CS 4820
Problem Set 8
Lillyan Pan

a. **Given a Turing machine M, and an input string x of length n, does M accept x in T(n) = 2n steps?**

Decidable, construct from M and x a universal machine U with input y. Construct y by concatenating x#x. We can simulate M on input y with a universal machine for 2n steps (as x is length n) and accept or reject depending on whether M has halted by that time.

b. **Given a Turing machine M, and an input string x of length n, does M accept x in T(n) = 2^n steps?**

Decidable, construct from M and x a universal machine U with input y. If the encodings of M and x are valid, the machine U does a step-by-step simulation of M. We can simulate M on x with a universal machine for 2^n steps (using a counting tape in binary on a separate track) accept or reject depending on whether M has halted by that time. Note, for n positions on the tape, each position represents a binary digit. With each step we add 1 in binary which is reflected on the counting tape, hence move left to the next position when there is overflow. We know if we have reached 2^n steps if after n positions there is overflow.

c. **Given a Turing machine M, does M accept the empty input (that is all tape positions blank, accept for the left end marker.)**

Undecidable. As given in Kozen's Notes, suppose we could decide whether a given machine accepts ε. We could then decide the halting problem as follows. Given a Turing machine M and a string x, we with to determine whether M halts on x. Construct from M and x a new machine M' that does the following on input y:

1. erases its input y
2. writes x on its tape (M' has x and description of M hard-wired in its finite control)
3. runs M on input x
4. accepts if M halts on x.

Note for all inputs y, M' has the same behavior: if M halts on x, then M' accepts its input y and if M does not halt on x, then M' does not halt on y, hence does not accept y. Thus,

$$L(M') = \begin{cases} \Sigma^* \text{ } if \text{ } M \text{ } halts \text{ } on \text{ } x \\ \emptyset \text{ } if \text{ } M \text{ } does \text{ } not \text{ } halt \text{ } on \text{ } x. \end{cases}$$

Thus, given M and x, and the constructed M', ask whether M' accepts ε. Based on our construction, M' answer YES if and only if M halts on x. Since we know the halting problem is undecidable, it must also be undecidable whether a given machine accepts the empty input.

d. **Given a Turing machine M, does M accept at least one input x, that has a 2 in some position?**

Undecidable. Assume there exists a universal machine U that takes in M and outputs YES if there exists a machine M such that M accepts x, where x has a 2 in some position. Construct machine U' such that U' solves the Halting Problem. Note, input into the Halting Problem machine must be in binary. Note U' takes in M and string x and decides if M halts on x. Construct M' such that if U(M') says YES then M halts on x. We want that if M halts on x, then the language of M' contains a string with a 2 in it. Hence, construct from M and x a new machine M' that does the following on input y:

1. Check if y = (2 prepended to some string y')
2. If YES, simulate M on x and if M halts then M' will accept
3. If NO, then loop forever.

Thus, M' accepts at least one y that has a 2 in some position, if and only if M halts on x. Note, if M' accepts y, that has a 2 in some position, then y = (2 prepended to y') and M halts on x. If M halts on x, then based on our construction the only way M is simulated on x is if M' accepts at least one y that has a 2 in some position. Therefore, in the language of this Turing Machine, there exists an input with a 2 by our construction, hence U' solves the Halting Problem. Since we know the halting problem is undecidable, it must also be undecidable whether a given machine accepts at least one input x, that has a 2 in some position.

e. **Given a Turing machine M, and input x, does M accept input x, without writing any symbols, or leaving the initial segment of the tape where input x is written.**

Decidable. If M never moves more than n tape cells away from the left endmarker, then it will either halt or loop in such a way that we can detect the looping after a finite time. This is because if M has k states and never moves more than n tape cells away from the left endmarker. Taking into account the left endmarker, there are only (n+1)k configurations it could possible ever be in. If it runs for any longer then that without moving more than n tape cells from the left endmarker, then it must be in a loop because it must have repeated a configuration. This can be detected by a machine that simulates M, counting the number of steps M takes on a separate track and declaring M to be in a loop if the bound of (n+1)k steps is ever exceeded.