Informatica en Rationalisme

Frits Vaandrager
CWI
Postbus 94079, 1090 GB Amsterdam
fritsv@cwi.nl
Vakgroep Programmatuur, Faculteit Wiskunde en Informatica, Universiteit van Amsterdam
Kruislaan 403, 1098 SJ Amsterdam

In het boek "Filosofie van de informatica" van Ollongren en Van den Herik [1] staan twee intrigerende uitspraken waarover ik hier, in het Liber Americorum voor Professor Alexander Ollongren, graag enkele opmerkingen zou willen maken. Op pagina 13 schrijven de auteurs:

> "Het gebied van de algoritmiek kan filosofisch tot het rationalisme gerekend worden, omdat empirie hier geen rol speelt."

en

> "Men zou zich zelfs de vraag kunnen stellen of de gehele informatica als discipline tot het rationalisme gerekend moet worden."

Beide uitspraken intrigeerden mij omdat ik mij bij mijn werk (verificatie van parallelle en real-time algoritmen) de laatste tijd steeds vaker een soort toegepast natuurkundige voel, die voor ieder probleem opnieuw moet proberen een adequaat model van de fysische werkelijkheid te construeren.

In de filosofie worden denkers als Plato, Descartes en Leibniz "rationalisten" genoemd omdat zij geloven in het bestaan van synthetische, a priori waarheden. Hiertegenover plaatst men de "empiristen", filosofen als Locke, Berkeley en Hume, die stellen dat de mens de wereld alleen kennen kan uit zijn ervaringen, door zien, tasten, herinneren en dergelijke. Wanneer Ollongren en Van den Herik de algoritmiek tot het rationalisme rekenen, dan lijken zij daarmee te bedoelen dat de denkwetten (spelregels) van dit vakgebied vastliggen en dat alle belangrijke resultaten daaruit verkregen kunnen worden louter door gebruik van het verstand. Wanneer men een voldoende geniaal iemand, zeg Donald Knuth, op jeugdige leeftijd zou opsluiten in een afgelegen en volstrekt geïsoleerd klooster met een paar definities waaronder die van een Turing machine, dan zou deze persoon de hele algoritmiek als het ware opnieuw kunnen uitvinden. Wanneer men de kloosterling na veertig jaar zou opzoeken, dan zou hij (of zij) mogelijk hebben ontdekt dat er zo iets is als een berekenbare functie, dat er diverse complexiteitsklassen van algoritmes zijn (zoals P en NP), en wie weet zelfs een driedelig werk hebben geschreven over "The art of computer programming". Al deze inzichten zouden verkregen zijn louter door gebruik van het verstand, aangezien er in de woestijnachtige en dorre omgeving van het klooster natuurlijk onvoldoende grondstoffen en hulpmiddelen zijn om zelfs ook maar een Babbage-achtige computer te construeren. Ik denk dat ik het eens ben met Ollongren en Van den Herik in de zin dat het hierbovengenoemde scenario althans in theorie mogelijk moet worden geacht. Echter, ik wil hierbij wel direct een aantal kanttekeningen plaatsen.

Ollongren en Van den Herik hanteren een tamelijk beperkte definitie van het begrip informatica. Aspecten zoals communicatietheorie en telecommunicatie worden voor de eenvoud buiten de deur gehouden. De computer wordt gezien als een "zwarte doos", die (gestuurd door een programma) in staat is autonoom handelingen te verrichten. Gegeven verder de nadruk die wordt gelegd op Turing machines, moeten we bij de inhoud van de zwarte doos denken aan een sequentiële machine die bij aanvang geïnitialiseerd wordt, vervolgens aan het rekenen slaat, en in het geval de rekenpartij eindigt een resultaat oplevert. Tijdens het rekenen is er geen sprake van interactie tussen de machine en haar omgeving. Dit model heeft het voordeel dat het simpel is, terwijl het toch in zeer veel situaties gebruikt kan worden. Tegenwoordig zien we echter steeds meer toepassingen waarin de computer zich in een permanente interactie bevindt met een complexe omgeving, bijvoorbeeld een netwerk met andere computers, een nucleaire reactor, een vliegtuig, etc. Bij het ontwerpen van programma's voor computers die in dit soort situaties opereren, en bij het redeneren over de correctheid van dergelijke programma's is het veelal noodzakelijk dat men gebruik maakt van een model waarin ook de omgeving van deze computer is verwerkt, en waarin soms zelfs het real-time gedrag van de processor wordt meegenomen. Om te bewijzen dat een bepaald netwerkprotocol correct is volstaat het niet het programma van een individuele processor te inspecteren: men moet de programma's voor alle processoren kennen, alsmede de fysische eigenschappen van de kabels die deze processoren onderling verbinden. Ik denk dat het wel buitengewoon onwaarschijnlijk is dat onze hypothetische kloosterling gedurende zijn/haar veertig jaar eenzaamheid een nuttige bijdrage aan het vakgebied van de gedistribueerde algoritmen zou kunnen leveren, eenvoudigweg omdat de informaticaproblemen die hier spelen voor een heel groot deel bepaald worden door de fysische beperkingen (zoals onbetrouwbare kanalen, uitvallen van processoren, etc.) waar ontwerpers tegen aanlopen. Dit geldt in nog sterkere mate voor de algoritmiek van "real-time embedded systems", zoals bijvoorbeeld chips in CD-spelers, auto's, etc. Samenvattend, zou ik willen stellen dat de bewering dat de algoritmiek tot het rationalisme gerekend kan worden, onhoudbaar wordt indien men een ruimere (en meer realistische) definitie van dit gebied hanteert dan Ollongren en Van den Herik in hun boek.

Maar ook wanneer men zich beperkt tot het klassieke Turing machine model, dan kan men nog een aantal plaatsen aanwijzen waar de empirie om de hoek komt kijken.

- Allereerst bij de definitie van een Turing machine zelf. In het oorspronkelijke artikel waarin de Turing machines worden geïntroduceerd [2] rechtvaardigt Turing zijn definitie via een klassieke en schitterende analyse van wat *mensen* doen wanneer zij iets uitrekenen en daarbij hun verstand gebruiken en aantekeningen maken op papier. In mijn ogen is dit een schoolvoorbeeld van wiskundige modelvorming.

- Zeer belangrijk binnen de algoritmiek is de complexiteit van algoritmes. Maar wat is die complexiteit? In de literatuur zijn naast de Turing machine een aantal andere berekeningsmodellen voorgesteld zoals de RAM (Random Access Machine) en de PRAM (Parallel Random Access Machine). Binnen deze modellen zijn dan nog weer varianten (uniforme kosten versus logaritmische kosten, CRCW (Concurrent Read and Concurrent Write) versus EREW (Exclusive Read and Exclusive Write). Deze modellen beogen op min of meer realistische wijze te beschrijven wat er in een werkelijke computers gebeurt, en kunnen dus vrijwel onmogelijk bedacht worden door onze kloosterling. Nu bestaan er wel polynomiale vertaalslagen tussen de diverse machine modellen, maar dat soort inzichten zijn volstrekt nutteloos (want te grof) wanneer men zich bijvoorbeeld afvraagt of een algoritme voor een CRCW PRAM verbeterd kan worden van $O(n^2)$ naar $O(n \log n)$.

- Steeds wanneer men de complexiteit van een algoritme bestudeert dient men te kijken of het gehanteerde model wel te verdedigen valt. Dit is een puur empirische vraag, waarbij men zich bijvoorbeeld moet afvragen of het wel realistisch is om aan te nemen dat optelling van natuurlijke getallen plaatsvindt in constante tijd, of dat variabelen in constante tijd geraadpleegd kunnen worden. (Vitányi [3] laat zien dat uit fysisch oogpunt deze aanname niet meer realistisch is wanneer het aantal variabelen te groot wordt.)

- Een analyse van de theoretische complexiteit van algoritmes dient hand in hand te gaan met experimenten met implementaties van deze algoritmes op echte machines. Ik noem een tweetal voorbeelden uit mijn eigen specialisme (protocolverificatie). Het beste algorithme voor het beslissen van sterke bisimulatie equivalentie op grafen is afkomstig van Paige en Tarjan, en heeft een theoretische complexiteit van $O(n \log m)$ (hier is n het aantal knopen en m het aantal takken van de graaf). Dit is beter dan de theoretische complexiteit $O(nm)$ van een algoritme dat bedacht is door Kanellakis en Smolka. Toch loopt in de praktijk het algoritme van Kanellakis en Smolka aanzienlijk sneller, eenvoudigweg omdat de constante overhead van het algoritme van Paige en Tarjan zo groot is dat het theoretische voordeel zich pas manisfesteert bij grafen die groter zijn dan wat er kan worden opgeslagen in het geheugen van een Sun Sparc werkstation. Een tweede voorbeeld is het beslissen van de zgn. failure equivalentie op grafen. Dit probleem is PSPACE compleet, dus uit theoretisch oogpunt hopeloos moeilijk. In de praktijk valt dit echter reuze mee en loopt een naïef algoritme voor failure equivalentie slechts twee keer zo langzaam als het Kanellakis-Smolka algoritme voor bisimulatie. Dit komt omdat de problemen bij het beslissen van failure equivalentie zich vooral voordoen bij grafen met veel nondeterminisme, en die zie je in de praktijk van de protocolverificatie niet zoveel.

De these dat de gehele informatica als discipline tot het rationalisme gerekend moet worden is denk ik onhoudbaar. Ollongren en Van den Herik onderschrijven dit, zo lijkt het, door te verwijzen naar Hofstadter's observatie dat een systeem dat foutloos functioneert op instructieniveau niet noodzakelijkerwijs correct werkt op een hoger niveau. Gegeven de extreme mate waarin het onderzoek binnen de informatica bepaald wordt door enerzijds de ontwikkelingen op hardware gebied en anderzijds de explosie aan praktische toepassingen, is de vraag of de informatica tot het rationalisme gerekend moet worden tamelijk academisch. De computersystemen zoals we die nu overal in de praktijk aantreffen zijn zo immens complex dat ze onmogelijk door het menselijk verstand in hun geheel doordacht kunnen worden. Bij het ontwikkelen van deze systemen gaan ratio en empirie dan ook hand in hand: men doet voortdurend waarnemingen aan het echte computersysteem om te testen of het waargenomen gedrag in overeenstemming is met dat wat door het berekeningsmodel wordt voorspeld. Hiermee wordt de idee van informatica als vorm van rationalisme tot een utopie, een ideaalbeeld dat Professor Ollongren zijn studenten (terecht!) altijd heeft voorgehouden maar dat vermoedelijk nooit gerealiseerd zal worden omdat de praktijk altijd mijlen zal blijven voorlopen op de theorie.

REFERENTIES

1. A. Ollongren en H.J. van den Herik. *Filosofie van de Informatica.* Serie wetenschapsfilosofie, Martinus Nijhoff, 1989.

2. A. Turing. On computable numbers, with an application to the entscheidungsproblem. *Proc. Lond. Math. Soc. Ser. 2*, 42:230–265, 1937. Corrections: *ibid* 43:544–546, 1937.

3. P.M.B. Vitányi. Locality, communication and interconnect length in multicomputers. *SIAM J. on Computing*, 17:659–672, 1988.