Afscheid van JKL

Als lichtvoetige afscheidsgroet aan Jan Karel gaat hierbij een poetische versie van een bewijs van een van de meest fundamentele inzichten uit de theoretische informatica: er zijn beslissingsproblemen die met geen computer zijn op te lossen. Het beroemdste onbeslisbare probleem is zonder twijfel het stop-probleem. Dat dit probleem onbeslisbaar is werd ruim voor de moderne computer werd uitgevonden aangetoond door Alan Turing.[1]

Turing's bewijs laat zien dat er geen algemene methode kan bestaan om van willekeurige programma's te zeggen of ze zullen stoppen, bij gegeven invoer. Om een voorbeeld te geven: het volgende Ruby programma stopt altijd:

```
print "Hoe heet je? "
input = gets.chop
if input == 'Jan Karel Lenstra'
    puts "Fantastisch"
else
    puts "Jammer"
end
```

Maar het volgende programma raakt bij foute invoer in een oneindige lus:

```
print "Hoe heet je? "
input = gets.chop
if input == 'Jan Karel Lenstra'
    puts "Fantastisch"
else
    puts "Jammer" while true
end
```

In 2000 publiceerde de Britse taalkundige Geoffrey Pullum een bewijs van de onbeslisbaarheid van het stopprobleem in dichtvorm. Mijn bijdrage is daarop een Nederlandse variant.

[1] A.M. Turing, 'On Computable Numbers, with an Application to the Entscheidungsproblem', *Proceedings of the London Mathematical Society*, 1936

De lusvinder genept — een logisch recept

een eenvoudig bewijs
van de onbeslisbaarheid van het stopprobleem

Geen methode voorspelt of mijn codes gaan lussen,
dat toon ik hier aan, en jij krijgt daar niets tussen.
Hoe je ook sputtert, m'n bewering gaat kloppen:
jij kunt niet voorspellen of mijn code zal stoppen.

Stel dat P, een programma, dat tóch kan voorspellen,
dus voor I, plus zijn invoer, weet te vertellen
'I stopt niet' als I in een lus raakt en doordraait en spint,
terwijl P schrijft 'I stopt' als hij géén lussen vindt.

Je geeft P mijn code, met data erbij.
En P leest en rekent, en checkt allebei.
Wat er ook in gaat, P stopt na een tijd:
P geeft altijd betrouwbaar voor alles bescheid.

Nu, het zit zo: zo'n P kan niet bestaan.
Want stel, jij schrijft hem en komt er mee aan.
Dan kan ik hem gebruiken voor een logische val
zo vilein dat je brein erop breekt met een knal.

De truc die ik uithaal vertel ik je nu.
Ik maak een recept, en ik noem het ding Q.
Q neemt een programma en gaat na of dat stopt
met P kan dat immers, dus je snapt dat dit klopt.

Voor een programma dat je Q aanreikt, zeg A,
vraagt Q P om raad, en gaat daarmee na
of wat ik A heb genoemd in een lus raken zal
als A werkt op zichzelf, een bijzonder geval!

En dit is niet alles. Als P zegt 'een lus'
dan zal Q het beamen, en Q stopt daarna dus.
Maar als P zegt 'dit stopt' gaat Q eindeloos voort
met dat steeds maar herhalen, in een lus die ontspoort!

En dit programma, dat ik Q heb gedoopt,
gaan we gebruiken, en jouw P wordt gesloopt.
Als Q opstart en vraagt om zijn invoer? Zeg: Q.
Q leest dan zichzelf, en . . . wat krijgen we nu?

Als Q in een lus raakt hebben we brokken.
P zegt dan 'een lus', want P zal niet jokken.
Maar dan moet Q dit beamen en stoppen.
Nu, je begrijpt, de methode zal floppen.

Maar als Q juist stopt, na verloop van een tijd
dan zal P dit zien, en vort met de geit,
Q moet dan een lus in, want dat was het recept.
Je ziet, ook dit kan niet, dus ik heb je genept.

Wat P doet is funest, 'k hoop maar dat je het ziet.
Want P zegt hoe het staat en Q doet dat teniet.
Als P eerlijk wil spreken dan liegt hij je wat voor.
En zo P wil liegen: 't klopt toch door en door.

Het recept dat je hier zag is bekend aan veel koks
uit de logische keuken, en het heet 'paradox'.
Toen je aannam dat die P kon bestaan
kreeg ik je te pakken, en toen ging je eraan.

Hoe kun je ontsnappen aan dit logisch debâcle?
Geloof niet in die P, want dat was het obstakel.
Van de tegenspraak die ik liet zien kun je leren:
dat zo'n P kan bestaan mag niemand beweren.

De moraal: niemand kan jou een methode vertellen
die het op 'tilt' gaan van je PC kan voorspellen.
Word niet boos en blijf vrolijk, is wat ik je brom,
en zoek zelf naar de fouten, want computers zijn dom.

Dit poëtisch bewijs van de onbeslisbaarheid van het stop-probleem is zeer vrij naar *Scooping the Loop Snooper* van Geoffrey Pullum, *Mathematics Magazine*, oktober 2000. Die versie was logisch niet helemaal accuraat; een gecorrigeerde versie is op internet te vinden: `http://www.lel.ed.ac.uk/~gpullum/loopsnoop.html`.

Jan van Eijck, augustus 2011