Rebutting the Sipser Halting Problem Proof

MIT Professor Michael Sipser has agreed that the following verbatim paragraph is correct (he has not agreed to anything else in this paper):

If simulating halt decider H correctly simulates its input D until H correctly determines that its simulated D would never stop running unless aborted then H can abort its simulation of D and correctly report that D specifies a non-halting sequence of configurations.

A simulating halt decider computes the mapping from its input finite strings to an accept or reject state on the basis of the actual behavior specified by this input as measured by its correct simulation of this input. The following shows how the correct value for the D and ⟨D⟩ diagonal in Sipser's Figure 4.6 is accept.

We start with Sipser's definitions of H and D:

> On input (M, w), where M is a TM and w is a string, H halts and accepts if M accepts w. Furthermore, H halts and rejects if M fails to accept w. In other words, we assume that H is a TM, where
>
> H(⟨M,w⟩ = { accept if M accepts w
> { reject if M does not accept w
>
> Now we construct a new Turing machine D with H as a subroutine. This new TM calls H to determine what M does when the input to M is its own description ⟨M⟩. Once D has determined this information, it does the opposite. That is, it rejects if M accepts and accepts if M does not accept.
>
> D(⟨M⟩) = { accept if M does not accept ⟨M⟩
> { reject if M accepts ⟨M⟩ (Sipser 1997:165)

We encode the Sipser D and define the behavior of Sipser H as C functions.

```
//
// Sipser_H returns 1 when its input would halt and return 1
// otherwise Sipser_H returns 0
//
int D(ptr2 M)
{
  if ( H(M, M) )
    return 0;
  return 1;
}

int main()
{
  Output((char*)"Input_Halts = ", D(D));
}
```

H bases its analysis of its input D on the behavior of its correct simulation of D. H finds that D remains stuck in infinitely recursive simulation (shown below) until H aborts its simulation of D.

(a) D calls H that simulates D with an x86 emulator
(b) that calls a simulated H that simulates D with an x86 emulator
(c) that calls a simulated H that simulates D with an x86 emulator ...
Until the executed H recognizes this repeating state, aborts its simulation of D and returns 0.

Complete halt deciding system (Visual Studio Project) Sipser version.
(a) x86utm operating system
(b) x86 emulator adapted from libx86emu to compile under Windows
(c) Several halt deciders and their sample inputs contained within Halt7.c
https://liarparadox.org/2022_10_08.zip

D calls simulating halt decider H which computes the mapping from its input D to an accept or reject state on the basis of the behavior of its correct simulation of D. When H correctly determines that this simulated input would remain stuck in recursive simulation H aborts this simulation and reports non-halting by returning 0. When D reverses this decision it returns 1. This is used to correctly fill in the “?” in the Sipser Figure 4.6 (see below) with “accept”.

Simulating halt decider H recognizes instances of recursive simulation using the same criteria that it uses in its dynamic behavior pattern that recognizes infinite recursion:

```
void Infinite_Recursion(u32 N)
{
  Infinite_Recursion(N);
}
```

	⟨M_1⟩	⟨M_2⟩	⟨M_3⟩	⟨M_4⟩ ...	⟨D⟩ ...
M_1	<u>accept</u>	reject	accept	reject	accept
M_2	accept	<u>accept</u>	accept	accept	accept
M_3	reject	reject	<u>reject</u>	reject	reject
M_4	accept	accept	reject	<u>reject</u>	accept
...					
D	reject	reject	accept	accept	<u>?</u>
...					

Figure 4.6 (Sipser 1997:167)

Sipser, Michael 1997. Introduction to the Theory of Computation. Boston: PWS Publishing Company (165-167)