Partiële Typetoekenning in Linkslineaire Applicatieve Termherschrijfsystemen

(Zin dat het heeft; een liber amicorum voor Jan van Bakel, pages 35-54, 1993)

Steffen van Bakel[†]

Introductie

Termherschrijfsystemen (Klop 1990) kunnen zich de laatste jaren in een toenemende belangstelling verheugen. Ze zijn niet alleen interessant voor degenen die zich bezig houden met de implementatie van functionele programmeertalen (Nöcker et al. 1991), ook binnen het onderzoek van natuurlijke talen spelen termherschrijfsystemen een rol van betekenis. Zo kan bijvoorbeeld het GRAMTSY-systeem (Coppen 1989), ontwikkeld voor de reductie van complexe lambda-formules van een Montague-achtige syntactische analyse, als een termherschrijfsysteem beschreven worden.

Typetoekenning op termherschrijfsystemen is een nog weinig verkend gebied. Voor programmeertalen gebaseerd op termherschrijfsystemen is de noodzaak voor een rigide typeringssysteem evident, al is het maar ter vermijding van run-time foutmeldingen en het verkrijgen van een efficiënte codegeneratie. Maar ook bij manipulaties van bomen die samenhangen met representaties van zinnen in een natuurlijke taal kan het toekennen van een type nuttig zijn, daar op die manier een intuïtieve semantiek kan worden gegeven en een waarborg voor het correct toepassen van de transformatieregels kan worden geformuleerd.

Typeringssystemen voor functionele programmeertalen werden tot nu toe nog geheel gemotiveerd vanuit noties van typering voor de Lambda Calculus (Barendregt 1984); de meeste van deze systemen zijn uitbreidingen van de meest primitieve, goed begrepen en duidelijk gedefinieerde notie van typetoekenning op lambdatermen, die bekend staat als het Curry typetoekenningssysteem (Curry & Feys 1958).

Bijvoorbeeld het typetoekenningssysteem voor de functionele programmeertaal ML (Milner 1978), zoals gedefinieerd door R. Milner is in feite een uitbreiding van Curry's systeem. (Het typetoekenningsalgorithme voor de functionele programmeertaal Miranda (Turner 1985) werkt ruw gezegd op dezelfde manier als dat voor ML. Een echt verschil tussen deze talen ligt in het feit dat Miranda ook een typecontrolealgorithme bevat, welke is gebaseerd op het typetoekenningssysteem voor een uitgebreide Lambda Calculus zoals gedefinieerd door A. Mycroft, een uitbreiding van Milners typetoekenningssysteem (Mycroft 1984, Kfoury et al. 1988).)

Om een formeel typesysteem voor alle talen die patroonherkenning gebruiken te leveren en voor het semantisch onderbouwen van transformatiesystemen, presenteert dit artikel een formele notie van typetoekenning op linkslineaire applicatieve termherschrijfsystemen. Zoals aangetoond in dit artikel voldoet typetoekenning in termherschrijfsystemen in het algemeen niet aan de subject-reductie eigenschap: het kan voorkomen dat een term (boom) die door middel van een herschrijfregel mag worden omgezet naar een andere term typeerbaar is,

[†] Ondersteund door de Nederlandse Organisatie voor Wetenschappelijk Onderzoek (N.W.O.).

terwijl het resultaat van de omzetting dat niet is, i.e. types zijn niet behouden onder herschrijving. Het belangrijkste resultaat van dit artikel, de formulering van een conditie op herschrijfregels die noodzakelijk en voldoende is voor het verkrijgen van subject-reductie, kan gebruikt worden om te bewijzen dat alle herschrijfregels die gedefinieerd kunnen worden in een functionele programeertaal zoals Miranda, veilig zijn vanuit die optiek. Ook als men op het eerste gezicht zou kunnen denken dat de generalisatie van de manier van typetoekenning zoals gebruikt in functionele programmeertalen naar een typetoekenningsysteem voor termherschrijfsystemen rechtstreeks zou zijn, is dit niet het geval, zoals volgt uit de condities die ter garantie van het behoud van types onder herschrijving gegeven zijn.

Het typetoekenningssysteem dat gepresenteert wordt in dit artikel is een partieel systeem in de zin van Pfenning (1988), omdat niet alleen gedefinieerd wordt hoe termen en herschrijfregels getypeerd kunnen worden, maar ook een type geleverd wordt voor elk functiesymbool. Daar zijn verscheidene redenen voor.

Voor symbolen die worden omschreven door een herschrijfregel (zulke symbolen heten 'gedefinieerde symbolen'), en voor symbolen waarvoor zo'n regel niet bestaat (constanten), moet een manier bestaan om te bepalen welk type gebruikt kan worden voor een voorkomen van zo'n symbool in een willekeurige context. Normaal levert voor gedefinieerde symbolen de structuur van de herschrijfregel de nodige informatie voor het bepalen van het type van zo'n symbool. In plaats van voor gedefinieerde symbolen het bij elk voorkomen bestuderen van hun regel ter verkrijging van het type, kan het type van het symbool in een afbeelding van symbolen naar types gebracht worden, en deze afbeelding voor de productie van het type gebruikt worden. Natuurlijk maakt het geen verschil het bestaan van zo'n afbeelding van symbolen naar types van het begin af aan aan te nemen, en typetoekenning te definiëren gebruikmakend van die afbeelding. Deze afbeelding is dan eveneens geschikt om te verzekeren dat types toegekend aan verschillende voorkomens van constanten niet onderling in conflict zijn.

In feite is de hier gevolgde aanpak sterk vergelijkbaar met die, welke genomen wordt door Hindley in Hindley (1969), waar hij het voornaamste typeschema van een object in Combinatorische Logica definieert. Zelfs zijn notie van typetoekenning kan beschouwd worden als een partiële. Aangezien systemen van combinatoren gemakkelijk vertaald kunnen worden naar linkslineaire applicatieve termherschrijfsystemen, zijn de resultaten van dit artikel, wanneer de toegestane herschrijfregels beperkt worden tot dewelke die corresponderen met combinatoren, hetzelfde als in Hindley (1969).

Een gedeelte van de resultaten zoals besproken in dit artikel werden voor het eerst gepresenteerd in Van Bakel et al. (1992). Lezers die geïnteresseerd zijn in een meer volledige bespreking (met aanvullende voorbeelden en een bespreking van een implementatie), worden naar dat artikel verwezen.

1 Linkslineaire applicatieve termherschrijfsystemen

In dit artikel worden linkslineaire applicatieve termherschrijfsystemen (LAT) bestudeerd, welke een subklasse vormen van de termherschrijfsystemen zoals gedefinieerd in Klop (1990). LAT's zijn gedefinieerd als de klasse van termherschrijfsystemen die een speciale binaire operator Ap (kunnen) bevatten, en in welke alle herschrijfregels linkslineair zijn. Om ze te kunnen onderscheiden van de termherschrijfsystemen die *alleen* het functiesymbool Ap bevatten, wordt de tweede categorie de *puur applicatieve termherschrijfsystemen* genoemd.

De motivatie voor het gebruik van applicatieve termherschrijfsystemen in plaats van de algemene termherschrijfsystemen kan worden geïllustreerd door het volgende voorbeeld: Als

Combinatorische Logica (CL)

$$\begin{aligned} S\ x\ y\ z &= x\ z\ (y\ z) \\ K\ x\ y &= x \\ I\ x &= x \end{aligned}$$

naar een termherschrijfsysteem vertaald moet worden, dan kan het eruit zien als (waarbij de impliciete applicatie van CL expliciet gemaakt is):

$$\begin{aligned} Ap(Ap(Ap(S,x),y),z) &\Rightarrow Ap(Ap(x,z),Ap(y,z)) \\ Ap(Ap(K,x),y) &\Rightarrow x \\ Ap(I,x) &\Rightarrow x \end{aligned}$$

Echter, de symbolen S, K en I kunnen ook als functies (met respectievelijk 3, 2 en 1 operanden) gezien worden. Als geprobeerd wordt die blik te vangen in de vertaling, dan moeten (om een stelsel met voldoende uitdrukkingskracht te krijgen) ook de ge-Curry-de versies van die symbolen gedefinieerd worden:

$$\begin{aligned} S(x,y,z) &\Rightarrow Ap(Ap(x,z),Ap(y,z)) \\ Ap(S_2(x,y),z) &\Rightarrow S(x,y,z) \\ Ap(S_1(x),y) &\Rightarrow S_2(x,y) \\ Ap(S_0,x) &\Rightarrow S_1(x) \\ K(x,y) &\Rightarrow x \\ Ap(K_1(x),y) &\Rightarrow K(x,y) \\ Ap(K_0,x) &\Rightarrow K_1(x) \\ I(x) &\Rightarrow x \\ Ap(I_0,x) &\Rightarrow I(x) \end{aligned}$$

In dit artikel worden de applicatieve herschrijfsystemen beschouwd, omdat deze meer algemeen zijn dan de subklasse van systemen waarin alleen het functiesymbool Ap bestaat. Omdat de linkslineaire puur applicatieve termherschrijfsystemen een subklasse vormen van de linkslineaire applicatieve termherschrijfsystemen, zijn alle resultaten verkregen in dit artikel ook van toepassing op die subklasse.

Dit artikel gaat ervan uit dat in een herschrijfregel een zeker symbool gedefinieerd wordt, en het is duidelijk dat de regels toegevoegd ter verkrijging van de ge-Curry-de versies van symbolen in de vertaling van CL naar een herschrijfsysteem niet bedoeld zijn als definities voor Ap, welk min of meer een 'voorgedefinieerd symbool' is, maar als definities voor de ge-Curry-de versies.

Echter, in het algemeen zijn termherschrijfsystemen niet gevoelig voor de namen die gebruikt zijn voor functiesymbolen. Dus het herschrijfsysteem zoals hierboven gegeven is in feite hetzelfde als hetgeen verkregen wordt na vervanging van alle Ap's door F, en dan kunnen alle herschrijfregels die beginnen met F gezien worden als regels die F definiëren. Om dit probleem te vermijden, worden die herschrijfsystemen beschouwd die een een 'voorgedefinieerd' binair functiesymbool hebben, Ap genaamd, welke dan niet hernoemd kan worden. Het symbool Ap wordt genegeerd wanneer gezocht wordt naar het symbool dat is gedefinieerd in een herschrijfregel.

In de Lambda Calculus bestaat een duidelijk verschil tussen vrije en gebonden variabelen van een term. In termherschrijfsystemen kan een termvariabele x die in de linkerkant van een herschrijfregel voorkomt, worden gezien als het bindende voorkomen van x, dat de voorkomens van x in de rechterkant bindt. Echter, in het algemeen kan x meerdere malen in de linkerkant voorkomen, waarbij de notie van *het* bindende voorkomen obscuur wordt. In dit

artikel worden linkslineaire herschrijfsystemen beschouwd, welke alleen herschrijfregels bevatten waarvoor de linkerkant lineair is (termvariabelen komen slechts één maal voor), omdat voor zulke regels het bindende voorkomen van een termvariabele uniek is.

De volgende definities zijn gebaseerd op definities zoals gegeven in Klop (1990). Definitie 1.1 definieert LAT's op de zelfde manier als de definitie gegeven door Klop voor termherschrijfsystemen, uitgebreid met deel (*i.c*) dat het bestaan van het voorgedefinieerde symbool Ap uitdrukt. Definitie 1.2 definieert een notie van herschrijving op LAT's op de zelfde manier als de definitie van herschrijving gegeven door Klop voor termherschrijfsystemen, uitgebreid met deel (*ii.a.2*) dat de linkslineairiteit van herschrijfregels uitdrukt, deel (*ii.a.4*) dat aangeeft dat het mogelijke gebruik van het symbool Ap in de linkerkant beperkt is, en deel (*iii*) voor de definitie van de notie van gedefinieerd symbool van een herschrijfregel. In feite zijn de delen (*ii.a.4*) en (*iii*) verbonden.

In sommige delen wordt wat betreft de ingevoerde notatie afgeweken van de door Klop gevolgde lijn, omdat enkele van de symbolen of definities zoals gegeven door Klop ook worden gebruikt in artikelen over typetoekenning, maar met een andere betekenis. Het woord 'vervanging' wordt bijvoorbeeld gebruikt voor de operatie die termvariabelen door termen vervangt, in plaats van het woord 'substitutie', dat gebruikt zal worden voor operaties die typevariabelen door types vervangen.

Substitutie en vervanging zijn ook operaties die gedefinieerd zijn in Curry & Feys (1958). Beide operaties worden daar gedefinieerd als operaties op termen; substitutie is gedefinieerd als de operatie die termvariabelen door termen vervangt, en vervanging is gedefinieerd als de operatie die subtermen door termen vervangt. Merk op dat de hier gebezigde definitie dus ook verschilt van die in Curry & Feys (1958).

Ter denotatie van een vervanging zullen hoofdletters zoals 'R' gebruikt worden, in plaats van griekse symbolen zoals 'σ', welke voor de denotatie van types gebruikt zijn. Het symbool '$\Rightarrow$' wordt gebruikt voor het herschrijvingssymbool, in plaats van '$\rightarrow$' welke wordt gebruikt als een typeconstructor. De notie 'constant symbool' wordt gebruikt voor een symbool dat niet herschreven kan worden, in plaats van voor een functie symbool met ariteit 0.

Definition 1.1 Een *Linkslineair Applicatieve Termherschrijfsysteem* (LAT) is een paar (Σ, **R**) bestaande uit een *alfabet* of *signatuur* Σ en een verzameling van *herschrijfregels* **R**.

i) Het alfabet Σ bestaat uit:
  a) Een aftelbaar oneindige verzameling van termvariabelen x, y, z, x', y', ...
  b) Een niet lege verzameling Σ_0 van *functie symbolen* F, G, ..., elk toegerust met een 'ariteit' (een natuurlijk getal), i.e. het aantal 'argumenten' dat het symbool wordt verondersteld te hebben.
  c) Een speciale binaire operator, *applicatie* genaamd (Ap).

ii) De verzameling van *termen* 'over' Σ is $\mathsf{Ter}(\Sigma)$ en wordt inductief gedefinieerd door:
  a) $x, y, z, \ldots \in \mathsf{Ter}(\Sigma)$.
  b) Als $F \in \Sigma_0 \cup \{Ap\}$ een n-air symbool is, en $T_1, \ldots, T_n \in \mathsf{Ter}(\Sigma)$ ($n \geq 0$), dan $F(T_1, \ldots, T_n) \in \mathsf{Ter}(\Sigma)$.
  De T_i ($i = 1, \ldots, n$) zijn de *argumenten* van deze term.

iii) Termen waarin geen variabele twee of meer keren voorkomt, heten *lineair*.

Definition 1.2 Laat (Σ, **R**) een LAT zijn.

i) Een *vervanging* R is een afbeelding van $\mathsf{Ter}(\Sigma)$ naar $\mathsf{Ter}(\Sigma)$ die voldoet aan $\mathsf{R}(F(T_1, \ldots, T_n))) = F(\mathsf{R}(T_1), \ldots, \mathsf{R}(T_n))$voor elk n-air functiesymbool F (waar $n \geq 0$). Dus, R is bepaald door zijn beperking tot de verzameling van termvariabelen. Ook wordt T^{R} geschreven in

plaats van $\mathsf{R}(T)$.

ii) *a*) Een *herschrijfregel* $\in \mathbf{R}$ is een paar (*Lhs*, *Rhs*) van termen $\in \mathsf{Ter}(\Sigma)$. Vaak krijgt een herschrijfregel een naam, bijv. **r**, en wordt $\mathbf{r} : Lhs \Rightarrow Rhs$ geschreven. Vier condities worden opgelegd:
   1) *Lhs* is niet een variabele.
   2) *Lhs* is lineair.
   3) De variabelen die voorkomen in *Rhs* zijn bevat in *Lhs*.
   4) Voor elke Ap in *Lhs* is het linker argument niet een variabele.

b) Een herschrijfregel $\mathbf{r} : Lhs \Rightarrow Rhs$ bepaalt een verzameling van *herschrijvingen* $Lhs^{\mathsf{R}} \Rightarrow Rhs^{\mathsf{R}}$ voor alle vervangingen R. De linkerkant Lhs^{R} wordt een *redex* genoemd; deze kan worden vervangen door zijn '*contractum*' Rhs^{R} binnen een context C[ ]; dit geeft aanleiding tot *herschrijfstappen*: C[Lhs^{R}] $\Rightarrow_{\mathbf{r}}$ C[Rhs^{R}].

c) De relatie $\Rightarrow_{\mathbf{r}}$ wordt de *een-staps herschrijfrelatie* gegenereerd door **r** genoemd. Herschrijfstappen concatenerend worden (mogelijk oneindige) *herschrijfreeksen* $T_0 \Rightarrow T_1 \Rightarrow T_2 \Rightarrow \cdots$ verkregen. Als $T_0 \Rightarrow \cdots \Rightarrow T_n$, dan wordt ook $T_0 \Rightarrow\!\!\!\!\Rightarrow T_n$ geschreven, en T_n is een *herschrijving* van T_0.

iii) *a*) In een herschrijfregel, is het uiterst linkse, meest buitenste symbool in de linkerkant dat niet een Ap is, het *gedefinieerde symbool* van die regel.

b) Als het symbool F het gedefinieerd symbool van **r** is, dan *definieert* **r** F.

c) F is *een gedefinieerd symbool*, als er een herschrijfregel is die F definieert.

d) $Q \in \Sigma_0$ is een *constant symbool* als Q niet een gedefinieerd symbool is.

Deel (*ii.a.4*) van definitie 1.2 is toegevoegd om herschrijfregels met linkerkanten zoals $Ap(x,y)$ te vermijden, omdat zulke regels geen gedefinieerd symbool zouden hebben.

Proposition 1.3 Laat F het gedefinieerde symbool van de herschrijfregel $\mathbf{r} : Lhs \Rightarrow Rhs$ zijn. Dan zijn er $n \geq j \geq 0$, en $T_1, \ldots, T_n$ zodat:

$$Lhs = Ap(Ap(\ldots Ap(F(T_1,\ldots,T_j),T_{j+1}),\ldots),T_n).$$

en $T_1, \ldots, T_n$ worden de patronen *van* **r** *genoemd.*

In dit artikel worden herschrijfsystemen beschouwd die *Curry-gesloten* zijn, i.e. voor elke herschrijfregel die het symbool F met ariteit $n \geq 0$ definieert, wordt aangenomen dat er n additionele herschrijfregels zijn die de functie symbolen F_0 tot en met F_{n-1} als volgt definiëren:

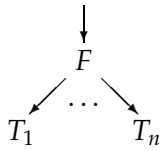
$$\begin{array}{lcl} Ap(F_{n-1}(x_1,\ldots,x_{n-1}),x_n) & \Rightarrow & F(x_1,\ldots,x_n) \\ Ap(F_{n-2}(x_1,\ldots,x_{n-2}),x_{n-1}) & \Rightarrow & F_{n-1}(x_1,\ldots,x_{n-1}) \\ \quad\vdots & & \\ Ap(F_0,x_1) & \Rightarrow & F_1(x_1) \end{array}$$

De toegevoegde regels met $F_{n-1}, \ldots, F_1, F_0$, etc., geven in feite de 'ge-Curry-de'-versies van F.

Het zou voldoende zijn geweest een afsluitoperatie te definiëren op LAT's, door toevoeging van herschrijfregels en uitbreiding van het alfabet Σ, maar het is eenvoudiger aan te nemen dat elke LAT Curry-gesloten is. Echter, wanneer een herschrijfsysteem gepresenteerd wordt, zullen alleen die regels getoond worden die essentieel zijn, en niet de regels die de 'ge-Curry-de'-versies definiëren.

2 Boomrepresentatie van termen en herschrijfregels

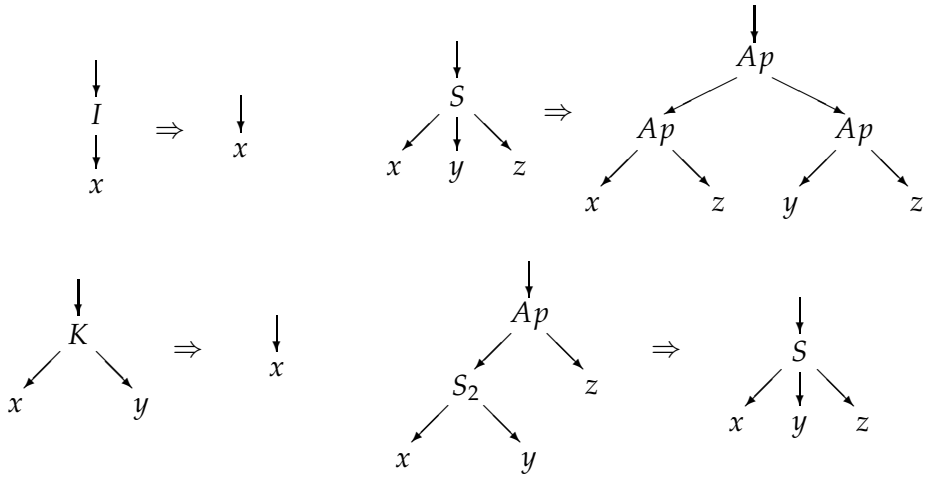
Definition 2.1 *i*) De boomrepresentatie van termen en herschrijfregels wordt verkregen op een rechtstreekse manier, door het representeren van een term $F(T_1, \ldots, T_n)$ door:



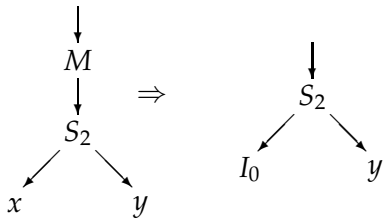
ii) De *ruggegraat* van een boom is zoals gebruikelijk gedefinieerd, i.e.: de wortelknoop van de boom is op de ruggegraat, en als een knoop op de ruggegraat is, dan is zijn meest linker onderdaan op de ruggegraat.

iii) In de boomrepresentatie van een herschrijfregel wordt de eerste knoop op de ruggegraat van de linkerkant, beginnende bij de wortelknoop, die geen Ap bevat, de *definiërende knoop* van die regel genoemd. (Merk op dat als F het gedefinieerde symbool van een herschrijfregel is, dan verschijnt het in de definiërende knoop van de boomrepresentatie van die regel.)

Enkele van de herschrijfregels van CL in boomrepresentatie:



Herschrijfregels kunnen natuurlijk meer complex zijn dan boven is geïllustreerd door de regels voor CL. In het algemeen, als de linkerkant van een herschrijfregel $F(T_1, \ldots, T_n)$ is, dan zijn de termen T_i niet noodzakelijk simpele variabelen maar kunnen gecompliceerde termen zijn, zoals bij voorbeeld in de herschrijfregel $M(S_2(x,y)) \Rightarrow S_2(I_0,y)$. In boomrepresentatie ziet deze regel er als volgt uit:



3 Typetoekenning in LAT's

In deze sectie wordt een notie van partiële typetoekenning op LAT's gepresenteerd, die gebaseerd is op het Milner typetoekenningssysteem (Milner 1978, Damas 1985). Het toekennen van types aan een LAT zal bestaan uit de etikettering van knopen en pijlen in de boomrepresentatie van termen en herschrijfregels met type-informatie. Types zijn toegekend aan knopen om de notie van 'type van een functie', 'type van een constante' of 'type van een variable' te vangen, en zijn toegekend aan pijlen om de notie van 'type van een subterm' (of boom) te vangen. De pijl die wijst naar de wortel van een term wordt de *wortelpijl* genoemd.

Er is een belangrijk verschil tussen de notie van typetoekenning zoals geïntroduceerd in dit artikel en Curry's typetoekenningssysteem. In dat systeem wordt een basis normaal gedefinieerd als een afbeelding van termvariabelen naar types, of, equivalent, als een verzameling van beweringen met verschillende termvariabelen als subjects. Echter, de bases die zijn toegestaan in het systeem zoals hier gepresenteerd kunnen een aantal verschillende beweringen voor dezelfde termvariabele bevatten. Dit komt overeen met de definitie van ML-typetoekenning als gegeven in Damas (1985), en wordt daar gebruikt voor de behandeling van de let-constructie. Het kan ook vergeleken worden met het gebruik van intersectietypes (Barendregt et al. 1983, Van Bakel 1992) op vrije variabelen van termen. In tegenstelling tot in de Lambda Calculus veroorzaakt dit geen problemen in termherschrijfsystemen, omdat er geen notie van 'abstractie' in deze wereld is. Daarbij, conditie (*i.b*) van definitie 3.5 vookomt deze ogenschijnlijke anomalie. Dus in het systeem zoals hier gepresenteerd is het mogelijk een type aan de term $Ap(x,x)$ toe te kennen.

3.1 Types

Het typesysteem zoals gedefinieerd in deze subsectie is gebaseerd op het Curry typesysteem, uitgebreid met typeconstanten.

Definition 3.1 *i*) $\mathcal{T}_C$, de verzameling van Curry types is inductief gedefinieerd door:
- *a*) Alle typevariabelen $\varphi_0, \varphi_1, \ldots \in \mathcal{T}_C$.
- *b*) Alle typeconstanten $c_0, c_1, \ldots \in \mathcal{T}_C$.
- *c*) Als $\sigma, \tau \in \mathcal{T}_C$, dan $\sigma \to \tau \in \mathcal{T}_C$.

ii) Een *bewering* is een expressie van de vorm $M{:}\sigma$, waar $M \in \mathsf{Ter}(\Sigma)$ en $\sigma \in \mathcal{T}_C$. M is het *onderwerp* en σ het *predikaat* van $M{:}\sigma$.

iii) Een *basis B* is een verzameling van beweringen met termvariabelen, niet noodzakelijk verschillend, als onderwerp. Als een basis beschreven is als $\{x_1{:}\rho_1, \ldots, x_n{:}\rho_n\}$, dan zijn de x_i verschillend.

In de notatie van types worden vaak de buitenste en de meest rechtse haakjes weggelaten. Het symbool φ zal gebruikt worden voor de notatie van een typevariabele en alle andere griekse tekens voor de notatie van arbitraire types. Ook zal vaak in plaats van φ geïndexeerd met een getal, het getal geschreven worden.

Definition 3.2 *i*) Een *substitutie* $S : \mathcal{T}_C \to \mathcal{T}_C$ is zoals gewoonlijk inductief gedefinieerd door:
- *a*) De substitutie $(\varphi := \alpha)$, waar φ een typevariabele is en $\alpha \in \mathcal{T}_C$, is gedefinieerd door:
  - *1*) $(\varphi := \alpha)(\varphi) = \rho$.
  - *2*) $(\varphi := \alpha)(\varphi_0) = \varphi_0$, als $\varphi \neq \varphi_0$.
  - *3*) $(\varphi := \alpha)(c_i) = c_i$.

4) $(\varphi := \alpha)(\sigma \to \tau) = (\varphi := \alpha)(\sigma) \to (\varphi := \alpha)(\tau)$.

b) Als S_1 en S_2 substituties zijn, dan is $S_1 \circ S_2$ eveneens een substitutie, waar $S_1 \circ S_2(\sigma) = S_1(S_2(\sigma))$.

ii) Als voor σ, τ er een substitutie S is zodat $S(\sigma) = \tau$, dan heet τ een *(substitutie) instantie* van σ.

iii) Als σ een instantie van τ is, en τ is een instantie van σ, dan heet σ een *triviale variant* van τ. Types die triviale varianten van elkaar zijn worden geïdentificeerd.

iv) $S(B) = \{x{:}S(\rho) \mid x{:}\rho \in B\}$.

v) $S(<B, \sigma>) = <S(B), S(\sigma)>$.

3.2 Typetoekenning

Typetoekenning op een LAT $(\Sigma, \mathbf{R})$ is gedefinieerd als de etikettering van knopen en pijlen in de boomrepresentatie van termen en herschrijfregels met types. Als een knoop of pijl is geëtiketteerd met een type σ, is deze *getypeerd* met σ, en is σ er aan *toegekend*.

In deze etikettering wordt gebruikt dat er een afbeelding is die een type in $\mathcal{T}_C$ levert voor elke $F \in \Sigma_0 \cup \{Ap\}$. Een dergelijke afbeelding heet een *omgeving*.

Definition 3.3 Laat $(\Sigma, \mathbf{R})$ een LAT zijn.

i) Een afbeelding $\mathcal{E} : \Sigma_0 \cup \{Ap\} \to \mathcal{T}_C$ heet een *omgeving* als $\mathcal{E}(Ap) = (1 \to 2) \to 1 \to 2$, en voor elke $F \in \Sigma_0$ met ariteit n geldt: $\mathcal{E}(F) = \mathcal{E}(F_{n-1}) = \cdots = \mathcal{E}(F_0)$.

ii) Voor $F \in \Sigma_0$ met ariteit $n \geq 0$, $\sigma \in \mathcal{T}_C$, en omgeving $\mathcal{E}$, wordt de omgeving $\mathcal{E}[F := \sigma]$ gedefinieerd door:

a) $\mathcal{E}[F := \sigma](G) = \sigma$, als $G \in \{F, F_{n-1}, \ldots, F_0\}$.

b) $\mathcal{E}[F := \sigma](G) = \mathcal{E}(G)$, in alle andere gevallen.

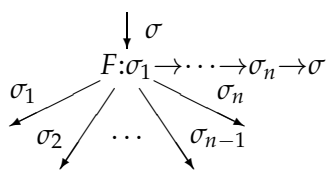
De conditie dat $\mathcal{E}(F) = \mathcal{E}(F_{n-1}) = \cdots = \mathcal{E}(F_0)$ is niet essentieel, maar wordt slecht ingevoerd, om redenen van efficiëncy; we willen aan alle ge-Curry-de verianten van een functiesymbol hetzelfde type toekennen. Deze beperking kan zonder verlies van resultaten verlaten worden.

Definition 3.4 Laat $(\Sigma, \mathbf{R})$ een LAT zijn.

i) $M \in \mathsf{Ter}(\Sigma)$ *is typeerbaar door* $\sigma \in \mathcal{T}_C$ *met betrekking tot* $\mathcal{E}$, als er een toekenning van types aan pijlen en knopen bestaat die voldoet aan de volgende beperkingen:

a) De wortelpijl van M is getypeerd met σ.

b) Als een knoop een symbool $F \in \Sigma_0 \cup \{Ap\}$ bevat met ariteit n $(n \geq 0)$, dan zijn er σ_1, $\ldots$, σ_n en σ, zodat deze knoop getypeerd is met $\sigma_1 \to \cdots \to \sigma_n \to \sigma$, de n uitgaande pijlen zijn van links naar rechts getypeerd met σ_1 tot en met σ_n, en de inkomende pijl is getypeerd met σ.



c) Als een knoop die een symbool $F \in \Sigma_0 \cup \{Ap\}$ bevat, getypeerd is met σ, dan er is een substitutie S zodat $S(\mathcal{E}(F)) = \sigma$.

ii) Laat $M \in \mathsf{Ter}(\Sigma)$ typeerbaar zijn door σ met betrekking tot $\mathcal{E}$. In het geval B een basis is die alle beweringen met variabelen als onderwerp bevat die verschijnen in de getypeerde boom voor $M{:}\sigma$, wordt $B \vdash_{\mathcal{E}} M{:}\sigma$ geschreven.

Merk op dat als $B \vdash_{\mathcal{E}} M{:}\sigma$, dan kan B meer beweringen bevatten dan nodig om $M{:}\sigma$ te verkrijgen.

Definition 3.5 Laat $(\Sigma, \mathbf{R})$ een LAT zijn.

i) $\mathbf{r} : \mathit{Lhs} \Rightarrow \mathit{Rhs} \in \mathbf{R}$ met gedefinieerd symbool F *is naïef typeerbaar met betrekking tot* $\mathcal{E}$, als aan de volgende beperkingen voldaan is:
   a) Er zijn $\sigma \in \mathcal{T}_C$ en basis B zodat $B \vdash_{\mathcal{E}} \mathit{Lhs}{:}\sigma$ en $B \vdash_{\mathcal{E}} \mathit{Rhs}{:}\sigma$.
   b) Alle knopen in $\mathbf{r}$ die dezelfde termvariabele x bevatten, zijn getypeerd met hetzelfde type.
   c) In $B \vdash_{\mathcal{E}} \mathit{Lhs}{:}\sigma$ en $B \vdash_{\mathcal{E}} \mathit{Rhs}{:}\sigma$, zijn alle knopen die F bevatten, getypeerd met $\mathcal{E}(F)$.

ii) $(\Sigma, \mathbf{R})$ *is naïef typeerbaar met betrekking tot* $\mathcal{E}$, als voor elke $\mathbf{r} \in \mathbf{R}$: $\mathbf{r}$ is naïef typeerbaar met betrekking tot $\mathcal{E}$.

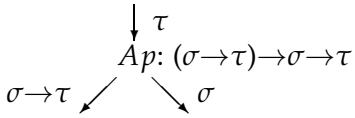
Conditie (*i.c*) is in feite toegevoegd om zeker te zijn dat het type dat geleverd wordt door de omgeving voor een functie symbool F niet in conflict is met de herschrijfregels die F definiëren. Door het type dat kan worden toegekend aan de definiërende knoop te beperken tot het type zoals geleverd door de omgeving, is zeker dat de herschrijfregel getypeerd wordt gebruik makend van dat type, en niet gebruik makend van een substitutieïnstantie.

In de rest van dit artikel zal ervan worden uitgegaan dat de omgeving vastligt, en wordt de subscript op $\vdash_{\mathcal{E}}$ weggelaten. Ook zal, in plaats van de frase '(naïef) typeerbaar met betrekking tot $\mathcal{E}$', gewoon 'typeerbaar' gebruikt worden.

Het is eenvoudig te controleren dat als F een functiesymbool is met ariteit n, en alle herschrijfregels die F definiëren typeerbaar zijn, er dan $\gamma_1, \ldots, \gamma_n, \gamma$ zijn zodat $\mathcal{E}(F) = \gamma_1 \to \cdots \to \gamma_n \to \gamma$.

Het gebruik van een omgeving correspondeert tot het gebruik van "axioma-schema's", en deel (*i.c*) van definitie 3.4 tot het gebruik van "axioma's" als in Hindley (1969).

Een typisch voorbeeld voor deel (*i.b*) van definitie 3.4 is het functiesymbool Ap, welke het type $(1\to2)\to1\to2$ heeft. Dus voor elke voorkomen van Ap in een boom zijn er σ en τ zodat het volgende deel is van die boom.



3.3 Het voornaamste paar voor een term

In deze subsectie wordt het voornaamste paar voor een typeerbare term M met betrekking tot $\mathcal{E}$ gedefinieerd, bestaande uit basis B en type σ, door het definiëren van de notie $pp(M)$ '*p*rincipal *p*air', gebruik makend van Robinsons unificatiealgorithme unify (Robinson 1965). (Merk op dat, van een formeel standpunt, de notie $pp_{\mathcal{E}}(M)$ gedefiniëerd zou moeten worden, maar dat opnieuw het subscript $\mathcal{E}$ wordt weggelaten.) Hieronder wordt aangetoond dat, voor elke typeerbare term, dit een legaal paar is en inderdaad het meest algemene.

We herinneren aan de volgende bekende eigenschap van unify.

Property 3.6 (ROBINSON 1965) *Als twee types een gemeenschappelijke instantie hebben, dan hebben ze een hoogste gemeenschappelijke instantie welke door* unify *geleverd wordt, dus voor alle* σ, τ: *als* S_1 = unify (σ, τ) *en* S_2 *is een substitutie zodat* $S_2(\sigma) = S_2(\tau)$ *dan er is een substitutie* S_3 *zodat* $S_2(\sigma) = S_3 \circ S_1(\sigma) = S_3 \circ S_1(\tau) = S_2(\tau)$.

Definition 3.7 Voor elke term M wordt de notie $pp(M) = <P, \pi>$ inductief gedefinieerd door:

i) Voor alle x, φ: $pp(x) = <\{x{:}\varphi\}, \varphi>$.

ii) Als voor elke $1 \leq i \leq n$: $pp(T_i) = <P_i, \pi_i>$, $\mathcal{E}(F) = \sigma_1 \to \cdots \to \sigma_n \to \sigma$, (kies indien noodzakelijk triviale varianten zodat de $<P_i, \pi_i>$ paarsgewijs disjunct zijn en deze paren geen typevariabelen met $\sigma_1 \to \cdots \to \sigma_n \to \sigma$ gemeenschappelijk hebben), en

$$S = \text{unify}\,(\sigma_1 \to \cdots \to \sigma_n \to \sigma,\ \pi_1 \to \cdots \to \pi_n \to \varphi),$$

waar φ niet voorkomt in een van de paren $<P_i, \pi_i>$, noch in $\sigma_1 \to \cdots \to \sigma_n \to \sigma$, dan: $pp(F(T_1, \ldots, T_n)) = <S(P_1 \cup \cdots \cup P_n), S(\sigma)>$.

Voor deze noties gelden de volgende eigenschappen:

Theorem 3.8 Betrouwbaarheid van substitutie.

i) *Als* $B \vdash M{:}\sigma$, *dan voor elke substitutie* S: $S(B) \vdash M{:}S(\sigma)$.

ii) *Laat* **r**: *Lhs* $\Rightarrow$ *Rhs een herschrijfregel zijn, typeerbaar met betrekking tot de omgeving* $\mathcal{E}$, *en laat* F *het gedefinieerde symbool van* **r** *zijn. Dan is* **r** *typeerbaar met betrekking tot* $\mathcal{E}[F := S(\mathcal{E}(F))]$.

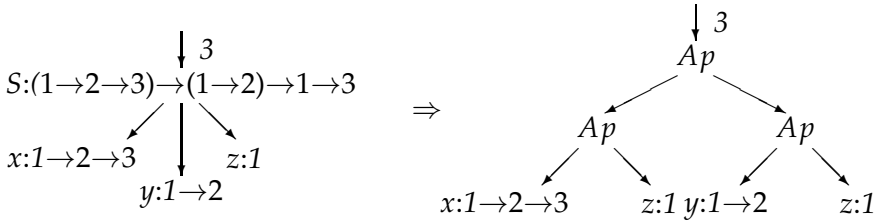
Het is eenvoudig na te gaan dat $pp(M) = <P, \pi>$ impliceert dat $P \vdash M{:}\pi$.

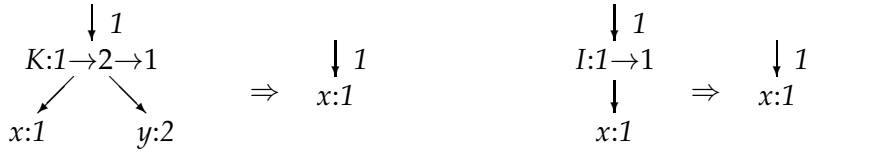
Theorem 3.9 Volledigheid van substitutie. *Als* $B \vdash M{:}\sigma$, *dan er zijn* P, π, *en een substitutie* S *zodat:* $pp(M) = <P, \pi>$, *en* $S(P) \subseteq B$, $S(\pi) = \sigma$.

3.4 Voorbeelden

Getypeerde varianten van sommige van de herschrijfregels voor CL. Ingevoegd zijn alleen die types die niet onmiddellijk duidelijk zijn. Merk op dat is aangenomen dat

$\mathcal{E}(S) = (1 \to 2 \to 3) \to (1 \to 2) \to 1 \to 3$,
$\mathcal{E}(K) = 1 \to 2 \to 1$, en
$\mathcal{E}(I) = 1 \to 1$.

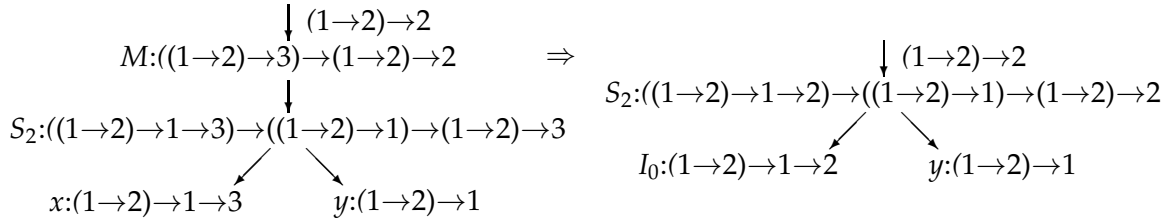




Gebruik makend van

$\mathcal{E}(M) = (((1\to2)\to3)\to(1\to2)\to2,$
$\mathcal{E}(S) = (1\to2\to3)\to(1\to2)\to1\to3,$ en
$\mathcal{E}(I) = 1\to1,$

kan de regel voor M kan als volgt getypeerd worden:

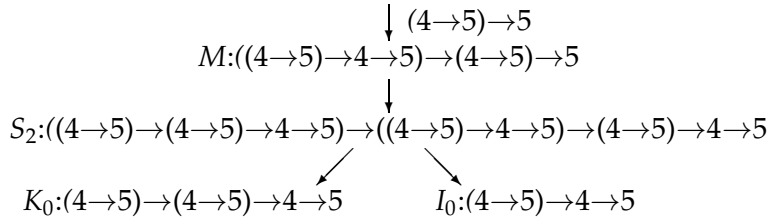


3.5 Patronen veroorzaken problemen

Definities 3.3, 3.4 en 3.5 definiëren wat een typetoekenning zou moeten zijn, gebruik makend van de strategie zoals gebruikt in functionele programmeertalen zoals bijvoorbeeld Miranda. Het wordt *naïef* genoemd, omdat het niet voldoende is om behoud van types na herschrijving te garanderen. (Dit wordt ook wel de 'subject-reductie eigenschap' genoemd.) Zelfs typeerbaarheid blijft niet behouden na herschrijving.

Neem bijvoorbeeld de definitie van M, welke getypeerd kan worden zoals hierboven gegeven. Het is eenvoudig in te zien dat deze herschrijfregel mag worden toegepast op de term $M(S_2(K_0, I_0))$, en dat deze term zal worden herschreven tot $S_2(I_0, I_0)$.

Ook al is de eerste term typeerbaar op de volgende manier:



de term $S_2(I_0, I_0)$ is niet typeerbaar met het type $(4\to5)\to5$. Deze term is zelfs in het geheel niet typeerbaar.

3.6 Een noodzakelijke en voldoende conditie voor behoud van types onder herschrijving

Uit definitie 1.2 volgt dat als een term M wordt herschreven tot de term M' gebruik makend van de herschrijfregel $Lhs \Rightarrow Rhs$, er een subterm M_0 van M is, en een vervanging R, zodat $Lhs^{\mathsf{R}} = M_0$. M' is verkregen door vervanging van M_0 door Rhs^{R}. Om de subject-reductie eigenschap te garanderen, zouden alleen die herschrijfregels $Lhs \Rightarrow Rhs$ geaccepteerd moeten worden die voldoen aan:

> Voor alle vervangingen R, bases B en types σ: als $B \vdash Lhs^{\mathsf{R}}{:}\sigma$, dan $B \vdash Rhs^{\mathsf{R}}{:}\sigma$.

omdat dan zeker is dat alle mogelijke herschrijvingen veilig zijn.

In de notie van typetoekenning zoals gepresenteerd in dit artikel, is het eenvoudig een conditie te formuleren waaraan herschrijfregels moeten voldoen om geaccepteerd te worden.

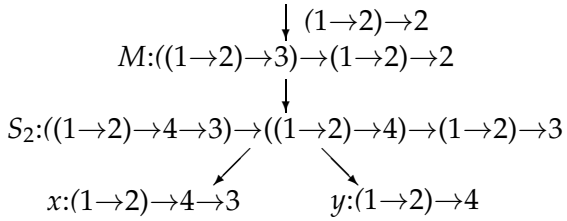
Definition 3.10 *i*) Een herschrijfregel $Lhs \Rightarrow Rhs$ heet *veilig* als:

Als $pp(Lhs) = <P, \pi>$, dan $P \vdash Rhs{:}\pi$.

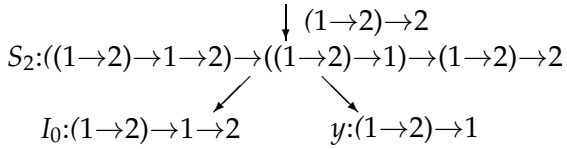
ii) De definitie van een *veilige typetoekenning met betrekking tot* $\mathcal{E}$ is vrijwel dezelfde als die voor een naïeve typetoekenning, en wordt daaruit verkregen door in definitie 3.5 de conditie (*i.a*) en (*i.b*) te vervangen door: Als $pp(Lhs) = <P, \pi>$, dan $P \vdash Rhs{:}\pi$.

Merk op dat de notie $pp(M)$ onafhankelijk van de definitie van typeerbare herschrijfregels is gedefinieerd.

Neem als voorbeeld van een regel die niet veilig is, de definitie van M. Ter verkrijging van $pp(M\,(S_2\,(x,y)))$, worden op de volgende manier types aan knopen in de boom toegekend:



Als de rechterkant getypeerd moet worden met $(1\to2)\to2$, dan is het type $(1\to2)\to1$ nodig voor de knoop die y bevat.



In de typetoekenning aan de herschrijfregel zijn de types toegekend aan de knopen die x en y bevatten dan ook niet de meest algemene types, nodig om het type voor de linkerkant van de herschrijfregel te vinden. Dus deze regel is niet veilig, en moet daarom verworpen worden.

Theorem 3.11 *i)* **De conditie is noodzakelijk.** *Laat Lhs, Rhs* $\in \mathsf{Ter}(\Sigma)$, *en* $\mathbf{r} : Lhs \Rightarrow Rhs$ *een typeerbare herschrijfregel zijn die niet veilig is. Dan bestaan er een vervanging* R, *en een type* μ, *zodat* $\vdash Lhs^{\mathsf{R}}{:}\mu \;\&\; \neg \vdash Rhs^{\mathsf{R}}{:}\mu$.

ii) **De conditie is voldoende.** *Laat Lhs, Rhs* $\in \mathsf{Ter}(\Sigma)$, *en* $\mathbf{r} : Lhs \Rightarrow Rhs$ *een veilige herschrijfregel zijn. Dan geldt voor elke vervanging* R, *basis B en een type* μ: $B \vdash Lhs^{\mathsf{R}}{:}\mu \;\Rightarrow\; B \vdash Rhs^{\mathsf{R}}{:}\mu$.

Bibliografie

(Bakel, S. van [1992]) : *Complete restrictions of the intersection type discipline* , in: *Theoretical Computer Science*, 102, p. 135–163, 1992.

(Bakel, S. van, S. Smetsers, and S. Brock. [1992]) : *Partial type assignment in Left Linear Applicative Term Rewriting Systems* , in: In J.-C. Raoult, editor, *Proceedings of CAAP '92. 17th Colloquim on Trees in Algebra and Programming, Rennes, France*, volume 581 of *Lecture Notes in Computer Science*, pp. 300–321. Springer-Verlag, 1992.

(Barendregt, H. [1984]) : The lambda calculus: its syntax and semantics. North-Holland, Amsterdam, revised edition, 1984.

(Barendregt, H., M. Coppo, M. Dezani-Ciancaglini [1983]) : *A filter lambda model and the completeness of type assignment* , in: *The Journal of Symbolic Logic*, **48(4)** 1983, pp. 931–940.

(Coppen, P.A. [1989]): *GRAMTSY 4.0 GRAMmaticaal Transformationeel SYsteem* , Intern Rapport Afdeling Computerlinguïstiek, Universiteit Nijmegen, 1989.

(Curry, H.B. & R. Feys [1958]): Combinatory Logic volume 1. North-Holland, Amsterdam, 1958.

(Damas, L.M.M. [1985]): Type Assignment in Programming Languages. PhD thesis, University of Edinburgh, Department of Computer Science, Edinburgh, 1985. Thesis CST-33-85.

(Hindley, J.R. [1969]): *The principal type scheme of an object in combinatory logic* , in: *Transactions of the American Mathematical Society*, 146, pp. 29–60, 1969.

(Klop, J.W. [1990]): *Term Rewriting Systems* , Report CS-R9073, Centre for Mathematics and Computer Science, Amsterdam, 1990.

(Kfoury, A.J., J. Tiuryn and P. Urzyczyn [1988]): *A proper extension of ML with an effective type-assignment* , in: *Proceedings of the Fifteenth Annual ACM SIGACT-SIGPLAN Symposium on Principles of Programming Languages*, pp. 58–69, San Diego, California, 1988.

(Milner, R. [1978]): *A theory of type polymorphism in programming* , in: *Journal of Computer and System Sciences*, 17, pp. 348–375, 1978.

(Mycroft, A. [1984]): *Polymorphic type schemes and recursive definitions* , in: *Proceedings of the International Symposium on Programming, Toulouse*, volume 167 of *Lecture Notes Computer Science*, pp. 217–239. Springer-Verlag, 1984.

(Nöcker, E.G.J.M.H., J.E.W. Smetsers, M.C.J.D. van Eekelen, and M.J. Plasmeijer [1991]): *Concurrent Clean* , in: *Proceedings of PARLE '91, Parallel Architectures and Languages Europe, Eindhoven, The Netherlands*, volume 506-II of *Lecture Notes in Computer Science*, pp. 202–219. Springer-Verlag, 1991.

(Pfenning, F. [1988]): *Partial Polymorphic Type Inference and Higher-Order Unification* , in: *Proceedings of the 1988 conference on LISP and Functional Programming Languages*, volume 201 of *Lecture Notes in Computer Science*, pp. 153–163. Springer-Verlag, 1988.

(Robinson, J.A. [1965]): *A machine-oriented logic based on the resolution principle* , in: *Journal of the ACM*, 12(1), pp. 23–41, 1965.

(Turner, D.A. [1985]): *Miranda: A non-strict functional language with polymorphic types* , in: *Proceedings of the conference on Functional Programming Languages and Computer Architecture*, volume 201 of *Lecture Notes in Computer Science*, pp. 1–16. Springer-Verlag, 1985.