Refuting the Sipser Halting Problem Proof

The Sipser proof is refuted on the basis of showing how Turing machine H correctly decides reject on ⟨D,⟨D⟩⟩ input while Turing machine D correctly decides accept on ⟨D⟩ input.

This x86utm operating system was created so that that halting problem could be examined concretely in the high level abstraction of the C programming language.

```
#define u32 uint32_t


int Simulate(u32 P, u32 I)
{
  ((void(*)(u32))P)(I);
  return 1;
}


int D(u32 P)   // P is a machine address
{
  if ( H(P, P) )
    return 0   // reject when H accepts
  return 1;    // accept when H rejects
}


int main()
{
  u32 HDD_Acceptance = H((u32)D, (u32)D);
  u32 DD_Acceptance  = D((u32)D);
  Output("H(D,D) Would_Halt = ", HDD_Acceptance);
  Output("D(D) Would_Halt = ",   DD_Acceptance);
}
```

A simulating halt decider that never stops simulating its input is simply a simulator on this input. If H() never stopped simulating D() then it can be seen that the halting behavior of D() would be the same as if D() invoked Simulate() instead of H(), thus D() would never terminate. The above analysis is confirmed by actual execution of the above function in the x86utm operating system:

(1) The first line of main() detects an infinitely repeating non-halting pattern that never reaches the second line of D it returns 0 for reject.

(2) The second line of main() returns 1 accept on the basis that H detects an infinitely repeating non-halting pattern and returns 0 for reject.

	$\langle M_1 \rangle$	$\langle M_2 \rangle$	$\langle M_3 \rangle$	$\langle M_4 \rangle$	...
M_1	accept		accept		
M_2	accept	accept	accept	accept	
M_3					
M_4	accept	accept			
...					

Original Figure 4.4

	$\langle M_1 \rangle$	$\langle M_2 \rangle$	$\langle M_3 \rangle$	$\langle M_4 \rangle$	...
M_1	accept	reject	accept	reject	
M_2	accept	accept	accept	accept	
M_3	reject	reject	reject	reject	
M_4	accept	accept	reject	reject	
...					

Original Figure 4.5

	$\langle M_1 \rangle$	$\langle M_2 \rangle$	$\langle M_3 \rangle$	$\langle M_4 \rangle$	...	$\langle D \rangle$
M_1	accept		accept			--
M_2	accept	accept	accept	accept		--
M_3						--
M_4	accept	accept				--
...						
D	reject	reject	accept	accept		accept
...						

Figure 4.4a (Figure 4.4 with row D and actual D(⟨D⟩) output added)

	$\langle M_1 \rangle$	$\langle M_2 \rangle$	$\langle M_3 \rangle$	$\langle M_4 \rangle$	...	$\langle D \rangle$ ...
M_1	accept	reject	accept	reject		--
M_2	accept	accept	accept	accept		--
M_3	reject	reject	reject	reject		--
M_4	accept	accept	reject	reject		--
...						
D	reject	reject	accept	accept		reject
...						

Figure 4.5a (Figure 4.5 with row D and actual H(D, ⟨D⟩) output added)

The one requirement of the diagonalization proof that is impossible to fulfill was that H obtains its value from D(⟨D⟩) and has the opposite value as D(⟨D⟩).

This requirement is rejected on the basis that:
(a) It requires an object with simultaneous mutually exclusive properties (always impossible).
(b) The actual return values of a computation supersede any estimates of what these values should be.

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