The Halting Problem

Soupfoo

Introduction

The Halting Problem is an unsolvable problem in logic and computability theory. It refers to the question of determining whether an arbitrary computer algorithm or a Turing Machine `Tm` will eventually halt on input `w` or keep on running indefinitely. There are no limitations on time and memory required for program execution. The goal is to just determine whether the program will halt or not.

Example

Let us take the following program as an example.

```
def count(n):
    while n > 0:
        n = n-1
    print("Countdown complete")

count(5)
```

We can formally verify that this program halts by using mathematical induction.

- **Base case**: If `n = 0`, the program halts immediately as the loop does not execute.
- **Induction hypothesis**: Let us assume that the program halts for `n = k`.
- **Inductive step**: If the program halts for `n = k` then it also halts for `n = k + 1` since it will reduce to `n = k` after one iteration.

Hence this program halts.

Let's modify the above program slightly.

```
def count(n):
    while n > 0:
        n = n+1
    print("Countdown complete")

count(5)
```

This program will never halt as `n` will never reach 0.

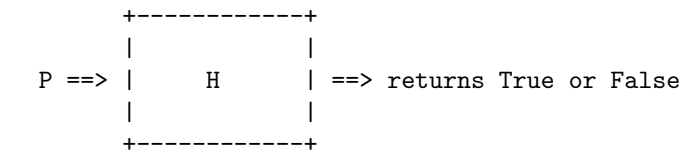
Deciding whether such small problems halt or not is simple. However, complex problems tend to be difficult to prove.

Turing's proof

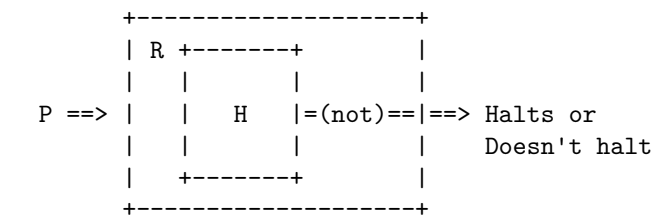
Alan Turing's proof of the halting problem is an example of **proof by contradiction** (reductio ad absurdum). He demonstrated that it is impossible to design an algorithm that can determine whether an arbitrary program halts or runs indefinitely for all possible inputs.

Proof:

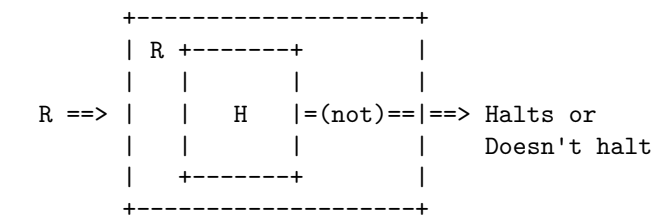
1. Assume there exists a program `H(P)` that returns `True` if program `P` halts and `False` if it runs forever on input `w`.



2. Define another program `R(P)` that complements the output of `H`, i.e. if `H(P)` returns `True`, `R` runs forever and if `H(P)` returns `False`, `R` halts.



3. Now feed `R` to itself as an input.



- If `H` predicts that `R` halts, `R` doesn't halt.
- If `H` predicts that `R` doesn't halt, `R` halts.

In both cases, a contradiction occurs.

The very existence of the halt checking program `H` leads to a logical paradox. Therefore, `H` cannot exist. Hence, the 'Halting Problem' is undecidable.

Conclusion

The Halting Problem is one of the foundational concepts in computability theory. Turing's proof by contradiction shows that it's impossible to construct an universal algorithm that can decide whether an arbitrary program halts or runs forever. Existence of such an algorithm leads to a paradox.