Is NP = P

- Are all NPC problems reducible to P, thus making NP=P?
- There are also problems which have been proven to be in NP but have not been proven to be in NPC. Solving this would mean that NP≠P.
  - Example: Finding if a number is a prime
Imperfect solutions to NPC

• There are algorithms to the NPC problems which yield an imperfect solution.
  – There is a solution to the TSP which guarantees (mathematically) a tour no longer than 1.5 times the optimal tour and runs in cubic time.
Noncomputability and undecidability

- It is not true that given enough time and enough computing resources, we can write software that does anything we want.
- Given enough time, money and brains some algorithms are simply non-computable.
- When given a finite number of inputs, we can simply write a program which says ‘yes’ when the input is a particular element.
- The problem arises with infinite input.
Word Correspondence Problem

• The word correspondence problem asks if given two groups of words over a finite alphabet, is it possible to concatenate corresponding words in each group to form the same word?

• This problem is undecidable since the number of words that need to be chosen is unbounded.
Syntactical Equivalence Problem

- It is impossible to write an algorithm such that one can prove that two different sets of syntactic rules describe the same language.
- This is important for compilers who might want to replace their current syntactic rule representation with another one.
The halting problem

• Is it possible to write a program X which given a description of another program Y and an input can decide if Y terminates?
• This is important for program verification, especially for critical systems.
• But how do we do it, how do we decide when to stop X and give up, since Y would have terminated in the next step !!
Proving undecidability

- The method is similar to proving that problems are in NPC.
- We just build reductions from a problem P to a known undecidable problem.
- We just assume that there exists an oracle which is able to solve an undecidable problem, then we take the input to problem P and convert it to a format that this oracle understands.
- Then we simply output the result from the oracle to problem P.
- Yet we need to prove a single problem to be undecidable.
Proving the halting problem

• We will prove that there is no program in language L that upon accepting any pair <R,X> where
  – R is a program in language L, and
  – X is a string of symbols
  will terminate after some finite time and output ‘yes’ if R halts and ‘no’ if R does not halt on X
• This program can take any finite amount of time and as much memory space as it requires.
• The method of proof is called ‘proof by contradiction’ whereby we assume that such a program, Q, exists and then will show that this is a contradiction.
Halting problem (cont.)

- Construct a program, S, which merely outputs 2 copies of its single input program W.
- Connect S such that Q will receive as its input program W and as its input string W also.
- Now since Q exists (assumption), the result will be yes or no if W terminates given itself as input.
  - W is not very strange, after all a compiler program can also compile itself !!
Halting Problem (cont.)

• Finally S will take the output of Q and if Q outputs ‘yes’ it will enter an infinite loop, and if it says ‘no’ S will terminate.
• Now make the input program $W = S$, thus Q operates on inputs S and S.
• A contradiction occurs
  – Assume that S when given itself terminates, but when Q says yes, S will not terminate!!
  – If S does not terminate, then Q will say no and S will terminate !!
Diagonalization Method

- Some people find the previous proof difficult, so here is an alternate proof.
- Build a matrix such that each row represents programs and the columns the input.
- Now each element will say ‘yes’ if the program halts on an input and ‘no’ if not.
- Imagine a program S which is the negative of the diagonal of the matrix
  - Thus if the j’th program halts on the j’th input, S when run on the j’th input will not halt and vice versa
Diagonalization Method (cont.)

- Such a program $S$ leads to a contradiction since it behaves as the negative of the diagonal.
- But if $S$ is the i’th program, what shall it do on the i’th input.
  - If it halts, it should not halt, and
  - If it does not halt, it should halt !!
Undecidable problems

• It turns out that some undecidable problems are polynomially equivalent.
• Thus the halting problem can be polynomially reduced to the halting problem.
• Yet there are others which are not and which are more undecidable than others !!