**CHAPTER 0-4**

**computability**

* We now identify a function that is not Turing computable and so, by the Church-Turing thesis, is widely believed to be noncomputable in the general sense.

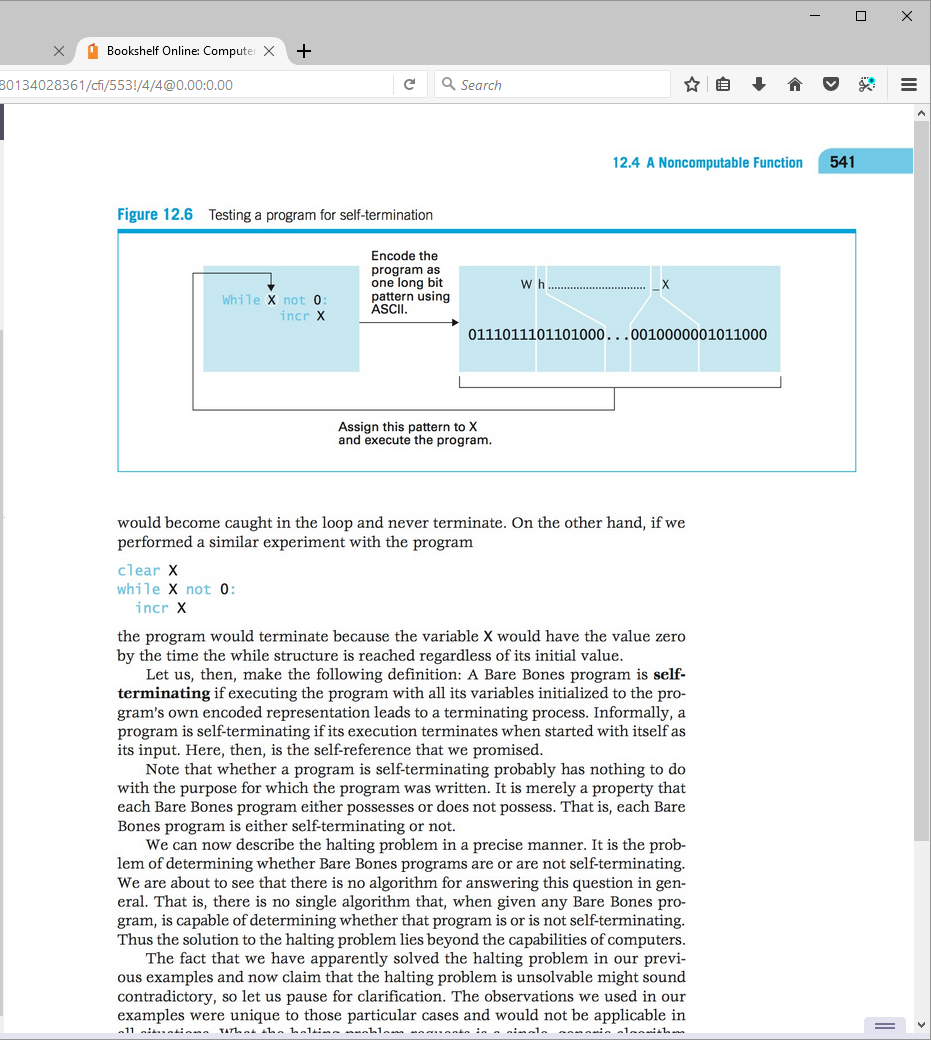
**The halting problem**

* The noncomputable function we are about to reveal is associated with a problem known as the halting problem, which is the problem of trying to predict in advance whether a program will terminate if started under certain conditions.
* First, observe that each Bare Bones program can be encoded as a single long bit pattern in a one-character-per-byte format using ASCII, which can then be interpreted as the binary representation for a nonnegative integer.
* Let us consider this simple program

while X not 0:

incr X

We want to know what would happen if we started this program with X assigned the integer value representing the program itself.



* Let us perform a similar experiment with the following program

clear X

while X not 0:

incr X

* Let us make the following definition: A Bare Bones program is self-terminating if executing the program with all its variables initialized to the program’s own encoded representation leads to a terminating process.
* Note that whether a program is self-terminating probably has nothing to do with the purpose for which the program was written. It is merely a property that each Bare Bones program either possesses or does not possess. That is, each Bare Bones program is either self-terminating or not.
* We can now describe the halting problem in a precise manner. It is the problem of determining whether Bare Bones programs are or are not self-terminating.

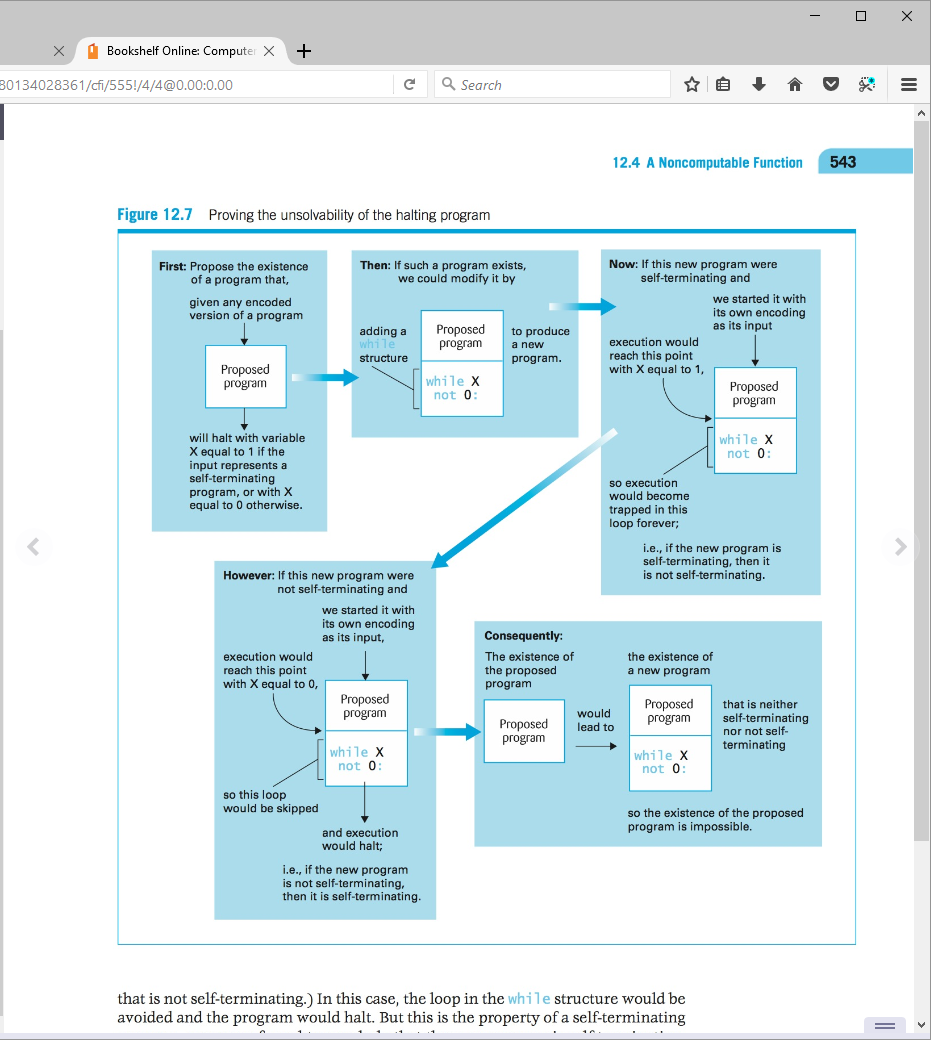
**The Unsolvability of the Halting Problem**

* The inputs of the halting function are encoded versions of a Bare Bones program. If the input represents a self-terminating program, the halting function produces the output value 1; if the input represents a not self-terminating program, the halting function produces the output value 0.
* Our task is to show that the halting function is not computable.
* Our approach is the technique known as “proof by contradiction.”
  + Assume the halting function is computable.
  + Then there must be a Bare Bones program P that computes it.
    - In other words, there is a Bare Bones program P that terminates with its output equal to 1 if its input is the encoded version of a self-terminating program and terminates with its output equal to 0 otherwise.
  + Assume that P's output variable is named X. Let us modify the program P by attaching the following statement at its end:

while X not 0:

Let us call this new program PP. Like all programs, this new program PP must be either self- terminating or not. However, we are about to see that it can be neither.

* Our entire argument is summarized below:



* We conclude that the halting function is not computable, and because the solution to the halting problem relies on the computation of that function, we must conclude that solving the halting problem lies beyond the capabilities of any algorithmic system. Such problems are called unsolvable problems.

**HW**

1. Is the following Bare Bones program self-terminating? Explain your answer.

incr X

decr Y

1. Is the following Bare Bones program self-terminating? Explain your answer.

Y=X

incr Y

incr Y

while X not 0:

decr X

decr X

decr Y

decr Y

decr Y

while Y not 0: