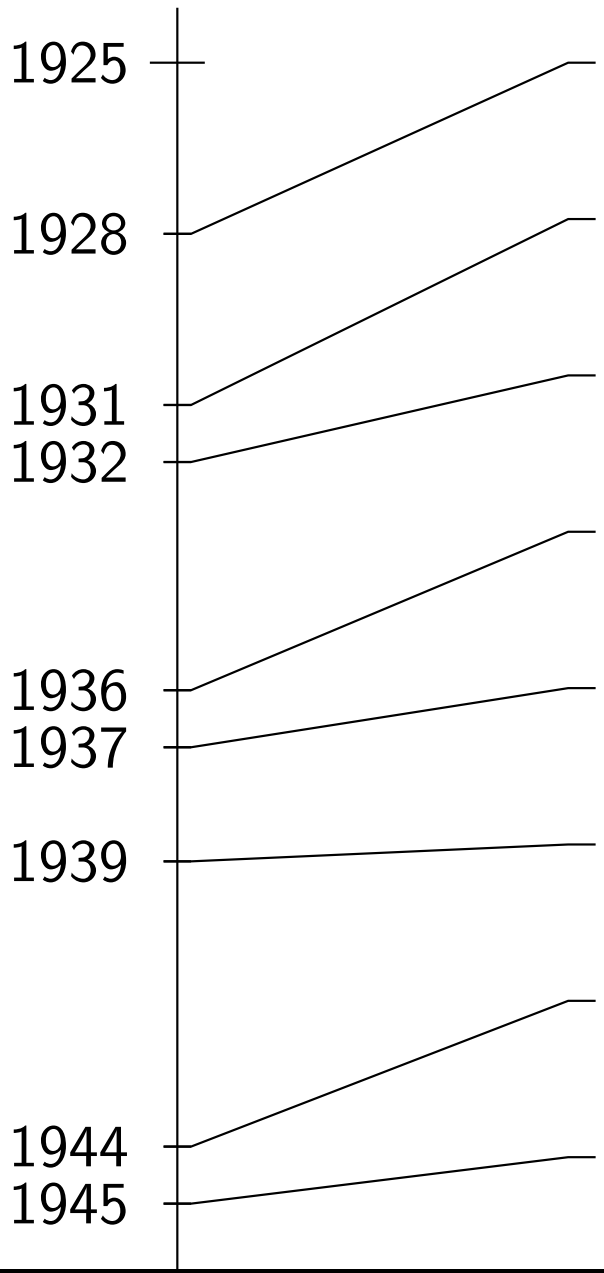
Church-Turing These

Een nieuw paradijs

Pieter van Engelen

Radboud Universiteit Nijmegen

03-06-2022; Fontys, Sittard



Formulering *Entscheidungsproblem* in *Grundzüge der theoretischen Logik*

Onvolledigheidsstellingen van Gödel

Introductie van de λ-calculus (A. Church)

Negatieve oplossing *Entscheidungsproblem*

Introductie *turing machine*, *halting problem* en negatieve oplossing *Entscheidungsproblem*

Systems of logics based on ordinals (Turing)

Introductie Turing-degree (E. Post)

First Draft of a Report on the EDVAC (von Neumann)

De These

Every *effectively calculable* function is *computable*

Church (1936), Turing (1937)

De protagonisten

Alonzo Church (1903 - 1995)

Princeton University, USA

- Logicus, wiskundige
- Van 1936 tot 1979 redacteur van *Journal of Symbolic Logic*
- 'Bedenker' van de λ-calculus
- Eerste-orde predicaat-logica is onbeslisbaar
- Peano-arithmetiek is onbeslisbaar

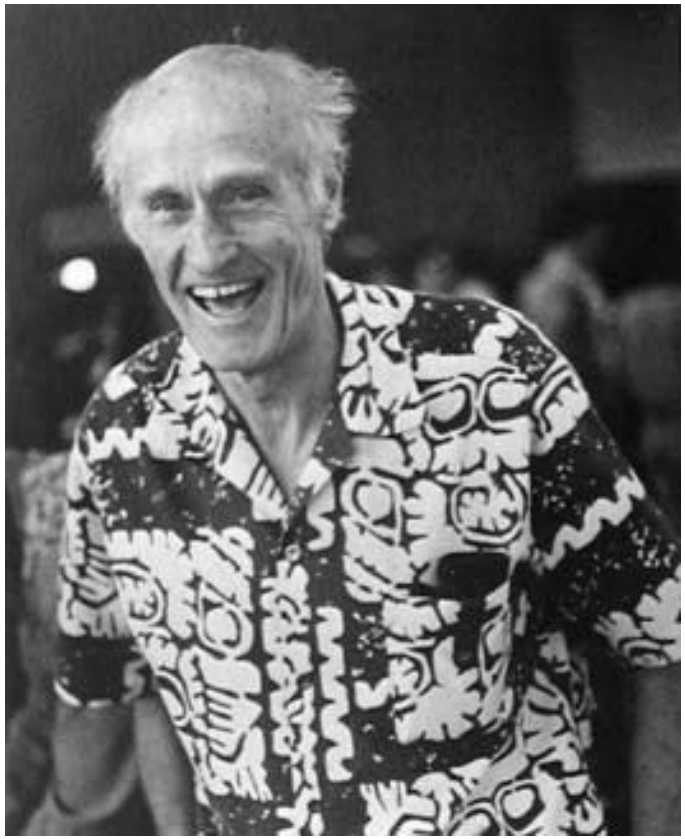
De protagonisten

Alan Turing (1912 - 1954)

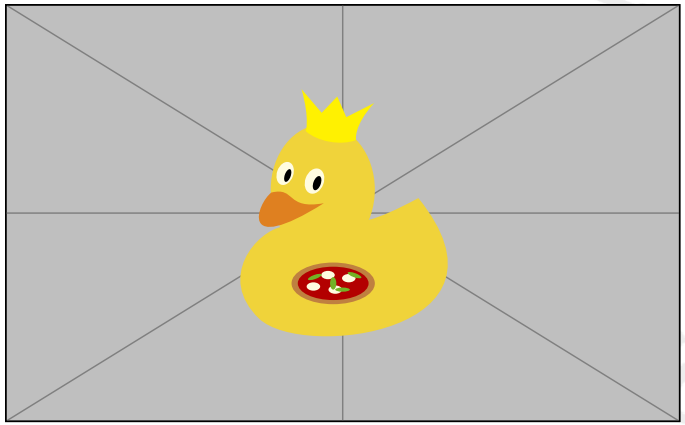
Cambridge & Manchester

- Grondlegger van
  - Informatica
  - Artificiële intelligentie
  - Morphogenetica
- Legendarisch codebreaker
- Marathonloper

De protagonisten



Stephen Kleene (1909-1994)



??? (1897 - 1954)

Das Entscheidungsproblem

Das Entscheidungsproblem

Vind een algoritme waarmee
de waarheid van een uitspraak in de eerste orde predikaatlogica
vast te stellen is.

(D. Hilbert & W. Ackermann, 1928, Grundzüge der theoretischen Logik)

Entscheidungsproblem

Eerste orde predikaatlogica
(extreem kort door de bocht)

Logica met

- variabelen
- de gebruikelijke operatoren $\wedge, \vee, \rightarrow, \neg, \ldots$
- predikaten $P(x)$
- universele en existentiële kwantificatie $\forall, \exists$

Voorbeelden:

$$\forall_{n \in \mathbb{N}} \exists_{m \in \mathbb{N}} [m > n]$$

$$\forall_{p,q \in \mathbb{Q}} \exists_{r \in \mathbb{Q}} [p < r < q]$$

$$\exists_x [P(x) \wedge \forall_y \forall_{y'} [P(y) \wedge P(y') \rightarrow y = y']]$$

Entscheidungsproblem

Eerste orde predikaatlogica

Afspraak:

We hebben het alleen over predikaten en kwantificatie over de natuurlijke getallen $\mathbb{N}$

Gezocht:

Algoritme wat gegeven een uitspraak roept of die uitspraak `WAAR` of `ONWAAR` is.

Probleem:

Wat is een algoritme?

Wat is een algoritme

Wat is een algoritme??

- Grootste-gemene-deler van Euclides

Wat is een algoritme

Wat is een algoritme??

- Grootste-gemene-deler van Euclides
- Zeef van Eratosthenes

Wat is een algoritme

Wat is een algoritme??

- Grootste-gemene-deler van Euclides
- Zeef van Eratosthenes
- Gauss-eliminatie

Wat is een algoritme

Probleem: Nog geen *formele* definitie van een *algoritme*.

Wat is een algoritme

Probleem: Nog geen *formele* definitie van een *algoritme*.

Terug naar 1936.

Wat is een algoritme

Probleem: Nog geen *formele* definitie van een *algoritme*.

Terug naar 1936-ish.

Wat is een algoritme

Probleem: Nog geen *formele* definitie van een *algoritme*.

Terug naar 1936-ish.

- Turing machines

Wat is een algoritme

Probleem: Nog geen *formele* definitie van een *algoritme*.

Terug naar 1936-ish.

- Turing machines
- Recursietheorie

Wat is een algoritme

Probleem: Nog geen *formele* definitie van een *algoritme*.

Terug naar 1936-ish.

- Turing machines
- Recursietheorie
- λ-calculus

De λ-calculus (*Church 1932*)

De programma's

$$x, y, \ldots \in \Lambda \text{ (Variabelen)}$$
$$M, N \in \Lambda \Rightarrow MN \in \Lambda \text{ (Applicatie)}$$
$$x, M \in \Lambda \Rightarrow (\lambda x.M) \in \Lambda \text{ (Abstractie)}$$

- $\lambda x.x$
- $\lambda xy.x$
- $\lambda pqr.pr(qr)$
- $(\lambda x.xx)A$
- $\lambda x.y$
- $\lambda fx.f(f(f(x))) \equiv \ulcorner 3 \urcorner$

De λ-calculus (*Church 1932*)

Actie

$$(\lambda x.M)N \longrightarrow_\beta M[x := N]$$

De λ-calculus (*Church 1932*)

Actie

$$(\lambda x.M)N \longrightarrow_\beta M[x := N]$$

Voorbeeld

$$(\lambda xyz.zxy)(\lambda x.xx)(\lambda x.x)(\lambda xy.x) \rightarrow_\beta$$

De λ-calculus (*Church 1932*)

Actie

$$(\lambda x.M)N \longrightarrow_\beta M[x := N]$$

Voorbeeld

$$(\lambda xyz.zxy)(\lambda x.xx)(\lambda x.x)(\lambda xy.x) \rightarrow_\beta$$
$$(\lambda yz.z(\lambda x.xx)y)(\lambda x.x)(\lambda xy.x) \rightarrow_\beta$$

De λ-calculus (*Church 1932*)

Actie

$$(\lambda x.M)N \longrightarrow_\beta M[x := N]$$

Voorbeeld

$$\begin{aligned}(\lambda xyz.zxy)(\lambda x.xx)(\lambda x.x)(\lambda xy.x) &\to_\beta \\ (\lambda yz.z(\lambda x.xx)y)(\lambda x.x)(\lambda xy.x) &\to_\beta \\ (\lambda z.z(\lambda x.xx))\lambda x.x(\lambda xy.x) &\to_\beta\end{aligned}$$

De λ-calculus (*Church 1932*)

Actie

$$(\lambda x.M)N \longrightarrow_\beta M[x := N]$$

Voorbeeld

$$\begin{aligned}
(\lambda xyz.zxy)(\lambda x.xx)(\lambda x.x)(\lambda xy.x) &\to_\beta \\
(\lambda yz.z(\lambda x.xx)y)(\lambda x.x)(\lambda xy.x) &\to_\beta \\
(\lambda z.z(\lambda x.xx))\lambda x.x(\lambda xy.x) &\to_\beta \\
(\lambda xy.x)(\lambda x.xx))\lambda x.x &\twoheadrightarrow_\beta \lambda x.xx
\end{aligned}$$

Recursietheorie (*Kleene 1935*)

Initiële functies

$$\mathcal{O}(x) = 0 \qquad \text{Nul}$$

$$\mathcal{S}(x) = x + 1 \qquad \text{Successor}$$

$$\mathcal{P}_i^n(x_1, \ldots, x_n) = x_i \qquad \text{Projectie}$$

$$f(\vec{x}) = h(g_1(\vec{x}), \ldots, g_m(\vec{x})) \qquad \text{Functie compositie}$$

Primitieve recursie

$$f(\vec{x}, 0) = g(\vec{x}) \qquad \text{0-geval}$$

$$f(\vec{x}, n+1) = h(\vec{x}, y, f(\vec{x}, y)) \qquad \text{Recursieve geval}$$

μ-recursie

$$f(\vec{x}) = \mu y[g(\vec{x}, y) = 0]$$

"De kleinste y zodat $g(\vec{x}, y) = 0$"

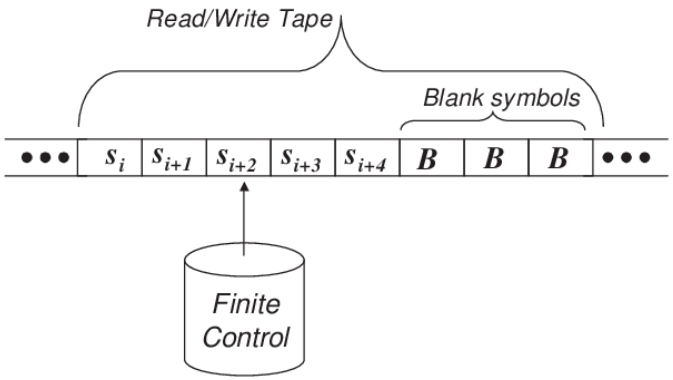
Recursietheorie (*Kleene 1935*)

Voorbeelden

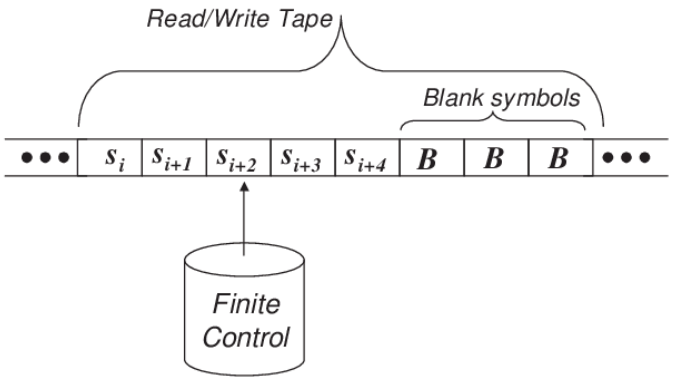
$$\mathcal{P}(0) = 0 \qquad \min(x, 0) = x$$
$$\mathcal{P}(n+1) = n \qquad \min(x, y+1) = \mathcal{P}(\min(x, y))$$

$$f(n) = \mu y[2y = n \vee 2y + 1 = n]$$

Turing machines (*Turing 1936*)



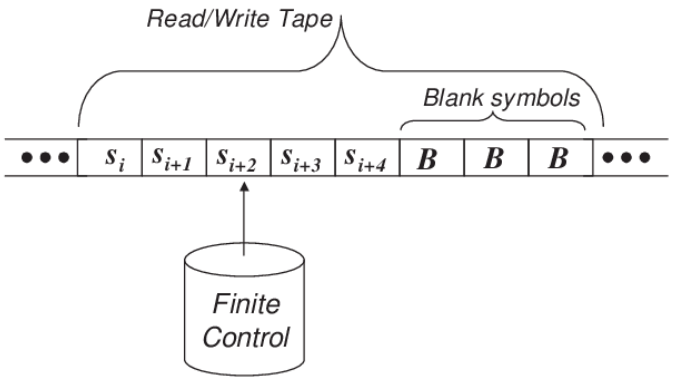
Turing machines (*Turing 1936*)



- Een eindig alfabet

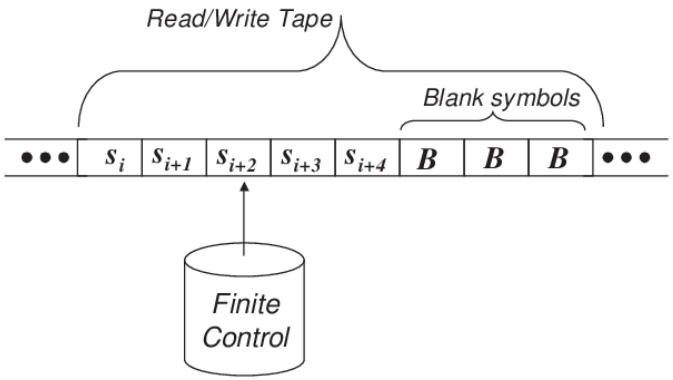
  $s_0, s_1, \ldots, s_n$

Turing machines (*Turing 1936*)



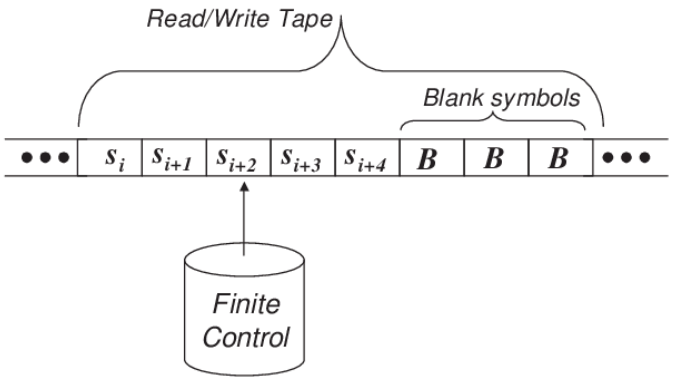
- Een eindig alfabet $s_0, s_1, \ldots, s_n$
- Een eindig aantal toestanden $q_0, q_1, \ldots, q_m$

Turing machines (*Turing 1936*)



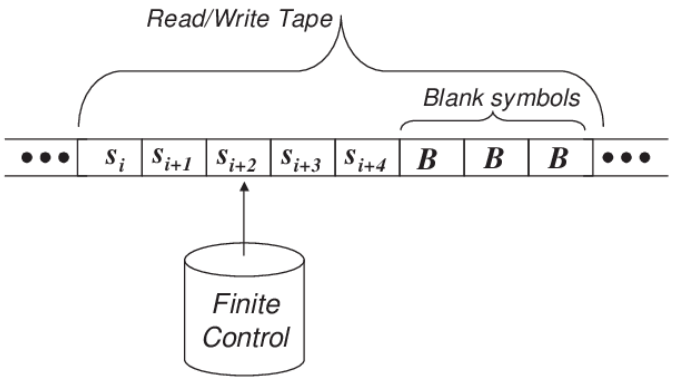
- Een eindig alfabet $s_0, s_1, \ldots, s_n$
- Een eindig aantal toestanden $q_0, q_1, \ldots, q_m$
- Een potentieel oneindige tape voor de symbolen

Turing machines (*Turing 1936*)



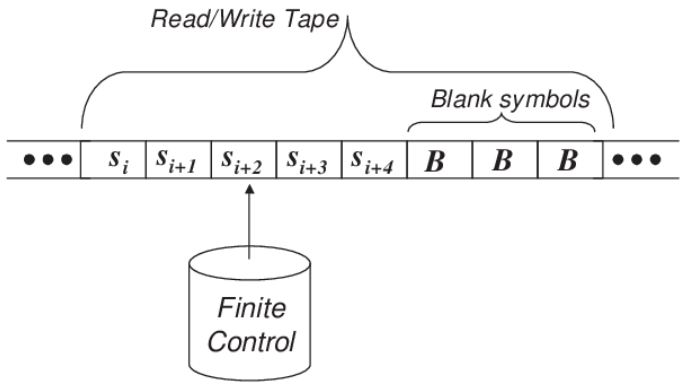
- Een eindig alfabet $s_0, s_1, \ldots, s_n$
- Een eindig aantal toestanden $q_0, q_1, \ldots, q_m$
- Een potentieel oneindige tape voor de symbolen
- De acties: L, R, s_i.

Turing machines (*Turing 1936*)



- Een eindig alfabet $s_0, s_1, \ldots, s_n$
- Een eindig aantal toestanden $q_0, q_1, \ldots, q_m$
- Een potentieel oneindige tape voor de symbolen
- De acties: L, R, s_i.
- Een eindige lijst van instructies

Turing machines (*Turing 1936*)



Voorbeeldinstructies

$q_0\mathbf{0}Rq_1$ Wanneer er in toestand q_0 een $\mathbf{0}$ op de tape staat, zet een stap naar rechts en ga in toestand q_1.

$q_4\mathbf{10}q_8$ Wanneer er in toestand q_4 een $\mathbf{1}$ op de tape staat, vervang de $\mathbf{0}$ door een $\mathbf{1}$ en ga in toestand q_8.

De equivalentie

$$\lambda - \text{definieerbaar} \overset{\text{(Turing 1937)}}{\Longrightarrow} \text{Turing berekenbaar}$$

$$\text{Turing berekenbaar} \overset{\text{(Turing 1937)}}{\Longrightarrow} \mu - \text{recursief}$$

$$\mu - \text{recursief} \overset{\text{(Kleene 1936)}}{\Longrightarrow} \lambda - \text{definieerbaar}$$

De equivalentie

De uitspraken:

- Een functie $f : \mathbb{N} \to \mathbb{N}$ is berekenbaar
- Er bestaat een λ-term F zdd $f(n) = m \Leftrightarrow F\ulcorner n \urcorner = \ulcorner m \urcorner$
- Er bestaat een μ-recursieve functie ϕ zdd $f(n) = m \Leftrightarrow \phi(n) = m$
- Er bestaat een T.M. zdd $f(n) = m \Leftrightarrow$ T.M.$_f$ geeft bij invoer $\ulcorner n \urcorner$ uitvoer $\ulcorner m \urcorner$

zijn synoniem met elkaar.

Halting Problem

lets met oneindigheid

Probleem:
Een algoritme kan eindeloos lang doorgaan, zonder een 'antwoord' te geven.

λ-calculus	$(\lambda x.xx)(\lambda x.xx)$
Recursietheorie	$f(n) = \mu y[y < 0]$
Turing machine	$\{q_0\mathbf{00}q_1, q_1\mathbf{00}q_0\}$

Halting Problem

lets met oneindigheid

Probleem:
Een algoritme kan eindeloos lang doorgaan, zonder een 'antwoord' te geven.

Oplossing:
Schrijf een algoritme wat van een gegeven algoritme P bepaalt of deze bij gegeven invoer n een antwoord geeft.

Halting Problem

lets met oneindigheid

Probleem:
Een algoritme kan eindeloos lang doorgaan, zonder een 'antwoord' te geven.

Oplossing:
Schrijf een algoritme wat van een gegeven algoritme P bepaalt of deze bij gegeven invoer n een antwoord geeft.

Computer says no...

Halting Problem

Theorem (Halting Problem)

Er bestaat geen algoritme wat bepaalt of een gegeven algoritme P stopt bij gegeven invoer n.

Proof.

Stel dat er een algoritme H bestaat (*), wat aan de voorwaarden voldoet. Maak een nieuw algoritme H' op de volgende manier:

$$H'(n) = \begin{cases} \uparrow & \text{als } H(P,n) = 1 \\ 1 & \text{als } H(P,n) \uparrow \end{cases}$$

Beschouw nu $H(H'(n))$. Wanneer H oneindig draait, is H' gedefinieerd, maar dan zou H juist niet oneindig moeten draaien. Tegenspraak. We geven de aanname (*) de schuld. □

Entscheidungsproblem

Feit: Er zijn overaftelbaar veel onoplosbare problemen

Entscheidungsproblem

Feit: Er zijn overaftelbaar veel onoplosbare problemen **Nog zo**

een: Het is *niet* beslisbaar om van een gegeven programma P te zeggen of het uitvoer x geeft.

Entscheidungsproblem

Feit: Er zijn overaftelbaar veel onoplosbare problemen **Nog zo**

een: Het is *niet* beslisbaar om van een gegeven programma P te zeggen of het uitvoer x geeft. **Gevolg:** Het *Entscheidungsproblem*

is niet oplosbaar.

Universaliteits principe

De These

Every *effectively calculable* function is *computable*

Church (1936), Turing (1937)

Elke *uitrekenbare* functie is *berekenbaar*

Hypercomputation

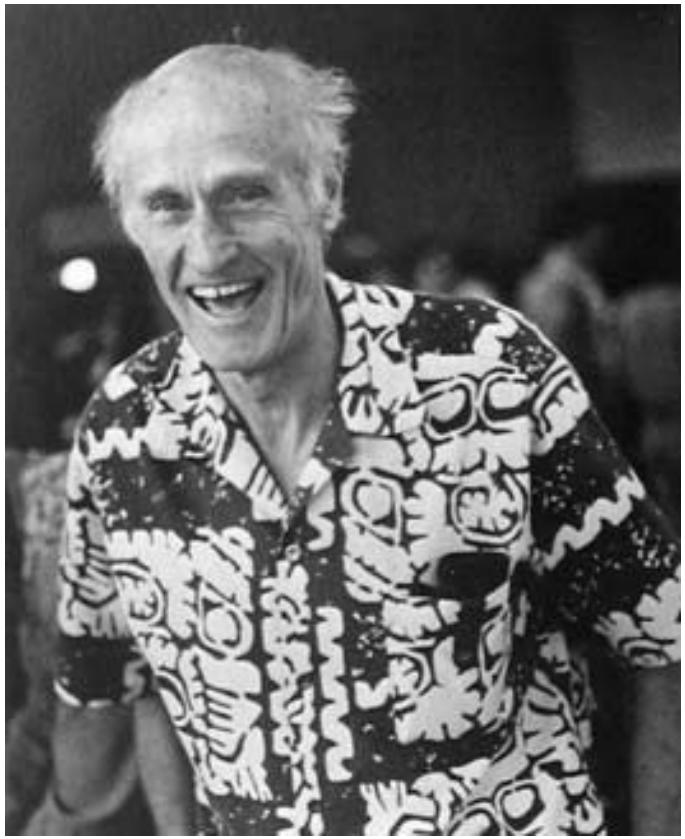
Oracle machines
Infinite state
Transfiniete recursie

Quantum computing

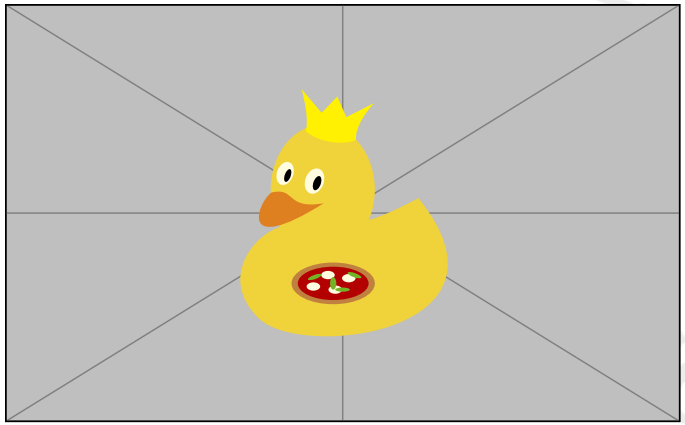
Church Turing Deutsch
Wat doet quantum computing

Tragiek in het paradijs

De protagonisten

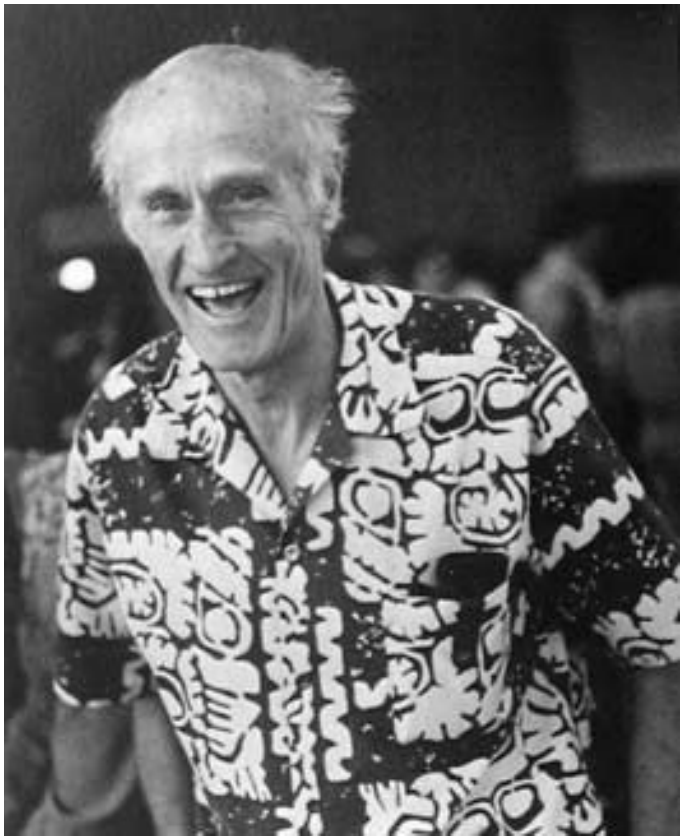


Stephen Kleene (1909-1994)

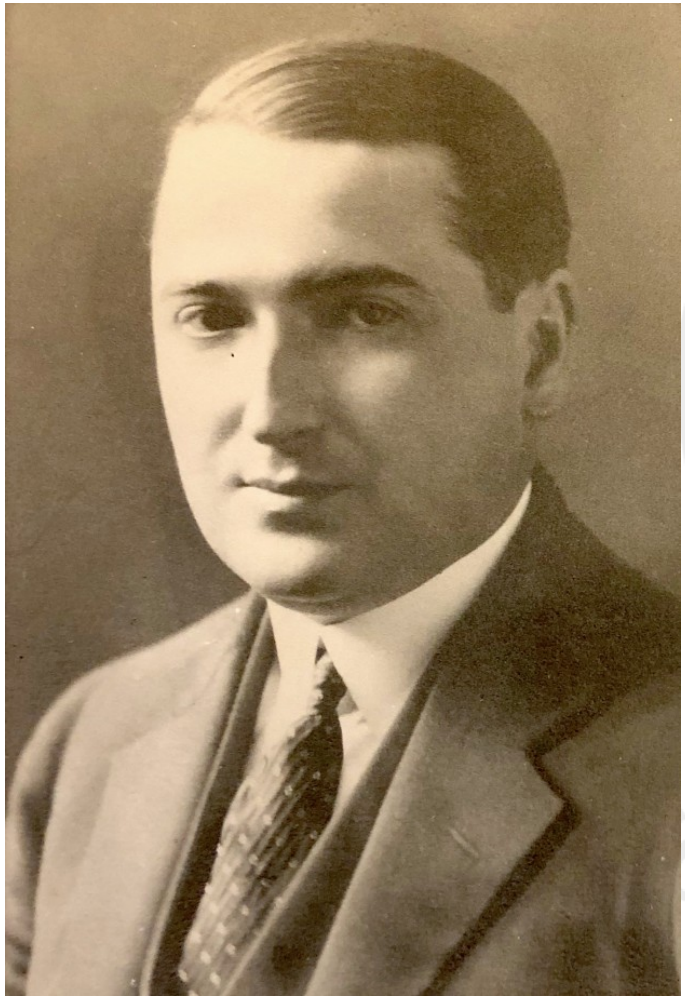


??? (1897 - 1954)

De protagonisten



Stephen Kleene (1909-1994)



Emil Post (1897 - 1954)