to the 2DFA based on the transition function of $M$. Not that there are only a finite number of feasible configuration pairs where the pertinent parts of each configuration are valid transitions (for example $aqb$ goes to $acq$ or something like that). Also note that no marking of the tape is needed, because the two heads can traverse the tape simultaneously, and it’s unnecessary to mark the tape in order to “remember” location.

So to summarize the reduction, here’s how to build a machine $S$ which “solves” $A_{TM}$ based on the specious assumption that $E_{2DFA}$ is decidable.

S=“On input $\langle M, w \rangle$, where $M$ is a TM and $w$ is a string:

1. Construct the 2DFA $T$ from $M$ and $w$ as described above.
2. Run the decider for $E_{2DFA}$ on $T$.
3. If the decider rejects, accept; if the decide accepts, reject.