Decidability

Recall that a function f is computable in a particular domain if there exists a Turing machine that computes the value of f for all arguments in its domain. Thus, a function f is uncomputable if no such Turing machine exists.

This concept of computability works very well with problems such as the optimization problems treated throughout this class. When we transform the problems from being optimization problems to being decision problems, we modify the terminology as well. Specifically, a computable decision problem is now referred to as a decidable problem, and an uncomputable decision problem is referred to as an undecidable problem. If we have a decision problem where we can determine "Yes" but not "No" (or vice-versa), then we have a semi-decidable problem.

A classic example of a semi-decidable decision problem is the following:

> Suppose we have a description of a Turing machine with input x and program M that we provide as input to another Turing machine. Can we use that second Turing machine to determine whether M, when started in state q_0 will ultimately reach either state q_Y or q_N? In other words, does there exist a Turing machine that can determine any other Turning machine will halt? This is called the halting problem, and it has been proven that no such Turing machine exists.