



Church-Turing These

Een nieuw paradijs

Pieter van Engelen

Radboud Universiteit Nijmegen

03-06-2022@Fontys, Sittard

De tijd

De protagonisten

De situatie

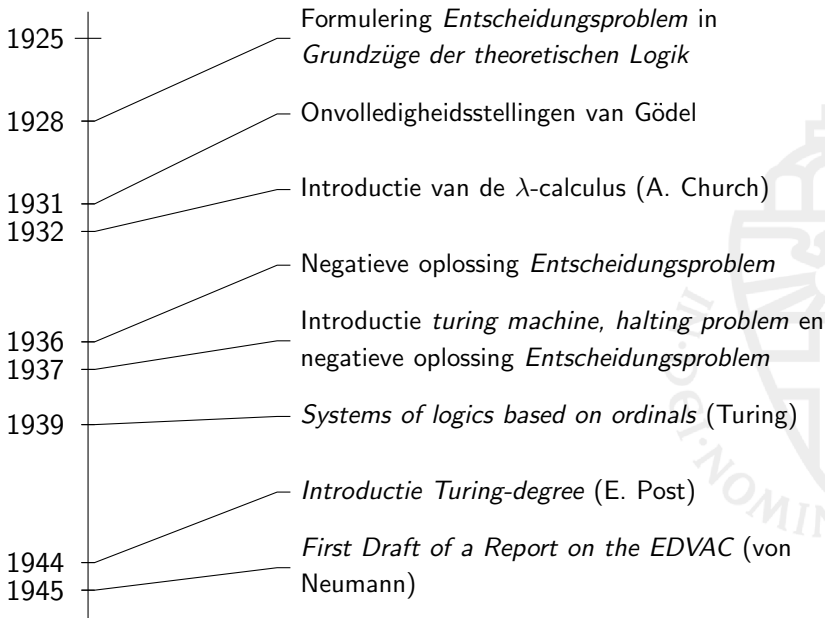
Entscheidungsproblem
Berekenbaarheidsmodellen
De kracht van berekenbaarheid

De these

Voorbij de these

Echte computers
Hypercomputation
Quantum computing



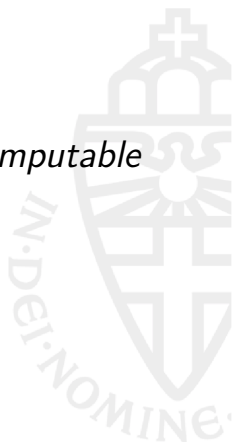




De These

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 λ -calculus
- Eerste-orde predicaat-logica is onbeslisbaar
- Peano-arithmetiek is onbeslisbaar

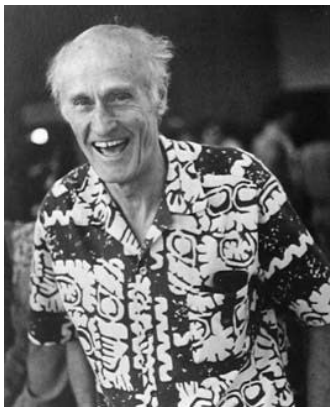
De protagonisten



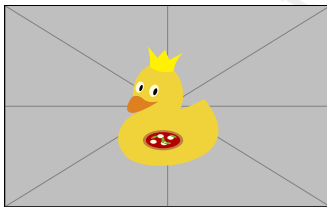
Alan Turing (1912 - 1954) *Cambridge & Manchester*

- Grondlegger van
 - Informatica
 - Artificiële intelligentie
 - Morphogenetica
- Legendarisch codebreaker
- Marathonloper
- Homosexueel in een tijd dat het strafbaar was

De protagonisten



Stephen Kleene (1909-1994)



??? (1897 - 1954)

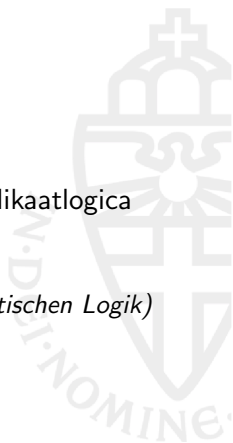


Das Entscheidungsproblem

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??

Reeds informeel bekend in de wiskunde

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





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

$x, y, \dots \in \Lambda$ (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 λ -calculus (*Church 1932*)

Actie

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

Voorbeeld

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

Nu jij: $(\lambda xyz.zxy)$ ("the force")("is")("strong in you")

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

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

Recursieve geval

μ -recursie

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

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



Recursietheorie (*Kleine 1935*)

Voorbeelden

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

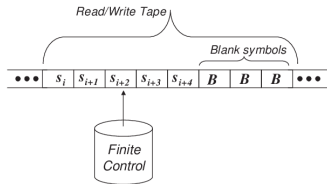
$$\mathcal{P}(n + 1) = n$$

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

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

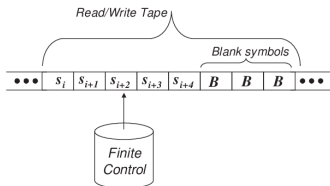
$$f(n) = \mu y[2y = n \vee 2y + 1 = n]$$

Turing machines (*Turing 1936*)



- Een eindig alfabet s_0, s_1, \dots, s_n
- Een eindig aantal toestanden q_0, q_1, \dots, 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

- $q_0 0 R q_1$ Wanneer er in toestand q_0 een **0** op de tape staat, zet een stap naar rechts en ga in toestand q_1 .
- $q_4 1 0 q_8$ Wanneer er in toestand q_4 een **1** op de tape staat, vervang de **0** door een **1** en ga in toestand q_8 .



De equivalentie

λ – definieerbaar $\stackrel{(\text{Turing } 1937)}{\implies}$ Turing berekenbaar

Turing berekenbaar $\stackrel{(\text{Turing } 1937)}{\implies}$ μ – recursief

μ – recursief $\stackrel{(\text{Kleene } 1936)}{\implies}$ λ – definieerbaar



De equivalentie

De uitspraken:

- Een functie $f : \mathbb{N} \rightarrow \mathbb{N}$ 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 $f(n) = m \Leftrightarrow \text{T.M.}_f$ geeft bij invoer $\ulcorner n \urcorner$ uitvoer $\ulcorner m \urcorner$

zijn synoniem met elkaar.



Halting Problem

Iets 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

Iets met oneindigheid

Probleem:

Een algoritme kan eendeloos 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

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

Beslisbaar probleem $P \implies$ T.M voor probleem P

The ugly: Erg veel T.M.'s
Gezocht:

One TM to rule them all



Universaliteits principe

One TM to rule them all

Hoe?

Proof.

Bereken een *beschrijvend getal* e per T.M. M
(een soort Gödelcodering van T.M.'s)

Maak een T.M. U , die een programma e en een invoer vraagt.

Nu geeft $U(e, n)$ het antwoord p op de tape precies wanneer M
antwoord p geeft bij invoer n . □

Machine U noemen we een *Universele Turingmachine*.
In effect een *Interpreter*.



Tussentijdse terugblik

Terugblik

- Formele definitie van een algoritme
- *Berekenbaarheid*
- Best veel onberekenbare problemen
- Reële getallen op een computer, foggeddaboudid.
- De droom van Hilbert in scherven
- One machine to rule them all

Op naar de These!



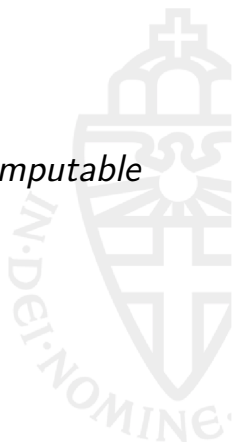


De These

Every *effectively calculable* function is *computable*

Church (1936), Turing (1937)

Elke *uitrekenbare* functie is *berekenbaar*

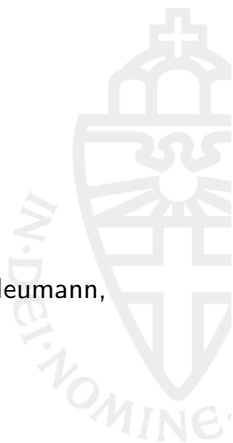




Concrete Turing Machines

- Colossus (1943)
- ENIAC (1946)
- Automatic Computing Engine (1945)
- EDVAC, EDSAC (1949)

Belangrijk concept: *Stored Program Computer* (von Neumann, 1945)



To boldly go where no man has gone before

- Orakel machines (Turing, 1939)
- Infinite state machines (discussie Gödel)
- Black hole
- Quantum computing





Quantum computing

- Superpositie
- Interferentie
- Verstrengeling

Gevolgen voor:

- Cryptografie
- Zoek algoritmen
- Computatieve biologie
- Machine learning?





Quantum computing

Church-Turing-Deutsch These

Een universele computer kan elk fysisch proces simuleren
Gandy 1980, Deutsch 1985

Stelling

De klasse van *berekenbare functies*
is *gelijk* aan
de klasse van *quantum-berekenbare functies*.



Terugblik

- Kennis over algoritmen en berekenbaarheid
- Van abstracte logica naar je broekzak
- Harde grenzen
- *Mer à boire*
- ??





The End

Dankjewel voor je aandacht

Contact: mail@pietervanengelen.nl

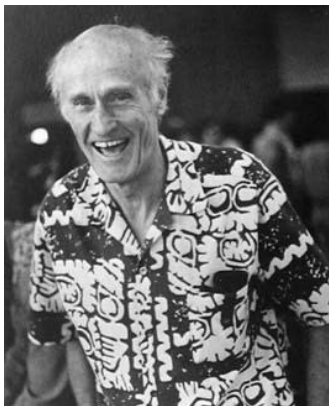




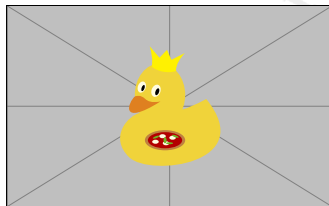
Tragiek in het paradijs



De protagonisten

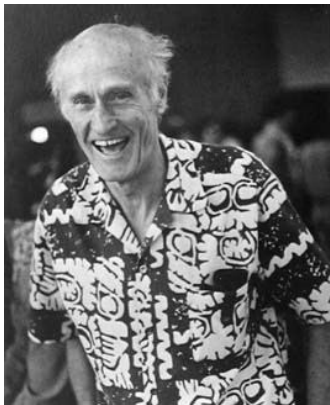


Stephen Kleene (1909-1994)



??? (1897 - 1954)

De protagonisten



Stephen Kleene (1909-1994)



Emil Post (1897 - 1954)

Emil Leon Post



- Logicus en wiskundige
- Bewijs van volledigheid propositiologica uit *Principia Mathematica* (1919)
- Vond in 1920 aanwijzingen voor de Onvolledigheidsstellingen van Gödel en onbeslisbaarheidsresultaten van Church en Turing
- Bipolaire stoornis
- Finite Combinatory Processes (1936)
- Turing-degree/Post's Theorem (1944)