Computers: What can't they do!
Computers: What can't they do?

Part 4: Undecidability

December 1, 2011

CS 42: Principles and Practice of Computer Science

A Strategy[1] for CS Programming Assignments[2]

Or: "How to succeed in CS 70 without programming at all!"

1. For each programming assignment, note that the desired program's behavior can be rephrased as a function from the integers $\mathbb{N}$ to the integers $\mathbb{N}$ by encoding the input and output appropriately.

[1] Please do not mention Prof. Stone when using the Strategy

[2] This strategy not applicable in CS 42.

A Strategy[1] for CS Programming Assignments[2]

Or: "How to succeed in CS 70 without programming at all!"

1. For each programming assignment, note that the desired program's behavior can be rephrased as a function from the integers $\mathbb{N}$ to the integers $\mathbb{N}$ by encoding the input and output appropriately.

2. Point out that **with 100% probability**, a function chosen from the set of all functions from $\mathbb{N}$ to $\mathbb{N}$ is not computable by any program at all.

[1] Please do not mention Prof. Stone when using the Strategy

[2] This strategy not applicable in CS 42.

A Strategy[1] for CS Programming Assignments[2]

Or: "How to succeed in CS 70 without programming at all!"

1. For each programming assignment, note that the desired program's behavior can be rephrased as a function from the integers $\mathbb{N}$ to the integers $\mathbb{N}$ by encoding the input and output appropriately.
2. Point out that **with 100% probability**, a function chosen from the set of all functions from $\mathbb{N}$ to $\mathbb{N}$ is not computable by any program at all.
3. Proceed to your choice of (a) the mountains, or (b) the beach.
4. Consult the fine print for disclaimers.

[1] Please do not mention Prof. Stone when using the Strategy

[2] This strategy not applicable in CS 42.

The TM Halting Problem

A specific undecidable problem. Is it obvious that it can't be solved?

P is an input program, i.e., a *turing machine, encoded suitably*.

w is P's input data.

P, w

input

The Halt Checker Program

output

Yes, P halts on w

No, P doesn't halt on input w.

Note that this blue halt checker is ***not*** allowed to run forever.

The DFA Halting Problem

Could we solve the problem for DFAs?

Note that this blue halt checker is ***not*** allowed to run forever.

The TM Halting Problem, Revisited

P is an input program, i.e., a *turing machine, encoded suitably*.

w is P's input data.

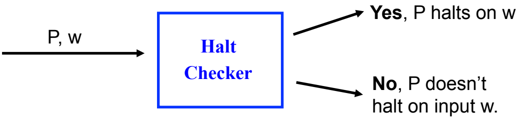
Note that this blue halt checker is ***not*** allowed to run forever.

Weird things. Are these ok?

- ✓ The blue program takes two inputs, one of which is a program.
- ✓ If the blue program runs $P(w)$, it may never finish!

Halting is Undecidable

Suppose a Halt-Checker exists:

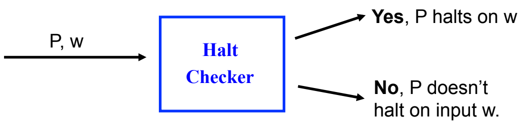


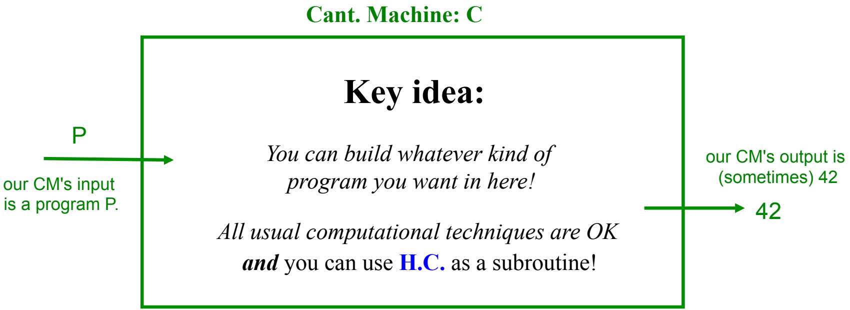
Our plan: Proof by Contradiction.

- ✓ Construct a new machine that uses this HC as a subroutine
- ✓ Show the new machine cannot possibly exist.

Halting is Undecidable

Suppose a Halt-Checker exists:

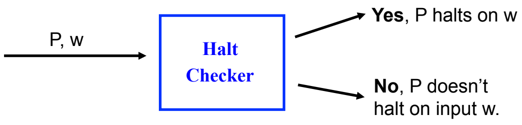


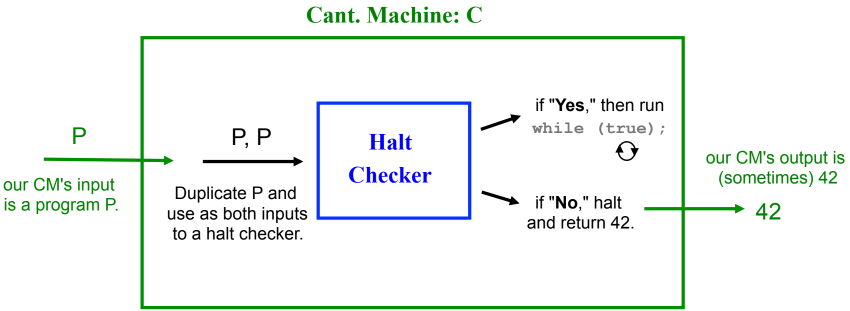


looking for a contradiction...

Halting is Undecidable

Suppose a Halt-Checker exists:





Claim: The Cant. Machine can't exist!

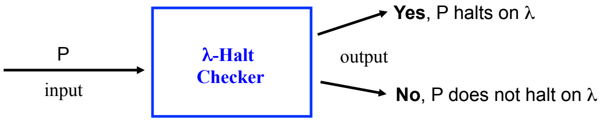
Beyond TM Halting

Now that we know that Halting is not decidable, we can use this to show other problems are undecidable.

Reduction: Show that *if* we could solve some problem P, then we could use that solution to build a Halt Checker HC! Hence, there's no TM that solves P.

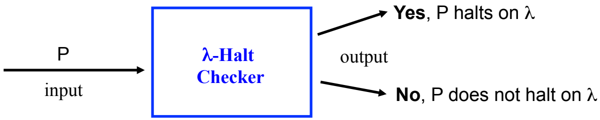
The Blank-Tape Halting Problem

P is a program that takes no input (a TM that starts with a blank tape).



The Blank-Tape Halting Problem

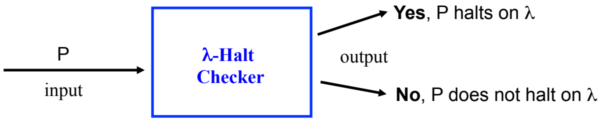
P is a program that takes no input (a TM that starts with a blank tape).

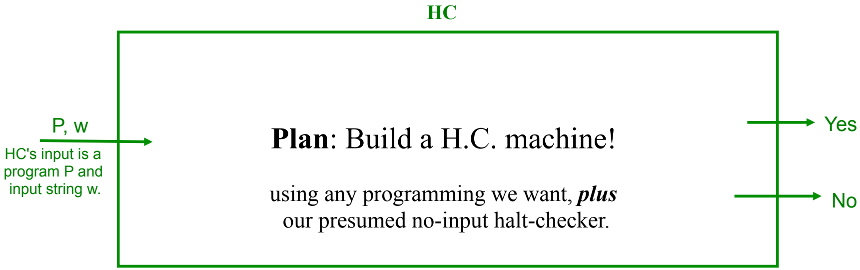


Claim: This machine cannot exist. Proof: Reduction *from* the Halting Problem: Use it to build HC.

The Blank-Tape Halting Problem

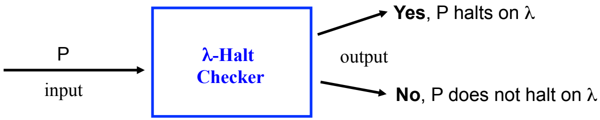
P is a program that takes no input (a TM that starts with a blank tape).

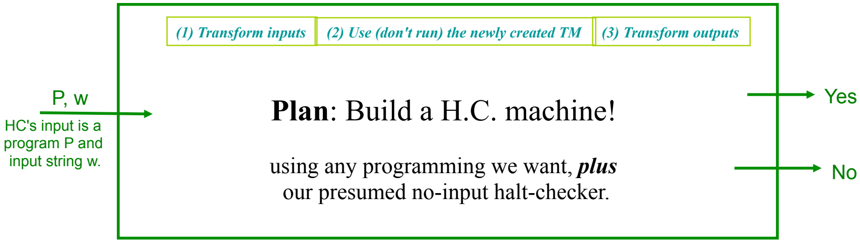




The Blank-Tape Halting Problem

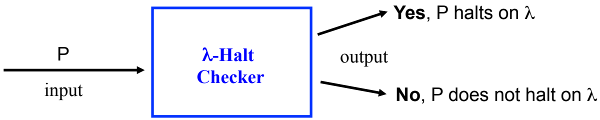
P is a program that takes no input (a TM that starts with a blank tape).

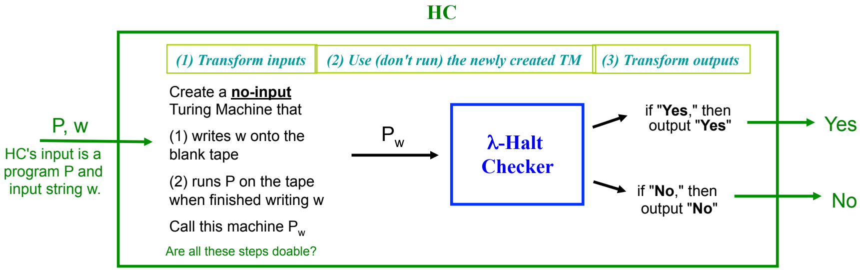




The Blank-Tape Halting Problem

P is a program that takes no input (a TM that starts with a blank tape).





Writeup for the Record

We show that a blank-tape-halt-checker can not exist by reduction from the Halting Problem. Assume that a blank-tape-halt-checker BT(P) does exist. We build a program HC(P,w), which takes a program P and a string w as input, as follows:

1. Build a Turing Machine that takes no inputs. It first writes the string w to its blank tape, and then runs P on that tape. Call this no-input turing machine TM. Note that TM effectively runs P on w.
2. Call our blank-tape-halt-checker on this TM: BT(TM)
3. If BT(TM) reports that TM halts, halt and output "Yes"
4. If BT(TM) reports that TM does not halt, halt and output "No"

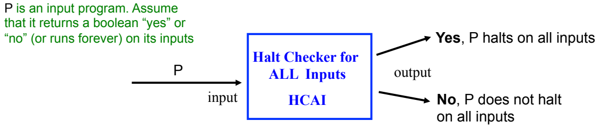
As long as BT exists, this constructed HC is a legitimate program. All of the steps are computable: writing a single, known string to a blank tape, running a turing machine's program, and conditional-checking. However, note that HC(P,w) is a halt checker!

- ✓ If P halts on w, HC(P,w) returns "Yes" (because TM will halt on no input, so BT(TM) returns "Yes")
- ✓ If P does not halt on w, HC(P,w) returns "No" (because TM will not halt on no input, so BT(TM) returns "No")

Since a halt-checker can not exist, we have reached a contradiction. Thus, our original assumption that the blank-tape-halt-checker exists was false. A blank-tape-halt-checker also can not exist.

NAME: "QUIZ"

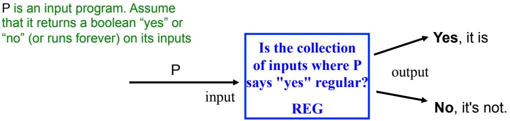
Show that All-Input Halting is Uncomputable



Claim: This cannot exist. Proof: By Reduction from Halting.

NAME: "QUIZ **EXTRA**"

Show that Says-Yes-To-A-Regular-Language is Uncomputable



Claim: This cannot exist. Proof: By Reduction from Halting.

Other Uncomputability Results

- ✓ Rice's Theorem: No nontrivial property of program behavior is decidable.
- ✓ Equivalence of programs is not decidable.
- ✓ Various tiling/matching problems are not decidable.