Pioniers van de Informatica: Alan Turing

Van Babbage naar Turing

Alan Turing (1912 – 1915) wordt wel de vader van de informatica genoemd; Charles Babbage is dan de grootvader. En die benamingen zijn correct.

Sinds het werk van Babbage aan de analytische machine in de eerste helft van de negentiende eeuw was er geen enkele vooruitgang geboekt in de theorie of de praktische constructie van een programmeerbare, laat staan een universeel programmeerbare, machine. Wel was er enorme vooruitgang geboekt op het terrein van de fijnmechanische en elektrotechniek. Die vooruitgang kwam praktisch tot uiting in (elektro)mechanische rekenmachines, die gedurende het eerste kwart van de twintigste eeuw steeds geavanceerder werden.

Turing maakte zich in eerste instantie niet druk over de praktijk van rekenmachines; hij sloot aan bij de basisideeën van Babbage over het programmeren van machines, die zo'n 100 jaar eerder werden geformuleerd.

Standbeeld van Turing, werkend in Bletchley Park, gemaakt door de kunstenaar Stephen Kettle. Foto van Jon Callas, San José, USA. Bron: Wikipedia.

Onbeslisbaarheid

Over het leven van Turing is veel bekend, zeker na de indrukwekkende film *The Imitation Game* uit 2014, waarin vooral zijn oorlogsjaren als kraker van de Duitse Enigmacode uitvoerig worden belicht. Maar voordat Turing in Bletchley Park aan het werk ging, had hij al de aandacht getrokken van de wiskundige wereld door een opzienbarend artikel over het *Entscheidungsproblem*. Dat artikel refereert aan het werk van Kurt Gödel, de wiskundige die bewees dat geen enkel axiomatisch systeem volledig gesloten is.

Zeker degenen onder ons die nog opgegroeid zijn met de klassieke Euclidische meetkunde weten hoe een axiomatisch wiskundig systeem is opgebouwd: uitgaande van een klein aantal axioma's worden steeds dieper gaandere stellingen afgeleid, die weer leiden tot stellingen, die weer leiden tot stellingen, enzovoort. Iedere stelling kan bewezen worden door de weg terug naar de axioma's te volgen.

Of omgekeerd kan men de onjuistheid van een stelling bewijzen door een tegenstrijdigheid aan te tonen, die uiteindelijk ook terugvoert naar de axioma's. Een mooi en zeer bekend voorbeeld is het bewijs dat $\sqrt{2}$ irrationaal is, waaruit volgt dat er getallen zijn die niet door een breuk met gehele teller en gehele noemer kunnen worden uitgedrukt.

Kun je nou van elke stelling, die gaat over de dingen van het axiomastelsel beslissen of die waar is, of onwaar? Het verbazingwekkende is dat Gödel mathematisch aantoonde dat je voor ieder wiskundig axiomastelsel stellingen kunt vinden, waarvan je niet kunt bewijzen dat ze waar of onwaar zijn: je kunt altijd stellingen vinden die *onbeslisbaar* zijn.

Turingmachines

Met *On Computable Numbers*, een artikel uit 1936, zette Turing een essentiële stap in de ontwikkeling van programmeerbare machines. Turing vroeg zich af wat machines wel en wat zij niet kunnen berekenen. Kort samengevat: hij bewees dat sommige berekeningen voor een machine onmogelijk zijn, net zoals Gödel bewees dat sommige stellingen niet herleidbaar zijn tot de axioma's van een stelsel. Turing beschreef daarmee dat er een grens bestaat aan berekenbaarheid, niet een praktische limiet, maar een theoretische.

Voor zijn bewijs bedacht een Turing een programmeerbare machine, waarvan de opbouw in essentie dezelfde was als die van de analytische machine van Babbage. Hij gebruikte in zijn bewijs de meest elementaire programmeertaal die je kunt bedenken, maar die de kenmerken van alle andere computertalen heeft.

Het HNF in Paderborn heeft een werkend exemplaar van zo'n turingmachine (TM), maar het denkwerk (het optellen van twee getallen) moet nog steeds gedaan worden door de bediener van het apparaat.

De universele turingmachine

De machine in het HNF is nog een geen Universele Turing Machine (UTM), een machine waarvan het programma zich in het geheugen van de machine bevindt. Het idee van een in een geheugen opgeslagen programma is tegenwoordig zo gangbaar dat we ons afvragen wat daar zo bijzonder aan is, maar in 1936 was dat een opzienbarend idee. Iedere hedendaagse computer is een UTM, natuurlijk technisch veel geavanceerder dan het model van Turing, maar theoretisch gezien niet fundamenteel anders. Eigenlijk is een computer een apparaat dat elk ander apparaat kan zijn, zolang je dat apparaat maar in de vorm van een algoritme kunt beschrijven. Een UTM kan zich als elke andere TM gedragen.

En dat is precies wat een TM is: de vertaling in materie van een algoritme. Bewezen kan worden dat ieder algoritme vertaald kan worden in een TM en dat iedere TM een algoritme vertegenwoordigt. TM's en algoritmen zijn fundamenteel identiek. Turing bewees dat er voor bepaalde wiskundige problemen geen TM kan bestaan en dus dat er voor die problemen geen algoritme bestaat.

Het idee van Turing om het algoritme van een machine in de machine zelf op te slaan, vormt het fundament van de hedendaagse computer. John von Neumann bouwde op dat fundament de eerste programmeerbare UTM, met het algoritme in het geheugen.

Referenties

[1] Pohl, I en Shaw, A, *The nature of computation*, Computer Science Press, Inc, Rockville, Maryland, 1981, pag. 261 - 293.
[2] HNF, *Museumsführer*, Paderborn, 2000, pag. 72 - 73.
[3] Zie voor de werking van een Enigmamachine: https://www.youtube.com/watch?v=ncL2Fl6prH8