Bird's boek gezien vanuit de HOP-activiteiten

Maarten Fokkinga, 19 mrt 1987.

Na aandachtige lezing (en bestudering) ben ik tot de volgende conclusies gekomen.

1. De tekst is prettig leesbaar, goed geschreven en de natuurlijke taal-formuleringen zijn heel exact. Menig collegedictaat schrijver kan ~~die~~ dit tot voorbeeld nemen.

2. Een groot deel van de onderwerpen die mijns inziens in "Programmeren I, II, III - nieuwe stijl" behandeld zouden moeten worden, komen aan bod. Maar niet alle onderwerpen; met name data structureringen ontbreken, en een expliciete behandeling van enige efficiëntie-verbeterings-technieken zoals "tupling", memoisation, en recursie-eliminatie (in verscheidene vormen).

3. Het grote verschil tussen wat Bird doet en wat wij in al onze schetsen hebben gedaan, zijn deze twee punten (verderop nog iets toegelicht):
   a. de nadruk op (bewijsbare) correctheid.
   b. het gebruik van (hogere orde) functies.

Door beide punten vergt Bird's boek een grotere

waardigheid in wiskundig redeneren dan wat uit "onze" teksten blijkt. Dit kan voor sommige (of veel?) studenten een struikelblok zijn. (Overigens ben ik van mening dat hierdoor alleen maar de "professionals" van de "amateurs" worden gescheiden. Programmeren kan nu ook ~~als~~ op wetenschappelijk niveau bedreven worden.) Hier is nog enige toelichting.

Ad a. De correctheid wordt aangetoond door het bewijzen van eigenschappen die middels gelijkheid zijn uitgedrukt. Bijvoorbeeld, gelijkheid met de specificatie. Ik denk dat deze correctheidsbeschouwingen, hoe wiskundig ze ook zijn, als veel zinvoller worden ervaren dan de invarianten-methode bij imperatieve programma's. (Daar waar een programma wordt "afgeleid" uit de specificatie, is de correctheidsbeschouwing impliciet – het is de constructie zelf. Bij de invarianten-methode staan wij zoiets niet toe: als een student een programma heeft "afgeleid" wordt hij bij het huidig programmeeronderwijs toch gedwongen invarianten op te stellen!)

Ad b. Bird programmeert zo te zien bij voorkeur op functie-niveau in plaats van object-niveau. Met andere woorden, programma's zijn vooral gedefinieerd als

samenstellingen van andere programma's (= functies) en niet zozeer ~~definities~~ gedefinieerd in termen van manipulaties op de elementaire data, zoals getallen en lijstelementen. Dit vereist een vermogen tot abstractie (en gewenning!). Ik vermoed dat velen van de wetenschappelijke staf van de Faculteit daar zelf ook moeite mee zullen hebben. (Overigens, Backus heeft zich in zijn beroemde Turing Award Lecture sterk gemaakt voor het programmeren op functie-niveau. Hij is daar heel dogmatisch in, en maakt programmeren op object-niveau gewoon onmogelijk.)

4. Of bovenstaande bezwaarlijk is hangt af van de doelstellingen die we kiezen. Een mogelijke doelstelling (van de soort "algemene attitude") luidt:

> de cursist is in staat (en voelt zich aangetrokken) tot het analyseren van programmeerproblemen op wiskundige manier.

Deze doelstelling wordt met het huidige programmeeronderwijs zeker niet gehaald (maar misschien ook niet nagestreefd). Het gebruik van Bird's boek lijkt mij een redelijke kans wanneer bovenstaande

doelstelling als een van de uitgangspunten wordt genomen.

5. Het is de vraag of het programmeren op object-nivo didactisch noodzakelijk is alvorens "geabstraheerd" kan worden tot het functie-nivo (met foldl, foldr, scan enzovoorts). Wanneer dit zo is – ik ben er niet zeker van – dan kan dat zelfs met Bird's boek nog bereikt worden, denk ik, door in het begin §2.5 en 2.6 (functies en sections) over te slaan en direct na § 3.3 (elementaire lijstoperaties) geschikte oefeningen in te voegen. De mogelijkheid van recursie moet dan wel uitvoerig gebruikt worden. Een gevaar is dat de studenten zich aanwennen om recursie te gebruiken zonder bijbehorende correctheidsargumentatie zoals in §4 geleerd wordt...

6. Wanneer er op grond van de doelstellingen gekozen wordt om Bird's boek te gebruiken, dan staat het voor mij vast dat je <u>niet</u> na één trimester kunt beginnen met "omzettingen naar imperatief programmeren". De vaardigheid in wishandig programmeren is dan te gering om er profijt van te hebben. <u>Bovendien</u> is er dan

nog net te weinig van wiskundig programmeren geleerd om het daadwerkelijk toe te kunnen passen (op 008-achtige problemen): o.a. datastructureringen ontbreken. Het onderwijs in wiskundig programmeren à la Bird moet minstens twee trimesters ononderbroken plaats vinden, wil het enige kans van slagen hebben (als aangeleerde attitude). Het lijkt mij onderwijskundig gezien wel doenlijk om parallel hieraan (beginnend in het eerste, tweede of derde trimester) het imperatief programmeren te onderwijzen. Misschien is de start met Pascal pas in het derde trimester zo gek nog niet...

7. Op het eerste gezicht lijkt het of Bird maar weinig programmeervoorbeelden geeft en maar weinig doet aan het systematisch ontwikkelen van programma's. Dit is bedrieglijke schijn. Hij geeft (na de nogal lange introductie over notatie, pag's 1..50) best wel veel niet-triviale voorbeelden; maar je ziet ze haast niet omdat ~~het~~ ze dankzij het functie-niveau vaak één-regelig zijn. Ook de programma-ontwikkeling komt heel expliciet aan bod, bijvoorbeeld al in het eerste grote

voorbeeld (§3.7) en ook later als apart deel-
hoofdstuk (§4.5).