

CSCI 303 Introduction to Algorithms  
Spring 2007  
April 11<sup>th</sup>, 2007 class notes

We have shown that  $K$  is undecidable. So the computer is limited. Meaning that there are problems you cannot solve on a computer.



Professor Adleman showing that the set of all decidable problems is so small compared to the set of all problems, that we would not even be able to see it on this diagram.

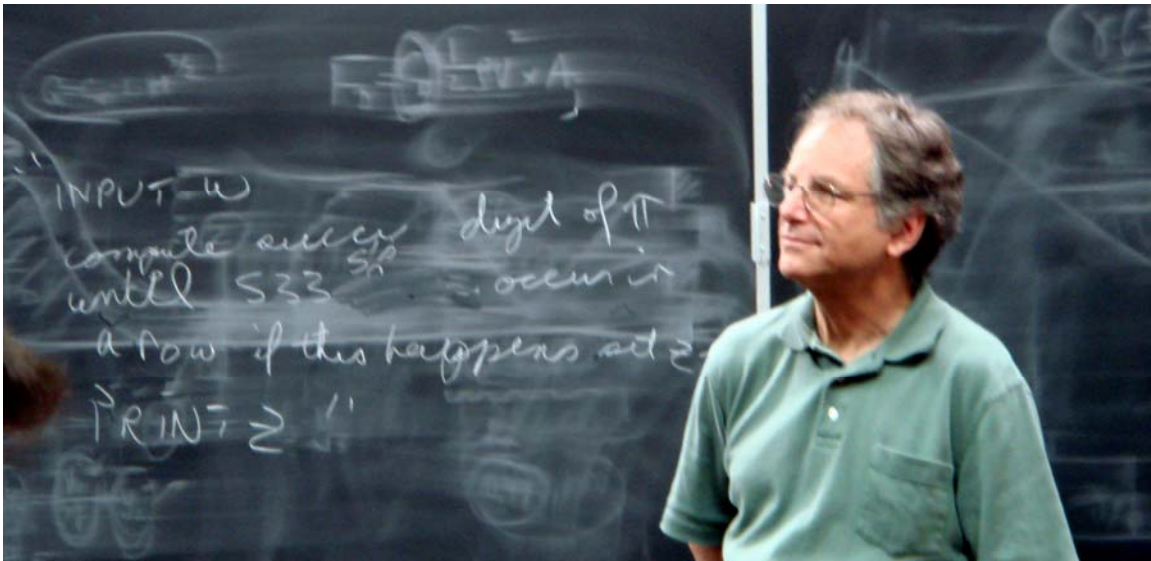
Examine this program:

“Input  $w$

Compute successive digits of  $\pi$  until 533 fives occur in a row. If this happens, set  $z = 0$ .

Print  $z$ .”

Does this program halt?



Are our brains different from computers?

It is wildly believed that S-BASIC programs (and Turing machines) capture the idea of an algorithm.

Let's talk about a robot that can build itself:

1. Find the pieces from a list.
2. Put them together according to a set of instructions.
3. Download entire memory to the new robot.
4. Turn on new robot.

The fact that the robot can access its own memory is related to the idea of the recursion theorem, due to Kleene.

Program:

"input w

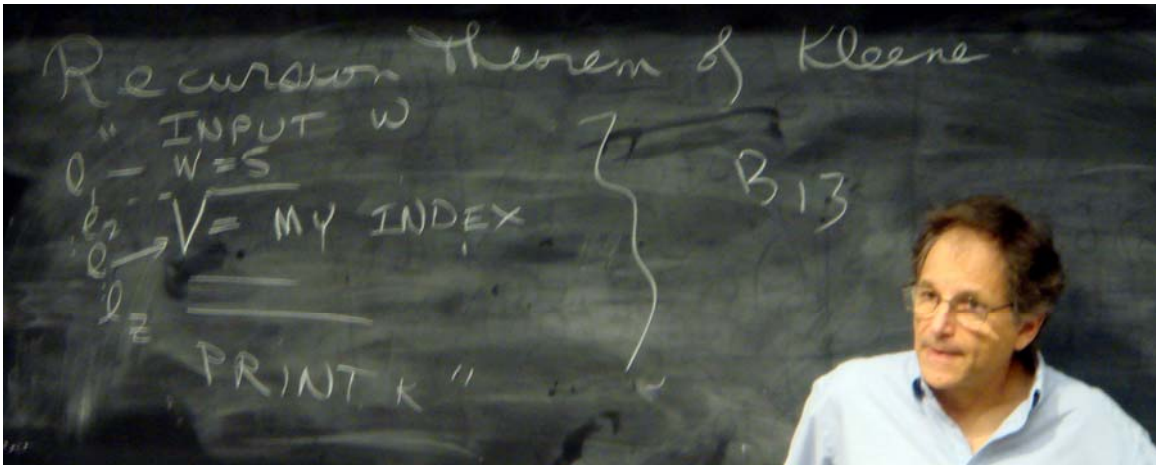
...

V = MY INDEX

...

print k"

MY INDEX returns  $i$  in program  $B_i$ .



An alternate proof that  $K$  is undecidable:

Assume  $K$  is decidable  $\Rightarrow C_K$  is computable  $\Rightarrow$  there exists a such that  $B_a = C_K$

Write a new program:

```

"Input w
V = MY INDEX
Call  $B_a$  (as a subroutine) with argument V
A: if  $B_a$  returns 1 then goto A
If  $B_a$  return 0 then L = 500
Print L"

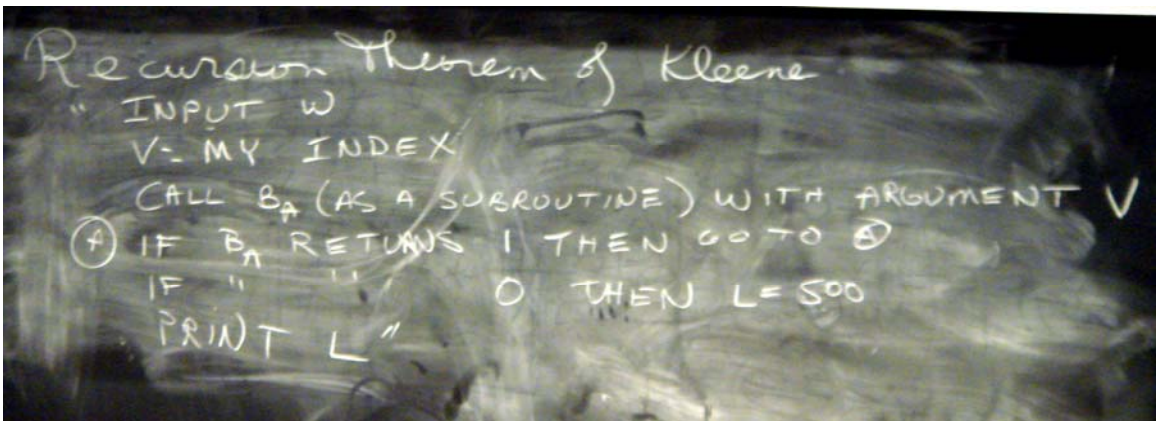
```

This is an SBasic program, so there exists some  $j$  such that the program is  $B_j$ . What does  $B_j$  do on input  $j$ ?

If  $B_j$  on  $j$  gets stuck on line A, then  $B_a$  on  $j$  outputs 1, thus  $j \in K$ . But by examining  $B_j$ , we see that it halts on  $j$ . Contradiction.

If  $B_j$  on  $j$  does not get stuck on line A, then  $B_a$  on  $j$  outputs 0, thus  $j \notin K$ . But by examining  $B_j$ , we see that it does not halt. Contradiction.

Therefore,  $K$  is undecidable.



So recursion theorem makes the game unfair. It says, given some program that can supposedly predict all programs' behavior, you get to ask it what it thinks you'll do, and then do the opposite.

Proof that S is undecidable:

Assume S is decidable  $\Rightarrow$  there exists a  $B_a$  that decides S

Write a new program:

```

    "Input w
    V = MY INDEX
    Call  $B_a$  (as a subroutine) with argument V
    A: if  $B_a$  returns 1 then goto A
    If  $B_a$  return 0 then L = 500
    Print L"
  
```

This is an SBasic program, so there exists some j such that the program is  $B_j$ . What does  $B_j$  do on input j?

If  $B_j$  on j gets stuck on line A, then  $B_a$  on j outputs 1, thus  $j \in S$ . But by examining  $B_j$ , we see that it halts. Contradiction.

If  $B_j$  on j does not get stuck on line A, then  $B_a$  on j outputs 0, thus  $j \notin S$ . But by examining  $B_j$ , we see that it does not halt. Contradiction.

Therefore, S is undecidable.

