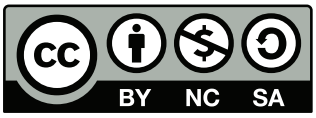
Module “Limieten van de berekenbaarheid”: antwoorden

Gilles Coremans

2018



1 Dr Turinglove or: How I Learned To Stop Worrying And Love The Machine

1.1 Wat is een effectieve procedure?

Een functie is effectief berekenbaar als en slechts als er een effectieve procedure bestaat om deze functie te berekenen.

Een procedure M is effectief als en slechts als:

- M bestaat uit een eindig aantal instructies.
- De uitvoering van M maar een eindig aantal stappen nodig heeft.
- M in principe door een mens met enkel potlood en papier uitgevoerd kan worden (men negeert fysieke limieten zoals tijd, hoeveelheid papier dat nodig is, enz).
- De uitvoering van M geen inzicht, intuïtie of dergelijke vergt. (Dus dat het domweg uitvoeren van de stappen goed genoeg is)

1.2 Wat is een Turingmachine?

Een Turingmachine is een wiskundig model voor een machine die berekeningen uitvoert. Informeel bestaat een Turingmachine uit een tape die oneindig lang is en ingedeeld is in vakjes die juist één teken bevatten. Het andere onderdeel is een schrijfkop die één teken per stap leest en op basis daarvan een nieuw teken kan schrijven, naar links of naar rechts kan gaan, en mogelijks naar een andere toestand overgaat.

Formeel bestaat een Turingmachine uit een aantal verzamelingen:

- Een eindige verzameling toestanden Q.
- Een begintoestand q_0.
- Een eindig alphabet Σ dat minstens uit $\{0, \triangleright\}$ bestaat.
- Een transitieafbeelding δ: $Q \times \Sigma \mapsto Q \times \Sigma \times \{L, R, N\}$ die de huidige toestand en het gelezen teken afbeeldt op de volgende toestand, het geschreven teken en de richting van beweging.

1.3 Wat stelt de Church-Turinghypothese?

De Church-Turinghypothese stelt dat elke functie f die effectief berekenbaar is, Turing-berekenbaar is.

Bemerk dat dit een implicatie is en geen equivalentie. Zie de vraag over de *converse* Church-Turinghypothese.

1.4 Welke argumenten zijn er voor de Church-Turinghypothese?

Deze vraag is een beetje raar en staat niet echt duidelijk in de cursus. Ik heb de papers waarop de tekst gebaseerd is bekeken om een idee te krijgen van wat hier het verwachte antwoord is.

Turing gaf in grote lijnen drie argumenten voor de these:[1]

1. **Intuïtie:** Turing stelde dat iedereen die het concept van een effectieve procedure begrijpt de intuïtie zou hebben dat zo een procedure steeds door een Turingmachine berekenbaar is.

2. **Equivalentie:** Elke poging om berekenbare functies te formaliseren is equivalent gebleken met Turingmachines. Elke uitbreiding kan exact dezelfde verzameling functies berekenen.

3. **Empirisch bewijs en generalisatie daarvan:** Alle effectief berekenbare functies die we kennen zijn inderdaad Turing-berekenbaar. Verder is er voor elke bekende methode om nieuwe effectief berekenbare functies te maken ook een verwante manier om nieuwe Turingmachines te maken uit bestaande, zodat elk zulke functie steeds een Turingmachine heeft.

1.5 Wat is de converse Church-Turinghypothese en hoe zou je ervoor argumenteren?

De converse Church-Turinghypothese is dat elke Turing-berekenbare functie f ook effectief berekenbaar is.

Dit is triviaal waar, aangezien de werking van een Turingmachine duidelijk voldoet aan alle eisen die gesteld worden van een effectief berekenbare functie:

[1] De module zelf schrijft drie argumenten toe aan Turing, waar de geciteerde bron er slechts twee bevat. De drie argumenten lijken uit een *andere* paper te komen waar deze worden toegeschreven aan Kleene. Also, er staan in de cursus citaties zonder bronvermeldingen. Wtf?

- Een Turingmachine heeft steeds een eindige hoeveelheid toestanden en transities, en bestaat dus uit een eindige hoeveelheid "instructies".
- De Turingmachine van een Turing-berekenbare functie eindigt per definitie in een eindig aantal stappen, anders was deze niet berekenbaar.
- Een mens kan op een blad de tape en staat van een Turingmachine simuleren.
- Er is geen inzicht nodig bij het simuleren van een Turingmachine, men moet enkel de aangegeven transities volgen.

Elke Turing-berekenbare functie voldoet dus aan onze definitie van een effectief berekenbare functie, en is dus effectief berekenbaar.

1.6 Voor welke doeleinden wordt de Church-Turinghypothese gebruikt?

De Church-Turinghypothese wordt in twee gevallen gebruikt.

De eerste is "luiheid"; indien de hypothese klopt kan men zonder enige moeite te doen zeggen dat een vage effectieve procedure om iets te berekenen ("pseudo-code") steeds berekenbaar is door een Turingmachine zonder deze Turingmachine te moeten construeren.

Het andere geval is iets filosofischer. Er bestaan functies die niet Turing-berekenbaar zijn. Als de Church-Turinghypothese waar is, zou dit ook betekenen dat er geen enkele effectieve procedure bestaat om die functies te berekenen. Dit is vrij interssant voor dingen zoals het halting problem.

1.7 Waarom zijn Turingmachines opsombaar?

Een Turingmachine bestaat uit een aantal eindige verzamelingen. De hoeveelheid mogelijke namen voor de objecten in die verzamelingen is wel oneindig, maar deze hebben geen effect op de uitvoering van de Turingmachine en kunnen dus weggelaten worden om een eindige voorstelling te bekomen.

Men kan Turingmachines dus voorstellen als strings uit een eindig alfabet, en deze op deze manier elk een nummer geven zodat elke string juist een uniek nummer heeft. Dan stelt niet elk nummer een Turingmachine voor, maar heeft elke Turingmachine wel een nummer, en kan men alle Turingmachines opsommen.

1.8 Halting problem

Beschouw de functie s:

$$s(e) = \begin{cases} 0 & \text{if machine } M_e \text{ does not halt for input } e \\ 1 & \text{if machine } M_e \text{ halts for input } e \end{cases}$$

Waarom is deze functie niet Turing-berekenbaar?

Bewijs is door contradictie. Stel dat s wel Turing-berekenbaar is. Dan is er een Turingmachine S die s berekent. We mogen aannemen dat S steeds eindigt op het eerste vakje. Neem nu een tweede Turingmachine J, die eindigt indien het start op een lege tape met enkel nullen, en anders naar rechts gaat en nooit eindigt.

$E = S \frown J$ is de machine die men verkrijgt door J uit te voeren na S. Dit is ook een Turingmachine en er bestaat dus ook een e zodat $M_e = E$ want de Turingmachine komt voor in de opsomming van alle Turingmachines. Start nu M_e met een invoer van e. Er zijn twee mogelijkheden, oftewel eindigt M_e, oftewel eindigt M_e niet.

1. Stel dat M_e eindigt. Dan is $s(e) = 1$. Als we $M_e = E = S \frown J$ ontbinden zien we dat S dus eindigt met een enkele 1 op de tape. Dus J start met een 1 op de tape, en eindigt niet. M_e eindigt dus niet voor invoer e. Dit is een contradictie.

2. Stel dat M_e niet eindigt voor invoer e. Dan is $s(e) = 0$, en S eindigt met een lege tape. Dus J stopt onmiddelijk. M_e moet dus eindigen voor invoer e, en dit is weer een contradictie.

Aangezien we in beide gevallen tot een contradictie komen, kunnen we besluiten dat s niet Turing-berekenbaar is en dat er geen Turingmachine s bestaat.

1.9 Wat is het beslissingsprobleem van Hilbert en wat zegt de Church-Turinghypothese hierover?

Het beslissingsprobleem van Hilbert (online vind je dit onder de naam "Entscheidungsproblem" en dat is ook de naam die ik gebruik, maar in de cursus spreekt men van "het beslisslingsprobleem") is het probleem van het bepalen of een zin in eerste-orde predikatenlogica logisch waar is (m.a.w. of het waar is in elke mogelijke structuur), en het vinden van een effectieve procedure om dit probleem op te lossen.

De Church-Turinghypothese zegt natuurlijk dat als zo'n procedure bestaat, er ook een Turingmachine bestaat die dezelfde berekening uitvoert. Dus kan het probleem gereduceerd worden tot de vraag of er een Turingmachine bestaat die steeds eindigt met uitvoer 1 of 0 en aangeeft of een zin logisch waar is of niet.

Turing en Church hebben onafhankelijk van elkaar bewezen dat dit probleem niet Turing-berekenbaar is. (Of dit betekent dat het ook niet effectief berekenbaar is hangt natuurlijk af van de waarheid van de Church-Turinghypothese.)

1.10 Wat is de bewijsstrategie van Church en Turing voor het Entscheidungsproblem?

Om te bewijzen dat het probleem niet effectief berekenbaar is, bewezen ze dat er geen Turingmachine bestaat die het probleem beslist (want volgens de Church-Turinghypothese, als het niet Turing-berekenbaar is, is het ook niet effectief berekenbaar).

Hiervoor reduceerden ze het halting problem naar het Entscheidungsproblem. Dit betekent dat ze bewezen dat de functie $h(e, w)$ (betekenis: 1 als M_e stopt met invoer w, anders 0) niet berekenbaar is. Dan bewezen ze dat als er een Turingmachine is die het Entscheidungsprobleem beslist, er ook een Turingmachine is die h beslist. Aangezien er geen Turingmachine bestaat die h beslist, kan er ook geen Turingmachine zijn die het Entscheidungsprobleem beslist.