1. We know A ≤m B, this implies the polynomial time computable function where for all x, x ϵ A if and only if f(x) ϵ B. and given we have transitive property B ≤m similarity we also have polynomial time computable function g such that for all y, y ϵ B if and only if g(y) ϵ C. Therefore we can prove A ≤m C by constructing a polynomial time computable function h such that for all x, x ϵ A if and only if H(x) ϵ C.
2. Input: a graph G

Guess: s: a path of nodes in G, denoted α, such that |α| ≤ |

Check: 1. α is a path

2. α passes every node in G exactly once.

Suppose this guess is nondeterministic, and the check must be a deterministic, polynomial time algorithm so that the question is NP.

Then: If (check() == true)

Return true.

Elses:

Crash;

Thus, the problem is in NP.

3. Proof by implication. Base of argument: if B is a subset of A, then if A then B. Here we know P is a subset of NP. To show the problem is NP, we can show the problem is P and deduce that it is also NP. Step 1: run the SCC algorithm on the given graph G. Step 2: to treat the SCC sets that the algorithm returns as big nodes. Step 3: ensure that the graph forms a straight line Then we can deduce there is a path G such that G is covered. The algorithm is a polynomial time linear algorithm, which means this problem is in P. Therefore, we can deduce the problem is NP.
4. We need to create a new large boolean circuit that contains both C1 and C2 circuits in order to solve this problem. Let the new circuit C' be equal to: C' = (C1 ^ ~C2) V (~C1 ^ C2) The circuit C' is equal to the logical and of C1 and negation of C2 logical OR'd with the negation of C1 logical and'd with C2. With this new circuit, we will run a deterministic polynomial time algorithm that will check to see if C' is satisfiable. If the

algorithm returns that C’ is satisfiable, that means C1 and C2 are not equivalent, else, they are equivalent.