Modellering van lijsten in 2de orde getypte λ-calculi

Maarten Fokkinga, 22 december 1987

In [Fokkinga 1987, § 5.3] wordt getoond hoe lijsten in de 2de orde (= Reynolds-Girard-) getypte λ-calculus gemodelleerd kunnen worden, door een specifiek programma-schema te geven. Henk Barendregt was geïnteresseerd in de formulering van een stelling waarvan dat programma-schema (een deel van) het bewijs vormt. Dit verhaal is een eerste poging daartoe:

> we geven een heel algemene definitie van het begrip (1ste en 2de orde) getypte λ-calculi, en presenteren (de essentie van) de constructie als een transformatie tussen λ-calculi die -in zekere zin- convertibiteit exact behoudt.

Behalve dit aspect van "modellering van datatypen" (modellering van lijsten is maar een specifiek geval), is er het aspect van "modulariteit", dat ook in [Fokkinga 1987, § 5.3 en 5.4] getoond wordt. Een nette formulering van 'modulariteit' is in de maak; de begrippen (term-)fragment -isolatie, scope-beperking, en fragment-abstractie komen daarbij aan bod in afzonderlijke stellingen.

Definitie van het begrip "een λ-calculus"

De diverse willekeurig gekozen verzamelingen moeten veiligheidshalve maar onderling disjunct zijn. Dat geef ik nergens meer expliciet aan.

1. $\mathbf{A}$ is een vast gekozen aftelbaar oneindige verzameling.
   $\alpha \in \mathbf{A}$ heet een <u>type-variabele</u>. □

2. Een <u>type-basis</u> $\mathbf{\Gamma}$ is een tweetal $\mathbf{\Gamma} = (\Gamma, arity)$ waarbij Γ een verzameling is, en arity een afbeelding van Γ naar natuurlijke getallen.
   $\gamma \in \Gamma$ heet een <u>type-constructor</u>.
   Wanneer $\gamma \in \Gamma$ schrijven we ook "$\gamma \in \mathbf{\Gamma}$".
   Wanneer $\Gamma = \emptyset$ schrijven we ook "$\mathbf{\Gamma} = \emptyset$". □

   Voorbeelden van mogelijke type-constructoren zijn:

<u>int</u>, <u>bool</u>, <u>char</u>	allen met arity 0
<u>list</u>, <u>stack</u>	met arity 1
×, +	met arity 2

   We nemen → nooit op in een type-basis maar bouwen deze expliciet in in de type-formatie-regels.

3. Een <u>orde</u>[ℓ] is een element van $\{1, 2\}$. □

   Bij orde 2 zullen Λ en $\forall$ (binding van type-variabelen) worden toegelaten. Dat is bij orde 1 niet het geval.

4. Zij $\mathbf{\Gamma} = (\Gamma, \text{arity})$ een type-basis en ℓ een orde.
De verzameling $\mathbb{T}(\mathbf{\Gamma}, \ell)$ van <u>$(\mathbf{\Gamma}, \ell)$-typen</u> wordt als volgt inductief gedefinieerd: (we schrijven $\mathbb{T}$ voor $\mathbb{T}(\mathbf{\Gamma}, \ell)$)
   - voor $\alpha \in \mathbf{A}$ is $\alpha \in \mathbb{T}$
   - voor $\tau, \sigma \in \mathbb{T}$ is $(\tau \to \sigma) \in \mathbb{T}$
   - voor $\gamma \in \Gamma$ met $\text{arity}(\gamma) = n$ en $\tau_1, \ldots, \tau_n \in \mathbb{T}$ is $\gamma(\tau_1, \ldots, \tau_n) \in \mathbb{T}$

   en ingeval $\ell = 2$:
   - voor $\alpha \in \mathbf{A}$ en $\tau \in \mathbb{T}$ is $(\forall \alpha . \tau) \in \mathbb{T}$.

5. FV en substitutie $(:=)$ zijn zoals gebruikelijk.

6. $\mathbf{X}$ is een vast gekozen oneindig aftelbare verzameling.
$x \in \mathbf{X}$ heet een <u>term-variabele</u>.

7. type is vast gekozen in $\prod_{\mathbf{\Gamma}} (\mathbf{X} \to \mathbb{T}(\mathbf{\Gamma}, 2))$, (m.a.w. voor iedere $\mathbf{\Gamma}$ is $\text{type}_{\mathbf{\Gamma}} \in \mathbf{X} \to \mathbb{T}(\mathbf{\Gamma}, 2)$ vast gekozen), zó danig dat voor will. $\mathbf{\Gamma}, \ell$:

$$\forall \tau \in \mathbb{T}(\mathbf{\Gamma}, \ell). \ \{x \in \mathbf{X} \mid \text{type}_{\mathbf{\Gamma}}(x) = \tau\} \text{ is oneindig,}$$

~~(6) $\forall \tau \in \mathbb{T}(\mathbf{\Gamma}, 1). \ \forall x \in \mathbf{X}. \ \text{type}_{\mathbf{\Gamma},1}(x) = \tau \Rightarrow \text{type}_{\mathbf{\Gamma}}(x) = \tau$.~~

~~(m.a.w. $\text{type}_{\mathbf{\Gamma},1} \restriction \mathbb{T}(\mathbf{\Gamma}, 1) = \text{type}_{\mathbf{\Gamma},2} \restriction \mathbb{T}(\mathbf{\Gamma}, 2)$.)~~

~~We~~ schrijven voortaan $\text{type}_{\mathbf{\Gamma}}$ i.p.v. $\text{type}_{\mathbf{\Gamma},\ell}$; zie conditie (6).

In een contekst waarin $\mathbf{\Gamma}$ bekend is, schrijven we soms x^τ voor $x \in \mathbb{X}$ met $\mathrm{type}_\Gamma(x) = \tau$. □

Omdat A, $\mathbb{X}$ en $\mathrm{type}(.)$ vast gekozen zijn hoeven ze niet expliciet als parameter vermeld te worden.

8. Zij Γ een type-basis en ℓ een orde.
Een (Γ,ℓ)-term-basis $\mathbb{C}$ is een tweetal $\mathbb{C} = (C, ts)$ waarbij C een verzameling is en

$$ts \in C \to \{ (\vec{\alpha}; \tau_0, \ldots, \tau_n) \mid n \geq 0,\ \vec{\alpha} \in A^*,\ \tau_i \in \mathbb{T}(\Gamma,\ell) \}.$$

$c \in C$ heet een (Γ,ℓ)-term-constructor.
$ts(c)$ heet het type-schema van c.
We schrijven soms $c \in \mathbb{C}$ wanneer $c \in C$. □

Voorbeelden van (Γ,ℓ)-term-constructoren en hun type-schema:

$\underline{zero}, \underline{one}, \ldots$	met	$ts(\underline{zero}) = ts(\underline{one}) = \cancel{\underline{nat}}\ (\emptyset; \underline{nat})$
$\underline{true}, \underline{false}$	met	$ts(\underline{true}) = ts(\underline{false}) = (\emptyset; \underline{bool})$
$\underline{nil}$	met	$ts(\underline{nil}) = (\alpha; \underline{list}(\alpha))$
$\underline{cons}$	met	$ts(\underline{cons}) = (\alpha; \underline{list}(\alpha), \alpha, \underline{list}(\alpha))$
$\underline{cons}^*$	met	$ts(\underline{cons}^*) = (\alpha; \alpha \to \underline{list}(\alpha) \to \underline{list}(\alpha))$
$\underline{lrec}$	met	$ts(\underline{lrec}) = (\alpha,\beta; \beta, \underline{list}(\alpha), \alpha\to\beta\to\beta, \beta)$
$\underline{lrec}^*$	met	$ts(\underline{lrec}^*) = (\alpha,\beta; \underline{list}(\alpha) \to (\alpha\to\beta\to\beta) \to \beta \to \beta)$
$\underline{lrec}'$	met	$ts(\underline{lrec}') = (\alpha; \underline{list}(\alpha) \to \forall\beta.(\alpha\to\beta\to\beta)\to\beta\to\beta)$
$\underline{lrec}''$	met	$ts(\underline{lrec}'') = (\emptyset; \forall\alpha.\underline{list}(\alpha) \to \forall\beta.(\alpha\to\beta\to\beta)\to\beta\to\beta)$

De type-basis Γ moet kennelijk minstens $\underline{nat}$, $\underline{bool}$, en $\underline{list}$ bevatten, en kennelijk is $\ell = 2$.

9. Een <u>basis</u> is een drietal (Γ, ℓ, C) van type-basis Γ, orde ℓ en (Γ,ℓ)-termbasis C. □

10. Zij (Γ, ℓ, C) een basis; we schrijven τ voor $\tau(\Gamma,\ell)$.
De collectie $\prod_{\tau \in \tau(\Gamma,\ell)} \mathbb{M}^\tau(\Gamma,\ell,C)$ (d.w.z. een verzameling $\mathbb{M}^\tau(\Gamma,\ell,C)$ voor elke $\tau \in \tau(\Gamma,\ell)$) wordt als volgt inductief –simultaan voor alle $\tau \in \tau$– gedefinieerd:
(we schrijven $\mathbb{M}^\tau$ voor $\mathbb{M}^\tau(\Gamma,\ell,C)$)

- voor $x \in X$ met $\text{type}_\Gamma(x) = \tau$ is $x \in \mathbb{M}^\tau$,
- voor $x \in X$ met $\text{type}_\Gamma(x) = \tau$ en $M \in \mathbb{M}^\sigma$ is $(\lambda x.M) \in \mathbb{M}^{(\tau\to\sigma)}$,
- voor $M \in \mathbb{M}^{\tau\to\sigma}$ en $N \in \mathbb{M}^\tau$ is $(MN) \in \mathbb{M}^\sigma$,
- voor $c \in C$ met $ts(c) = (\vec{\alpha}; \tau_0, \ldots, \tau_n)$ en $\vec{\rho} \in \tau$ en voor $M_i \in \mathbb{M}^{\tau_i[\vec{\alpha}:=\vec{\rho}]}$ $(i = 1, \ldots, n)$ is $c(M_1, \ldots, M_n) \in \mathbb{M}^{\tau_0[\vec{\alpha}:=\vec{\rho}]}$,

en ingeval $\ell = 2$:

- voor $\alpha \in A$ en $M \in \mathbb{M}^\tau$ zodanig dat
$$\neg\, \exists \sigma \in \tau.\ \alpha \in FV(\sigma) \wedge \exists x^\sigma \in X.\ x \in FV(M)$$
(m.a.w. $\forall \sigma \in \tau.\ \forall x^\sigma \in X.\ x \in FV(M) \Rightarrow \alpha \notin FV(\sigma)$),
is $(\Lambda\alpha.\, M) \in \mathbb{M}^{(\forall\alpha.\tau)}$,
- voor $\tau \in \tau$ en $M \in \mathbb{M}^{\forall\alpha.\sigma}$ is $(M\tau) \in \mathbb{M}^{\sigma[\alpha:=\tau]}$.

$M \in \mathbb{M}^\tau$ heet een <u>(Γ,ℓ,C)-term van type</u> τ, kortweg: <u>term</u>.
$\mathbb{M}(\Gamma,\ell,C) = \bigcup_{\tau\in\tau} \mathbb{M}^\tau$ □

Merk op dat $\mathbb{M}^\tau(\Gamma,1,C) \subseteq \mathbb{M}^\tau(\Gamma,2,C)$ voor $\tau \in \tau(\Gamma,1)$.
en dat voor $\tau \neq \sigma$ toch $\mathbb{M}^\tau(\Gamma,\ell,C) \cap \mathbb{M}^\sigma(\Gamma,\ell,C) \neq \emptyset$ kan zijn!

11. FV (hierboven al gebruikt!) en substitutie (:=) op termen zijn zoals gewoonlijk, (en onafhankelijk van bases!).

12. Zij $B = (\Gamma, \ell, C)$ en $B' = (\Gamma', \ell', C')$ ieder een basis.
De basis $\underline{B + B'}$ is $(\Gamma \cup \Gamma', \max(\ell, \ell'), C \cup C')$. $\square$

13. Zij $B = (\Gamma, \ell, C)$ een basis.
Een $\underline{\text{B-contractie}}$ R is een element van

$$\prod_{\text{basis } B' = (\Gamma', \ell', C')} \prod_{\tau \in \mathbb{T}(\Gamma' \cup \Gamma, \max(\ell, \ell'))} \mathcal{P}(\mathbb{M}^\tau(B+B') \times \mathbb{M}^\tau(B+B'))$$

d.w.z.

voor iedere basis $B' = (\Gamma', \ell', C')$ en iedere $\tau \in \mathbb{T}(\Gamma \cup \Gamma', \max(\ell, \ell'))$ is R een relatie op $\mathbb{M}^\tau(B + B')$. $\square$

14. De $(\emptyset, 1, \emptyset)$-contractie $\underline{(\beta)}$ is als volgt gedefinieerd:

$(\beta) = \{ ((\lambda x . M) N), M[x := N]) \mid$
$B' = (\Gamma', \ell', C')$ is een basis; $\sigma, \tau \in \mathbb{T}(\Gamma', \max(\ell', \ell))$;
$x^\sigma \in \mathbb{X}$, $M \in \mathbb{M}^\tau((\emptyset, 1, \emptyset) + B')$, $N \in \mathbb{M}^\sigma((\emptyset, 1, \emptyset) + B') \}$

De $(\emptyset, 2, \emptyset)$-contractie $\underline{(\beta')}$ is als volgt gedefinieerd:

$(\beta') = \{ (((\Lambda \alpha . M) \sigma), M[\alpha := \sigma]) \mid$
$B' = (\Gamma', \ell', C')$ is een basis; $\tau \in \mathbb{T}(\Gamma', \underline{\max}(\ell, \ell'))$,
$M \in \mathbb{M}^\tau((\emptyset, 2, \emptyset) + B')$ z.d.d. $\forall \rho \forall x^\rho \in FV(M).\ \alpha \notin FV(\rho)$,
$\sigma \in \mathbb{T}(\Gamma', \underline{\max}(\ell, \ell')) \}$ $\square$

Merk op dat als B, B' bases zijn, en R een B-contractie, dan is R ook een $B+B'$-contractie. Dit geldt met name voor (β) en (β'). Ook: R, R' beide B-contractie $\Rightarrow R \cup R'$ is B-contractie.

15. Een $\underline{\lambda\text{-calculus}}$, kortweg: $\underline{\text{calculus}}$, λ is een 4-tal (Γ, ℓ, C, R) waarbij (Γ, ℓ, C) een basis is en R een

(Γ, ℓ, C)- contractie. $\square$

16. Zij $\lambda = (\Gamma, \ell, C, R)$ een calculus.

We definiëren

$\mathbb{T}_\lambda = \mathbb{T}(\Gamma, \ell)$

$\mathbb{M}^\tau_\lambda = \mathbb{M}^\tau(\Gamma, \ell, C)$ voor alle $\tau \in \mathbb{T}_\lambda$

$\mathbb{M}_\lambda = \mathbb{M}(\Gamma, \ell, C)$

$(\rightarrow_\lambda)$ = de compatibele afsluiting van $R^{\cup(\beta)\cup(\beta')}$ beperkt tot termen in $\mathbb{M}_\lambda$

$(\twoheadrightarrow_\lambda)$ = de reflexieve, transitieve afsluiting van $(\rightarrow_\lambda)$

$(=_\lambda)$ = de refl., symm., transitieve afsluiting van $(\rightarrow_\lambda)$

$\Gamma_\lambda = \Gamma$

$C_\lambda = C$

$\ell_\lambda = \ell$

$R_\lambda = R$. $\square$

13ª Zij B een basis en R een B-contractie.

Wanneer $(M.N) \in R$ noteren we $M \rightarrow_R N$. $\square$

~~16. Zij $\lambda = (\Gamma, \ell, C, R)$ een calculus.~~

We definiëren

$$\text{orde}(\lambda) = \ell$$

$$\mathbb{M}_\lambda = \mathbb{M}(\Gamma, \ell, C), \qquad \mathbb{M}^\tau_\lambda = \mathbb{M}^\tau(\Gamma, \ell, C) \text{ voor } \tau \in \Pi_\lambda$$

$$\Pi_\lambda = \pi(\Gamma, \ell)$$

$$(\twoheadrightarrow_\lambda) = (\twoheadrightarrow_R)$$

$$(\rightarrow_\lambda) = (\rightarrow_R) \cup (\beta) \cup (\beta')$$

$$\cancel{(=_\lambda) = (=_R).} \qquad \square$$

17. Zij voor $i=1,2$ $\lambda_i = (\Gamma_i, \ell_i, C_i, R_i)$ een calculus.
We definiëren $\underline{\lambda_1 + \lambda_2}$ als de calculus

$$(\Gamma_1 \cup \Gamma_2,\ \max(\ell_1, \ell_2),\ C_1 \cup C_2,\ R_1 \cup R_2) \qquad \square$$

18. Een <u>calculustransformatie</u> $\ldots^*$ is een afbeelding die willekeurige calculus λ afbeeldt op een drietal, bestaande uit

- een calculus, zeg λ^*
- een afbeelding $\ldots^* : \Pi_\lambda \rightarrow \Pi_{\lambda^*}$
- een stel afbeeldingen $\ldots^* : \mathbb{M}^\tau_\lambda \longrightarrow \mathbb{M}^{\tau^*}_{\lambda^*}$ (voor $\tau \in \Pi_\lambda$). $\square$

Merk op dat in def. 18 <u>niet</u> geëist wordt dat

- voor $\gamma \in \Gamma$ ook $\gamma^* \in \Gamma^*$ (wel dat $\gamma^* \in \Pi_{\lambda^*}$)
- voor $c \in C$ ook $c^* \in C^*$ (wel dat $c^* \in \mathbb{M}_{\lambda^*}$)

waarbij $(\Gamma, \ell, C, R) = \lambda$ en $(\Gamma^*, \ell^*, C^*, R^*) = \lambda^*$.

19. Een calculustransformatie $\ldots^*$ heet <u>syntactisch homomorf</u> indien voor willekeurige calculus λ geldt:

er is voor iedere samenstellingswijze van typen en termen een context zó dat het $\ldots^*$-beeld van een samengesteld type of term gevormd wordt door in de context de beelden er onderdelen te plaatsen.

Dus er zijn bij gegeven $\lambda = (\Gamma, \ell, C, \underline{R})$ contexten $\mathcal{C}_\gamma$ $(\gamma \in \Gamma)$, $\mathcal{C}_\forall$, $\mathcal{C}_\to$, $\mathcal{C}_\alpha$, $\mathcal{C}_\lambda$, $\mathcal{C}_{\text{applicatie}}$, $\mathcal{C}_c$ $(c \in \mathfrak{C})$, $\mathcal{C}_\Lambda$, $\mathcal{C}_{\text{type-applicatie}}$, $\mathcal{C}_x$, zo dat, bijvoorbeeld

$$(M\,N)^* = \mathcal{C}_{\text{applicatie}}[M^*, N^*]$$

$$c(M_1, \ldots, M_n)^* = \mathcal{C}_c[M_1^*, \ldots, M_n^*]$$

De preciese definitie van context wordt hier achterwege gelaten. $\square$

20. Een calculustransformatie $\ldots^*$ heet <u>conversie-isomorf</u> hortweg <u>(=)-isomorf</u>, indien voor willekeurige calculus λ geldt:

$$\forall \tau \in \mathbb{T}_\lambda \quad \forall M, N \in \mathbb{M}_\lambda^\tau : \qquad M =_\lambda N \iff M^* =_{\lambda^*} N^* \;.\square$$

Merk op dat Def. 20 <u>niet</u> eist dat

$$\forall M, N \in \mathbb{M}_\lambda . \quad M =_\lambda N \iff M^* =_{\lambda^*} N^* .$$

Het is volgens Def. 20 toegelaten dat er $\tau, \sigma \in \mathbb{T}_\lambda$ zijn en $M \in \mathbb{M}_\lambda^\tau$, $N \in \mathbb{M}_\lambda^\sigma$ met

$$M \neq_\lambda N \quad \wedge \quad M^* =_{\lambda^*} N^* .$$

Kennelijk is dan $\tau \neq \sigma$ maar $\tau^* = \sigma^*$. Dit doet zich voor wanneer de transformatie de constanten *zero* en *false* (van type *nat* en *bool*) afbeeldt op de 2-de orde getypte Church-numeral voor nul resp. de Church-representatie voor 'false': *zero* en *false* zijn van verschillend type en niet convertibel maar hun beelden zijn identiek.

Bij ongetypeerde calculi zouden we willen definiëren: een transformatie heet *semantiek-isomorf* indien

- convertibiliteit behouden blijft onder de afbeelding
- de afbeelding geen "ongewenste" convertibiliteit introduceert.

Bijvoorbeeld, $\underline{zero}^* =_{\lambda^*} \underline{false}^*$ is een conversie die best geïntroduceerd mag worden, ook al zal $\underline{zero} \neq_\lambda \underline{false}$, maar $\underline{true}^* =_{\lambda^*} \underline{false}^*$ is een conversie die ongewenst is.

Een eerste, eenvoudige, stelling

De volgende stelling zegt dat term-constructoren met arity ≥ 1 niet nodig zijn.

Stelling 1

Er is een syntactisch homomorfe, $(=)$-isomorfe calculustransformatie $\ldots^*$ zodat voor willekeurige calculus $\lambda = (\Gamma, \ell, \mathbb{C}, R)$ geldt $\lambda^* = (\Gamma, 2, \mathbb{C}^*, R^*)$ met $\mathbb{C}^* = (C, ts^*)$ en $ts^*(c) \in \{(\emptyset; \tau_0) \mid \tau_0 \in \mathcal{T}_{\lambda^*}\}$.

(aangenomen $\mathbb{C} = (C, ts)$)

Bewijs

Definieer voor $c \in C$

$$ts^*(c) = (\emptyset;\ \forall\vec{\alpha}.\ \tau_1 \to \cdots \to \tau_n \to \tau_0)$$

$$\text{indien}\quad ts(c) = (\vec{\alpha};\ \tau_0, \ldots, \tau_n).$$

Definieer $\ldots^*$ voor λ-termen als de homomorfe extensie van: voor $c(M_1, \ldots, M_n) \in \mathbb{M}_\lambda^{\tau_0[\vec{\alpha}:=\vec{\rho}]}$ en $M_i \in \mathbb{M}_\lambda^{\tau_i[\vec{\alpha}:=\vec{\rho}]}$

$$c(M_1, \ldots, M_n)^* = (c'\ \vec{\rho}\ M_1^* \ldots M_n^*).$$

Definieer

$$R^* = \{M^* \to N^* \mid (M, N) \in R\}.$$

Definieer $\ldots^*$ voor λ-typen als de identiteit.

Het is nu duidelijk dat (i) $\ldots^*$ een calculustransformatie is, (ii) $\ldots^*$ syntactisch homomorf is, en (iii) ts^* aan de gewenste conditie voldoet. Bovendien is de transformatie een $(\to)$- en dus ook $(\twoheadrightarrow)$- en $(=)$-isomorfie. $\square$

Een tweede, nu niet-triviale, stelling

We laten zien dat een specifieke calculus-modellering van "lijsten" getransformeerd kan worden, (=)-isomorf, naar de calculus zonder speciale lijstvoorzieningen.

Van horen zeggen ken ik de volgende bewering:

> Alle XXX-bewijsbare totale recursieve functies over een inductief gedefinieerde verzameling (datatype?) zijn uit te drukken middels primitieve recursie van hogere orde (en compositie e.d.?)

Een dergelijke bewering schijnt door Gödel bewezen te zijn. Op dit moment ontbreekt mij de tijd om ~~precies~~ te achterhalen hoe de bewezen bewering precies luidt. De bewering motiveert om in het geval van "lijsten" de onderstaande calculus te beschouwen.

Definieer $\quad lists = (\Gamma_{lists}, 1, C_{lists}, R_{lists})\quad$ met

$\Gamma_{lists} = (\{\underline{list}\},\ (\underline{list} \mapsto 1)\}$ $\qquad$ dus arity$(\underline{list}) = 1$

$C_{lists} = (C,\ ts)$ beide hieronder opgesomd:

$$ts(\underline{nil}) = (\alpha;\ \underline{list}(\alpha)\ )$$

$$ts(\underline{cons}) = (\alpha;\ \ \alpha \to \underline{list}(\alpha) \to \underline{list}(\alpha))$$

$$ts(\underline{lrec}) = (\alpha, \beta;\ \ \underline{list}(\alpha) \to (\alpha \to \beta \to \beta) \to (\beta \to \beta))$$

$$\begin{aligned} \mathrm{Rlists} = \{\ & \underline{\mathrm{lrec}}\ \underline{\mathrm{nil}}\ F\,A \to A, \\ & \underline{\mathrm{lrec}}\ (\underline{\mathrm{cons}}\ X\,Y)\ F\,A \to F\,X\,(\underline{\mathrm{lrec}}\ Y\,F\,A) \end{aligned}$$

$$\begin{aligned} \mid\ & \text{er is basis } B',\ \sigma,\tau \in \mathbb{T},\ X \in \mathbb{M}^{\sigma},\ Y \in \mathbb{M}^{\underline{list}(\sigma)}, \\ & F \in \mathbb{M}^{\sigma\to\tau\to\tau},\ A \in \mathbb{M}^{\tau} \\ & \text{waarbij}\quad \mathbb{T} \text{ staat voor } \mathbb{T}(\Gamma_{lists} \cup \Gamma_{B'},\ \max(1, \ell_{B'})) \\ & \qquad\qquad \mathbb{M}^{\tau} \text{ staat voor } \mathbb{M}^{\tau}((\Gamma_{lists}, 1, C_{lists}) + B')\quad (\tau \in \mathbb{T})\ \}. \end{aligned}$$

De constante $\underline{\mathrm{lrec}}$ modelleert primitieve recursie over lijsten, want

(i) $\mathrm{primrec}(\mathrm{cons}(x_1, \ldots, \mathrm{cons}(x_n, \mathrm{nil})\ldots), f, a) = f(x_1, \ldots, f(x_n, a)\ldots)$

(ii) $\underline{\mathrm{lrec}}\ (\underline{\mathrm{cons}}\ X_1 \ldots (\underline{\mathrm{cons}}\ X_n\ \underline{\mathrm{nil}})\ldots)\ F\,A =_{lists} (F\,X_1 \ldots (F\,X_n\,A)\ldots)$

<u>Stelling 2</u>

Er is een syntactisch homomorfe, $(=)$-isomorfe calculustransformatie $\ldots^*$ zó dat voor willekeurige calculus $\lambda = (\Gamma, \ell, C, R)$ geldt:

$$((\Gamma, \ell, C, R) + lists)^* = (\Gamma, 2, C, R)$$

<u>Bewijs</u>

We definiëren $\ldots^*$ als volgt.

Voor willekeurige Γ, ℓ, C en R is

$$\Gamma^* = \Gamma \setminus \{\underline{list}\}$$

$$\ell^* = 2$$

$$C^* = C \setminus \{\underline{\mathrm{nil}}, \underline{\mathrm{cons}}, \underline{\mathrm{lrec}}\}$$

$R^* = R \setminus R_{lists}$. We zetten $\lambda' = (\Gamma, 2, C, R)$ en voorts definiëren we $\ldots^*$ op $\mathbb{T}_{\lambda+lists}$ als de homomorfe extensie van

$$\underline{list}(\tau)^* = (\forall\alpha.\ (\tau^* \to \alpha \to \alpha) \to \alpha \to \alpha\ ) \quad \in \mathbb{T}_{\lambda'}$$

$$\gamma(\tau_1, \ldots, \tau_n)^* = \gamma(\tau_1^*, \ldots, \tau_n^*) \qquad \text{voor } \gamma \neq \underline{list}$$

en tenslotte definiëren we voor $\tau \in \mathbb{T}_{\lambda+list}$ de afbeelding $\ldots^*$ als de homomorfe extensies van

- voor $\underline{nil} \in \mathbb{M}_{\lambda+lists}^{\underline{list}(\tau)}$ (dit is dus voor álle $\tau \in \mathbb{T}_{\lambda+list}$ waar)

$$\underline{nil}^* = (\Lambda\alpha.\ \lambda f^{\tau^* \to \alpha \to \alpha}\ a^{\alpha}.\ a) \quad \in \mathbb{M}_{\lambda'}^{\underline{list}(\tau)^*}$$

- voor $\underline{cons} \in \mathbb{M}_{\lambda+lists}^{\tau \to \underline{list}(\tau) \to \underline{list}(\tau)}$

$$\underline{cons}^* = (\lambda x^{\tau^*}\ l^{\underline{list}(\tau)^*}.\ \Lambda\alpha\ \lambda f^{\tau^* \to \alpha \to \alpha}\ a^{\alpha}.\ f\, x\, (l\, \alpha\, f\, a)\ ) \quad \in \mathbb{M}_{\lambda'}^{(\tau \to \underline{list}(\tau) \to \underline{list}(\tau))^*}$$

- voor $\underline{lrec} \in \mathbb{M}_{\lambda+lists}^{\underline{list}(\tau) \to (\tau \to \sigma \to \sigma) \to \sigma \to \sigma}$

$$\underline{lrec}^* = (\lambda l^{\underline{list}(\tau)^*}.\ l\ \sigma^*) \quad \in \mathbb{M}_{\lambda'}^{(\underline{list}(\tau) \to (\tau \to \sigma \to \sigma) \to \sigma \to \sigma)^*}$$

Het is eenvoudig te verifiëren dat $\ldots^*$ een calculustransformatie is. Per constructie geldt $(\lambda + lists)^* = \lambda'$ en is $\ldots^*$ syntactisch homomorf. Van de "$\Leftrightarrow$" die geëist wordt opdat $\ldots^*$ $(=)$-isomorf is, is "$\Rightarrow$" eenvoudig te verifiëren en is "$\Leftarrow$" iets moeilijker. Het bewijs van deze laatste implicatie zal wel gebaseerd zijn op het feit dat in iedere calculus iedere reductie ook in een specifieke volgorde (left most) gedaan kan worden. [Dit heb ik niet verder geverifieerd, dd. dec '87.] $\square$

Op voor de handliggende wijze kunnen we calculi nats en bools definiëren. Er geldt dan de volgende stelling

(i) er is een calculustransformatie $\ldots^*$ zodat voor will. calculus (Γ, ℓ, C, R): $\quad ((\Gamma, \ell, C, R) + \text{nats})^* = (\Gamma, 2, C, R)$,

(ii) idem met: $((\Gamma, \ell, C, R) + \text{bools})^* = (\Gamma, 2, C, R)$,

en de transformaties zijn syntactisch homomorf en $(=)$-isomorf.

De stelling waarvoor (Fokkinga 1987, §5.3) een bewijs geeft

De transformatie van Stelling 2 is enerzijds erg elegant te noemen omdat hij syntactisch homomorf is, maar andertijds erg onhandig omdat de pure-calculus-modellering van lists nu verspreid door/over alle termen komt. Zouden we de modellering ietwat anders willen maken, dan moeten we – conceptueel – alle termen en subtermen doorzoeken en de transformaties van lists-typen en -termen opsporen. (En let wel, sommige voorkomens van $(\Lambda\alpha.\,\lambda f.\,\lambda a.\,a) \in \mathbb{M}_{\lambda}^{\forall\sigma}$ met $\sigma = \forall\alpha.(\tau \to \alpha \to \alpha) \to \alpha \to \alpha$ zijn wél ontstaan als transformatie van <u>nil</u>, en sommige wellicht niet!)

De volgende Stelling geeft een "modulaire" calculustransformatie van λ+lists naar λ. (Het begrip "modulariteit" werk ik later op fundamentele wijze uit; helaas kost het me enige moeite een nette en bevredigende formulering te krijgen, zodat ik nu alvast deze specifieke toepassing van "modulariteit" geef.)

Stelling 3 (Modulaire lijst-modellering)

Zij list de calculus voor lijsten; met voor het gemak $ts(\underline{lrec}) = (\alpha; \forall\beta.(\alpha\to\beta\to\beta)\to\beta\to\beta)$ in plaats van $ts(\underline{lrec}) = (\alpha,\beta; (\alpha\to\beta\to\beta)\to\beta\to\beta)$.

Zij $\lambda = (\Gamma, \ell, \mathbb{C}, R)$ een calculus en zij $\lambda' = (\Gamma, 2, \mathbb{C}, R)$. $\emptyset' = (\emptyset, 2, \emptyset, \emptyset)$

Er zijn $\emptyset'$-contexten $\mathcal{C}[\dots]$ en $\mathcal{D}[\cdot,\cdot,\cdot]$ die voldoen aan de volgende eigenschap:

Voor willekeurige syntactisch juiste invulling op $\cdots$ (met name zodanig dat het linkerlid in $\mathbb{M}_{\lambda'}$ zit), en willekeurige $\sigma_i, \tau_i \in \mathbb{T}_{\lambda'}$ *) en $M_i \in \mathbb{M}_{\lambda'}^{\tau_i}$ geldt:

$$\mathcal{C}[\dots\ \mathcal{D}[\sigma_i, \tau_i, \Lambda list_i.\lambda nil_i.\lambda cons_i.\lambda lrec_i.M_i] \dots]$$

$$=_{\lambda' + lists} \quad \dots\ M_i[list_i, nil_i, cons_i, lrec_i := \underline{list}(\sigma_i), \underline{nil}, \underline{cons}, \underline{lrec}]\ \dots$$

waarbij $list_i \in \mathbb{A}$ en $nil_i, cons_i, lrec_i \in \mathbb{X}$ met $type(nil_i) = list_i$, $type(cons_i) = \sigma_i \to list_i \to list_i$, $type(lrec_i) = list_i \to (\forall\beta.(\sigma_i\to\beta\to\beta)\to\beta\to\beta)$.
De subscript i geeft aan dat er verscheidene voorkomens zijn, zeg $1 \le i \le n$; de voorkomens van $\mathcal{D}[\dots]$ mogen genest zijn.

[Wanneer $\lambda' + lists$ een conservatieve uitbreiding van λ'

---

*) met $list_i \notin FV(\tau_i)$

is, levert deze gelijkheid dus dat de lijst-constructoren en ~~reductie~~ contractie-regels -onder behoud van convertibiliteit- ingeruild kunnen worden tegen zuivere calculus-termen en -typen.]] Dit kunnen we wellicht iets directer formuleren als volgt:

Er is een <u>(=)-isomorfe</u> calculustransformatie $\ldots^*$ met $(\lambda + lists)^* = \lambda'$ die voldoet aan de volgende eigenschap:

Voor willekeurige $N \in \mathbb{M}_{\lambda + lists}$ is $N^* = \mathcal{C}[\tilde{M}]$
waarbij
$\tilde{N}$ een term is die uit N ontstaat door herhaaldelijk (totdat $\tilde{N} \in \mathbb{M}_{\lambda'}$) een voorkomen i $(i \in 1..n)$ van de vorm

$$M_i[list_i, nil_i, cons_i, lrec_i := \underline{list}(\sigma_i), \underline{nil}, \underline{cons}, \underline{lrec}]$$

met
willekeurige $list_i \in \mathbb{A}$, $\sigma_i \in \mathbb{T}_{\lambda + list_i}$, $\tau_i \in \mathbb{T}_{\lambda + list_i}$
met $list_i \notin FV(\tau_i)$, $nil_i, cons_i, lrec_i \in \mathbb{X}$ met
$type(nil_i) = list_i$, $type(cons_i) = \sigma_i \to list_i \to list_i$, $type(lrec) =$
$(list_i \to \forall \beta.\, (\sigma_i \to \beta \to \beta) \to \beta \to \beta)$ en $M_i \in \mathbb{M}^{\tau_i}_{\lambda'}$
(dus $M_i[\ldots := \ldots] \in \mathbb{M}^{\tau_i[list_i := \underline{list}(\sigma_i)]}_{\lambda' + lists}$)

te vervangen door

$$\mathcal{D}[\sigma_i, \tau_i, (\Lambda list_i.\, \lambda nil_i.\, \lambda cons_i.\, \lambda lrec_i.\, M_i)]$$

NB. deze eigenschap legt $\ldots^*$ ~~nog~~ niet uniek vast.
(Einde van de formulering van Stelling 3) $\square$

<u>Bewijs</u> (van Stelling 3)

Kies

$$\mathcal{C}[\dots] = (\lambda\, genList.\ [\dots])\ (\Lambda\alpha \Lambda\beta\, \lambda p.\ p\ (\underline{list}(\alpha))^*\ \underline{nil}^*\ \underline{cons}^*\ \underline{lrec}^*)$$

$$\mathcal{D}[\sigma,\tau,P] = \quad genlist\ \sigma\ \tau\ P$$

waarbij

- genlist $\in X$ geheel nieuw is en niet voorkomt in de termen die aan de transformatie worden onderworpen (zodat genlist in $\mathcal{D}$ gebonden wordt door λgenlist in $\mathcal{C}$) (dus eigenlijk hangen $\mathcal{C}$ en $\mathcal{D}$ nog enigszins van de te transformeren term af en is de formulering van de stelling onjuist!),
- met type (genlist) =

$$(\forall\alpha.\ \forall\beta.\ (\ \forall list.\ (list \to (\alpha \to list \to list) \to (list \to \forall\beta.(\alpha\to\beta\to\beta)\to\beta\to\beta) \\ \to \beta) \\ \to \beta)$$

- en $(\underline{list}(\alpha))^*$, $\underline{nil}^*$, $\underline{cons}^*$, $\underline{lrec}^*$ zijn de transformaties van $\underline{nil} \in \mathbb{M}^{\underline{list}(\alpha)}_{\lambda+lists}$, $\underline{cons} \in \mathbb{M}^{\alpha\to\underline{list}(\alpha)\to\underline{list}(\alpha)}_{\lambda+lists}$ en $\underline{lrec} \in \mathbb{M}^{\underline{list}(\alpha)\to\forall\beta.(\alpha\to\beta\to\beta)\to\beta\to\beta}_{\lambda+lists}$ die door Stelling 2 gegeven worden.

Het is nu eenvoudig te verifiëren dat

(i) de syntactische constructies (i.e. de geformeerde termen) correct zijn, met name wat de typering betreft;

(ii) de afbeelding $\dots^*$ een calculustransformatie is; en

(iii) de beweerde convertibiteit (op pag 15) en implicaties tussen convertibiteiten (i.e. (=)-isomorfie) waar zijn.

Punt (iii) is enerzijds vrij triviaal, anderzijds (i.e. de andere implicatie) moeilijker, cf. Stelling 2. □

Fokkinga, M.M.: Programming Language Concepts – The Lambda Calculus Approach. In Essays on Concepts, Formalisms and Tools, (eds. P.R.J. Asveld and A. Nijholt), CWI Tract 42, 1987, pp 129-162.