3.20) the only difference between a TM and a DFA/NFA is that the head of the TM can move to the left (DFA and NFA can only move right) and that the head can write after the input (there is no stack for DFA/NFA). If we can prove that a TM gives no extra computing power, we prove that it recognizes no more than regular languages. If we prove that TM gives no less computing power, we prove that it does recognize regular languages. Being able to move past the input and continue gives us no extra power. We are not able to store this information and we cannot return to the information in state, so it will not give us any more power. In addition, being able to move left, but not to the input string is exactly what a DFA gives. We are able to move between new states however we want but we are not allowed to take in more input.

4.12) Step 1: construct DFA that accepts every string with odd # of 1's. This is trivial (just a two state DFA with 1's going between and 0's on loop). Step 2: Construct new DFA D' that is the cross product of the odd # of 1's DFA and the DFA M. Step 3: check if L(D') ~= empty using L4. Step 4: if L4 accepts, accept, otherwise reject. This proves that A is decidable.

4.14) 1. Construct DFA that recognizes 1*. This is trivial. G can be converted into a PDA as described in class. Convert CFG G into PDA P, then use the fact that L(D) ^ L(P) is context free, so G cross P = P'. Convert P' into a CFG G'. L(G') != empty like previous problem all the L4 stuff and accept or reject. Then the lang is decidable

4.17) D' intersect D'' using cross product construction. Size of D' ^ D'' is A * B where A is the size of D' and B is the size of D''. Every string with length AB traverse the DFA and call it E. If string accepts both or rejects both check the other string, otherwise reject. If all strings up to length AB - 1, then L(D') == L(D''). No need to check past AB - 1 because pumping lemma. Every string longer than JK will go through each state at least one more time and all the stuff about looping.

4.27) 1. Construct grammar G accepts all strings with more 1s than 0s. Turn this into PDA. CFL intersect with regular language is a CFL, so P ^ M = P'. Using Cartesian product construction. Convert P' into CFG D'. L(G') ~= empty using L4 and all that. E is decidable.