Defining a Decidability Decider for the Halting Problem

When we understand that every potential halt decider must derive a formal mathematical proof from its inputs to its final states previously undiscovered semantic details emerge.

When-so-ever the potential halt decider cannot derive a formal proof from its input strings to its final states of Halts or Loops, undecidability has been decided.

The formal proof involves tracing the sequence of state transitions of the input TMD as syntactic logical consequence inference steps in the formal language of Turing Machine Descriptions.

$\exists$H $\in$ Turing_Machines_Descriptions
$\forall$b $\in$ Turing_Machines_Descriptions
$\forall$c $\in$ Finite_Strings
~((b,c) ⊢ H.Halts ∨ (b,c) ⊢ H.Loops) ↔ H.Pathological-Self-Reference

Every element of the infinite set of (b,c) is divided into one of these subsets:
(1) H can derive a formal proof from input (b,c) finite strings to H.Halts
(2) H can derive a formal proof from input (b,c) finite strings to H.Loops
(3) else H.Pathological-Self-Reference(Olcott 2004)

This meets the Rice Criteria shown below thereby providing the single valid counter-example required to refute Rice's Theorem.

http://kilby.stanford.edu/~rvg/154/handouts/Rice.html
Rice's theorem: Any nontrivial property about the language recognized by a Turing machine is undecidable.

A property about Turing machines can be represented as the language of all Turing machines, encoded as strings, that satisfy that property. The property P is about the language recognized by Turing machines.

https://www.tutorialspoint.com/cgi-bin/printpage.cgi
ACCEPTED LANGUAGE & DECIDED LANGUAGE
A TM accepts a language if it enters into a final state for any input string w.
A language is recursively enumerable if it is accepted by a Turing machine.

A TM decides a language if it accepts it and enters into a rejecting state for any input not in the language. A language is recursive if it is decided by a Turing machine.