Formal Verification

How to check if a designed algorithm meets the formal specification?

Testing/Typing are not sufficient

- Easy to argue that a given input will produce a given output (though the halting problem is already undecidable).
- Easy to argue that a property always holds at a single program point
- Also easy to argue that all constructs in the language will preserve some property (like when we proved type soundness).
- Much harder to prove general properties of the behavior of a program on all inputs.

Undecidability of Program Verification

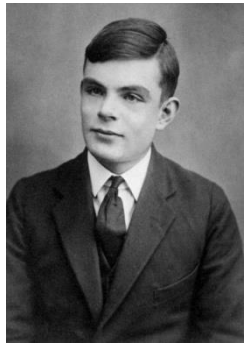
Rice's Theorem (1951): Every *nontrivial* semantic property of recursively enumerable languages is *undecidable*.

- Recursively enumerable languages are equivalent to Turing machines (and almost all languages you program).

Proof: Reduce from the halting problem of Turing machines.

Formal Verification

1949



A. M. Turing

Friday, 24th June [1949]

Checking a large routine by Dr A. Turing.

How can one check a routine in the sense of making sure that it is right?

In order that the man who checks may not have too difficult a task the programmer should make a number of definite assertions which can be checked individually, and

1967

Robert W. Floyd

ASSIGNING MEANINGS TO PROGRAMS[1]

Introduction. This paper attempts to provide an adequate basis for formal definitions of the meanings of programs in appropriately defined programming languages, in such a way that a rigorous standard is established

1979

DeMillo, Lipton and Perlis

Reports and Articles

Social Processes and Proofs of Theorems and Programs

Richard A. De Millo
Georgia Institute of Technology

Richard J. Lipton and Alan J. Perlis
Yale University

Robert W. Floyd

2009

Hoare, Misra, Leavens, Shankar

The Verified Software Initiative: A Manifesto

C.A.R. HOARE
Microsoft Research

JAYADEV MISRA
The University of Texas at Austin

GARY T. LEAVENS
Iowa State University

and

NATARAJAN SHANKAR
SRI International Computer Science Laboratory

1. INTRODUCTION

We propose an ambitious and long-term research program toward the construction of error-free software systems. Our manifesto represents a consensus position that has emerged from a series of national and international meetings, workshops, and conferences held from 2004 to 2007. The research project, the Verified Software Initiative,