Problems, Computability, and Decidability

A function $f$ with domain $A$ is computable iff a TM with any input $a \in A$ halts with $f(a)$ on its tape.

problem = compute function for whole domain
instance = compute function for a single input

If a function $f$ is computable, then some TM solves every instance.

If a function $f$ is uncomputable, then a TM might solve some instances, but not all of them.

A decision problem is when $f$’s range is $\{\text{yes, no}\}$ (or $\{1, 0\}$ or any two-element set).

A decision problem $f$ is decidable iff $f$ is computable. Otherwise, $f$ is undecidable.

Note that a decidable problem corresponds to a recursive language.

The halting problem is to determine whether TM $M_i$ halts on input $x_j$. 
Undecidability of the Halting Problem

Theorem: The halting problem is undecidable.

Proof by contradiction:
Assume that $M_H$ solves the halting problem.
Construct $M'$ as follows.
On input $x_i$, $M'$ runs $M_H$ on $M_i$ and $x_i$.
If $M_H$ says yes, $M'$ enters an infinite loop.
If $M_H$ says no, $M'$ halts.

$M'$ is machine $M_n$ for some integer $n$.
When $M_n$ runs on $x_n$, $M_H$ runs on $x_n$ and $M_n$.
If $M_H$ says yes, then $M_n$ does not halt.
If $M_H$ says no, then $M_n$ halts.
$M_H$ is wrong, which is a contradiction.
Therefore, no TM solves the halting problem.
Reductions

Problem $P$ is undecidable if a solution for problem $P$ can be used to solve the halting problem. This is a reduction of $P$ to the halting problem.

Suppose $M_P$ is a TM for problem $P$. Try to reduce $P$ to halting problem by:

Examples:

Problem: Does $M$ halt on blank input?
Reduction: Map $M, x$ to a TM that writes $x$ and runs $M$.

Problem: Does $M$ enter state $q$?
Reduction: Map $M, x$ to a TM that enters $q$ when $M$ halts on $x$. 