Betrouwbaarheid van programma's.

Mathematische uitspraken plegen zich door drie eigenschappen te onderscheiden:
a) ze zijn algemeen in de zin, dat ze op een groot aantal gevallen van toepassing zijn; (men formuleert een eigenschap van "alle niet ontaarde driehoeken")
b) ze zijn heel precies; (dit integenstelling tot alle uitspraken, die hun algemeenheid aan vaagheid ontlenen)
c) er bestaat een bewijstraditie, waardoor de kans op onjuistheid van de uitspraak drastisch gereduceerd kan worden; (hierbij tekenen we aan, dat er voor het bereiken van een dergelijk betrouwbaarheidsniveau geen alternatieven bekend zijn).

Zodra wij rekenautomaten aan het werk zetten met de bedoeling aan de uitkomsten consequenties te verbinden, dan zijn wij daartoe slechts gemachtigd voor zover wij ons geloof kunnen rechtvaardigen, dat de geproduceerde uitkomsten inderdaad het antwoord op het door ons gestelde probleem zijn. Een eerste vereiste daartoe is, dat wij ons ervan kunnen overtuigen, dat het programma correct is. Hoe kunnen wij deze overtuiging bereiken?

In de begintijd heeft men dit door "programma testen" geprobeerd te bereiken: men onderwerpt het programma aan een aantal bekende testgevallen, proefberekeningen, waarvan het antwaord bekend is: als in deze testgevallen het juiste resultaat bereikt werd, nam men aan, dat het programma in alle gevallen het correcte resultaat zou produceren. De bittere ervaring heeft geleerd, dat deze laatste aanname ongerechtvaardigd is.

Het punt is, datde klasse mogelijk bedoelde rekenprocessen, dat onder controle van een niet helemaal triviaal programma zo enorm groot is, dat men met de testgevallen slechts een absoluut verwaarloosbare fractie metterdaad proberen kan en hele subklassen van in een of andere zin "critische gevallen" kan en zal missen. Sterker: de klasse van de mogelijke berekeningen is een klasse van zo sterk gestructureerde objecten, dat als van de 1000 testgevallen er 1 is misgegaan, de verwachting dat ook in de toekomst van de 1000 testgevallen er gemiddeld 1 zal misgaan, helemaal ongerechtvaardigd is: de berekeningen zijn onvoldoende gelijksoortig om hier zinvol statistiek op te kunnen bedrijven. (Dit is iets, wat allelei mensen maar heel moeilijk kunnen geloven.) Om het vereiste betrwoubaarheidsniveau te halen, rest ons slechts een ding: nl. bewijzen, dat het programma correct is.

Om dit te kunnen doen, moet aan drie voorwaarden voldaan zijn:
a) de vereiste eigenschappen moeten voldoende scherp en hanteerbaar geformuleerd zijn (anders hoeven we over "correctheid" helemaal niet te praten)
b) er moet een bewijstraditie zijn (inclusief stellingen, axioma's en beproefde redeneerpatronen)
c) de voor een correctheidsbewijs benodigde hoeveelheid "manipulatie" moet binnen de grenzen van het mogelijke blijven.

De eerste pogingen om correctheidsbewijzen te leveren, waren niet bemoedigend. In retrospectie is dat ook heel begrijpelijk.

Om van een programma de correctheid te bewijzen, zal men een formele definitie van de semantiek van de programmeertaal, waarin het programma is uitgedrukt, moeten hebben en de eerste pogingen, zulke formele definities te

geven, hebben, hoewel tot ondubbelzinnige definities -en dat was in die tijd al heel wat!- niet geleid tot axiomastelsels, die zich soepel leenden tot basis van correctheidsbewijzen. In dit opzicht lijken de tijden inmiddels veranderd.

Een tweede oorzaak van de aanvankelijk ontmoedigende ervaringen was gelegen in het feit, dat men zich in zijn optimisme stortte in het leveren van correctheidsbewijzen van programma's geschreven in toen gangbare programmeertalen, die meestal niet vrij waren van anomalieen, die een zekere barokheid niet ontzegd kon worden en die zeker niet ontworpen waren in een tijd, dat men zich over bewijsbaarheid van correctheid al veel zorgen maakte.

De grootste doorbraak is echter gekomen door de ontdekking, dat de hoeveelheid vereiste manipulatie sterk kon afhangen van de structuur van het programma en dat een van de dingen, die men met structurering van een programma na kon streven juist vermindering van de bewijslast was. Een onmiddellijk gevolg van deze ontdekking is geweest, dat men het correctheidsprobleem constructiever ging benaderen. In plaats van dat men eerst een programma ging schrijven en dan ging proberen om te bewijzen, dat het goed was, gingen mensen correctheidsbewijs en programma hand in hadn ontwikkelen: op het moment, dat het correctheidsbewijs vorm kreeg, schreef men een programma, waarop dit correctheidsbewijs van toepassing was. Een extra "benefit" van deze benaderingswijze was, dat op dat moment de correctheids overwegingen zich ontpopten tot een vruchtbaar heuristisch hulpmiddel.

Een van de machtigste hulpmiddelen is de invariantiestelling voor loops van C.A.R.Hoare:

Indien voor de statement S de voorwaarde P and B een voldoende preconditie is om te garanderen dat, mits de uitvoering van S eindigt, aan de post-conditie P dan voldaan zal zijn, dan is voor de statement

    while B do S od

de pre-conditie P voldoende om in geval van beeindiging de post-conditie P and non B te garanderen.

Het belang van deze stelling is daarin gelegen, dat hij zich zeer wel blijkt te lenen voor de constructie van programma's. Om een bepaalde relatie R te bewerkstelligen, kiest men een P en een B, zodat P and non B de eindrelatie R impliceert. Het programma krijgt dan de algemene vorm

    vestig de geldigheid van P;
    while B do onder invariantie van P een "stap"
                in de richting van "non B" od .

De stelling is in bijzonder machtig, omdat we ons bij de bewijsvoering er niet over hoeven uit te laten, hoe effectief de stap in de richting "non B" nou wel precies is: zolang we er ons van kunnen overtuigen dat de toestand "non B" in een eindig aantal stappen bereikt zal worden, is dit stuk van het correctheidsbewijs compleet. Doordat de stelling van Hoare zich er niet over uitspreekt, hoevaak de herhaalbare statement herhaald wordt, hebben we hier een stuk gereedschap dat ons in staat stelt programma's met een deterministisch netto effect te construeren onder gebruikmaking van het semi-deterministische primitivum "een stap in de goede richting". Het is dit gebruik van semi-deterministische primitiva, dat onder de naam "strategische abstractie" bezig is burgerrecht te verkrijgen.

Bijna alle programma's, die ik ken, om bij een gegeven puntenwolk in het platte vlak het convex omhulsel te vinden, zijn via strategische abstractie

bijvoorbeeld afbeeldbaar op hetzelfde abstracte programma van de volgende vorm, waarin de mogelijke waarden van de variabele "convex omhulsel" het convex omhulsel van een niet lege deelverzameling van de gegeven punten zijn.

```
initialiseer het convex omhulsel, zodat het een of meer
punten omvat;
while er bestaat een punt buiten het convex omhulsel do
      kies een punt buiten het convex omhulsel en noem dat q;
      pas het convex omhulsel aan, zodat ook punt q omvat is
                                                        od
```

In hun uiteindelijke uitwerking kunnen de hierop geente programma's nog enorm verschillen, bv. doordat ze zich in de keuze van het punt q altijd zullen beperken tot een punt dat op het uiteindelijke contour zal blijken te liggen of zich deze beperking niet opleggen.

Een tweede techniek begint burgerrecht te verkrijgen onder de naam "representationele abstractie". Hier scheidt men de feitelijk gekozen techniek om in het geheugen verschillende waarden van een (abstracte) variabele te representeren van de voor de bewijsvoering meest conveniente wijze om deze verschillende waarden te karakteriseren, daarmee het correctheidsbewijs in dit opzicht factoriserend. Aan de hand van enige simpele voorbeelden zal een en ander geïllustreerd worden.

2 juli 1973

prof.dr.Edsger W.Dijkstra