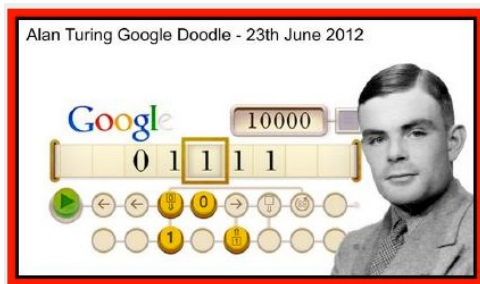


Turing Machines V

Monday, October 27



Handouts:
Assignment 18

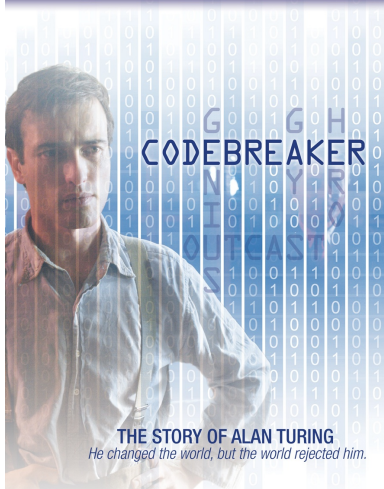
PREVIEWS OF COMING ATTRACTIONS



Tonight's Film

"powerful...imaginative"

The Sunday Times



7:30 – 9:00 PM

Warner 010

Preview of Coming Attractions I

- ▶ Deadline to Drop Course or Switch to Credit/No Credit
5 PM This Friday
- ▶ Registration For Winter Term
7 AM on Tuesday, November 4
- ▶ Spring Term Registration:
November 13 – 18
Individual Advising Sessions:
November 3 – November 12

Preview of Coming Attractions II

- ▶ Monday and Wednesday:
Turing's "On Computable Numbers"
- ▶ Friday:
Transition to Turing's "Computing Machinery and Intelligence" Paper in *Mind*
- ▶ Next Week: *Mind*
Reread *Mind* Paper
Searle's Response
- ▶ Friday, November 7:
ZOOM Session with David Leavitt

Preview of Coming Attractions III

- ▶ Week of November 11:
Team Presentations on Responses to Mind Paper
- ▶ Friday, November 20:
Team Written Reports

David Hilbert



Born: January 23, 1862 in Königsberg, Prussia
(now Kaliningrad, Russia)

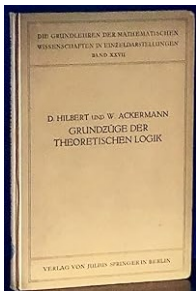
Died: February 14, 1943 in Göttingen, Germany

[Hilbert Biography Link](#)

"No one shall expel us from the Paradise that Cantor has created."



David Hilbert
1862 – 1943



Book
1928



Wilhelm Ackerman
1896 – December 1962

Complete

**Consistent
Decidable**

Self-Reference

I am not provable

Gödel (in his famous "Incompleteness Theorem") actually showed that in any "sufficiently large" system of mathematics (containing the natural numbers or "axioms of Peano arithmetic", for example), there are statements that are true but not provable. He did this by essentially encoding the sentence "I am not provable" within his mathematical system using "Gödel numbers".

Kurt Gödel



Born: April 28, 1906 in Brünn, Austria-Hungary
(now Brno, Czech Republic)

Died: January 14, 1978 in Princeton, New Jersey

[Gödel Biography Link](#)



Gödel with Einstein, Princeton, 1950

Gödel's First Incompleteness Theorem: No reasonable formal proof system for Number Theory can prove all "true sentences". So, in any such system, there are going to be true sentences that cannot be proven to be true (true but not provable sentences).

Gödel : No sufficiently powerful system of mathematics can prove its own "consistency" unless it is inconsistent. So, one of the things you cannot do within such a system of mathematics is to prove its own consistency. You need a "meta-system to do so (going "above" the system).

An "inconsistency" is a statement p in the system with the property that both p and not p are true.

From *Breaking The Codes*



Turing Explains Gödel and Turing

ON COMPUTABLE NUMBERS, WITH AN APPLICATION TO THE ENTSCHIEDUNGSPROBLEM

By A. M. TURING.

[Received 28 May, 1936.—Read 12 November, 1936.]

The “computable” numbers may be described briefly as the real numbers whose expressions as a decimal are calculable by finite means. Although the subject of this paper is ostensibly the computable *numbers*, it is almost equally easy to define and investigate computable functions of an integral variable or a real or computable variable, computable predicates, and so forth. The fundamental problems involved are, however, the same in each case, and I have chosen the computable numbers for explicit treatment as involving the least cumbersome technique. I hope shortly to give an account of the relations of the computable numbers, functions, and so forth to one another. This will include a development of the theory of functions of a real variable expressed in terms of computable numbers. According to my definition, a number is computable if its decimal can be written down by a machine.

Turing used Self-Reference to
attack the
"Entscheidungsproblem"

We can also use the notion of "self-reference" to prove the **Halting Problem**. (Turing's Entscheidungs or "decision" problem).

Does an algorithm exist to "decide" the following problem:
Given a TM M and an input string x , does M eventually halt when it is run with x on its tape?

Given an algorithm $H()$ and its input x , does $H()$ halt when run on x ?
(does $H(x)$ halt?)

We will show the Halting Problem is *undecidable* in the sense that no Turing machine can ever answer "yes" or "no" to the above question(s).