Theoreme für lau!

Tim Baumann

Curry Club Augsburg

16. Mai 2017

System F a.k.a. Girard–Reynolds polymorphic λ-calculus

Ein Typkontext ist eine endliche Menge $\Delta = \{X_1, \ldots, X_n\}$ von Typvariablen. Die Typen sind durch die induktive Definition

$$\tau := X \mid \mathsf{Bool} \mid \tau_1 \to \tau_2 \mid \forall X.\, \tau'$$

gegeben. Die Typen im Typkontext Δ sind die Typen, deren freie Variablen in Δ liegen. Formal:

$$\boxed{\Delta \vdash \tau \text{ type}} \qquad \frac{}{\Delta \vdash \mathsf{Bool} \text{ type}} \qquad \frac{X \in \Delta}{\Delta \vdash X \text{ type}}$$

$$\frac{\Delta \vdash \tau_1 \text{ type} \quad \Delta \vdash \tau_2 \text{ type}}{\Delta \vdash \tau_1 \to \tau_2 \text{ type}} \qquad \frac{\Delta \cup \{X\} \vdash \tau \text{ type}}{\Delta \vdash \forall X.\, \tau \text{ type}}$$

Church-Kodierung in System F

$$
\begin{aligned}
\mathrm{Product}(A,B) &:= \forall Y.\,(A \to B \to Y) \to Y \\
\mathrm{Sum}(A,B) &:= \forall Y.\,(A \to Y) \to (B \to Y) \to Y \\
\mathrm{Nat} &:= \forall Y.\,Y \to (Y \to Y) \to Y \\
\mathrm{List}(A) &:= \forall Y.\,Y \to (A \to Y \to Y) \to Y
\end{aligned}
$$

Terme in System F

Ein Wertekontext ist eine endliche Menge $\Gamma = \{x_1 : \tau_1, \dots, x_m : \tau_m\}$, wobei $x_1, \dots, x_m$ paarweise verschiedene Typvariablen sind und $\tau_1, \dots, \tau_m$ Typen sind. Er bildet zusammen mit Δ einen Kontext $\Delta; \Gamma$ falls die freien Variablen der Typen $\tau_1, \dots, \tau_m$ in Δ liegen. Formal:

$$\boxed{(\Delta; \Gamma) \text{ ctx}} \qquad \frac{}{(\Delta; \bullet) \text{ ctx}} \qquad \frac{(\Delta; \Gamma) \text{ ctx} \qquad \Delta \vdash \tau \text{ type}}{(\Delta; \Gamma \cup \{x : \tau\}) \text{ ctx}}$$

Terme in System F

Die Terme t in System F sind induktiv definiert durch

$$t := \mathrm{true} \,|\, \mathrm{false} \,|\, \textbf{if } t_1 \textbf{ then } t_2 \textbf{ else } t_3 \,|\, \lambda x{:}\tau.\, t \,|\, t_1\, t_2 \,|\, \Lambda A.\, t \,|\, t\, \tau$$

Ein Term t hat Typ τ in einem Kontext $\Delta; \Gamma$ (notiert $\Delta; \Gamma \vdash t : \tau$), falls

$$\boxed{\Delta; \Gamma \vdash t : \tau \textit{ vorausgesetzt } (\Delta; \Gamma) \text{ ctx}}$$

$$\frac{(x : \tau) \in \Gamma}{\Delta; \Gamma \vdash x : \tau}$$

$$\frac{v \in \{\mathrm{true}, \mathrm{false}\}}{\Delta; \Gamma \vdash v : \mathrm{Bool}} \qquad \frac{\Delta; \Gamma \vdash b : \mathrm{Bool} \quad \Delta; \Gamma \vdash t : \tau \quad \Delta; \Gamma \vdash e : \tau}{\Delta; \Gamma \vdash \textbf{if } b \textbf{ then } t \textbf{ else } e : \tau}$$

$$\frac{\Delta; \Gamma \cup \{x : \tau_1\} \vdash t : \tau_2}{\Delta; \Gamma \vdash \lambda x{:}\tau_1.\, t : \tau_1 \to \tau_2} \qquad \frac{\Delta; \Gamma \vdash t_1 : \tau_1 \to \tau_2 \quad \Delta; \Gamma \vdash t_2 : \tau_1}{\Delta; \Gamma \vdash t_1\, t_2 : \tau_2}$$

$$\frac{\Delta \cup \{A\}; \Gamma \vdash t : \tau}{\Delta; \Gamma \vdash \Lambda A.\, t : \forall A.\, \tau} \qquad \frac{\Delta; \Gamma \vdash t : \forall A.\, \tau' \quad \Delta \vdash \tau \text{ type}}{\Delta; \Gamma \vdash t\, \tau : \tau'[\tau/A]}$$

Beispielterme

$$
\begin{aligned}
\mathsf{null} &: \quad \forall A.\ \mathsf{List}(A) \to \mathsf{Bool} \\
\mathsf{null} &:= \quad \Lambda A.\, \lambda xs{:}\, \mathsf{List}(A). \\
&\qquad\quad xs\, \mathsf{Bool}\, \mathsf{true}\, (\lambda a{:}A.\, \lambda b{:}\mathsf{Bool}.\, \mathsf{false})
\end{aligned}
$$

$$
\begin{aligned}
\mathsf{pair} &: \quad \forall A.\, \forall B.\, A \to B \to \mathsf{Product}(A, B) \\
\mathsf{pair} &:= \quad \Lambda A.\, \Lambda B.\, \lambda a{:}A.\, \lambda b{:}B. \\
&\qquad\quad \Lambda Y.\, \lambda f{:}A \to B \to Y. \\
&\qquad\qquad f\ a\ b
\end{aligned}
$$

$$
\begin{aligned}
\mathsf{append} &: \quad \forall A.\ \mathsf{List}(A) \to \mathsf{List}(A) \to \mathsf{List}(A) \\
\mathsf{append} &:= \quad \Lambda A.\, \lambda xs{:}\, \mathsf{List}(A).\, \lambda ys{:}\, \mathsf{List}(A). \\
&\qquad\quad \Lambda Y.\, \lambda y{:}Y.\, \lambda f{:}A \to Y \to Y. \\
&\qquad\qquad ys\ Y\ (xs\ Y\ y\ f)\ f
\end{aligned}
$$

$$
\begin{aligned}
\mathsf{map} &: \quad \forall A.\, \forall B.\, (A \to B) \to \mathsf{List}(A) \to \mathsf{List}(B) \\
\mathsf{map} &:= \quad \Lambda A.\, \Lambda B.\, \lambda g{:}A \to B.\, \lambda xs{:}\, \mathsf{List}(A). \\
&\qquad\quad \Lambda Y.\, \lambda y{:}Y.\, \lambda f{:}B \to Y \to Y. \\
&\qquad\qquad xs\ Y\ y\ (\lambda a{:}A.\, f\ (g\ a))
\end{aligned}
$$

Reduktion von Termen in System F

Die Reduktionsrelation $\rightsquigarrow$ auf der Menge der Terme ist die kleinste reflexive, transitive, kongruente Relation mit

$$\begin{aligned}
\textbf{if}\ \mathsf{true}\ \textbf{then}\ t\ \textbf{else}\ e &\rightsquigarrow t \\
\textbf{if}\ \mathsf{false}\ \textbf{then}\ t\ \textbf{else}\ e &\rightsquigarrow e \\
(\lambda x.\, t)s &\rightsquigarrow t[s/x] \\
(\Lambda X.\, t)\tau &\rightsquigarrow t[\tau/X]
\end{aligned}$$

Interpretation von Typen

Eine Typumgebung für einen Typkontext Δ ist eine Abbildung

$$\vec{A} : \Delta \to \mathsf{Types}_\bullet$$

(wobei $\mathsf{Types}_{\widetilde{\Delta}} := \{\tau \mid \widetilde{\Delta} \vdash \tau\}$)
Jede Typumg. $\vec{A}$ induziert für jeden disjunkten Typkontext Δ' eine Abb.

$$[\![-]\!]_{\vec{A}} : \mathsf{Types}_{\Delta \sqcup \Delta'} \to \mathsf{Types}_{\Delta'}$$

rekursiv definiert durch

$$\begin{array}{rcll}
[\![\mathsf{Bool}]\!]_{\vec{A}} & := & \mathsf{Bool} & \\
[\![X]\!]_{\vec{A}} & := & \vec{A}(X) \text{ falls } X \in \Delta & \\
[\![X]\!]_{\vec{A}} & := & X \text{ falls } X \in \Delta' & \\
[\![\tau_1 \to \tau_2]\!]_{\vec{A}} & := & [\![\tau_1]\!]_{\vec{A}} \to [\![\tau_2]\!]_{\vec{A}} & \\
[\![\forall X.\, \tau]\!]_{\vec{A}} & := & \forall X.\, [\![\tau]\!]_{\vec{A}} & \qquad (\text{Œ } X \notin \Delta \cup \Delta')
\end{array}$$

Interpretation von Termen

Eine Termumgebung $\vec{a}$ für einen Kontext $\Delta;\Gamma$ bzgl. einer Typumgebung $\vec{A}$ für Δ ist eine Familie

$$(\vec{a}(x) : [\![\tau]\!]_{\vec{A}})_{(x:\tau)\in\Gamma}.$$

von Termen. Jede solche Termumgebung induziert für jeden von $\Delta;\Gamma$ disjunkten Kontext $\Delta';\Gamma'$ eine Abbildung

$$[\![-]\!]_{\vec{A},\vec{a}} : \mathsf{Terms}_{\Delta\sqcup\Delta',\Gamma\sqcup\Gamma'} \to \mathsf{Terms}_{\Delta',\Gamma'}$$

rekursiv definiert durch

$$\begin{array}{rcll}
[\![v]\!]_{\vec{A},\vec{a}} & := & v & (v \in \{\mathsf{true},\mathsf{false}\}) \\
[\![\mathbf{if}\ b\ \mathbf{then}\ t\ \mathbf{else}\ e]\!]_{\vec{A},\vec{a}} & := & \mathbf{if}\ [\![b]\!]_{\vec{A},\vec{a}}\ \mathbf{then}\ [\![t]\!]_{\vec{A},\vec{a}}\ \mathbf{else}\ [\![e]\!]_{\vec{A},\vec{a}} & \\
[\![\lambda x{:}\tau.\, t]\!]_{\vec{A},\vec{a}} & := & \lambda x{:}[\![\tau]\!]_{\vec{A}}.\, [\![t]\!]_{\vec{A},\vec{a}} & (\text{Œ}\ x \notin \Gamma \cup \Gamma') \\
[\![t_1\ t_2]\!]_{\vec{A},\vec{a}} & := & [\![t_1]\!]_{\vec{A},\vec{a}}\, [\![t_2]\!]_{\vec{A},\vec{a}} & \\
[\![\Lambda X.\, t]\!]_{\vec{A},\vec{a}} & := & \Lambda X.\, [\![t]\!]_{\vec{A},\vec{a}} & (\text{Œ}\ X \notin \Delta \cup \Delta') \\
[\![t\ \tau]\!]_{\vec{A},\vec{a}} & := & [\![t]\!]_{\vec{A},\vec{a}}\, [\![\tau]\!]_{\vec{A}} &
\end{array}$$

Relationen zwischen Typen

Eine Relation $\mathcal{A} : A \Leftrightarrow A'$ zwischen zwei Typen $A, A' \in \text{Types}_\bullet$ ist eine Relation zwischen den Termen dieser beiden Typen, für die gilt:

$$t_1 \rightsquigarrow t_2, \;\; t_1' \rightsquigarrow t_2', \;\; \mathcal{A}(t_2, t_2') \implies \mathcal{A}(t_1, t_1').$$

Eine Relation $\vec{\mathcal{A}} : \vec{A} \Leftrightarrow \vec{A}'$ zwischen Typumgebungen $\vec{A}$ und $\vec{A}'$ für Δ ist eine Familie

$$(\mathcal{A}_X : \vec{A}(X) \Leftrightarrow \vec{A}'(X))_{X \in \Delta}$$

von Relationen zwischen den Typen $\vec{A}(X)$ und $\vec{A}'(X)$

Funktionen als Relationen

Sei A ein Typ. Dann definiert

$$t \sim_{\mathrm{obs}} t' \quad :\Longleftrightarrow \quad \text{für alle } f : A \to \mathsf{Bool} \text{ gilt } f\ t \sim_{\mathsf{Bool}} f\ t'$$

eine Relation (sogar eine Äquivalenzrelation) $\sim_{\mathrm{obs}}: A \Leftrightarrow A$.
Seien A und B Typen und $f : A \to B$. Dann definiert

$$\mathcal{R}_f(x, y) \quad :\Longleftrightarrow \quad f\ x \sim_{\mathrm{obs}} y$$

eine Relation $\mathcal{R}_f AB$.

Relationen zwischen Typen

Solch eine Relation $\vec{\mathcal{A}} : \vec{A} \Leftrightarrow \vec{A'}$ induziert für jeden Typ τ im Typkontext Δ eine Relation $[\![\tau]\!]_{\vec{\mathcal{A}}} : [\![\tau]\!]_{\vec{A}} \Leftrightarrow [\![\tau]\!]_{\vec{A'}}$ wie folgt:

$$
\begin{array}{rcl}
[\![\mathsf{Bool}]\!]_{\vec{\mathcal{A}}} & := & \sim_{\mathsf{Bool}} \text{ mit } t \sim_{\mathsf{Bool}} t' \text{ genau dann wenn} \\
 & & \quad t \rightsquigarrow \mathsf{true} \iff t' \rightsquigarrow \mathsf{true} \\
[\![X]\!]_{\vec{\mathcal{A}}} & := & \mathcal{A}_X \\
[\![\tau_1 \to \tau_2]\!]_{\vec{\mathcal{A}}} & := & [\![\tau_1]\!]_{\mathcal{A}} \to [\![\tau_2]\!]_{\mathcal{A}} \\
[\![\forall X.\, \tau]\!]_{\vec{\mathcal{A}}} & := & \sim \text{ mit } g \sim g' \text{ genau dann wenn} \\
 & & \quad \text{für alle } A, A' \in \mathsf{Types}_\bullet \\
 & & \quad \text{und Relationen } \mathcal{A} : A \Leftrightarrow A' \\
 & & \quad \text{gilt } [\![\tau]\!]_{(\vec{\mathcal{A}} \cup \{X \mapsto \mathcal{A}\})}(g\, A, g'\, A')
\end{array}
$$

mit $(\mathcal{R}_1 \to \mathcal{R}_2)(f, f') :\Longleftrightarrow$ für alle a, a' mit $\mathcal{R}_1(a, a')$ gilt $\mathcal{R}_2(f\, a, f'\, a')$.

Relationen zwischen Typen – Ein Beispiel

Sei $f : A \to B$ eine Funktion. Dann ist

$$[\![\mathsf{List}(X)]\!]_{\{X \mapsto \mathcal{R}_f\}} : [\![\mathsf{List}(X)]\!]_{\{X \mapsto A\}} \Leftrightarrow [\![\mathsf{List}(X)]\!]_{\{X \mapsto B\}}$$

eine Relation zwischen $\mathsf{List}(A)$ und $\mathsf{List}(B)$.

Satz (Parametrizität)

Für alle $\tau \in$ *Types*$_\bullet$ *und* $t : \tau$ *gilt* $[\![\tau]\!]_\bullet(t, t)$

Parametrizität

Seien $\vec{A}$ und $\vec{A'}$ Typumgebungen für Δ, $\vec{\mathcal{A}} : \vec{A} \Leftrightarrow \vec{A'}$ eine Relation. Zwei Termumg. $\vec{a}$ und $\vec{a'}$ für $\Delta; \Gamma$ bzgl. $\vec{A}$ bzw. $\vec{A'}$ sind relatiert bzgl. $\vec{\mathcal{A}}$, falls

$$[\![\tau]\!]_{\vec{\mathcal{A}}}(\vec{a}(x), \vec{a'}(x)) \quad \text{für alle } (x : \tau) \in \Gamma.$$

$$\begin{aligned}\Delta; \Gamma \models t : \tau :\Longleftrightarrow \text{ für } & \text{ alle Typumgebungen } \vec{A}, \vec{A'} \text{ von } \Delta \text{ und} \\ & \text{ alle Relationen } \vec{\mathcal{A}} : \vec{A} \Leftrightarrow \vec{A'} \text{ und} \\ & \text{ alle relatierten Termumg. } \vec{a} \text{ bzgl. } \vec{A} \text{ und } \vec{a'} \text{ bzgl. } \vec{A'} \\ \text{gilt } & [\![\tau]\!]_{\vec{\mathcal{A}}}([\![t]\!]_{\vec{A},\vec{a}}, [\![t]\!]_{\vec{A'},\vec{a'}})\end{aligned}$$

Satz (Parametrizität)

Aus $\Delta; \Gamma \vdash t : \tau$ *folgt* $\Delta; \Gamma \models t : \tau$

Kurz: Interpretationen in relatierten Umgebungen sind relatiert.

Ein Isomorphismus

Sei A ein Typ. Definiere $\widehat{A} := \forall Y.\,(A \to Y) \to Y$. Wir haben Funktionen

$$\begin{aligned} i &: A \to \widehat{A}, \quad i := \lambda x{:}A.\,\Lambda Y.\,\lambda g{:}A \to Y.\,g\,x \\ j &: \widehat{A} \to A, \quad j := \lambda h{:}\widehat{A}.\,h\,A\,(\lambda x{:}a.\,x) \end{aligned}$$

Es gilt $j\,(i\,x) \rightsquigarrow x$, also $j\,(i\,x) \sim_{\mathrm{obs}} x$, aber stimmt auch $i\,(j\,h) \sim_{\mathrm{obs}} h$?
Parametricity to the rescue! Für $h : \widehat{A}$, $f : A \to B$ und $b : B \to B'$ gilt

Ein Isomorphismus

Sei A ein Typ. Definiere $\widehat{A} := \forall Y.\,(A \to Y) \to Y$. Wir haben Funktionen

$$
\begin{aligned}
i : A \to \widehat{A}, &\quad i := \lambda x{:}A.\, \Lambda Y.\, \lambda g{:}A \to Y.\, g\, x \\
j : \widehat{A} \to A, &\quad j := \lambda h{:}\widehat{A}.\, h\, A\, (\lambda x{:}a.\, x)
\end{aligned}
$$

Es gilt $j\,(i\,x) \rightsquigarrow x$, also $j\,(i\,x) \sim_{\mathrm{obs}} x$, aber stimmt auch $i\,(j\,h) \sim_{\mathrm{obs}} h$?
Parametricity to the rescue! Für $h : \widehat{A}$, $f : A \to B$ und $b : B \to B'$ gilt

$$
\begin{aligned}
& [\![\widehat{A}]\!]_\bullet(h, h) \\
\iff \quad & \text{für alle } S, S' \text{ und } \mathcal{S} : S \Leftrightarrow S' \text{ gilt } [\![(A \to Y) \to Y]\!]_{\{Y \mapsto \mathcal{S}\}}(h\,S, h\,S') \\
\implies \quad & [\![(A \to Y) \to Y]\!]_{\{Y \mapsto \mathcal{R}_b\}}(h\,B, h\,B') \\
\iff \quad & \text{für alle } g : A \to B,\ g' : A \to B' \text{ mit } [\![A \to Y]\!]_{\{Y \mapsto \mathcal{R}_b\}}(g, g') \\
& \text{gilt } [\![Y]\!]_{\{Y \mapsto \mathcal{R}_b\}}(h\,B\,g, h\,B'\,g') \\
\implies \quad & [\![Y]\!]_{\{Y \mapsto \mathcal{R}_b\}}(h\,B\,f, h\,B'\,(b \circ f)) \\
\iff \quad & b\,(h\,B\,f) \sim_{\mathrm{obs}} h\,B'\,(b \circ f)
\end{aligned}
$$

Ein Isomorphismus

Sei A ein Typ. Definiere $\widehat{A} := \forall Y.\,(A \to Y) \to Y$. Wir haben Funktionen

$$\begin{aligned} i &: A \to \widehat{A}, & i &:= \lambda x{:}A.\,\Lambda Y.\,\lambda g{:}A \to Y.\,g\,x \\ j &: \widehat{A} \to A, & j &:= \lambda h{:}\widehat{A}.\,h\,A\,(\lambda x{:}a.\,x) \end{aligned}$$

Es gilt $j\,(i\,x) \rightsquigarrow x$, also $j\,(i\,x) \sim_{\mathrm{obs}} x$, aber stimmt auch $i\,(j\,h) \sim_{\mathrm{obs}} h$?
Parametricity to the rescue! Für $h : \widehat{A}$, $f : A \to B$ und $b : B \to B'$ gilt

$$b\,(h\,B\,f) \sim_{\mathrm{obs}} h\,B'\,(b \circ f).$$

Ein Isomorphismus

Sei A ein Typ. Definiere $\widehat{A} := \forall Y.\, (A \to Y) \to Y$. Wir haben Funktionen

$$\begin{aligned} i : A \to \widehat{A}, &\quad i := \lambda x{:}A.\, \Lambda Y.\, \lambda g{:}A \to Y.\, g\, x \\ j : \widehat{A} \to A, &\quad j := \lambda h{:}\widehat{A}.\, h\, A\, (\lambda x{:}a.\, x) \end{aligned}$$

Es gilt $j\,(i\,x) \rightsquigarrow x$, also $j\,(i\,x) \sim_{\mathrm{obs}} x$, aber stimmt auch $i\,(j\,h) \sim_{\mathrm{obs}} h$?
Parametricity to the rescue! Für $h : \widehat{A}$, $f : A \to B$ und $b : B \to B'$ gilt

$$b\,(h\,B\,f) \sim_{\mathrm{obs}} h\,B'\,(b \circ f).$$

Es folgt mit $B = A$, $B' = X$, $b = g$ und $f = (\lambda x{:}A.\, x)$:

$$\begin{aligned} i\,(j\,h) &\sim_{\mathrm{obs}} \Lambda X.\, \lambda g{:}A \to X.\, g\,(h\,A\,(\lambda x{:}A.\, x)) \\ &\sim_{\mathrm{obs}} \Lambda X.\, \lambda g{:}A \to X.\, h\,X\,(g \circ (\lambda x{:}A.\, x)) \\ &\sim_{\mathrm{obs}} \Lambda X.\, \lambda g{:}A \to X.\, h\,X\,g \\ &\sim_{\mathrm{obs}} h \end{aligned}$$

- `http://twitter.com/parametricity`
- Philip Wadler, *Theorems for Free!*, 4'th International Conference on Functional Programming and Computer Architecture, London, September 1989
- Robert Harper, *Practical Foundations for Programming Languages*, zweite Auflage, Kapitel 48