Max Shi
Professor Bhatt
CS 334
November 22, 2019
I pledge my honor that I have abided by the Stevens Honor System.

Problem Set 9

1.
    a. Let there be three languages A,B,C such that $A \leq_m B$ and $B \leq_m C$. Therefore, there must also exist computable functions f and g s.t. for all strings w, w is in A iff f(w) is in B, and for all strings v, v is in B iff g(v) is in C. Thus, we can combine the two functions to create the function g(f(x)) = h(x). We can then create a TM that takes input x and simulates the function f on x, then takes the output of that function (we'll call it y) and simulates g on y. Let this output be z. This z is equivalent to h(x), and thus, h(x) is computable. Furthermore, through the TM, because x is in A iff f(x) is in B, and y is in B iff g(y) is in C, then x is in A iff g(f(x)) is in C, and g(f(x)) = h(x). Thus, $A \leq_m C$.
    b. If $A \leq_m \bar{A}$, this means there is a computable function that for all strings w, w is in A iff f(w) is in $\bar{A}$. Because this function is computable, it has an inverse computable function that we will call g. Thus we can reverse this relation: for all strings w, w is in $\bar{A}$ iff g(w) is in A. Therefore, $\bar{A} \leq_m A$. Because A is Turing recognizable, then $\bar{A}$ is also Turing recognizable, by the theorem is in the book, and by the principles of reducibility. Therefore, because A and $\bar{A}$ are both Turing recognizable, this implies that a TM will halt on all strings in A and all strings in $\bar{A}$, making A Turing decidable.
2. Assume that there exists a TM D is a TM that takes two TMs as input and decides $DISJOINT_{TM}$. We will reduce this TM to make a decidable E_{TM}, (we'll call it A), the TM whose language is all TMs with an empty language, to prove that DISJOINT is undecidable, as we have proven before that E_{TM} is undecidable.
   Let TM A accept as input a TM M. Then, let I be a TM with the language of all strings, i.e. L(I) = Σ^*. Therefore, the language of this TM is not the empty set. Then, let A run D(M, I). If D accepts, then let A accept. If D rejects, then let A reject. By doing this, if the language of M is empty, the language will be completely disjoint from L(I), as no elements overlap as there are none in L(M), and therefore, D will accept, A will accept, and A reproduces E_{TM} in this case. If M contains at least one string in this language, D will reject, as any string will overlap with L(I), and the machines will not have disjoint languages. Therefore, D rejects, A rejects, and A reproduces E_{TM} again. Thus, we have created a decidable E_{TM}, which is impossible, thus D cannot exist and DISJOINT is undecidable.
3. This language is in P if there is an algorithm that runs in polynomial time to find a solution. Thus, we can adopt the following algorithm:
   Begin by choosing a vertex and remembering that vertex. Then, visit every adjacent vertex to that first vertex. Then, for each adjacent vertex found, visit all adjacent vertices that does not include the remembered node. Then, if any of those vertices connects back to the first vertex, then that graph contains a triangle. If at any point during this algorithm there are no valid adjacent vertices, we can continue to the next vertex in the graph. We will repeat this process for every vertex in the graph. If this condition is never met, then the graph does not contain a

triangle.

This algorithm is a breadth first search of length 3, which has a maximum runtime of all vertices and all edges in the graph. We also run this for each vertex in the graph, giving us a maximum possible runtime of V * (V+E). This simplifies to V^2 + VE, which is polynomial. Because each edge and each vertex is represented in a finite amount of data in the input bits n, this represents finding the solution to the problem in polynomial time with regards to the length of the data.

4.

a. My algorithm will be for each x_i for 1 <= i <= n, we will append this onto the front of the formula, like so: $x_i \wedge \varphi(x_1, x_2, \ldots x_n)$. This new formula p_i is also in conjunctive normal form, and thus, we can give this to the genie. If the genie says this new formula p_i is not satisfiable, then we know that x_i has to be false, as p_i being satisfiable would guarantee that x_i can be set to true. It follows that if the genie says p_i is satisfiable, then x_i can be set to true. We continue doing this for all x's, and therefore, this algorithm runs in polynomial time, as each iteration happens in 3 steps each time (append, check with genie, set true/false), and runs once for all n, thus happening in time linear to n.
b. The maximum number of queries is n. As a matter of fact, it will always query n times.
c. Because the formula is already satisfiable, we know there exists at least one truth assignment that satisfies it. On each run of the algorithm, we always come out with an assignment for each variable, whether it is true or false. Finally, each variable assignment will be valid: if $x_i \wedge \varphi(x_1, x_2, \ldots x_n)$ is satisfiable, then we know that the first term, x_i can be set to true, as there are no other options to satisfy this clause. Otherwise, we know that x_i cannot be true, and it must be false, as we know the formula is satisfiable and there are no other options if true is not an option. Therefore, we create a valid truth assignment for all variables.