The Logical Possibility of Halting Oracles

Eric Holloway

DOI: 10.33014/issn.2640-5652.1.1.holloway.3

A halting oracle is a kind of function that can determine whether any computer program will halt. It is commonly thought that halting oracles are logically impossible. This is due to the common proof presented for the halting problem, which is as follows.

Imagine we have a halting oracle function $H(p,i)$, which returns true if program p halts on input i, and false otherwise.

We can also have a function defined as $G(p)$: if $H(p,p)$ is true then loop forever, else halt.

If we run $G(G)$, then we end up with a contradiction, because if $H(G,G)$ halts then $G(G)$ will not halt, and visa versa. Consequently, the function H is impossible.

This proof introduces a confusion because it appears that a halting oracle is logically impossible. However, the proof only shows that it is impossible for a halting oracle to be a finite Turing machine. It is still logically possible to have a halting oracle that is not a finite Turing machine.

To see that halting oracles are a logical possibility, we can construct our own halting oracle from logically possible components.

First, note that all finite Turing machines form a countable infinite set. This means that we can match each finite Turing machine with a positive integer. Imagine this as an infinitely long index, with a numbered entry for each Turing machine.

The second step is to note that each finite Turing machine has a halting status. Each machine either halts or runs forever. Consequently, we can take the infinite index from the first step, and, for each Turing machine, add an entry with its halting status. Thus, our index now consists of three elements: an index number, the finite Turing machine definition, and the machines halting status.

The final step is to imagine a search machine that has access to this index. The difference between the search machine and a finite Turing machine is that since the index is infinite the search machine is also infinite, and thus the search machine does not show up in the index itself. The operation of the search machine is to incrementally search the index for a given finite Turing machine and return the finite machines halting status. Since the set is countably infinite and each finite Turing machine is paired with a finite number, the search machine is guaranteed to halt with the halting status for every finite Turing machine. Thus, the search machine is a halting oracle.

Now we can address the question whether a halting oracle is logically impossible. Since the search machine is a halting oracle, then if the search machine is not logically impossible, neither is the general concept of a halting oracle.

The search machine is composed of two main components. If neither component is logically impossible, then neither is their composition in the form of the search machine. The components are:

1. The countable infinite index of finite Turing machine halting statuses
2. The search procedure

Countable infinite sets are used regularly in mathematics and are not logically impossible, although some branches of mathematics deny their existence as an axiomatic decision. Thus, component #1 is not logically impossible.

The search procedure can be encoded with a finite Turing machine that counts upwards incrementally, and can call out to an external index to retrieve the finite Turing machine definition and halting status. Since the search performs an exact match with the input finite Turing machine and the retrieved finite Turing machine, then the match can be encoded as a finite Turing machine as well. Since finite Turing machines are not logically impossible, then neither is component #2, the search procedure.

Finally, the combination of the infinite index and the search procedure is not logically impossible, since no further concepts are added. Consequently, the search machine demonstrates that halting oracles are logically possible.