Aan het eind van het Turingjaar 2012 gaan we in op een van de meest bijzondere problemen van de twintigste eeuw: wanneer is iets 'berekenbaar'? De Brit Alan Turing (1912-1954) gaf een antwoord door middel van Turingmachines.

■ door Klaas Pieter Hart

BEREKENBAARHEID VOLGENS TURING

In 1936 publiceerde de Engelse wiskundige Alan Turing een artikel met de titel *On Computable Numbers, with an Application to the Entscheidungsproblem*. Hierin stelde hij het probleem aan de orde wat nou eigenlijk het begrip 'berekenbaar' inhoudt. Die vraag is moeilijker dan je misschien zou denken. Als je om je heen kijkt, zijn alle getallen die je ziet het resultaat van berekeningen, en dus kennelijk berekenbaar – in die zin lijkt alles berekenbaar en lijkt een definitie niet eens echt nodig.

Maar, denk eens aan het volgende: neem een rechthoekige driehoek met rechthoekszijden van lengte 1 en bepaal de lengte van de hypotenusa. De stelling van Pythagoras geeft het antwoord: $\sqrt{1^2+1^2}=\sqrt{2}$. Klaar, zou je zeggen, we hebben de lengte berekend. Maar daar waren Turing en zijn tijdgenoten niet tevreden mee, want $\sqrt{2}$ is alleen maar een afkoring voor 'het positieve getal waarvan het kwadraat gelijk is aan 2', en dat vertelt ons eigenlijk niets over de grootte van dat getal.

Je zou kunnen proberen $\sqrt{2}$ als een quotiënt van twee natuurlijke getallen te schrijven; dan weet je *exact* hoe groot dat getal eigenlijk is. Maar, zoals al vaak ook in dit blad gememoreerd, $\sqrt{2}$ kán niet als zo'n quotiënt worden geschreven. We zullen daarom met *benaderingen* genoegen moeten nemen.

Om die benaderingen te maken, zijn al heel wat manieren bedacht. Een van de simpelste is de volgende: Bereken 1^2, $1{,}1^2$, $1{,}2^2$, $1{,}3^2$, $1{,}4^2$, ... Het blijkt dat $1{,}4^2 < 2 < 1{,}5^2$. Begin opnieuw, maar nu met $1{,}41^2$, $1{,}42^2$, ...; je vindt $1{,}41^2 < 2 < 1{,}42^2$. Vervolgens vind je $1{,}414^2 < 2 < 1{,}415^2$ enzovoort. Met voldoende geduld kun je zo net zoveel decimalen van $\sqrt{2}$ bepalen als je maar wilt. Je zou $\sqrt{2}$, ook al is het niet als breuk te schrijven, toch wel berekenbaar willen noemen.

Afbeelding: caeliaes.wordpress.com/2012/06/30/alan-turing-100-aniversary-t-shirt

Maar... als je als wiskundige zinnige dingen over berekenbare getallen wilt zeggen en bewijzen – en dat wilde men in de tijd van Turing – dan zul je toch eerst een precieze definitie van berekenbaarheid moeten geven.

Turing bedacht zo'n definitie en hij goot die in termen van *machines*. Zijn idee was dat een getal pas berekenbaar mocht heten als je de decimalen automatisch door een machine zou kunnen laten berekenen.

Opgave 1. Beschrijf de bovenstaande berekening van de decimalen van $\sqrt{2}$ in een computerprogrammaatje.

TURINGMACHINES Turing stelde zich een rekenaar voor die niet veel kon en oneindig veel schrijfruimte tot zijn beschikking had. Die schrijfruimte bevond zich op een oneindig lange strook papier, verdeeld in vierkantjes. Een rekenaar ziet op elk moment precies één vierkantje en kan het volgende doen

- het gelezen symbool uitgummen;
- een symbool schrijven;
- één vakje naar links gaan;
- één vakje naar rechts gaan.

Wat de rekenaar doet, hangt helemaal af van hoe de machine in elkaar is gezet.

Nu bouwde Turing deze machines niet echt; hij schreef telkens een tabel op die precies specificeerde wat de machine zou moeten doen. Hier is zo'n tabel, uit het artikel van Turing:

toestand	symbool	operaties	toestand
𝔟	geen	P0, R	𝔠
𝔠	geen	R	𝔢
𝔢	geen	P1, R	𝔣
𝔣	geen	R	𝔟

Hoe moeten we dit lezen? Onze machine kan in vier toestanden verkeren; Turing gaf deze aan met de gotische letters 𝔟, 𝔠, 𝔢 en 𝔣. Dankzij die toestanden kunnen we de rekenaar verschillende dingen laten doen met hetzelfde gelezen symbool. Deze machine doet echter niets met het gelezen symbool; daarom staat in de kolom 'symbool' telkens het woord 'geen'. In de kolom 'operaties' staat P voor print, R voor rechts en L voor links.

De machine begint met een lege tape en in toestand 𝔟. De eerste regel van de tabel zegt: als je in toestand 𝔟 bent, schrijf dan een 0 op de tape, ga een vakje naar rechts en ga over naar toestand 𝔠. De tweede regel van de tabel zegt: als je in toestand 𝔠 bent, ga dan een vakje naar rechts en ga over naar toestand 𝔢. De derde regel van de tabel zegt: als je in toestand 𝔢 bent, schrijf dan een 1 op de tape, ga een vakje naar rechts en ga over naar toestand 𝔣. De vierde regel van de tabel zegt: als je in toestand 𝔣 bent, ga dan een vakje naar rechts en ga over naar toestand 𝔟.

Het resultaat van de werking van de machine is dat de tape gevuld wordt met om en om nullen en enen, gescheiden door lege vakjes:

$$|0|\ |1|\ |0|\ |1|\ |0|\ |1|\ |0|\ |1|\ \ldots$$

De machine wordt nooit gevraagd iets van de tape te lezen. Dat hoeft ook niet, want de tape was leeg verondersteld. Het geeft echter ook aan dat de berekening nogal simpel is: er hoeft niets onthouden te worden. Bij ingewikkelder berekeningen moet er wél iets onthouden worden en daar zijn de andere vakjes voor: het resultaat van de berekening staat in de vakjes met een oneven nummer en de even vakjes dienen als kladpapier. In het kader op pagina 23 staat een wat ingewikkelder machine beschreven die inderdaad de even vakjes als klad gebruikt.

Wat is nu eigenlijk het resultaat van bovenstaande berekening? Dat is het reële getal waarvan de binaire ontwikkeling in de oneven vakjes staat en dat is, in ons geval, de som van alle machten van $\frac{1}{4}$, het getal $\frac{1}{3}$ dus.

Opgave 2. Bouw een Turingmachine die de binaire ontwikkeling van $\frac{1}{5}$ genereert.

ONEINDIGE LUS Het aantal tabellen dat we kunnen opschrijven is natuurlijk oneindig. Hier is een flauwe machine:

toestand	symbool	operaties	toestand
𝔞	iets	P0, R, R	𝔟
𝔟	iets	P1, L, L	𝔞

Deze machine doet, ongeacht wat hij leest, niets anders dan een 0 printen, twee vakjes naar rechts, een 1 printen, twee vakjes naar links, een 0 printen, twee vakjes naar rechts, een 1 printen, twee vakjes naar links, ... Deze machine komt in een oneindig voortlopende lus terecht: dezelfde rijen toestanden en acties worden oneindig vaak herhaald terwijl er maar eindig veel symbolen op de tape komen.

Turing noemde een reëel getal *berekenbaar* als er een machine is die de binaire ontwikkeling van dat getal kan produceren en die *niet* in een oneindige lus terecht komt. De machine in de linkerko-

lom van pagina 21 laat zien dat het getal $\frac{1}{3}$ berekenbaar is.

EEN UNIVERSELE MACHINE Turing bedacht vervolgens een Universele Machine. Dit is een machine die het werk van élke Turingmachine kan uitvoeren. Om te zien wat dat betekent, gaan we eerst alle machines coderen en op tapes zetten.

Om te beginnen bekijken we alleen tabellen met maar één operatie per regel. Dat is geen beperking. We hoeven bijvoorbeeld maar weinig aan onze eerste machine te veranderen:

toestand	symbool	operaties	toestand
𝔞	geen	P0	𝔟
𝔟	geen	R	𝔠
𝔠	geen	R	𝔡
𝔡	geen	P1	𝔢
𝔢	geen	R	𝔣
𝔣	geen	R	𝔞

De toestandletters zijn natuurlijk niet belangrijk; in plaats van 𝔞, 𝔟, 𝔠, 𝔡, 𝔢 en 𝔣 gebruiken we rijtjes enen. De te lezen en printen symbolen geven we met nullen en enen weer: 0 voor 'leeg', 1 voor 0 en 11 voor 1. De instructies kunnen we ook door getallen vervangen, of beter: door rijtjes nullen en enen. Bijvoorbeeld: 1, 11, 111 en 1110 voor L, R, E, en P (de E staat voor 'erase', uitgummen dus).

Verder schrijven we elk getal door middel van een rij enen op en gebruiken we : als scheider in een regel en ; als 'einde regel'. De eerste regels van onze machine worden dan

1:0:11101:11;
11:0:11:111;
111:0:11:1111;
1111:0:111011:11111;
11111:0:11:111111;
111111:0:11:1;

Deze rijen zetten we achter elkaar op een tape als invoer voor de universele machine.

In het artikel van Turing vind je een beschrijving van een machine, noem hem *U*, die, gegeven zo'n tape, precies doet wat de bijbehorende machine gedaan zou hebben. Dit is op zich al een mooi idee: een machine die je, door hem een programma (de tape) te geven, elke berekening kunt laten uitvoeren.

HET ENTSCHEIDUNGSPROBLEM Turing deed met dit idee echter iets anders: hij liet zien dat bepaalde problemen *onoplosbaar* waren. De wiskundige David Hilbert (1862-1943) had het probleem opgeworpen een algoritme te maken dat van elke bewering zou kunnen laten zien of die bewijsbaar was of niet; dat is het *Entscheidungsproblem* uit de titel van Turings artikel.

Wat Turing deed, was eerst laten zien dat er geen machine is die, gegeven een tape van de bijbehorende machine, kan laten zien of deze tijdens zijn werk in een oneindige lus raakt of niet. Het idee is om, uitgaande van zo'n machine, noem hem *W*, een andere te bouwen, genaamd *L*, die een tape leest, beslist of de machine in een oneindige lus komt en als dat zo is dan onze eerste machine laat draaien; als de machine niet in een oneindige lus komt laat hij onze flauwe machine draaien.

Dat is wel wat flauw, maar als je nu de tape van de machine *L* zélf als invoer gebruikt, dan krijg je de rare situatie dat de machine in een oneindige lus komt precies dan als hij dat niet doet. Die tegenspraak laat zien dat zo'n wondermachine *W* niet bestaat.

In 1931 had Kurt Gödel (1906-1978) zijn beroemde *onvolledigheidsstelling* bewezen. Deze zegt dat er in de rekenkunde altijd uitspraken zullen zijn die niet bewijsbaar zijn, maar waarvan de ontkenning óók niet bewijsbaar is. Dat doorkruiste een ander streven van Hilbert, namelijk laten zien dat het tegendeel van de stelling van Gödel waar was. Turing liet zien dat zelfs beslissen of iets bewijsbaar is in het algemeen ook niet lukt.

Het bewijs van die laatste bewering gaat door elke rekenkundige uitspraak aan een machine te koppelen en omgekeerd, en wel zó, dat een machine die detecteert of een uitspraak bewijsbaar is meteen ziet of de bijbehorende machine in een oneindige lus komt. Maar voor dat laatste is geen machine te maken en dus voor het eerste ook niet.

FUNCTIES In zijn bewijs gebruikte Turing het idee dat zijn machines in feite ook functies van de

In het Centrum Wiskunde & Informatica in Amsterdam is dit jaar een werkende Turingmachine van LEGO gemaakt.
Afbeelding: www.legoturingmachine.org

natuurlijke getallen naar zichzelf berekenen. Als je de afgedrukte nullen als scheiders beschouwd en de blokjes enen als natuurlijke getallen, dan kun je de werking van onze eerste machine ook opvatten als een berekening van de constante functie c: $n \to 1$.

De ingewikkelde machine uit het kader hiernaast laat zien dat de functie f: $n \to n$ berekenbaar is en de machine van opgave 3 berekent de functie k: $n \to n^2$. Het is misschien wat moeilijk te bevatten, maar de functies die we als berekenbaar willen beschouwen zijn inderdaad berekenbaar volgens Turings definitie, zoals $n \to 2^n$, $n \to n!$, en de functie die de priemgetallen nummert. Zelfs dingen als priemfactorontbindingen kun je door Turingmachines laten doen.

EEN ECHTE MACHINE In het Centrum Wiskunde & Informatica in Amsterdam staat een werkende Turingmachine, gemaakt van LEGO; deze telt twee getallen bij elkaar op. De machine werkt iets anders dan de machines van Turing zelf; als je op de website van het CWI kijkt, zul je zien dat de machine een rij instructies uitvoert (zie www.legoturingmachine.org). Deze zijn net zo eenvoudig als hierboven beschreven, alleen zijn de overgangen naar andere toestanden vervangen door zogeheten GOTO-statements. ■

In *Pythagoras* 49-4 (februari 2010) verscheen een uitgebreid artikel van Arnout Jaspers over Turing: 'Alan Turing (1912-1954): vader van de computer'. Dit artikel is te vinden in het archief op www.pythagoras.nu.
Pythagoras-redacteur Paul Levrie schreef met zijn collega Rudi Penne over Turing op hun blog weetlogs.scilogs.be, zie http://goo.gl/Df1mp.

0,01011011101111011111011111110...

Is er een Turingmachine die de bovenstaande binaire ontwikkeling (de bedoeling is dat elke groep één 1 meer heeft dan de vorige) genereert? Ja, namelijk:

toestand	symbool	operaties	toestand
b		Pe, R, Pe, R, P0, R, R, P0, L, L	o
o	1	R, Px, L, L, L	o
o	0		q
q	0	R, R	q
q	1	R, R	q
q	geen	P1, L	p
p	x	E, R	q
p	e	R	f
p	geen	L, L	p
f	iets	R, R	f
f	geen	P0, L, L	o

De eerste regel initialiseert de tape met |e|e|0| |0| en plaatst de leeskop bij de eerste 0; de letters e zijn er om het begin van het getal af te bakenen. De overige regels printen de nullen en enen en af en toe een x om aan te geven hoe groot de komende groep enen moet zijn; als zo'n groep af is wordt de x uitgegumd (de E staat voor 'Erase') en wordt verderop een nieuwe x opgeschreven.

Opgave 3. Maak een tabel voor een Turingmachine die groepen enen print, gescheiden door nullen en zó dat de n-de groep uit n^2 enen bestaat.