

# Church-Turing These

## Een nieuw paradijs

Pieter van Engelen

Radboud Universiteit Nijmegen

03-06-2022

## De tijd

De protagonisten

## De situatie

Entscheidungsproblem

Berekenbaarheidsmodellen

De kracht van berekenbaarheid

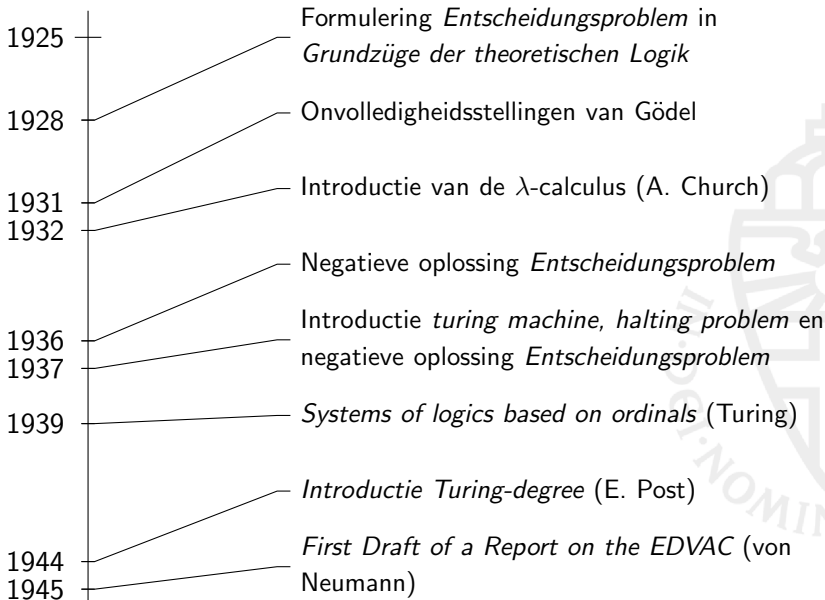
## De these

## Voorbij de these

Hypercomputation

Quantum computing

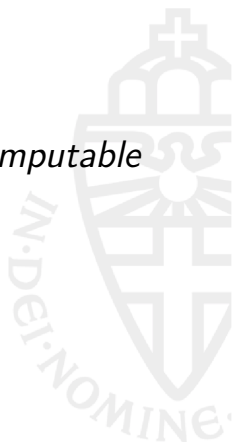




# De These

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  $\lambda$ -calculus
- Eerste-orde predicaat-logica is onbeslisbaar
- Peano-arithmetiek is onbeslisbaar

# De protagonisten

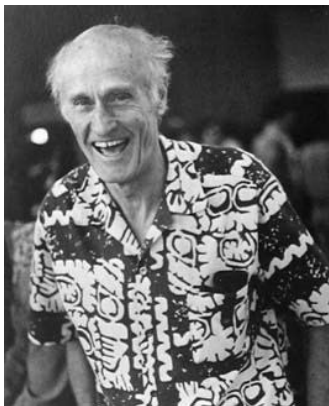


Alan Turing (1912 - 1954)

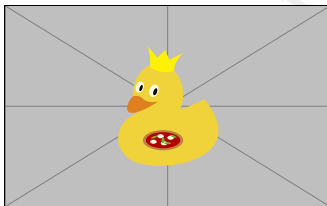
*Cambridge & Manchester*

- 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  $\wedge, \vee, \rightarrow, \neg, \dots$
- 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

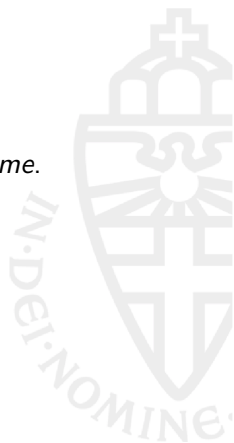
Wat is een algoritme??

- 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
- $\lambda$ -calculus



# De $\lambda$ -calculus (*Church 1932*)

## De programma's

$x, y, \dots \in \Lambda$  (Variabelen)

$M, N \in \Lambda \Rightarrow MN \in \Lambda$  (Applicatie)

$x, M \in \Lambda \Rightarrow (\lambda x.M) \in \Lambda$  (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 $\lambda$ -calculus (*Church 1932*)

## Actie

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



# De $\lambda$ -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 $\lambda$ -calculus (*Church 1932*)

## Actie

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

## Voorbeeld

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



# De $\lambda$ -calculus (*Church 1932*)

## Actie

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

## Voorbeeld

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





# De $\lambda$ -calculus (*Church 1932*)

## Actie

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

## Voorbeeld

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



# Recursietheorie (*Kleene 1935*)

## Initiële functies

$$\mathcal{O}(x) = 0$$

Nul

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

Successor

$$\mathcal{P}_i^n(x_1, \dots, x_n) = x_i$$

Projectie

$$f(\vec{x}) = h(g_1(\vec{x}), \dots, g_m(\vec{x}))$$

Functie compositie

## Primitieve recursie

$$f(\vec{x}, 0) = g(\vec{x})$$

0-geval

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

Rekursieve geval

## $\mu$ -recursie

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

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

# Recursietheorie (*Kleene 1935*)

## Voorbeelden

$$\mathcal{P}(0) = 0$$

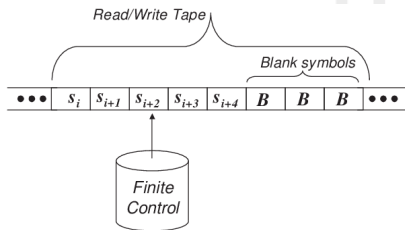
$$\mathcal{P}(n + 1) = n$$

$$\min(x, 0) = x$$

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

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

# Turing machines (*Turing 1936*)

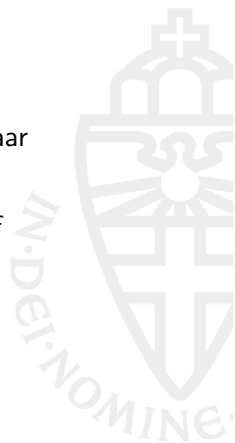


# De equivalentie

$\lambda$  – definieerbaar  $\xRightarrow{\text{(Turing 1937)}} \text{Turing berekenbaar}$

$\text{Turing berekenbaar} \xRightarrow{\text{(Turing 1937)}} \mu$  – recursief

$\mu$  – recursief  $\xRightarrow{\text{(Kleene 1936)}} \lambda$  – definieerbaar



# De equivalentie

De uitspraken:

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

zijn synoniem met elkaar.



# Halting Problem





# Universaliteits principe



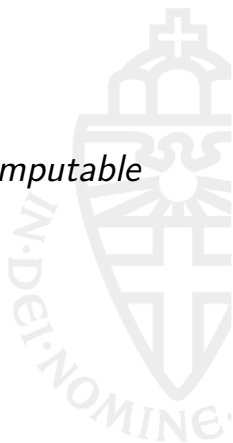


# De These

Every *effectively calculable* function is *computable*

Church (1936), Turing (1937)

Elke *uitrekenbare* functie is *berekenbaar*



# Hypercomputation

Oracle machines  
Infinite state  
Transfinitete recursie



# Quantum computing

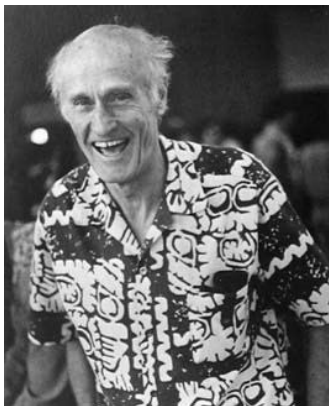
Church Turing Deutsch  
Wat doet quantum computing



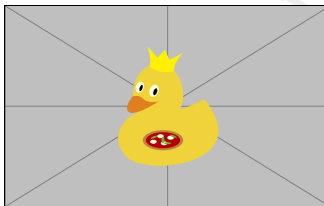
# Tragiek in het paradijs



# De protagonisten

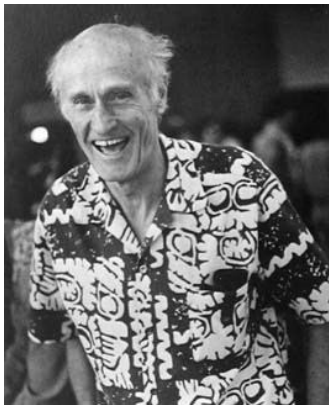


Stephen Kleene (1909-1994)



??? (1897 - 1954)

# De protagonisten



Stephen Kleene (1909-1994)



Emil Post (1897 - 1954)

