

Theory of computation

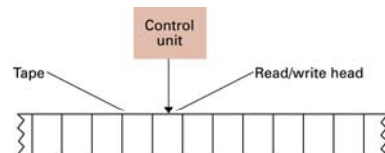
Can computer do anything?

- ▶ Which computer are we talking about?
 - ▶ Supercomputers, Cloud, PC, iPhone, quantum computer, ...
 - ▶ We will use an abstract model (Turing machine).
- ▶ There are two important problems
 - ▶ Can computer solve all kinds of problems?
 - ▶ There are some problems unsolvable by today's machines or any future algorithmic machine.
 - Ex: The halting problem
 - ▶ Which problems can be solved efficiently by computers?
 - ▶ There are problems too complex to be solvable in practice.
 - The P-NP classification



Turing machines

- ▶ Introduced by Alan M. Turing in 1936
- ▶ Conceptual device that consists
 - ▶ A control unit that can read and write symbols on a tape
 - ▶ The tape extends indefinitely at both ends
 - ▶ Each cell on the tape can store a finite set of symbols

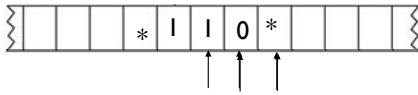


- ▶ At any time, it must be in one of a finite number of states
 - ▶ Computation starts in the start state, and stops in the halt state

Turing machine operation

- ▶ Inputs at each step
 - ▶ State
 - ▶ Value at current tape position
- ▶ Actions at each step
 - ▶ Write a value at current tape position
 - ▶ Move read/write head
 - ▶ Change state

An example of a Turing machine



Current state	Current cell content	Value to write	Direction to move	New state to enter
START	*	*	Left	ADD
ADD	0	1	Right	RETURN
ADD	1	0	Left	CARRY
ADD	*	*	Right	HALT
CARRY	0	1	Right	RETURN
CARRY	1	0	Left	CARRY
CARRY	*	1	Left	OVERFLOW
OVERFLOW	*	*	Right	RETURN
RETURN	0	0	Right	RETURN
RETURN	1	1	Right	RETURN
RETURN	*	*	No move	HALT

Church-Turing thesis

- ▶ Church-Turing thesis: a Turing machine can compute any **computable function**.
 - ▶ Not proven, but generally accepted
- ▶ Function
 - ▶ A mapping of a set of input values and a set of output values.
 - ▶ Each input is assigned a single output
- ▶ Computing a function
 - ▶ Determining the output value associated with a given input
- ▶ Noncomputable function
 - ▶ A function that cannot be computed by any algorithm

Which problems cannot be solved by TM?

- ▶ Any problem that can be solved on a computer has a solution expressed in some language
 - ▶ Any programming language comprising the features of this language can surely express a solution to the problem
- ▶ The halting problem: for a given program (encoded as a bit stream), return 1 if the program will eventually halt, or 0 if the program will run forever
- ▶ A wrong algorithm: run the program to see if it can halt.
 - ▶ If the program halts, then return 1.
 - ▶ If the program doesn't halts for 10 years,
- ▶ A problem is solvable means it needs be answered in a finite number of operations.

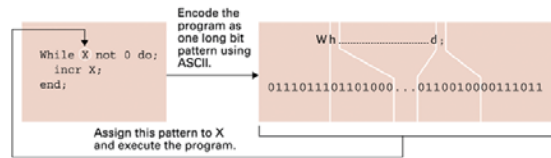
An example

```
While x is not 0
  x = x + 1
end
```

- ▶ Consider this program
 - ▶ x is a positive integer
 - ▶ Not considering the finite representation
 - ▶ If the input x is 0, the program halts.
 - ▶ Otherwise, it does not halt.

Self-reference and self-terminating

- ▶ Self-reference: use the encoded program as the input

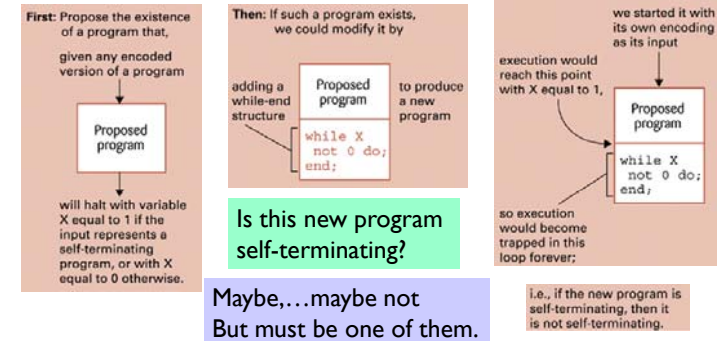


- ▶ Self-terminating: if a program with self-reference can halt, then it is called self-terminating.
 - ▶ The example program is not self-terminating.

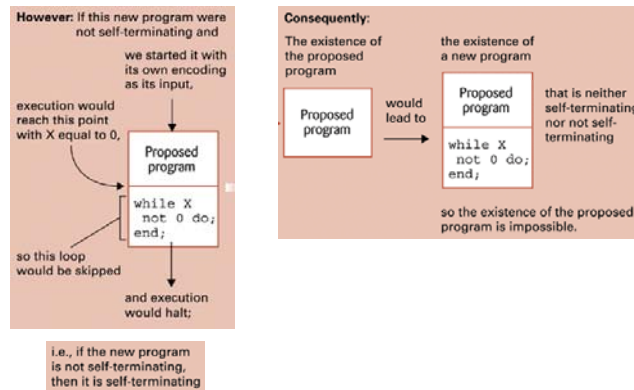
The undecibility of the halting program

Theorem: the halting problem is noncomputable.

Proof:



Proof (continue)



Problem classification

- ▶ Among solvable problems, some problems appear easier than the others.
- ▶ How to classify problems based on their difficulties?
 - ▶ Classification may be based on time, space, or other computing resources.
 - ▶ Unless otherwise noted, "complexity" means "time complexity."
 - ▶ Answer: The complexity of a problem is measured by the time complexity of the "best" algorithm to solve it.
- ▶ Unfortunately, finding a best solution or knowing it is the best is difficult for most problems.
 - ▶ Ex: The complexity of "searching a list" is $O(N)$.

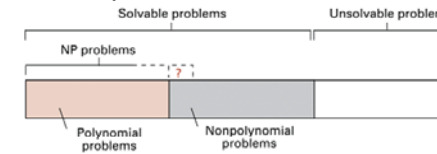
Class P

- ▶ Class P is the set of **decision problems** that can be solved by a Turing machine in a **polynomial time**.
 - ▶ Decision problem is a problem whose answer is either yes or no.
 - ▶ The halting problem is a decision problem.
 - ▶ If the problem size is N , polynomial time means the running time is dominated by a polynomial function of N
 - ▶ Exponential function $f(N)=2^N$ is always larger than the polynomial $p(N)=N^k$ for any constant k if N is large enough.
- ▶ Most computer scientists consider the problems in class P can be solved practically



Class NP

- ▶ Class NP is the set of problems that the “yes”-answers can be verified by a Turing machine in polynomial time.
 - ▶ The halting problem is in not NP.
- ▶ A million dollar question: **P=NP?**



- ▶ The Clay Math Institute's first millennium prize problem
- ▶ A new proof by Vinay Deolalikar (Aug 2010)
 - ▶ <http://www.win.tue.nl/~gwoegi/P-versus-NP/Deolalikar.pdf>

