Computability/Decidability and Halting problem

Ali, Bayatpoor
Grant, Gurvis
Holly, McCoy

What is the Halting Problem

Computers are unable to declare if a program will halt or Infinitely Loop

An “undecidable” problem

Halt or Loop

Halt Case:

```
print("Hello World");
```

Loop Case:

```
x = 1;

While x == 1:

    print("Hello World");
```

What Could Work Intuitively

Build program called: “The Decider”

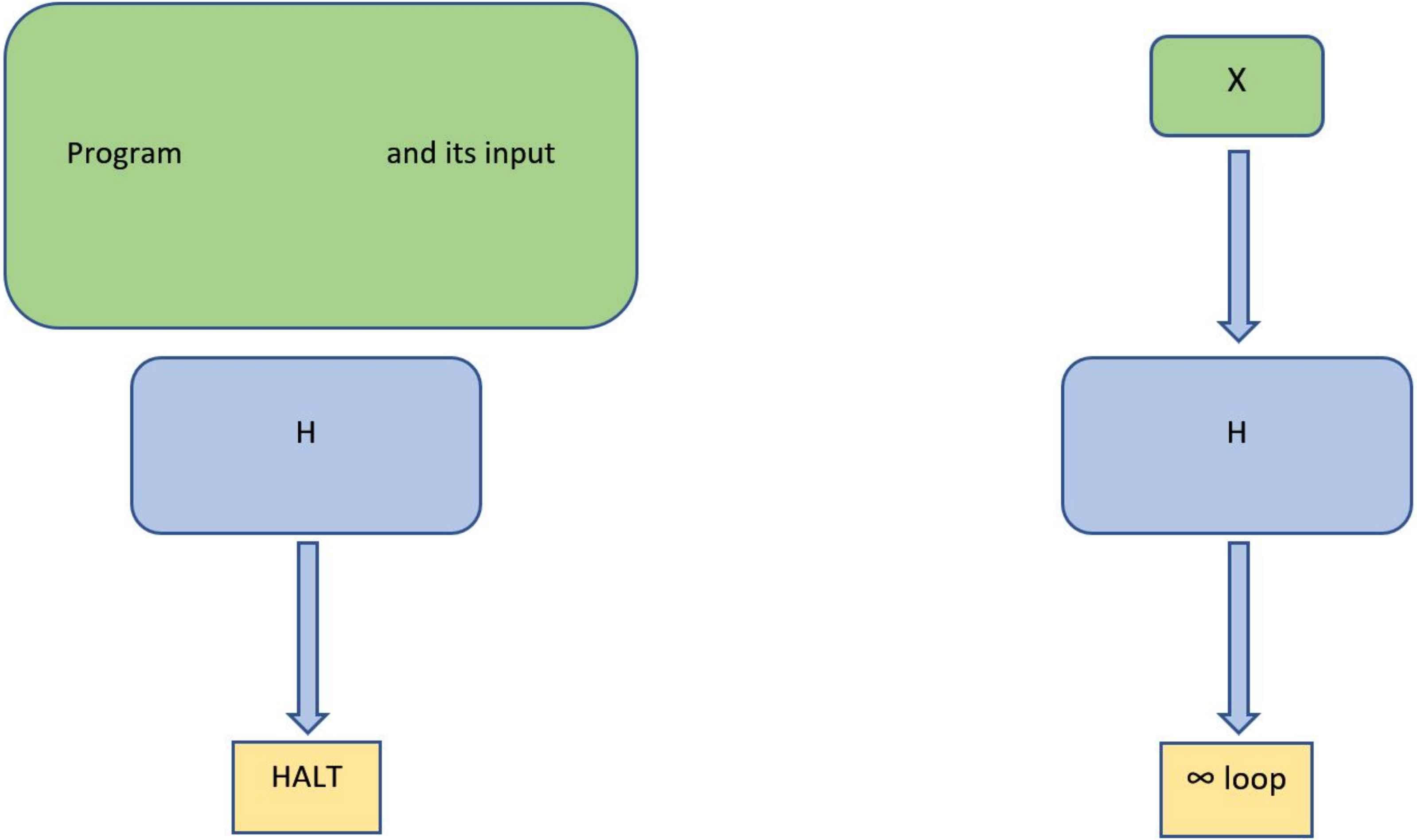
The Decider takes 2 inputs

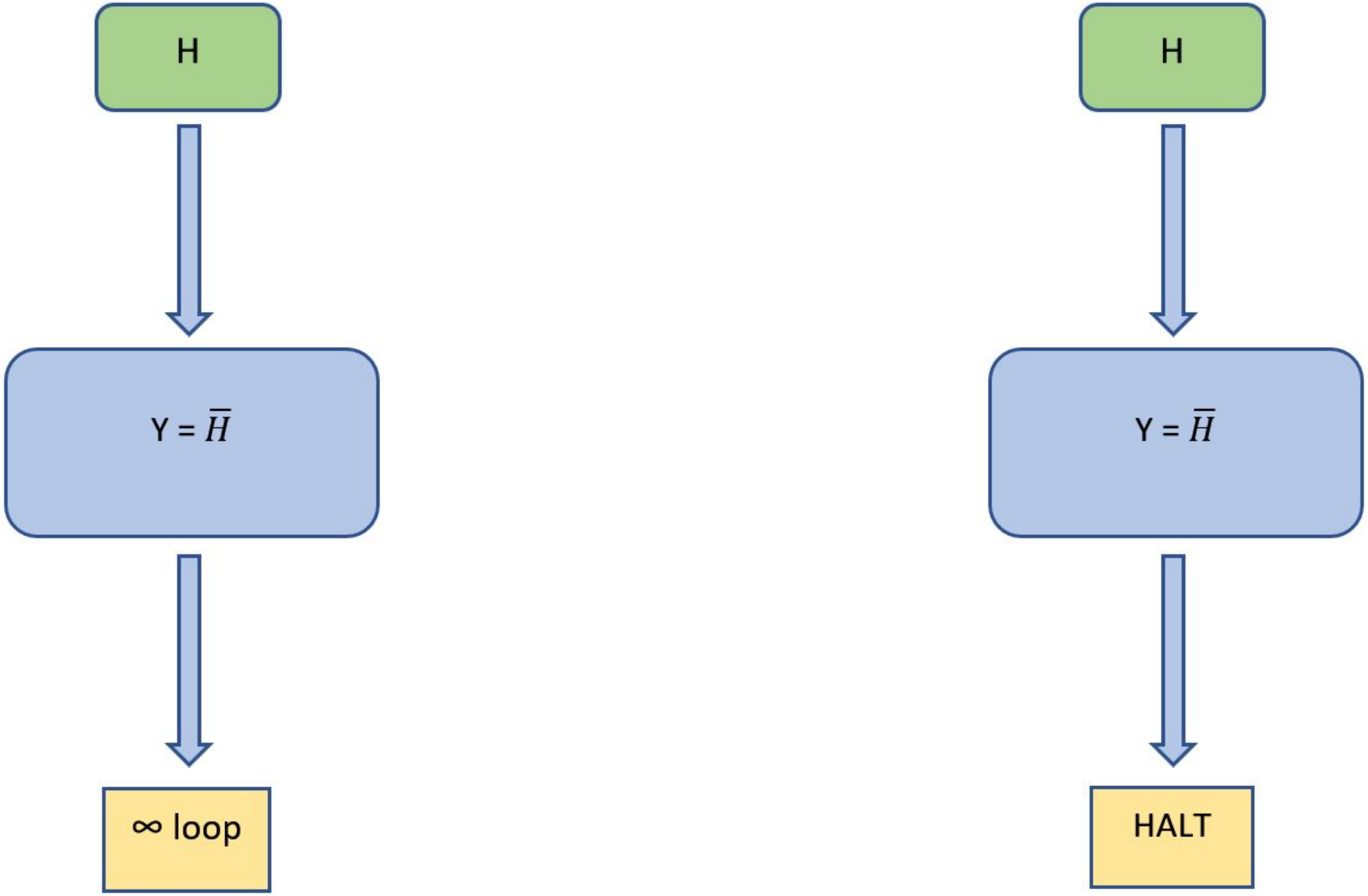
1) Another Program called “Program X”
2) Input to be fed into Program X call it “Input X”

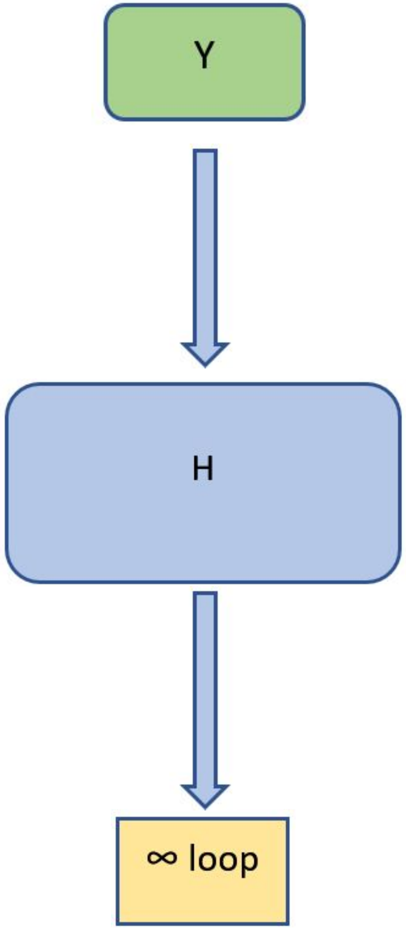
Deciders output will be either Loops or Halts

Problems with this Method:

How long to wait for infinity result?


Program
and its input
H
HALT
X
H
∞ loop


H
$Y = \overline{H}$
∞ loop
H
$Y = \overline{H}$
HALT



Y
H
∞ loop

Busy Beaver

The Busy Beaver function the maximum number of steps that a n-state m-symbol halting Turing machine can take, thus if the Busy Beaver of a n-state turing machine was computable then so would the halting problem.

For a given Turing machine if it ran for more steps than the Busy Beaver function of n states then it would not be halting.

It is uncomputable for n and it is currently known to not be able to be computed with n ≥ 1919, so it is a finite number that is not computable

Busy Beaver

1, 2, 3, and 4 state Turing machines can be shown to be halting using the Busy Beaver function.

They number of steps per busy beaver: 1, 6, 21, 107

Computing lower bound for 5 takes about a minute

Computing lower bound for 6 would take 10^36527 times longer or 10^36521 years

Halting Solver

Only proveably works for 2 symbols 1-4 states

First computes busy beaver steps for a given turing machine

Then computes turing machine in question to see if it halts or runs longer than busy beaver (then it does not halt)

Now to show it working...

Implications of Halting Problem

Turing machines have been constructed that if they halt proves/disproves the Riemann hypothesis, Goldbach's conjecture, and ZFC consistency. These open problems in math could simply be solved by the halting problem.

Could also easily verify if a program works as expected as each part could be shown to halt or not. Such as if a function halts as expected or will loop forever.

References

P. Linz, An Introduction to Formal Languages and Automata.Jones & Bartlett Learning, 2016.[

T. Rado, “On non-computable functions,” The Bell System Technical Journal, vol. 41, no. 3, pp. 877–884, 1962.

“The on-line encyclopedia of integer sequences,” 2020. [Online]. Available: https://oeis.org/A060843

G. Gurvis, “cot4210-halting-problem,” 2020. [Online]. Available: https://github.com/grant0417/cot4210-halting-problem