Honours projecten BSc Informatica: twee voorstellen

mogelijk ook geschikt voor BSc Kunstmatige Intelligentie

Alban Ponse

section Theory of Computer Science

Informatics Institute, University of Amsterdam

https://staff.fnwi.uva.nl/a.ponse/

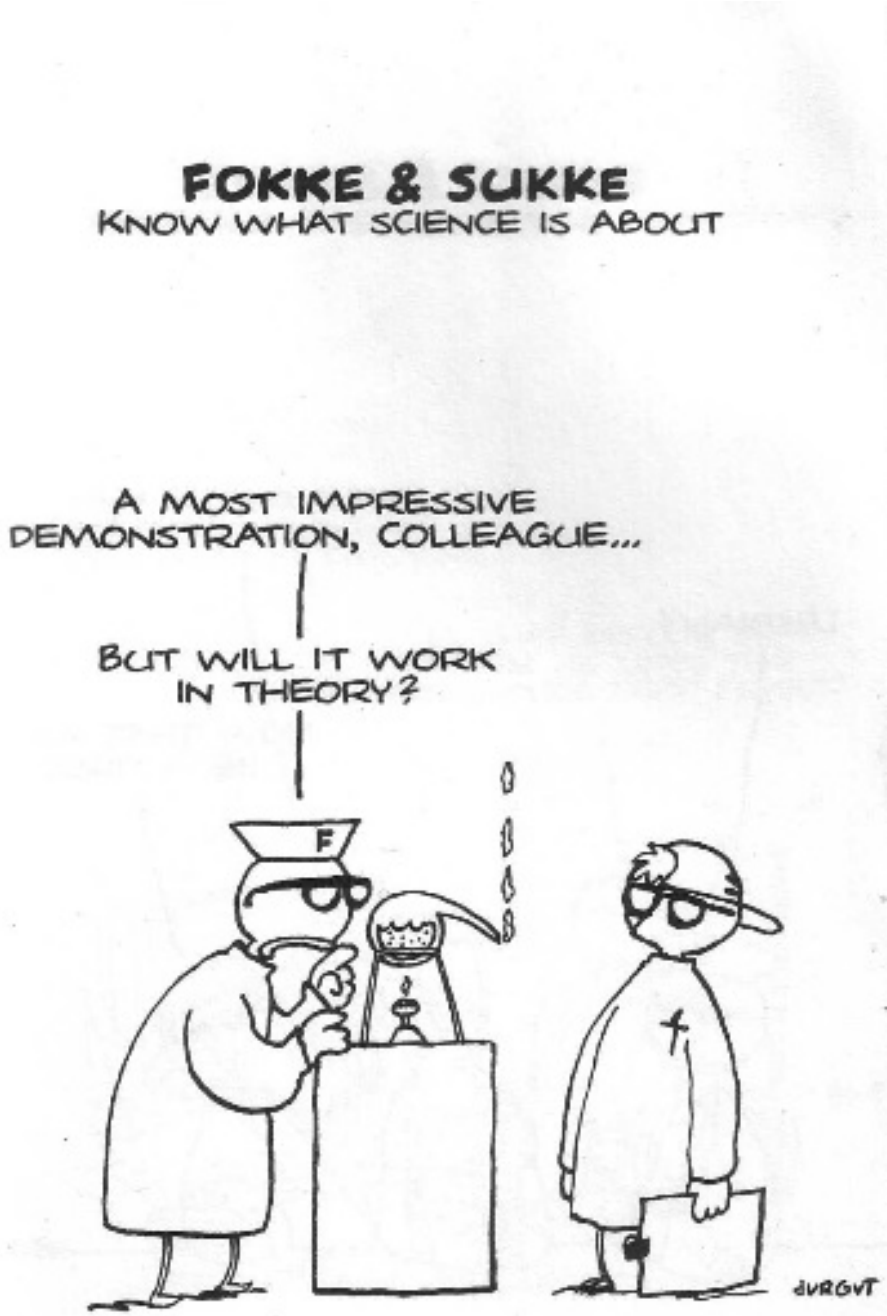
28 Augustus, 2015

Twee voorstellen voor
Honoursprojecten

Begeleiding
Wekelijkse, korte bijeenkomsten

Begeleiders
Inge Bethke & Alban Ponse

Onderzoeksgroep / context
Theory of Computer Science,
Instituut voor Informatica



Voorstel 1: Getalsrepresentaties en termherschrijven

Honoursproject voor INF (evt KI) voor **twee studenten** (mogelijk **één student**)

Getalsrepresentaties (binair, decimaal) zijn fundamenteel in de aritmetische programmatuur.

Interesse: normaalvormen voor diverse representaties van $\mathbb{Z}$ met $+$, $\cdot$ en de additieve inverse functie $-(\_)$.

Gezocht: *termherschrijfsystemen* die deze normaalvormen opleveren **en** die wenselijke (wiskundige) eigenschappen hebben.

Idee: Een decimaal getal als 132 zien we als normaalvorm met de waarde

$$132 = 10 \cdot 13 + 2 \quad \text{en} \quad 13 = 10 \cdot 1 + 3$$

dus, algemener, $wd = 10 \cdot w + d$ met $d \in D = \{0, 1, 2, ..., 9\}$ en $w \in D^*$.

Twee perspectieven op de structuur, uitgaande van D.

Eerste perspectief:

$$(1 \mathbin{\hat{d}} 3) \mathbin{\hat{d}} 2$$

representeert het decimale getal 132 en de functie $\_ \mathbin{\hat{d}} \_$ (infix notatie) is een *tree-constructor*.

Typische herschrijfregels:

$$\begin{aligned} 0 \mathbin{\hat{d}} x &\to x \\ x \mathbin{\hat{d}} (y \mathbin{\hat{d}} z) &\to (x + y) \mathbin{\hat{d}} z \\ x \cdot (y \mathbin{\hat{d}} 2) &\to ((x \cdot y) \mathbin{\hat{d}} 0) + (x \cdot 2) \end{aligned}$$

Vb: $2 \mathbin{\hat{d}} (1 \mathbin{\hat{d}} 5) = (2 + 1) \mathbin{\hat{d}} 5 = 3 \mathbin{\hat{d}} 5 \quad (= 35)$

Ook: $-(1 \mathbin{\hat{d}} 3) \mathbin{\hat{d}} 2)$ is normaalvorm, maar wat is die van $x \mathbin{\hat{d}} (-(y \mathbin{\hat{d}} z))$?

Tweede perspectief: Tien functies $\mathbb{Z} \to \mathbb{Z}$ (postfix notatie)

$$\_ :_d 0, \quad \_ :_d 1, \quad \_ :_d 2, \ \ldots, \quad \_ :_d 9,$$

decimal append zero, ..., en *decimal append nine*.

Vb: $(1 :_d 3) :_d 2$ representeert het decimale getal 132.

Typische herschrijfregels:

$$0 :_d 2 \to 2$$
$$x \cdot (y :_d 2) \to ((x \cdot y) :_d 0) + (x \cdot 2)$$

Vraagstelling: Bewijs *ground completeness* van de herschrijfsystemen in: *Three Datatype Defining Rewrite Systems for Datatypes of Integers each extending a Datatype of Naturals*, http://arxiv.org/abs/1406.3280v2

Ander relevant herschrijfsysteem voor $\mathbb{Z}$ staat beschreven op p.10 in: http://arxiv.org/pdf/1411.4410v1.pdf

Voorstel 2: Normalisatie van Boolese condities

Honoursproject voor INF (evt KI), bij voorkeur voor **twee studenten** (mogelijk **drie**)

Context: Welke logische wetten karakterizeren *short-circuit* evaluatie van Boolese condities in de setting van het imperatief programmeren?

$\Rightarrow$ vaststelling van de mogelijke zij-effecten die in de evaluatie van zulke condities kunnen optreden: een eenvoudig voorbeeld met && ("Logical and") is de conditie

$$(f(x) > 17) \;\&\&\; (y < 3)$$

want als evaluatie van $f(x)$ van invloed kan zijn op de waarde van y, kan deze conditie een **ander** resultaat (true of false) opleveren dan

$$(y < 3) \;\&\&\; (f(x) > 17)$$

en geldt commutativiteit van && dus **niet.**

Tegenvoorbeeld: links-distributiviteit van `&&` over `||` ("Logical or").

1) For program variable `i`, atom `(i==`k`)` with $k \in \mathbb{Z}$ is a test, and

2) Boolean evaluation of assignment `(i=e)` yields `false` iff `e`'s value is 0.

Then, if `i` has initial value 2,

`(i=i+1) && ((i==2) || (i==3))` evaluates to `true`, and

`((i=i+1) && (i==2)) || ((i=i+1) && (i==3))` evaluates to `false`

Logische wetten die **niet** gelden: (equationeel)

- alle varianten van distributiviteit, bv.
  x `&&` $(y$ `||` $z)$ $=$ $(x$ `&&` $y)$ `||` $(x$ `&&` $z)$ (als hierboven)
- alle varianten van absorptie, bv. x `&&` $(x$ `||` $y) = x$
- x `&&` $x = x$ en x `||` $x = x$

Aanpak: druk samengestelde condities uit mbv *Hoare's conditional* :

$$P \triangleleft Q \triangleright R$$

staat voor "*If* Q *then* P *else* R" en definitie van $\&\&$ is dan

$$P \,\&\&\, Q = Q \triangleleft P \triangleright F$$

met F een constante voor "false".

Dit is zgn. *short-circuit evaluation*: er wordt niet meer ge-evalueerd dan nodig: als P "false" oplevert, dan ook $P \,\&\&\, Q$ zonder dat Q wordt ge-evalueerd.

Short-circuit disjunctie $||$ (de "logical or") is gedefinieerd als

$$P \,||\, Q = T \triangleleft P \triangleright Q$$

met T een constante voor "true", en negatie $\neg$ is gedefinieerd als

$$\neg P = F \triangleleft P \triangleright T$$

Mogelijke zij-effecten van Boolese (atomaire) condities worden gekarakterizeerd door verschillende semantieken.

Free valuation congruence (fvc): alle mogelijke zij-effecten (met evaluatie naar true / false)

⋮ (Minstens drie andere semantieken)

Static vc (svc): als in propositielogica (geen zij-effecten)

Vraagstelling: Definieer normalisatie van condities volgens de diverse semantieken, en bespreek het gebruik van tools hierbij (er is een termherschrijftool beschikbaar)

Strategie: syntactische transformatie op condities als beschreven in http://arxiv.org/abs/1504.08321v2

N.B.

1. Varianten van `&&` en `||` die volledige ("strict") evaluatie voorschrijven, zijn definieerbaar in termen van short-circuit evaluatie:

$$P \,\&\, Q = (P \mid\mid (Q \,\&\&\, F)) \,\&\&\, Q$$
$$P \mid Q = (P \,\&\&\, (Q \mid\mid T)) \mid\mid Q$$

2. Voor atomen a, b, c kan

$$a \triangleleft b \triangleright c$$

niet worden uitgedrukt mbv `&&` en `||` en $\neg$ als er (sterke) zij-effecten mogen optreden.

3. Meer informatie beschikbaar op
`https://staff.fnwi.uva.nl/a.ponse/scl.php`