

Over Fields en Meadows

Jan Bergstra

Programming Research Group
University of Amsterdam
janb@science.uva.nl

April 27, 2006

*Voor René Tönissen!
Bij zijn afscheid van Ivl-HvA,
Op een voorspoedige toekomst op de HAN.*

René is een **wiskundige** net als ik verdwaald in de informatica.
Gaaf dat wel goed?

Leren wij het ooit af om als wiskundigen te denken?

René wel.

Ik niet.

Abstracte Data Typen: ADTs

In jaren 70 in de mode, nu gewoon. *Wat is het?*

Gegevens in de vorm gieten van gewone wiskundige structuren.

Gegevens is een vage term. Ik beperk met tot de zogenaamde **values**.

datastructuren zijn ook ADTs maar dat ligt moeilijker

Wie kan er een definitie geven van een **ARRAY**?

Abstracte Data Typen: ADTs

In jaren 70 in de mode, nu gewoon. *Wat is het?*

Gegevens in de vorm gieten van gewone wiskundige structuren.

Gegevens is een vage term. Ik beperk met tot de zogenaamde **values**.

datastructuren zijn ook ADTs maar dat ligt moeilijker

Wie kan er een definitie geven van een **ARRAY**?

Meteen 250 Euro waard. Ik heb nooit iets betrouwbaars op papier gezien.

Een definitie staat op papier en de auteur heeft niet de wens om 'm zelf te veranderen, ook niet na een beetje kritiek.

Informatici zijn vaak vaag over hun onderwerp wat het onderwerp ook is. René kan daar beter tegen dan ik.

Gebruik ADTs om een onderwerp uit de wiskunde te bekijken.

Het is in de mode om over wiskunde na te denken.
Men wil weer meer wiskunde op school.

*De wiskunde is door de grafische rekenmachine onderuit
gehaald.*

Gebruik ADTs om een onderwerp uit de wiskunde te bekijken.

Het is in de mode om over wiskunde na te denken.
Men wil weer meer wiskunde op school.

*De wiskunde is door de grafische rekenmachine onderuit
gehaald.*

Een beetje informatica was onze schoolwiskunde al te machtig.
Dan maar veel meer informatica erbij.

Nat, een mooie verzameling van waarden

Een mooie verzameling van waarden is:

nul, één, twee, drie, vier, . . .

Hoe wordt dit een data type? Maak een **algebra**:

$(Nat \mid 0, 1, +)$

Nat, een mooie verzameling van waarden

Een mooie verzameling van waarden is:

nul, één, twee, drie, vier, . . .

Hoe wordt dit een data type? Maak een **algebra**:

$(Nat \mid 0, 1, +)$

Nu schrijven we de elementen van *Nat*:

$0 = \text{nul}$

$1 = \text{een}$

$1 + 1 = 2 = \text{twee}$

$1 + (1 + 1) = 3 = \text{drie}$

$1 + (1 + (1 + 1)) = 4 = \text{vier}$

$1 + (1 + (1 + (1 + 1))) = 5 = \text{vijf}$

. . .

Wat bereiken we zo?

Het **data type** ($Nat \mid 0, 1, +$) gaat over tellen . **Maar:**

- Je ziet af van nederlandse namen voor getallen.
- en zelfs van het gebruik van cijfers (behalve 0 en 1),
- en van de logaritmische notatie ($89 = 10 \cdot 8 + 9$).
- en de essentie blijft over.

Is dit alles? Nee, er zijn ook drie belangrijke vergelijkingen:

Wat bereiken we zo?

Het **data type** ($Nat \mid 0, 1, +$) gaat over tellen . **Maar:**

- Je ziet af van nederlandse namen voor getallen.
- en zelfs van het gebruik van cijfers (behalve 0 en 1),
- en van de logaritmische notatie ($89 = 10 \cdot 8 + 9$).
- en de essentie blijft over.

Is dit alles? Nee, er zijn ook drie belangrijke vergelijkingen:

equations *Vergelijkingen Voor Nat*

$$(x + y) + z = x + (y + z)$$

$$x + y = y + x$$

$$x + 0 = x$$

Deze vergelijkingen vertellen ons dat:

- nullen meestal kunnen worden weggelaten,
- haakjes (hier) overbodig zijn,
- je gewoon $1 + 1 + 1 + 1$ kunt schrijven en het niet uitmaakt hoe je haakjes zet.

Dus: een taal $(0, 1, +)$ met drie vergelijkingen vertelt ALLES wat we willen weten over natuurlijke getallen nadat we afzien van een serie bijzaken.

Dit is abstractie in de informatica.

Negatieve getallen

Voeg toe de operatie '–' dan hebben we meer termen zoals dat heet:

$$(Geh \mid 0, 1, +, -)$$

Haakjes zijn nu ineens van groot belang. Er zijn nu veel meer termen (uitdrukkingen, expressies, vormen, formules) zoals:

$$1 + -(1 + 0) \quad \text{en} \quad (1 + (-0)) + -0$$

equations *VergVoorGehelen*

$$(x + y) + z = x + (y + z)$$

$$x + y = y + x$$

$$x + 0 = x$$

$$x + (-x) = 0$$

Haakjes mogen niet meer weggelaten worden!

$$1 + -(1 + 1) \neq (1 + -1) + 1$$

Ook hier geldt: deze vergelijkingen leveren alle informatie die je wilt.

Voorbeeld: $1 + -4 = -3$

$$\begin{aligned} & 1 + -(1 + (1 + (1 + 1))) \\ &= (1 + 0) + -(1 + (1 + (1 + 1))) \\ &= (1 + ((1 + (1 + 1)) + -(1 + (1 + 1)))) + -(1 + (1 + (1 + 1))) \\ &= ((1 + (1 + (1 + 1))) + -(1 + (1 + 1))) + -(1 + (1 + (1 + 1))) \\ &= (-(1 + (1 + 1)) + (1 + (1 + (1 + 1)))) + -(1 + (1 + (1 + 1))) \\ &= -(1 + (1 + 1)) + ((1 + (1 + (1 + 1))) + -(1 + (1 + (1 + 1)))) \\ &= -(1 + (1 + 1)) + 0 \\ &= -(1 + (1 + 1)) \end{aligned}$$

Dat valt NIET MEE! Maar wie dit kan heeft het definitief begrepen.
Zo snap je ook **waarom** decimale notatie handig is.

Cijfers in een data type?

Je voegt een aantal **nieuwe namen** toe: **2,3,...,9** met eigen **equations** *GehelenMetDigits*

$$(x + y) + z = x + (y + z)$$

$$x + y = y + x$$

$$x + 0 = x$$

$$x + (-x) = 0$$

$$2 = 1 + 1$$

$$3 = 2 + 1$$

$$4 = 3 + 1$$

$$5 = 4 + 1$$

$$6 = 5 + 1$$

$$7 = 6 + 1$$

$$8 = 7 + 1$$

$$9 = 8 + 1$$

- Nu kun je bewijzen: $1 + -4 = -3$.
- abstracte datatypen zijn flexibel, je kunt precies toevoegen wat je wilt. Niet meteen de decimale notatie, op dit moment tenslotte overbodig.
- Je kunt alleen maar zien hoe handig iets is als je er ook vanaf kunt zien.
- De gewone $-$ voor aftrekken is nu een afkorting: $x - y = x + (-y)$, we werken verder met de 'minus' omdat dat eenvoudiger is.

Vermenigvuldigen

De gehelen bestaan met 0, 1, + en -. Je **KUNT** ze vermenigvuldigen, dat **MOET** niet. We voegen \cdot toe aan het data type (en vergeten de DIGITS verder):

(Geh | 0, 1, +, -, \cdot)

equations *Commutatieve Ringen*

$$(x + y) + z = x + (y + z)$$

$$x + y = y + x$$

$$x + 0 = x$$

$$x + (-x) = 0$$

$$(x \cdot y) \cdot z = x \cdot (y \cdot z)$$

$$x \cdot y = y \cdot x$$

$$x \cdot 1 = x$$

$$x \cdot (y + z) = x \cdot y + x \cdot z$$

CR is een heel algemeen systeem

- Het vertelt hoe je moet vermenigvuldigen als je kunt optellen en negatief maken.
- Het kan ergens tegen: je mag aannemen dat $5 = 0$ en dan werkt het ook uitstekend.
- Nu bewijzen we '2 maal 2 = 4', immers voor informatici doet het er toe of je dat uit je hoofd met leren (opzoeken in een tabel, al dan niet per computer) of dat het uit de axioma's volgt (een kwestie van AI):
- $2 \cdot 2 = (1 + 1) \cdot (1 + 1) = (1 + 1) \cdot 1 + (1 + 1) \cdot 1 = (1 + 1) + (1 + 1) = 1 + (1 + (1 + 1)) = 1 + (1 + (1 + 2)) = 1 + 3 = 4.$

CR is een heel algemeen systeem

- Het vertelt hoe je moet vermenigvuldigen als je kunt optellen en negatief maken.
- Het kan ergens tegen: je mag aannemen dat $5 = 0$ en dan werkt het ook uitstekend.
- Nu bewijzen we '2 maal 2 = 4', immers voor informatici doet het er toe of je dat uit je hoofd met leren (opzoeken in een tabel, al dan niet per computer) of dat het uit de axioma's volgt (een kwestie van AI):
- $2 \cdot 2 = (1 + 1) \cdot (1 + 1) = (1 + 1) \cdot 1 + (1 + 1) \cdot 1 = (1 + 1) + (1 + 1) = 1 + (1 + (1 + 1)) = 1 + (1 + (1 + 2)) = 1 + 3 = 4.$

Conclusie: met abstracte data typen in de hand is '2 maal 2 = 4' iets wat je kunt begrijpen (en niet alleen weten).

Nu komt een **afwijking** van het normale verhaal. Het idee van **deling** is dit:

gegeven x en y zoek een z zodat $x \cdot z = y$.

Deling is een operatie x/y die deze z oplevert. Je bent al een heel eind als je $x = 1$ neemt. We schrijven ook x^{-1} of $\frac{1}{x}$ voor $1/x$.

Splits syntax en semantiek: $1/x$ is wel syntax ongeacht de semantiek.

Wat is de semantiek (betekenis) van $(1 + (1 + 1)) + (1 + 1)$? Het is niet 5 maar 'het idee van 5 los van de notatie'. Daar kun je preciezer over zijn, via verzamelingen (wiskunde), of via congruenties (informatica).

$\|t\|$ staat voor de semantiek van t , als we de semantiek als onderwerp hebben. Anders staat t direct voor z 'n semantiek.

Syntax is wat je opschrijft. **Semantiek** is wat het betekent.

Een programma p is syntax, uit de semantiek $\|p\|$ kan volgen dat het een virus is. Voor informatica is het onderscheid tussen syntax en semantiek **essentieel**.

Informatici willen EERST syntax kunnen opschrijven en DAARNA de vraag stellen wat het betekent.

- VOORBEELD: is tekst t een 'correct' Java programma?
- Om de vraag te stellen moet je t wel op mogen schrijven.
- DUS: t is syntax, ook als ie niet syntactisch correct is.

Er is (CR) geen z met $0 \cdot z = 1$ dus de vraag "**wat is $\|1/0\|$?**" is serieus.

Voor wiskundigen een FILOSOFISCH ISSUE!

Voor informatici een kwestie van DESIGN!

Vraag: wat is $1/\text{FIETS}$?

Is het **IETS** of is het **NIETS**, of **geen van beide**?

Wat zegt een wiskundige: als je $1/\text{FIETS}$ opschrijft zul je er wel iets mee bedoelen, als je er niks mee bedoelt moet je het ook niet opschrijven.

Als je er wel wat mee bedoelt dan moet je het maar zeggen, je kan niet verwachten dat een ander er iets zinnigs over zegt.

wiskundige: $1/0$ is even zinloos als $1/\text{FIETS}$!

ADT: een kwestie van design!

Delen door 0 verbieden is curieus: je verbiedt door een rood stoplicht rijden toch alleen omdat het kan!

Wat is $\|1/0\|$?

Vanuit *de informatica*: we mogen best $1/0$ bij de syntax tellen, zonder te weten waar het voor staat. Dat moeten we dan daarna zeggen.

Daarmee is de vraag:

wat is volgens jou $1/0$?

een adequate vraag geworden.

een design decision (JAB & JVT)

Stel: $\|1/0\| = 0$.

Waarom is de 'wiskundige visie' moeilijk?

Neem de volgende vergelijking:

$$\frac{0}{(1 + x \cdot x)} + 1 = 1$$

Is deze juist (= mag je 'm opschrijven)? **Ja**, want je deelt nooit door nul. En nu

$$\frac{0}{((1 + x^2)^{500.000} + (1 + y^2)^{500.000} - (1 + z^2)^{500.000})} = 0$$

Ook goed, **want...** laatste stelling van Fermat (Andrew Wiles).
Tenslotte:

$$0/(1 + x^3) = 0$$

is mis. **Kies** $x = -1$.

Is er een algemeen verhaal over deze kwestie? Nee, open probleem over de logica van de breuken (sinds 1900/Hilbert's 10e probleem).

\mathbb{Q} bevat de rationale getallen (breuken). Rationaal staat tegenover irrationaal, zoals de wortel uit 2.

$$(\mathbb{Q} \mid 0, 1, +, -, \cdot, ^{-1})$$

Syntax

signature Σ

sorts *field*

operations

0: \rightarrow *field*;

1: \rightarrow *field*;

+: *field* \times *field* \rightarrow *field*;

-: *field* \rightarrow *field*;

\cdot : *field* \times *field* \rightarrow *field*;

$^{-1}$: *field* \rightarrow *field*

De axioma's die we al hadden.

equations *CR*

$$(x + y) + z = x + (y + z)$$

$$x + y = y + x$$

$$x + 0 = x$$

$$x + (-x) = 0$$

$$(x \cdot y) \cdot z = x \cdot (y \cdot z)$$

$$x \cdot y = y \cdot x$$

$$x \cdot 1 = x$$

$$x \cdot (y + z) = x \cdot y + x \cdot z$$

Verder...

De mooiste vergelijkingen die we voor de inverse operatie kunnen bedenken:

equations SIP (*Strong Inverse Properties*)

$$\begin{aligned}(-x)^{-1} &= -(x^{-1}) \\(x \cdot y)^{-1} &= x^{-1} \cdot y^{-1} \\(x^{-1})^{-1} &= x\end{aligned}$$

equations SIP (*andere notatie*)

$$\begin{aligned}\frac{1}{-x} &= -\frac{1}{x} \\ \frac{1}{x \cdot y} &= \frac{1}{x} \cdot \frac{1}{y} \\ \frac{1}{\left(\frac{1}{x}\right)} &= x\end{aligned}$$

$$(1 + x^2 + y^2 + z^2 + u^2)/(1 + x^2 + y^2 + z^2 + u^2) = 1$$

Uit SIP volgt $0^{-1} = 0$. Nu volgt nog één axioma L.

L (voor Lagrange, een Frans wiskundige uit de 19e eeuw):

$$\frac{1 + x^2 + y^2 + z^2 + u^2}{1 + x^2 + y^2 + z^2 + u^2} = 1 \quad (\text{L})$$

L levert de informatie dat voor een groot aantal getallen (uitdrukkingen om preciezer te zijn) q i.h.b. alle sommen van 4 kwadraten plus 1,

$$\frac{q}{q} = 1$$

Sommen van kwadraten zijn niet negatief, plus 1 zijn ze zeker positief, en dus kun je er zeker door delen met 1 als resultaat.

Jan Bergstra & John Tucker 2005:

*De vergelijkingen **CR**, **SIP** en **L** leveren alle gewenste informatie over de rationale getallen.*

Precies: steeds als twee breuken gelijk zijn kun je dat hiermee ook bewijzen. Dit is de eerste algebraïsche specificatie van de rationale getallen met inverse operatie.

In de software engineering zijn al lang zulke specificaties bekend, maar alle met tal van extra operaties en daardoor veel gecompliceerder.

Een field (lichaam) is een CR waarin IL geldt:

IL, Inverse Law

Voor elke $x \neq 0$ is er een y is zodat $x \cdot y = 1$.

Wat we nu weten is dat je aan het lichaam $(\mathbb{Q} \mid 0, 1, +, -)$ een inverse operatie \cdot^{-1} kunt toevoegen en dat dat een mooie set van vergelijkingen levert die het rekenen vastlegt.

Hier is een fraaie vergelijking in de wereld van de lichamen met deling.

RIL, Restricted Inverse Law

$$x \cdot (x \cdot x^{-1}) = x$$

RIL volgt uit IL. In de ADT's wil je met vergelijkingen werken, maar IL is geen vergelijking. Voorstel: vervang IL door RIL (naast CR en SIP).

Een CR die voldoet aan SIP en RIL is een **meadow** (beetje minder dan een field).

Er zijn veel meer meadows dan fields. Je kunt rekenen met $6 = 0$ (modulo 6) en dat gaat prima. Dan hebben we $2 \cdot 3 = 0$. Nuldelers, dat mag niet in een field maar wel in een meadow.

Open Probleem

Bestaat een vergelijking die geldt in alle fields (met deling) maar niet in alle meadows?

wat kunnen we hiermee?

- mooie puzzels formuleren
- leren rekenen met eerst syntax daarna semantiek
- leren rekenen langs de lijn van termherschrijven en ADT's
- van de grond af aan het wiskunde onderwijs opzetten vanuit de modernere concepten van de informatica (object-oriëntatie op de basisschool)
- het begrip van het rekenen leren en de techniek/uitvoering aan computers overlaten
- met meer detail de begrippen in de informatica bekijken: wat is nu een computervirus? Volgens sommige auteurs vergt een C-virus voor verspreiding juist de acties van een **MENS!**