

Hoe abstract wiskundig kan/moet/mag hedendaags academisch fp-onderwijs zijn?

Jan Terlouw

Landelijke FP-dag, 9 januari 2004

1 Inleiding

Hier volgen twee teksten die als referentie dienen bij mijn FP-dag-voordracht met de hierboven vermelde titel. Eerst: een beschrijving van de abstracte functionele taal **FEM**, met als toegift: een stelling over deze taal en een programmeeropdracht. Vervolgens: de inleiding van het dictaat dat ik destijds vervaardigd heb voor een cursus *Inleiding Theoretische Informatica* (RuG-Informatica, 1984/1985).

De hierna volgende beschrijving van **FEM** is in zoverre incompleet dat achterwege gelaten zijn: (i) de precieze definitie van de operationele (herschrijf)semantiek van **FEM** en (ii) de precieze conventies aangaande de weergave van FEM-programma's als ASCII-strings. (Er worden echter wel twee concrete voorbeelden van FEM-programma's in ASCII-vorm gegeven.) De aanvullende details zijn elders te vinden, en wel op:

www.cs.rug.nl/~terlouw/FEMdictaat.pdf

De tekst op die webpagina is de uiteindelijke versie van het dictaat dat gebruikt is bij de afgelopen 2003/2004-versie van de RuG-Informatica-cursus *Functioneel Programmeren*. Dat dictaat werd bij die cursus gebruikt naast het boek *Haskell; The Craft of Functional Programming* van Simon Thompson (Addison-Wesley, 1999 [second edition]) **N.B.** Een expositorisch verbeterde versie van het dictaat is in de maak. Doelgroep van de cursus: tweedejaars studenten Informatica.

De opbouw van mijn FP-dag-voordracht is als volgt:

- (1) Korte uitleg van **FEM** aan de hand van de hierna volgende tekst over **FEM**.
- (2) Uiteenzetting van wat ik in onderwijsverband wilde doen met **FEM** en van wat ik in onderwijsverband verder nog wil doen met **FEM**.
- (3) Een verslag van mijn ervaringen tot nu toe in onderwijsverband met **FEM**, inclusief een uiteenzetting van hoe ik sommige dingen in dit verband anders wil gaan doen.
- (4) Verzoek aan het publiek om feedback te leveren, mede op basis van eigen ervaringen waar het gaat om de verwerking van extra theorie in (o.a.) fp-onderwijs.
- (5) In samenhang hiermee: aanzet tot een verdere discussie naar aanleiding van de titelvraag van de voordracht, mede met als referentie: de bijgevoegde inleiding van mijn vroegere dictaat *Inleiding Theoretische Informatica*.

2 De abstracte functionele taal FEM

FEM-termen zijn opgebouwd uit zgn. *constanten* en *variabelen*. Er zijn om te beginnen drie speciale constanten, die we hier noteren als: **s**, **p** en **t**. Voorts is er voor elk natuurlijk getal n een

speciale bijbehorende constante, die we hier noteren als \bar{n} . De verzameling van al deze constanten $s, p, t, \bar{0}, \bar{1}, \bar{2}, \bar{3}, \bar{4}, \dots$ bij elkaar geven we aan met Con . Wat, vervolgens, de *variabelen* betreft: we nemen eenvoudigweg aan dat hiervan een aftelbaar oneindige verzameling Var voorhanden is. We veronderstellen hierbij uiteraard dat $\text{Var} \cap \text{Con} = \emptyset$.

De verzameling Ter van de *FEM-termen* wordt als volgt inductief opgebouwd:

- (1) Constanten zijn FEM-termen.
- (2) Variabelen zijn FEM-termen.
- (3) Als A en B FEM-termen zijn, dan is (AB) ook een FEM-term.

Willekeurige variabelen worden aangegeven met de (eventueel geïndexeerde) letters x, y, z ; willekeurige FEM-termen worden aangegeven met de (eventueel geïndexeerde) letters A, B, C, D, \dots

De intuïtieve semantiek van FEM-termen is als volgt.

- (1) De constanten $\bar{0}, \bar{1}, \bar{2}, \bar{3}, \bar{4}, \bar{5}, \bar{6}, \bar{7}, \dots$ representeren de natuurlijke getallen, de constanten s en p representeren respectievelijk de opvolgerfunctie en de voorgangerfunctie op de verzameling van de natuurlijke getallen (waarbij we ad hoc 0 als voorganger van 0 nemen ten einde die voorgangerfunctie totaal te laten zijn). De constante t representeert — op een dadelijk nog nader aan te geven wijze — een operatie die test of een getal gelijk aan 0 is.
- (2) Variabelen hebben waarden die van buitenaf gegeven worden, en wel door zgn. *declaraties*; zie verderop.
- (3) In een samengestelde FEM-term (AB) zien we A als een functie die toegepast wordt op een argument B ; de waarde van (AB) is het resultaat van die functieapplicatie. N.B. Dit is vooralsnog een intuïtieve — nog niet wiskundig exact gepreciseerde — semantiek, waarbij de betekenis van “vreemde termen” als (AA) , met verzamelingstheoretisch onmogelijke “zelfapplicatie”, of $(\bar{0}B)$, met verzamelingstheoretisch zinloze “getal-applicatie”, vooralsnog open gelaten wordt.

Samengestelde termen (AB) heten *applicatieve termen*; deze formele terminologie wordt gemotiveerd door bovenstaande informele uiteenzetting. Notationele conventies: de buitenste haakjes van applicatieve termen mogen weggelaten worden — dus AB is voluit: (AB) — en verder is het zo dat applicatie naar links associëert — dus ABC is voluit: $((AB)C)$, en $ABCD$ is voluit: $((((AB)C)D))$.

Vervolg van de intuïtieve semantiek: de waarde van een term $tABC$ is gelijk aan de waarde van B als A waarde 0 heeft, en gelijk aan de waarde van C als A een (getal)waarde anders dan 0 heeft. Kortom: een term $tABC$ is te lezen als “if $A = 0$ then B else C ”.

Een *FEM-declaratie* — of, kortweg, *declaratie* — is per definitie een expressie van de vorm

$$x\vec{y} = B,$$

waarbij x een variabele is en \vec{y} een (eventueel lege) rij is van onderling verschillende variabelen, anders dan x , en waarbij B een of andere FEM-term is. (Het linkerlid van de formele gelijkheid dient ook als een FEM-term gelezen te worden, en wel overeenkomstig de hierboven vermelde haakjesconventie.) In dit geval heet x de *gedeclearerde variabele* en heten de (eventuele) variabelen uit \vec{y} ook wel de *formele parameters* van de declaratie.

Extreem geval: als x een variabele is en B een FEM-term, dan is de expressie $x = B$ een *parameterloze* declaratie van x . Een declaratie met, bijvoorbeeld, twee formele parameters is van de vorm $xyz = B$ oftewel voluit, met haakjes, $((xy)z) = B$.

De intuïtieve semantiek van declaraties is als volgt. Door een declaratie wordt — eventueel in samenspel met andere declaraties — de waarde vastgelegd van de gedeclearerde variabele.

Een parameterloze declaratie $x = B$ bepaalt dat x dezelfde waarde heeft als B . Een declaratie $xy_1 \cdots y_k = B$ met k parameters, waarbij $k \geq 1$, bepaalt dat x een functie f voorstelt (of “is”) die achtereenvolgens op k argumenten p_1, \dots, p_k kan werken en daarvoor dan oplevert: de waarde van B als daarin elke y_i waarde p_i krijgt ($1 \leq i \leq k$).

Een declaratie $x\vec{y} = B$ heet *recursief* als de gedeclareerde variabele x ook in het rechterlid B voorkomt. In een rij van declaraties kan eventueel ook sprake zijn van *wederzijdse recursie*; voorbeeld: een rij van declaraties $x_1\vec{y} = B, x_2\vec{z} = C$ van (onderling verschillende) variabelen x_1 en x_2 , waarbij het zo is dat x_1 in C voorkomt (maar niet in \vec{z}) en x_2 in B voorkomt (maar niet in \vec{y}).

Een *context* is per definitie een eindige rij van declaraties van verschillende variabelen. Een *FEM-programma* is per definitie een paar

$$(\Gamma, A),$$

waarbij Γ een context is en A een FEM-term is, de zgn. *hoofdterm* van het programma.

Intuïtieve semantiek: de waarde van een FEM-programma (Γ, A) is gelijk aan: de waarde van de FEM-term A als daarin elke variabele zijn door Γ bepaalde waarde krijgt.

Voorbeelden. Hier volgen twee verschillende FEM-programma's voor de berekening van 2^{10} .

```
( dubbel x = t x 0 (s(s(dubbel(p x)))) ,
  macht x = t x 1 (dubbel(macht(p x))) ,
  macht 10 )
```

```
( iteratie f n x = t n x (f(iteratie f (p n) x)) ,
  dubbel x = iteratie s x x ,
  macht x = iteratie dubbel x 1 ,
  macht 10 )
```

Definitie.

- (i) Zij $k \in \mathbb{N} \setminus \{0\}$. Onder een (totale) k -aire getaltheoretische functie verstaan we: een functie van \mathbb{N}^k naar \mathbb{N} .
- (ii) Zij $k \in \mathbb{N} \setminus \{0\}$ en zij f een k -aire getaltheoretische functie. Onder een *FEM-implementatie van f* verstaan we: een FEM-programma (Γ, A) met de eigenschap dat voor alle $n_1, \dots, n_k \in \mathbb{N}$ geldt, dat het FEM-programma

$$(\Gamma, A\overline{n_1} \cdots \overline{n_k})$$

de waarde $f(n_1, \dots, n_k)$ heeft.

Stelling. Zij $k \in \mathbb{N} \setminus \{0\}$ en zij f een k -aire getaltheoretische functie. f is precies dan FEM-implementeerbaar als f Turingmachine-berekenbaar is.

Programmeeropdracht voor Haskell-kenners. Construeer in Haskell definities van functies `testFEM` en `gwaardeFEM` van typen `String -> Bool`, resp. `String -> Int` met de eigenschap dat het volgende geldt voor elke willekeurige ASCII-string xs .

- (1) Als xs een FEM-programma is, dan heeft `(testFEM xs)` waarde `True`. Als xs geen FEM-programma is, dan heeft `(testFEM xs)` waarde `False`.
- (2) Als xs een FEM-programma is met getal n als waarde, dan is de waarde van `(gwaardeFEM xs)` gelijk aan n .

3 Inleiding van een theorie-dictaat uit 1984/1985

In de theoretische informatica houden we ons bezig met de wiskundige beschrijving en bestudering van de fundamentele begrippen van de informatica. Op deze wijze willen we vanuit een algemeen theoretisch kader een beter inzicht verkrijgen in die begrippen en hun eigenschappen. Twee interessante aspecten van dit onderzoek zijn de volgende:

1. Het verkregen inzicht komt (hetzij direct, hetzij indirect) ten goede aan de praktische omgang met die begrippen en kan, sterker nog, onontbeerlijk zijn indien het gaat om de beheersing van meer complexe praktijksituaties, waar ad hoc methoden falen (vanwege gebrek aan gestructureerdheid en efficiëntie) en een systematische, theoretisch gefundeerde aanpak vereist is. (Denk hierbij bijvoorbeeld aan de toepassing van de automaten-theorie bij het ontwerpen van vertalers.)
2. Het onderzoek geeft aanleiding tot theorievormingen die ook vanuit algemeen wiskundig standpunt interessant zijn, in het bijzonder vanwege de gehanteerde methoden; deze kunnen ontleend zijn aan de traditionele wiskunde, maar kunnen ook van een nieuw karakter zijn. Die theorievormingen in de ruimere zin (dit wil zeggen: in een algemeen wiskundig kader, niet noodzakelijk steeds in direct verband met de informatica in de strikte zin) worden eveneens tot het gebied van de theoretische informatica gerekend. Hoewel ze geïnspireerd zijn vanuit de informatica, kunnen ze zich soms ontwikkelen tot een hoog abstractieniveau. Vanuit wetenschappelijk standpunt gezien is die verre gaande abstractie echter nog steeds zinvol voor een beter begrip van de informatica; zij kan dit op directe wijze zijn (doordat zij concrete toepassingen induceert), maar ook op indirecte wijze, doordat zij vruchtbaar is voor de manier van denken over begrippen uit de informatica. Immers, door oefenen in geschikte vormen van abstract denken leren wij algemene, verhelderende structuren en verbanden onder-

kennen, die verborgen blijven indien we vasthouden aan louter ad hoc benaderingswijzen.

Om terug te keren tot de eerste zin van deze inleiding: de wiskundige beschrijving op zich speelt al een essentiële rol: de begrippen in kwestie zoals die ons vanuit de praktijk aangereikt worden ("computer", "programma", "datastructuur", etc.) hebben in eerste instantie een nogal vage of complexe inhoud en moeten daarom, opdat ze op gerichte, systematische wijze bestudeerd kunnen worden, voorgesteld (gemodelleerd) worden door hanteerbare, welomschreven wiskundige objecten. Daarbij moeten ze ontdaan worden van die details welke voor het doel van het betreffende onderzoek niet relevant zijn. (Om een voorbeeld te noemen, vooruitlopend op hoofdstuk 5 van dit dictaat: als we het verschijnsel recursie in programmeertalen willen bestuderen, dan doen we dit aan de hand van een modeltaal die eenvoudig te definiëren is (veel eenvoudiger dan de praktijktaalen, zoals ALGOL, met hun ingewikkelde syntaxisdefinities), maar die ondanks nog wel de in dit verband essentiële trekken vertoont.) Wat dan overblijft is een wiskundige representatie van de begrippen waarin precies die aspecten weerspiegeld worden welke relevant zijn voor het beoogde onderzoek. (In feite is dit een algemeen roost van wiskundige modellering, dat we aantreffen bij uiteenlopende toepassingen van de wiskunde.)

Het college Inleiding Theoretische Informatica beoogt een eerste indruk te geven van wat het hierboven omschreven vak theoretische informatica daadwerkelijk inhoudt; van wat de onderwerpen zijn waarmee het zich bezighoudt en van wat de daarbij gebruikte wiskundige methoden zijn. Uiteraard kan hierbij slechts een zeer beperkt deel van het vak belicht worden; wat de student dan ook gegeven wordt is in de eerste plaats, zoals gezegd, een eerste indruk en niet zozeer een representatief beeld van het vak.

Een nader idee van wat het vak in zijn algemeen-

heid omvat, krijgt de student door het volgende citaat uit de zogenaamde Nieuwsbrief van de Werkgemeenschap Theoretische Informatica:

Het terrein van onderzoek omvat de fundamentele begrippen die betrekking hebben op representatie van informatie, algoritmen, talen en automaten.

Tot de theoretische informatica wordt meer in het bijzonder gerekend:

- theorie van automaten en formele talen
- theorie der berekenbaarheid
- analyse van algoritmen en complexiteit van berekeningen
- mathematische aspecten van programmeertaaldefinitie
- semantiek en bewijstheorie van programmeertalen
- theorie van programmaspecificatie en -ontwerp
- theorie van datastructuren
- theorie van databanken
- theorie van parallelle processen en VLSI.

De werkgemeenschap Theoretische Informatica verwacht uitbreiding van haar terrein van onderzoek met de verdere ontwikkeling van de informatica.

De beste aanwijzing wat onderzoek in de Theoretische Informatica nu wel behelst, wordt gevonden in de onderwerpen welke worden gepresenteerd op de volgende colloquia:

- de colloquia van de Europese Associatie van Theoretische Informatica,
- de Oosteuropese symposia Mathematical Foundations of Computer Science,
- de Amerikaanse symposia Theory of Computing, Foundations of Computer Science and Principles of Programming Languages.

In deze inleiding mag niet onvermeld blijven, dat het vak theoretische informatica een nogal eigen karakter heeft in vergelijking met de informaticavakken die de student tot nu toe heeft gehad en dat daardoor het gevaar bestaat, dat deze student een aantal aspecten ervan niet op de juiste wijze inschat. De volgende opmerkingen (deels waarschuwingen) zijn daarom op hun plaats:

1. De aard van het vak brengt met zich mee dat er een grote waarde gehecht wordt aan het zorgvuldig (en dus volledig) opschrijven van wiskundige redeneringen, en wel in het bijzonder redeneringen die zich afspelen op een nogal abstract niveau. In feite is wat dit betreft ervaring met wiskunde (dat wil zeggen: training in abstract denken en redeneren) belangrijker dan praktische ervaring in de informatica. De student dient hieraan de nodige aandacht te besteden; de werkcolleges zijn hem daarbij tot steun.
2. Opdat de student de voor hem tot nu toe betrekkelijk onbekende benaderingswijzen daadwerkelijk eigen maakt, is het van groot belang dat hij, naast de passieve ervaringen op het college, serieuze tijd besteedt aan actieve, zelfstandige studie en zichzelf daarbij kritisch test op werkelijk

begrip van de stof. (Gelegenheid tot het stellen van vragen zal uiteraard in voldoende mate geboden worden.) Een dergelijke opmerking geldt natuurlijk algemeen, voor het leren van ieder vak, maar is hier wel in het bijzonder van toepassing. Soms zal de zelfwerkzaamheid extra benadrukt worden doordat bepaalde delen van dit dictaat die zich daartoe lenen (meer dan voor een weergave op het college) aan de student zelf ter bestudering overgelaten worden.

3. Minstens vindt de student sommige delen van de stof al te abstract en geeft hij op deze af, omdat hij het "nut" ervan voor de praktische informatica niet direct inziet. Hij dient zich dan echter te realiseren dat die neiging tot abstractie (waarover reeds gesproken is in het begin van deze inleiding, in punt 2 aldaar) onlosmakelijk verbonden is met het wetenschappelijke karakter van de informatica zoals die op de universiteit doceert wordt. Een wezenlijk kenmerk van een academische opleiding (en daarin onderscheidt zij zich duidelijk van een beroepsopleiding) is juist de algemene theorievorming als abstractie van het geheel van louter concrete en praktische zaken; dat laatste is hier: de zogenaamde praktische informatica, dus, zeg maar, het schrijven van programma's en wat daar direct mee te maken heeft, zonder een wetenschappelijke uitbouw.

Wat de tentamenstof betreft: deze wordt ruimschoots omvat door dit dictaat. In principe is het dus mogelijk de stof uit het dictaat te leren, onafhankelijk van het college, mits men voldoende in staat is zelfstandig te werken (ken uzelf!). Te zijner tijd zal precies worden aangegeven worden welke gedeelten uit het dictaat overgeslagen mogen worden.

Tenslotte: op- en aanmerkingen die kunnen bijdragen tot een betere presentatie van de stof worden ten zeerste op prijs gesteld; zowel student als docent hebben daar hun voordeel bij.