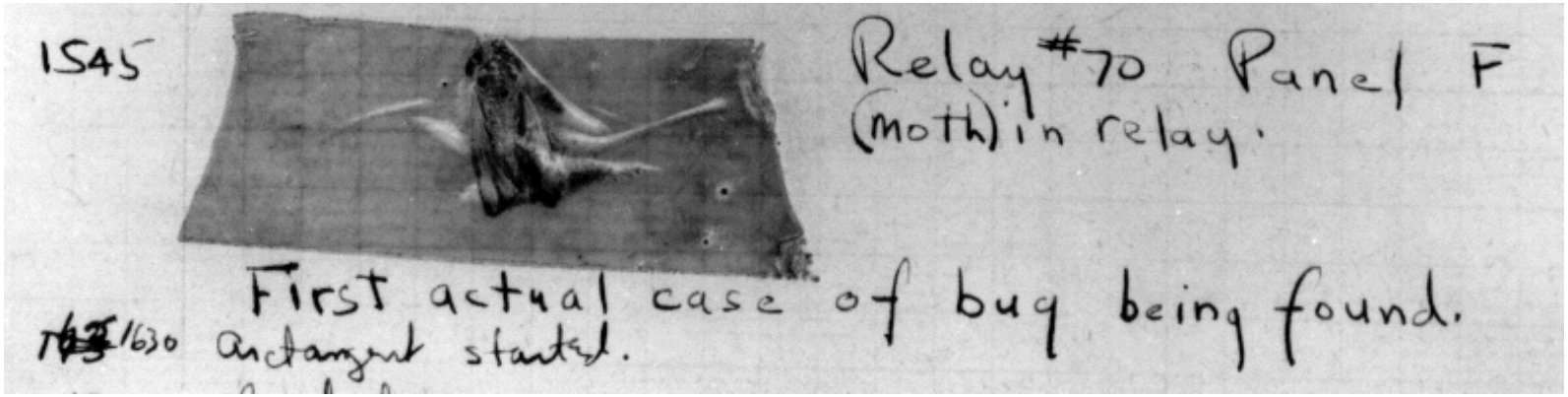
Programm- und Systemverifikation

184.741

Vorlesung: H. Veith G. Weissenbacher I. Konnov

Der erste Bug



Gefunden von einem Techniker im Harvard Mark II Rechner am 9. September, 1947. Heute ausgestellt im Smithsonian Museum, Washington

A problem has been detected and Windows has been shut down to prevent damage
to your computer.

DRIVER_IRQL_NOT_LESS_OR_EQUAL

If this is the first time you've seen this Stop error screen,
restart your computer, If this screen appears again, follow
these steps:

Check to make sure any new hardware or software is properly installed.
If this is a new installation, ask your hardware or software manufacturer
for any Windows updates you might need.

If problems continue, disable or remove any newly installed hardware
or software. Disable BIOS memory options such as caching or shadowing.
If you need to use Safe Mode to remove or disable components, restart
your computer, press F8 to select Advanced Startup Options, and then
select Safe Mode.

Technical information:

*** STOP: 0x000000D1 (0x0000000C,0x00000002,0x00000000,0xF86B5A89)

*** gv3.sys - Address F86B5A89 base at F86B5000, DateStamp 3dd991eb

Beginning dump of physical memory
Physical memory dump complete.
Contact your system administrator or technical support group for further
assistance.

A problem has been detected and Windows has been shut down to prevent damage to your computer.

DRIVER_IRQL_NOT_LESS_OR_EQUAL

If this is the first time you've seen this Stop error screen, restart your computer, If this scre[illegible] appears again, follow these steps:

Check to make sure any new hardwar[illegible]
If this is a new installation, ask[illegible]er for any Windows updates you might [illegible]

If problems continue, disable or r[illegible] or software. Disable BIOS memory c[illegible]
If you need to use Safe Mode to remove or disable comp[illegible]ents, restart your computer, press F8 to select Advanced Startup Opti[illegible]ns, and then select Safe Mode.

Technical information:

*** STOP: 0x000000D1 (0x0000000C,0x00000002,0x00000000,0xF86B[illegible])

*** gv3.sys - Address F86B5A89 base at F86B5000, DateSta[illegible] 3dc[illegible]91eb

Beginning dump of physical memory
Physical memory dump complete.
Contact your system administrator or technical support group fo[illegible] furt[illegible]er assistance.

Looks like you want to know what

DRIVER_IRQL_NOT _LESS_OR_EQUAL

means ...

Software Verifikation: Wozu?

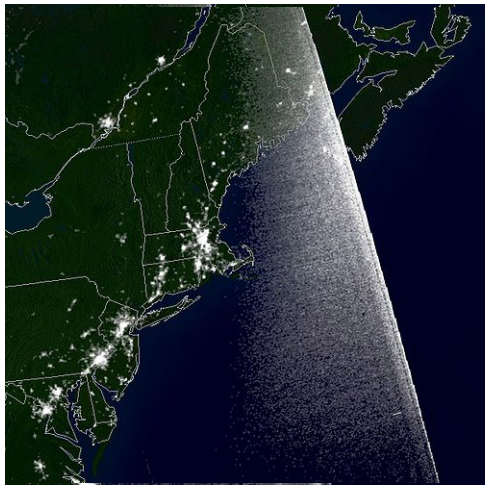
Meine Fluginformation

F16 (Nördliche Hemisphäre)

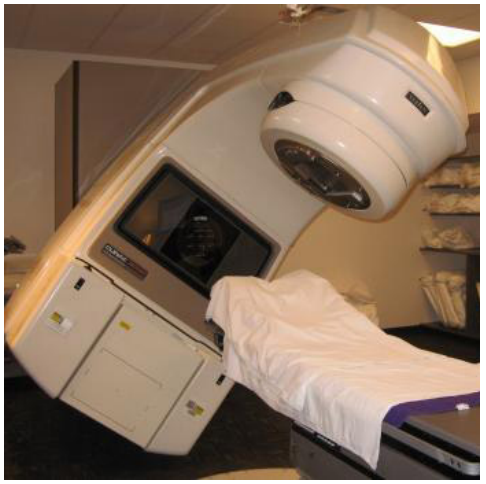
F16 (Südliche Hemisphäre)

Software Verifikation: Wozu?

Ariane 5 – flight 501



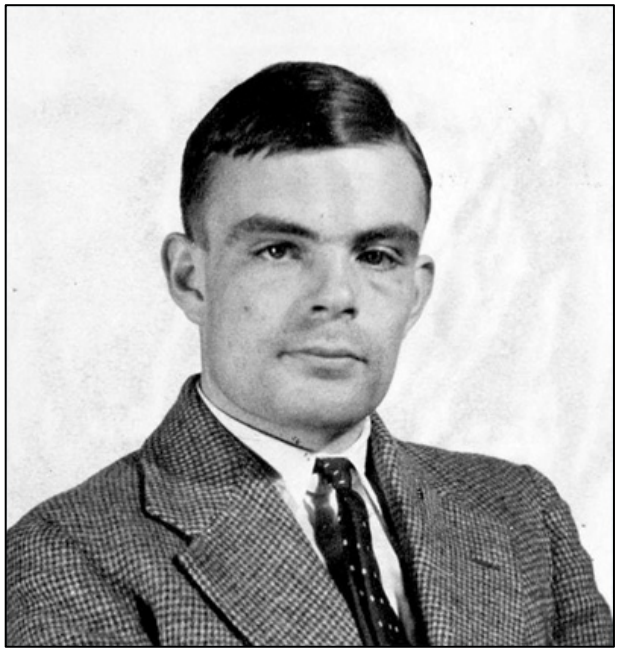
“Northeastern Blackout” 2003



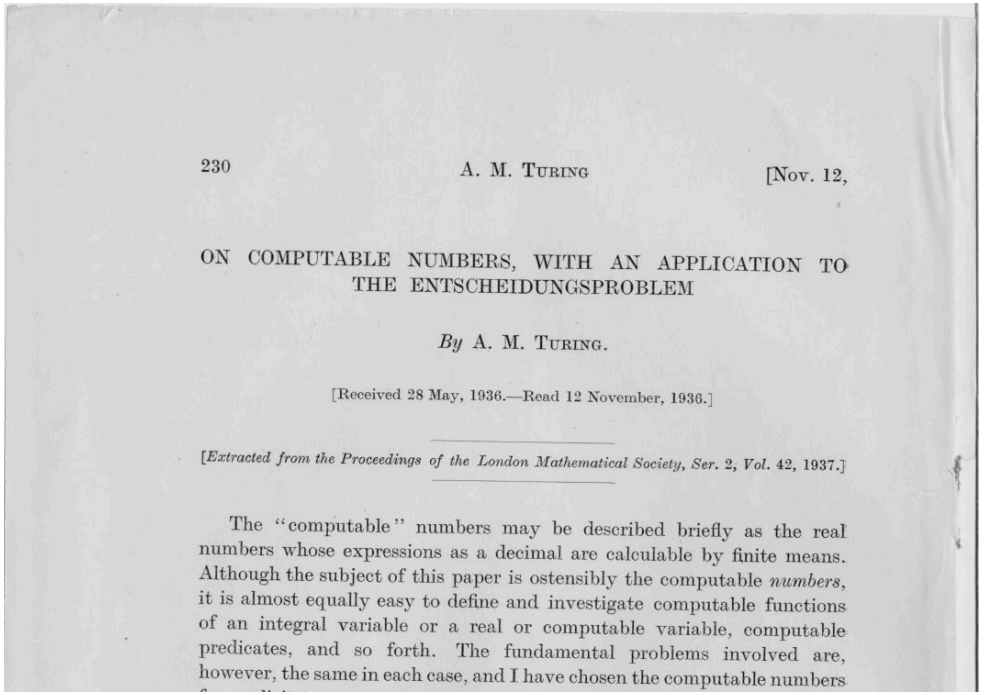
Therac-25 Vorfall

Können wir solche Fehler verhindern?

Automatisierte Fehlersuche?



A. Turing (1912-1954)



230 A. M. TURING [Nov. 12,

ON COMPUTABLE NUMBERS, WITH AN APPLICATION TO THE ENTSCHEIDUNGSPROBLEM

By A. M. TURING.

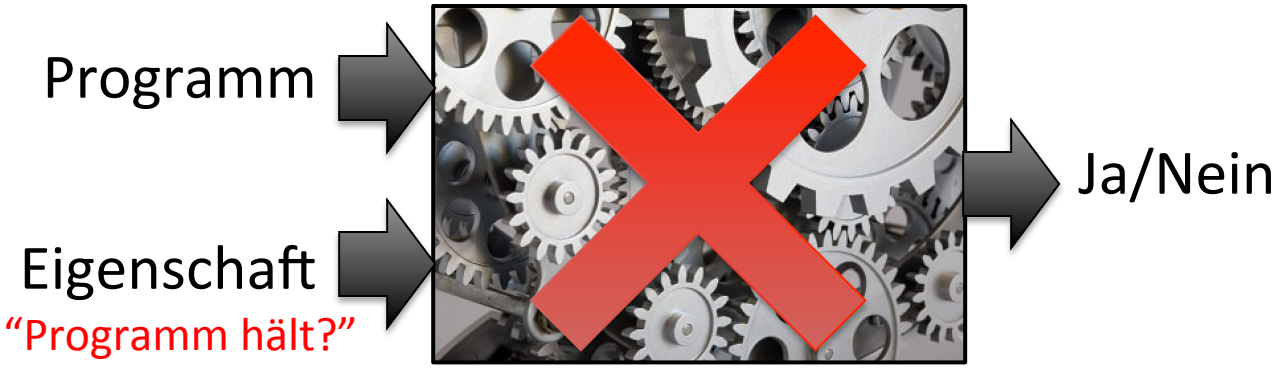
[Received 28 May, 1936.—Read 12 November, 1936.]

[*Extracted from the Proceedings of the London Mathematical Society, Ser.* 2, *Vol.* 42, 1937.]

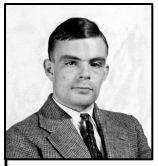
The "computable" numbers may be described briefly as the real numbers whose expressions as a decimal are calculable by finite means. Although the subject of this paper is ostensibly the computable *numbers*, it is almost equally easy to define and investigate computable functions of an integral variable 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

1912 2012

Automatisierte Fehlersuche?



Mission impossible…

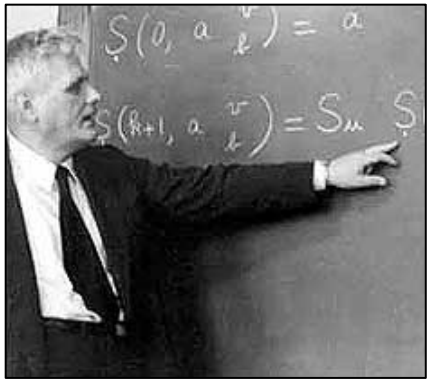


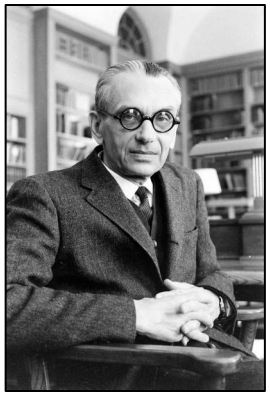
A. Turing

1936 Halting Problem

1912 2012

Fehler automatisch finden?

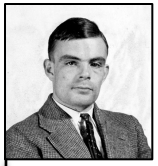




Alonzo Church, 1936:
An Unsolvable Problem of Elementary Number Theory

Kurt Gödel, 1931:
Über formal unentscheidbare Sätze der Principia Mathematica und verwandter Systeme

Mission Impossible…

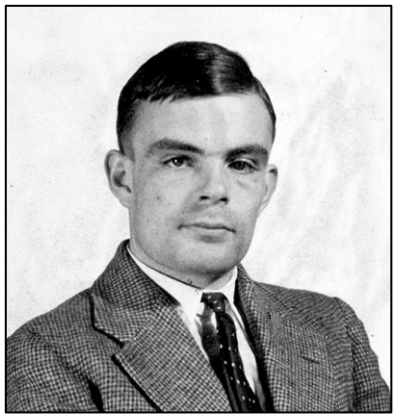


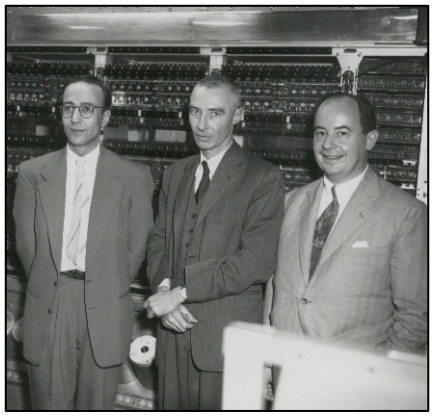
A. Turing

1931 1936 Halting Problem

1912 2012

“Software” im 2. Weltkrieg

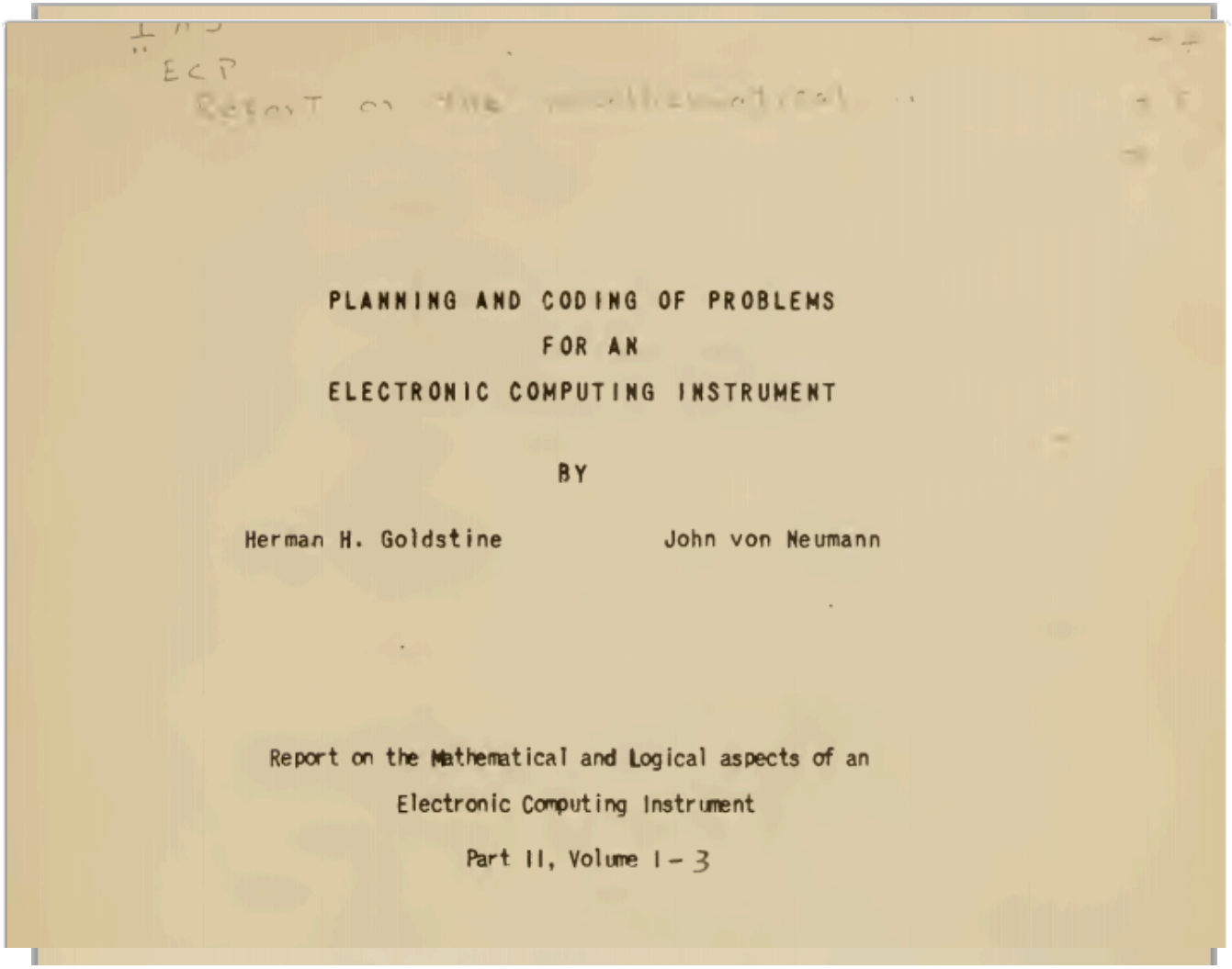




Herman Goldstine
J. Robert Oppenheimer
John von Neumann

A. Turing — 1931 — 1936 Halting Problem

1912 — 2012



PLANNING AND CODING OF PROBLEMS
FOR AN
ELECTRONIC COMPUTING INSTRUMENT

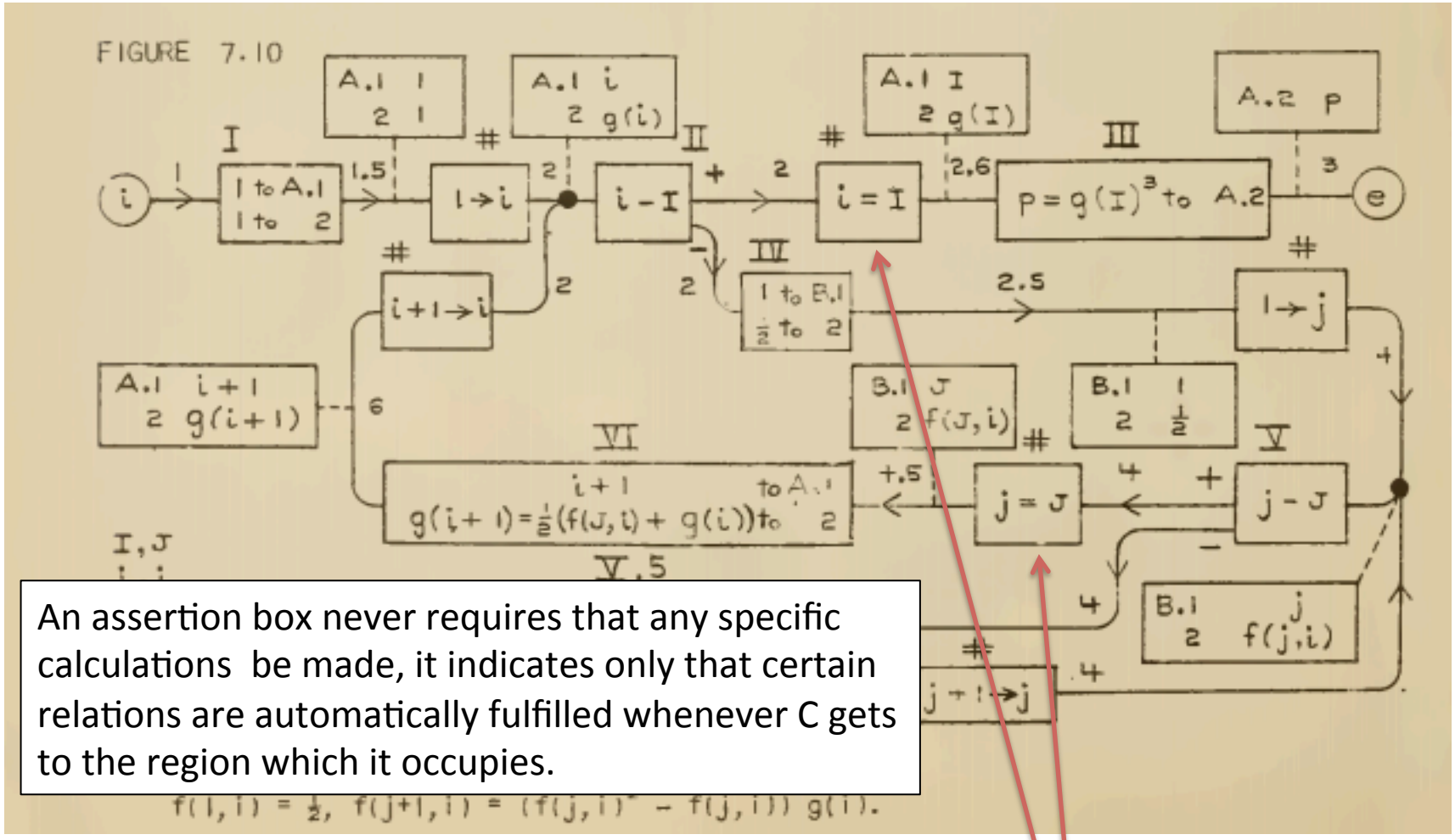
BY

Herman H. Goldstine John von Neumann

Report on the Mathematical and Logical aspects of an
Electronic Computing Instrument
Part II, Volume I - 3

A. Turing 1931 1936 Halting Problem 1947

1912 2012



An assertion box never requires that any specific calculations be made, it indicates only that certain relations are automatically fulfilled whenever C gets to the region which it occupies.

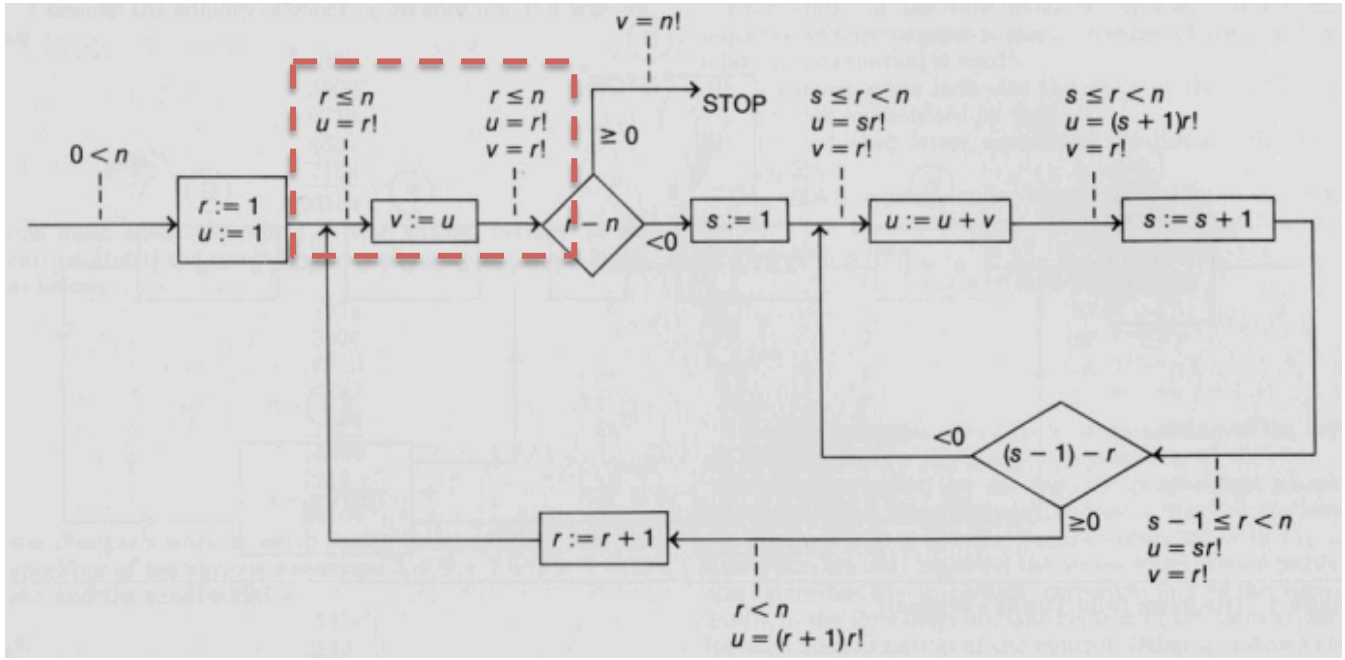
assertion boxes

Friday, 24th June.

Checking a large routine. by Dr. A. Turing.

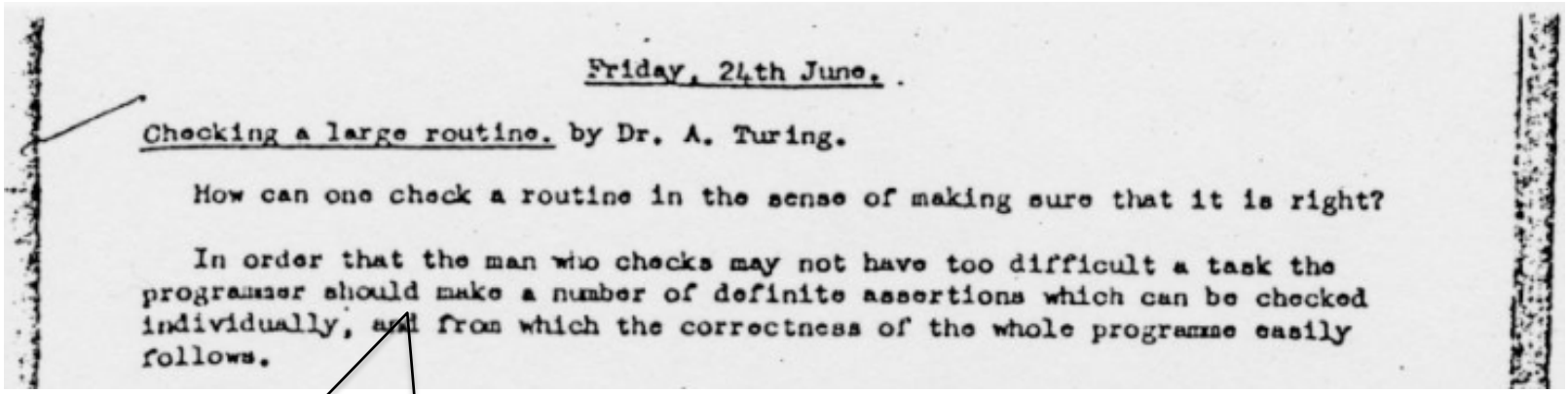
How can one check a routine in the sense of making sure that it is right?

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 from which the correctness of the whole programme easily follows.



A. Turing 1931 1936 Halting Problem 1947 1949 Assertions

1912 2012



Friday, 24th June.

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 from which the correctness of the whole programme easily follows.

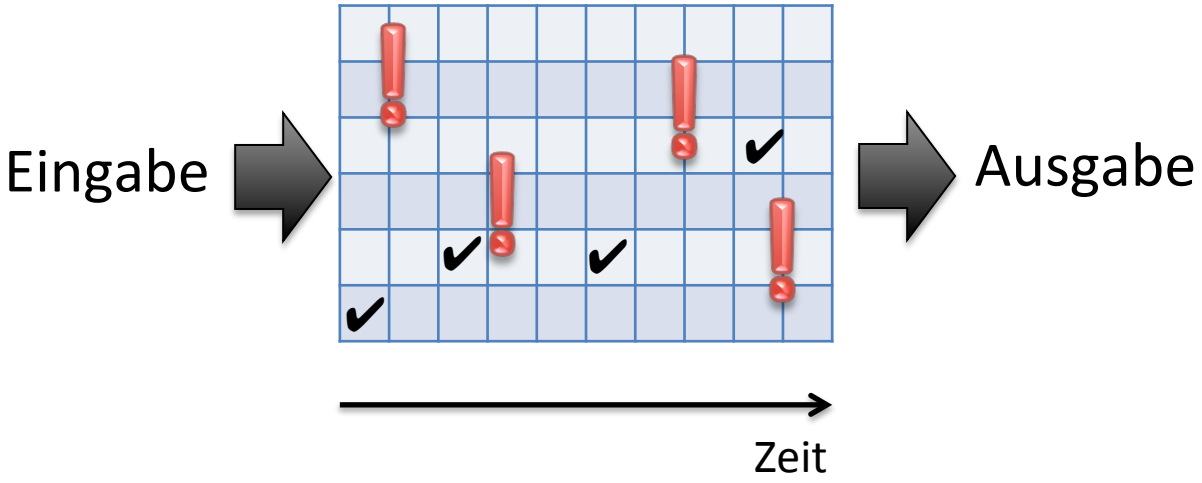
der Programmierer sollte eine Reihe eindeutiger Behauptungen („assertions“) machen, die unabhängig voneinander überprüft werden können...

Kurze Umfrage:

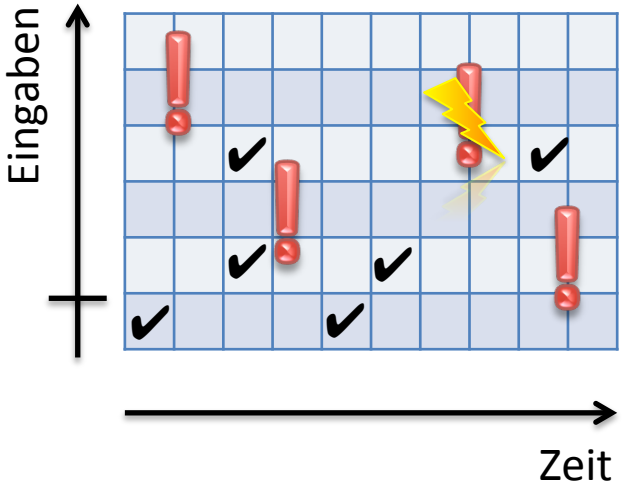
- ✓ Wer von Ihnen programmiert?
- ✓ Wer weiß, was „Assertions“ sind?
- ✓ Wer verwendet Assertions?

~~Automatisierte~~ Fehlersuche?

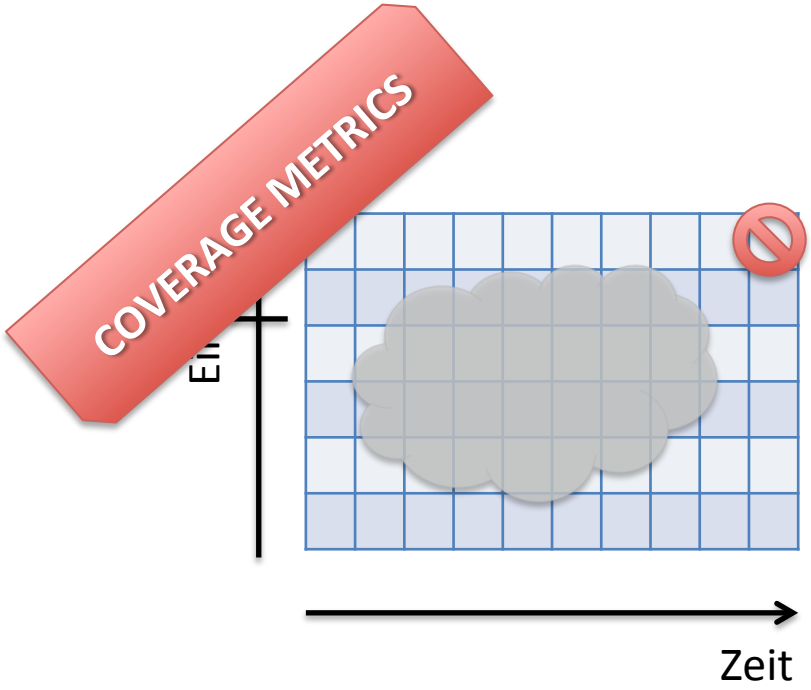
~~Automatisierte~~ Fehlersuche?



~~Automatisierte~~ Fehlersuche?



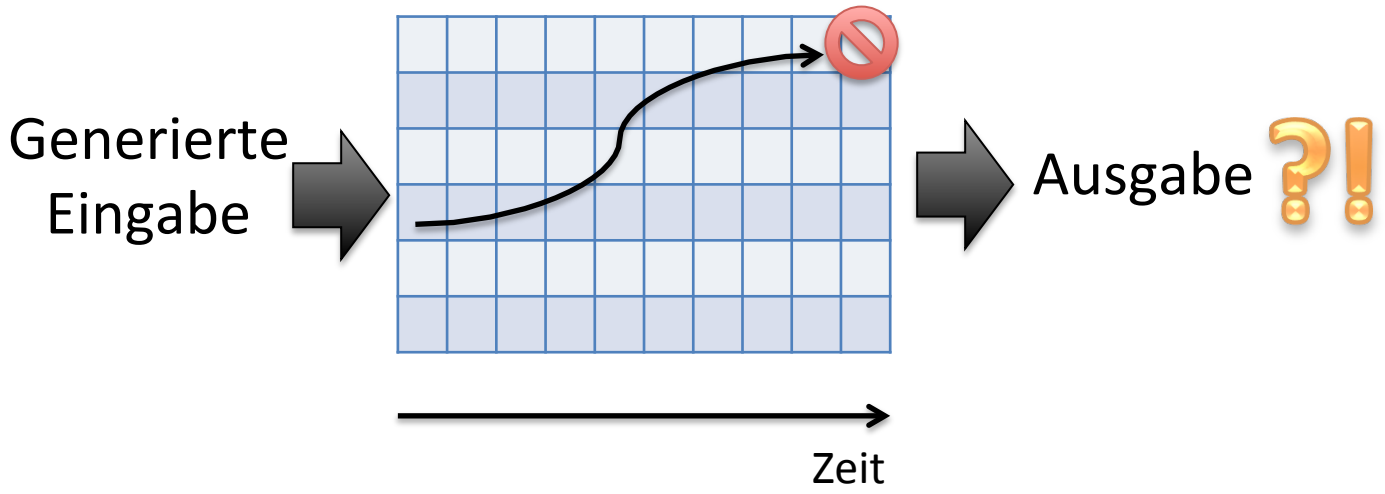
~~Automatisierte~~ Fehlersuche?



Kurze Umfrage:

- ✓ Wer von Ihnen testet (systematisch)?
- ✓ Welche Coverage Metriken kennen Sie?

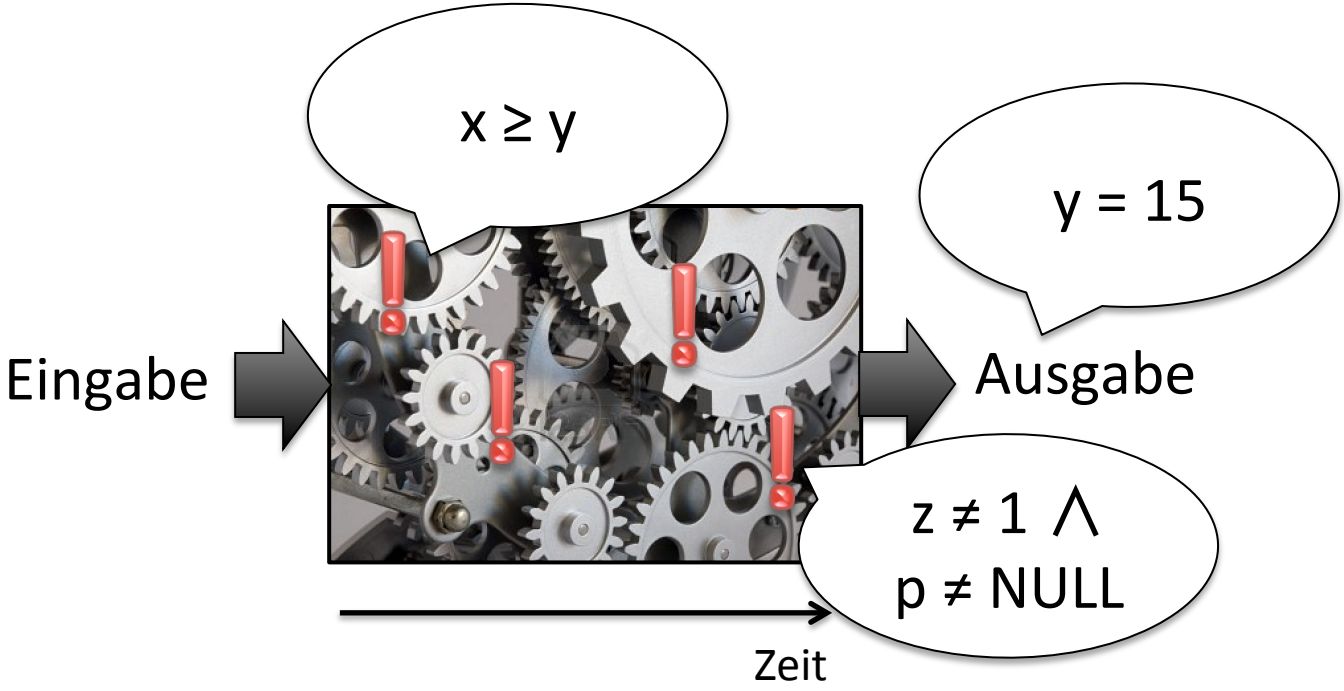
Automatisierte Testfallgenerierung



Vorlesungsinhalte

- Teil 1: Testen und Test-Tools
  - Assertions und Coding Styles
  - Coverage Kriterien und Metriken
  - Testfallgenerierung

Spezifikationen/Assertion Languages



Spezifikationen und Assertions

$x \geq y$

$y = 15$

Kurze Umfrage:

- ✓ Wann müssen Assertions halten?
- ✓ Wenn alle Assertions halten, ist das Programm korrekt?

$z \neq 1 \wedge p \neq \text{NULL}$

Spezifikationen und Assertions

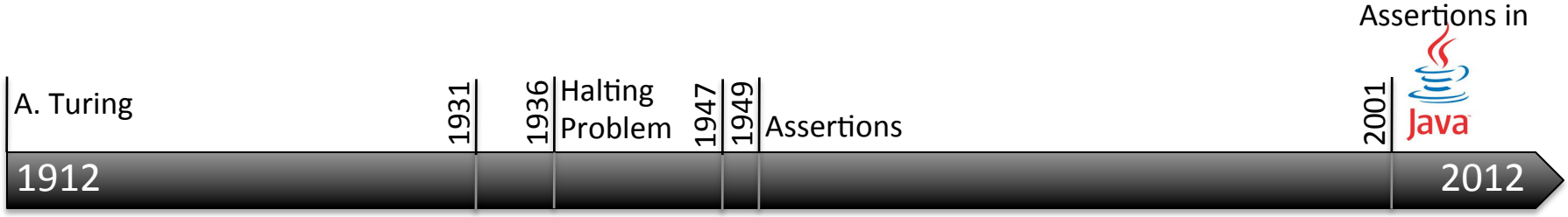
$x \geq y$

$y = 15$

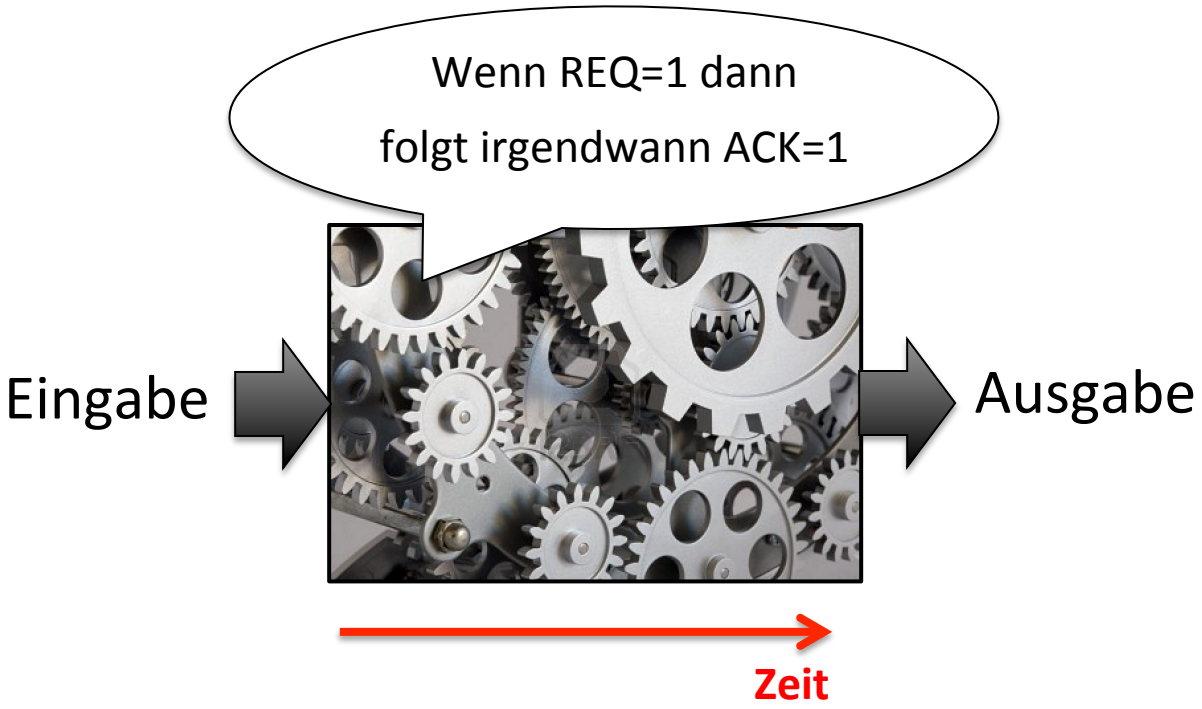
Noch eine Umfrage:

- ✓ In welcher Sprache sind Assertions geschrieben?
- ✓ Was kann man mit Assertions nicht ausdrücken?

$z \neq 1 \wedge p \neq \text{NULL}$



Spezifikationen/Assertion Languages



Spezifikationen/Assertion Languages

Wenn REQ=1 dann

folgt irgendwann ACK=1

Umfrage:

- ✓ Wie kann diese Assertion verletzt werden?
- ✓ Können wir das durch Testen feststellen?

Amir Pnueli, 1977:
Linear Temporal Logic

Spezifikationen/Assertion Languages

Vorlesungsinhalte

- Teil 1: Testen
  - Assertions und Coding Styles
  - Coverage Kriterien und Metriken
  - Testfallgenerierung

März

- Teil 2: Tools aus der Logik
  - Aussagenlogik (und SAT Solver)
  - Prädikatenlogik (und SMT Solver)
  - Temporallogik

April

Model Checking

Edmund Clarke (Dr.hc. TU)
Allen Emerson
Joseph Sifakis

- Idee:
  - Assertions in Temporaler Logik für Programme mit endlichem Zustandsraum verifizierbar
  - ➔Modelle statt Programme
  - Alle erreichbaren Zustände durchsuchen!
  - Funktioniert auch für nebenläufige Modelle


State Space Explosion

Symbolic Model Checking

Ken McMillan

- Idee:
  - Zustände durch Formeln darstellen (eigentlich Binary Decision Diagrams)
  - Implementierung: SMV Model Checker

$$x \vee y$$

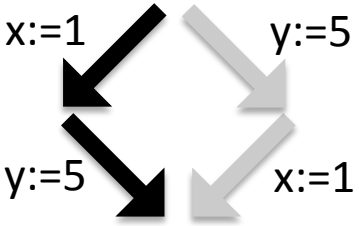
X	Y
0	1
1	0
1	1

Partial Order Reduction


Peled


Godefroid


Valmari




G. Holzmann

- Idee:
  - Nicht alle Programmabfolgen relevant
  - Implementierung: SPIN Model Checker
    - Modellierungssprache: Promela

Vorlesungsinhalte

- Teil 1: Testen
  - Assertions und Coding Styles
  - Coverage Kriterien und Metriken
  - Testfallgenerierung

März

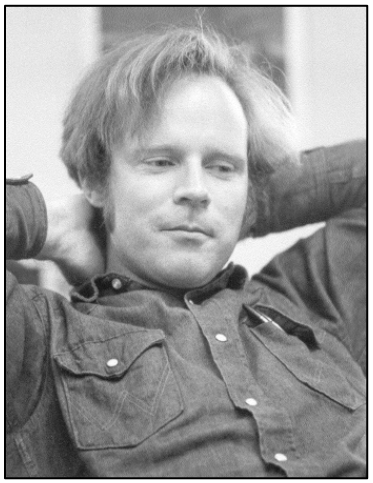
- Teil 2: Tools aus der Logik
  - Aussagenlogik (und SAT Solver)
  - Prädikatenlogik (und SMT Solver)
  - Temporallogik

April

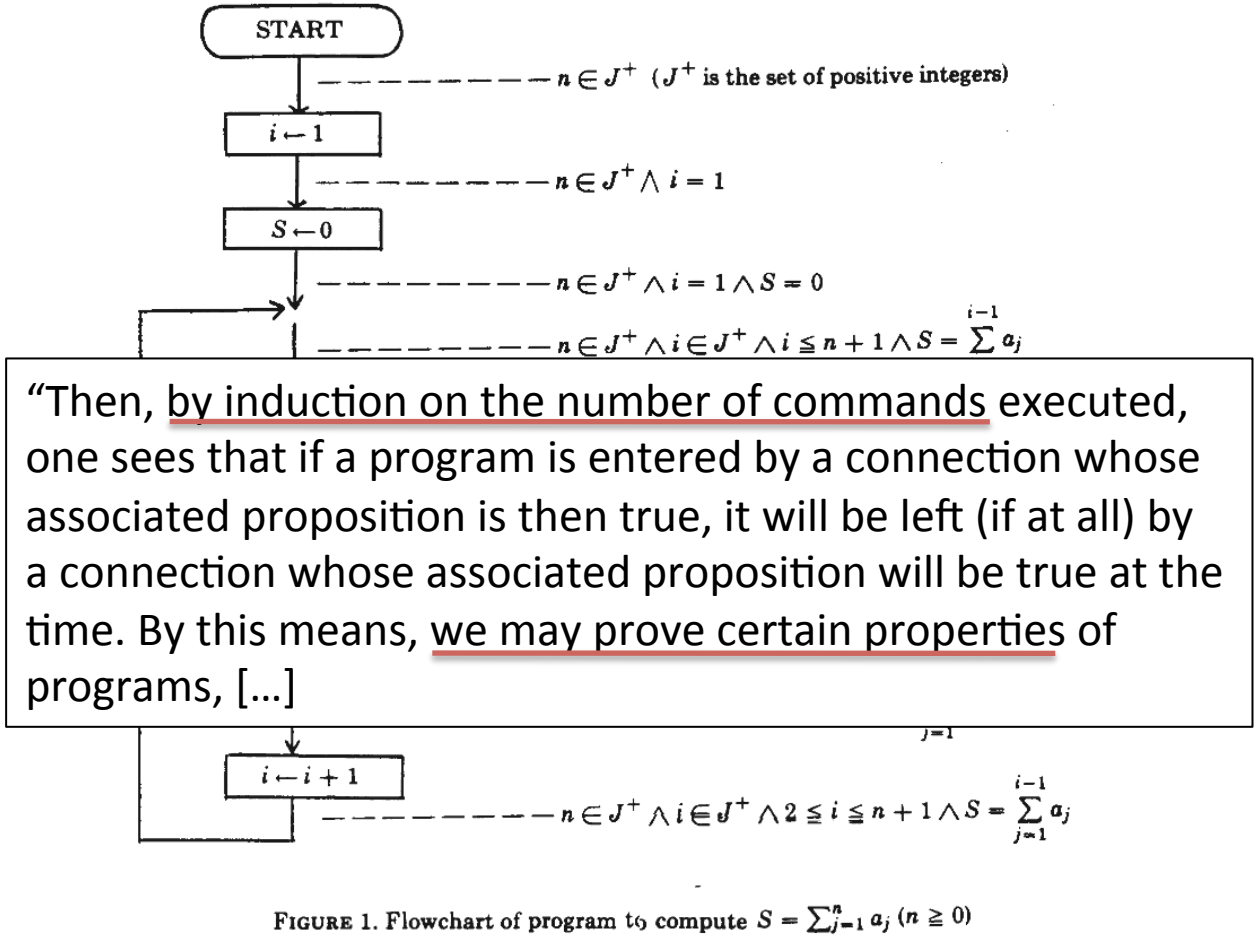
- Teil 3: Verifikation von Modellen
  - SMV (symbolisches Model Checking)
  - SPIN (Partial Order Reduction)

Mai

Assertions und Programmsemantik



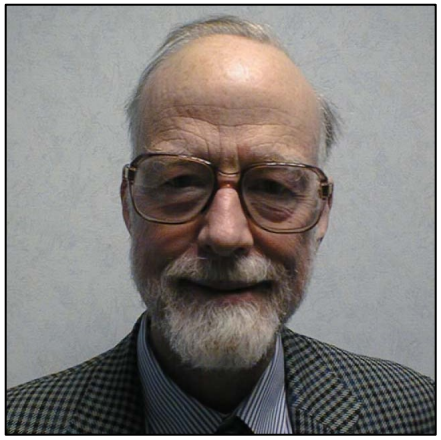
Robert W. Floyd



"Then, by induction on the number of commands executed, one sees that if a program is entered by a connection whose associated proposition is then true, it will be left (if at all) by a connection whose associated proposition will be true at the time. By this means, we may prove certain properties of programs, […]

FIGURE 1. Flowchart of program to compute $S = \sum_{j=1}^{n} a_j$ $(n \geqq 0)$

Floyd-Hoare Logik: Axiome für Programme



Sir C.A.R. Hoare

Instruktion

$$\{ P \} S \{ Q \}$$

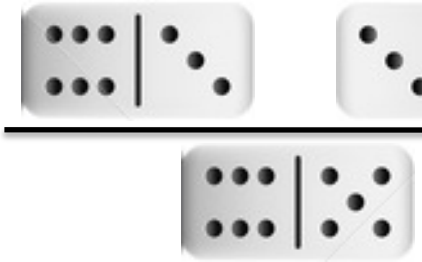
Vorbedingung

Nachbedingung

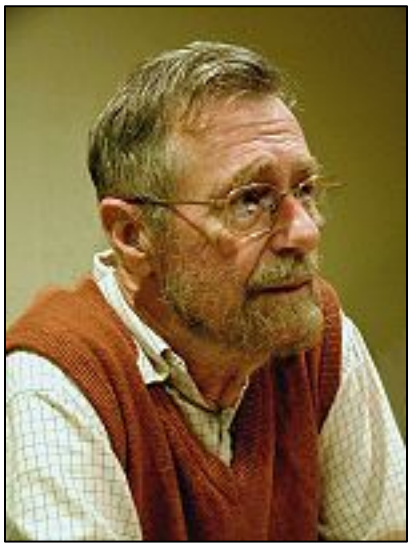
$$\frac{}{\{ Q[x/e] \}\ x := e\ \{ Q \}}$$

$$\frac{\{ P \} S \{ Q \} \qquad \{ Q \} T \{ R \}}{\{ P \} S ; T \{ R \}}$$

$$\frac{}{\{ x = 1 \}\ y := 1\ \{ x = y \}}$$



Dijkstra's Prädikatenkalkül



Edsger W. Dijkstra

Welchen *Effekt* hat eine Instruktion auf eine Assertion?

$$\{ x < 10 \}\ x := x + 1\ \{ ? \}$$

$$sp(x := E, P) = \exists x_0.\ x = E[x/x_0] \wedge P[x/x_0]$$

$$sp(x := x+1, x < 10) = \exists x_0.\ (x = x_0 + 1) \wedge (x_0 < 10)$$

"vorheriger" Wert von x

Bounded Model Checking (BMC)

- Automatisches Beweisen von Korrektheit lt. Turing nicht möglich!
- Aber: Können mit Hoare Logik und symbolischer Simulation Programme bis zu beschränkter Anzahl von Schritten simulieren.
- Tool: C Bounded Model Checker CBMC
  - Für Fehlersuche in ANSI-C Programmen

Daniel Kröning (Oxford)

Vorlesungsinhalte

- Teil 1: Testen
  - Assertions und Coding Styles
  - Coverage Kriterien und Metriken
  - Testfallgenerierung

  } März
- Teil 2: Tools aus der Logik
  - Aussagenlogik (und SAT Solver)
  - Prädikatenlogik (und SMT Solver)
  - Temporallogik

  } April
- Teil 3: Verifikation von Modellen
  - SMV (symbolisches Model Checking)
  - SPIN (Partial Order Reduction)

  } Mai
- Teil 4: Verifikation von Programmen
  - Einführung Hoare Logik
  - Bounded model Checking von C Programmen

  } Ende Mai

Vorlesungsmodus

- 6 ECTS (4.5 Stunden)
- Nächste Vorlesung: 5. März 2015 (siehe TISS)
- VU: Vorlesung mit Übung
  Anwendung von Verifikations- und Testwerkzeugen, theoretische Aufgabenblätter
  Übungsabgaben werden bewertet, Teil der Note (50%)
  Voraussichtlich **5 bis 6 Übungsabgaben**
  - **Assertions: Ausgabe 18./19. März, Abgabe Anfang April**
  - **Coverage/Test Case Generation: Ausgabe 25./ 26. März, Abgabe nach Ostern**
  - **Logik/SAT/SMT: Ausgabe 15./16. April, Abgabe Anfang Mai**
  - **Temporal Logic & Model Checking**
  - **Hoare Logik/BMC: Ausgabe 20/21. Mai, Abgabe Anfang/Mitte Juni**
- Prüfung
  Voraussichtlich Mitte Juni 2015 (17. oder 18. Juni)