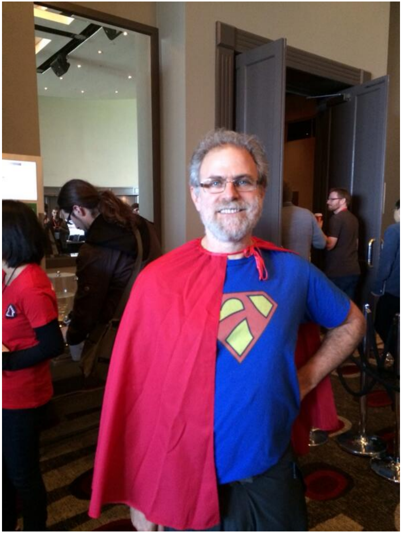
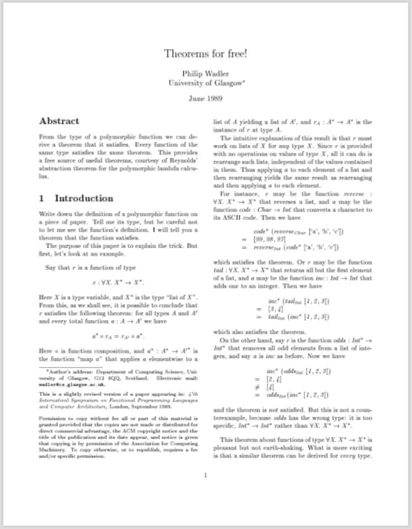
Theoreme für lau!

Tim Baumann

Curry Club Augsburg

18. Mai 2017





Theorems for free!

Philip Wadler
University of Glasgow*

June 1989

Abstract

From the type of a polymorphic function we can derive a theorem that it satisfies. Every function of the same type satisfies the same theorem. This provides a free source of useful theorems, courtesy of Reynolds' abstraction theorem for the polymorphic lambda calculus.

1 Introduction

Write down the definition of a polymorphic function on a piece of paper. Tell me its type, but be careful not to let me see the function's definition. I will tell you a theorem that the function satisfies.

The purpose of this paper is to explain the trick. But first, let's look at an example.

Say that r is a function of type

$$r : \forall X.\ X^* \to X^*.$$

Here X is a type variable, and X^* is the type "list of X". From this, as we shall see, it is possible to conclude that r satisfies the following theorem: for all types A and A' and every total function $a : A \to A'$ we have

$$a^* \circ r_A = r_{A'} \circ a^*.$$

Here $\circ$ is function composition, and $a^* : A^* \to A'^*$ is the function "map a" that applies a elementwise to a list of A yielding a list of A', and $r_A : A^* \to A^*$ is the instance of r at type A.

The intuitive explanation of this result is that r must work on lists of X for *any* type X. Since r is provided with no operations on values of type X, all it can do is rearrange such lists, independent of the values contained in them. Thus applying a to each element of a list and then rearranging yields the same result as rearranging and then applying a to each element.

For instance, r may be the function $reverse : \forall X.\ X^* \to X^*$ that reverses a list, and a may be the function $code : Char \to Int$ that converts a character to its ASCII code. Then we have

$$\begin{aligned} & code^*\ (reverse_{Char}\ [\text{'a'}, \text{'b'}, \text{'c'}]) \\ = \ & [99, 98, 97] \\ = \ & reverse_{Int}\ (code^*\ [\text{'a'}, \text{'b'}, \text{'c'}]) \end{aligned}$$

which satisfies the theorem. Or r may be the function $tail : \forall X.\ X^* \to X^*$ that returns all but the first element of a list, and a may be the function $inc : Int \to Int$ that adds one to an integer. Then we have

$$\begin{aligned} & inc^*\ (tail_{Int}\ [1, 2, 3]) \\ = \ & [3, 4] \\ = \ & tail_{Int}\ (inc^*\ [1, 2, 3]) \end{aligned}$$

which also satisfies the theorem.

On the other hand, say r is the function $odds : Int^* \to Int^*$ that removes all odd elements from a list of integers, and say a is inc as before. Now we have

$$\begin{aligned} & inc^*\ (odds_{Int}\ [1, 2, 3]) \\ = \ & [2, 4] \\ \neq \ & [4] \\ = \ & odds_{Int}(inc^*\ [1, 2, 3]) \end{aligned}$$

and the theorem is *not* satisfied. But this is not a counterexample, because *odds* has the wrong type: it is too specific, $Int^* \to Int^*$ rather than $\forall X.\ X^* \to X^*$.

This theorem about functions of type $\forall X.\ X^* \to X^*$ is pleasant but not earth-shaking. What is more exciting is that a similar theorem can be derived for *every* type.

*Author's address: Department of Computing Science, University of Glasgow, G12 8QQ, Scotland. Electronic mail: wadler@cs.glasgow.ac.uk.

This is a slightly revised version of a paper appearing in: *4'th International Symposium on Functional Programming Languages and Computer Architecture*, London, September 1989.

Assume $a : A \to A'$ and $b : B \to B'$.

$$head : \forall X.\ X^* \to X$$
$$a \circ head_A = head_{A'} \circ a^*$$

$$tail : \forall X.\ X^* \to X^*$$
$$a^* \circ tail_A = tail_{A'} \circ a^*$$

$$(+\!\!+) : \forall X.\ X^* \to X^* \to X^*$$
$$a^*\ (xs +\!\!+_A ys) = (a^*\ xs) +\!\!+_{A'} (a^*\ ys)$$

$$concat : \forall X.\ X^{**} \to X^*$$
$$a^* \circ concat_A = concat_{A'} \circ a^{**}$$

$$fst : \forall X.\ \forall Y.\ X \times Y \to X$$
$$a \circ fst_{AB} = fst_{A'B'} \circ (a \times b)$$

$$snd : \forall X.\ \forall Y.\ X \times Y \to Y$$
$$b \circ snd_{AB} = snd_{A'B'} \circ (a \times b)$$

$$zip : \forall X.\ \forall Y.\ (X^* \times Y^*) \to (X \times Y)^*$$
$$(a \times b)^* \circ zip_{AB} = zip_{A'B'} \circ (a^* \times b^*)$$

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

Die Typen von System F sind durch die induktive Definition

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

gegeben, wobei X für eine Typvariable steht.

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

Die Typen von System F sind durch die induktive Definition

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

gegeben, wobei X für eine Typvariable steht. Ein Typkontext ist eine endliche Menge $\Delta = \{X_1, \ldots, X_n\}$ von Typvariablen.

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

Die Typen von System F sind durch die induktive Definition

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

gegeben, wobei X für eine Typvariable steht. Ein Typkontext ist eine endliche Menge $\Delta = \{X_1, \ldots, X_n\}$ von Typvariablen. 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

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

Church-Kodierung in System F

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

Terme in System F

Die Terme t in System F sind induktiv definiert durch

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

Terme in System F

Die Terme t in System F sind induktiv definiert durch

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

Ein Wertekontext ist eine endliche Menge $\Gamma = \{x_1 : \tau_1, \ldots, x_m : \tau_m\}$, wobei $x_1, \ldots, x_m$ paarweise verschiedene Typvariablen sind und $\tau_1, \ldots, \tau_m$ Typen sind.

Terme in System F

Die Terme t in System F sind induktiv definiert durch

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

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

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

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

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

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

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

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

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

Beispielterme

$$
\begin{array}{lll}
\mathsf{id} & : & \forall A.\, A \to A \\
\mathsf{id} & := & \Lambda A.\, \lambda x{:}A.\, x
\end{array}
$$

Beispielterme

$$
\begin{array}{lll}
\mathsf{id} & : & \forall A.\, A \to A \\
\mathsf{id} & := & \Lambda A.\, \lambda x{:}A.\, x
\end{array}
$$

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

Beispielterme

$$
\begin{aligned}
\mathsf{id} \quad &: \quad \forall A.\, A \to A \\
\mathsf{id} \quad &:= \quad \Lambda A.\, \lambda x{:}A.\, x
\end{aligned}
$$

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

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

Beispielterme

$$
\begin{aligned}
\mathsf{id} &: \quad \forall A.\, A \to A \\
\mathsf{id} &:= \quad \Lambda A.\, \lambda x{:}A.\, x
\end{aligned}
$$

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

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

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

Beispielterme

$$
\begin{array}{lcl}
\mathsf{id} & : & \forall A.\, A \to A \\
\mathsf{id} & := & \Lambda A.\, \lambda x{:}A.\, x
\end{array}
$$

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

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

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

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

Reduktion und Äquivalenz von Termen in System F

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

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

Reduktion und Äquivalenz von Termen in System F

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

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

Zwei Terme $b, b' : \mathsf{Bool}$ heißen Kleene-äquivalent, notiert $b \simeq b'$, falls

$$b \simeq b' \quad :\Longleftrightarrow \quad (b \rightsquigarrow \mathsf{true} \iff b' \rightsquigarrow \mathsf{true}).$$

Reduktion und Äquivalenz von Termen in System F

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

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

Zwei Terme $b, b' : \mathsf{Bool}$ heißen Kleene-äquivalent, notiert $b \simeq b'$, falls

$$b \simeq b' \quad :\Longleftrightarrow \quad (b \rightsquigarrow \mathsf{true} \iff b' \rightsquigarrow \mathsf{true}).$$

Sei A ein Typ. Zwei Terme $t, t' : A$ heißen beobachtungsäquivalent, falls

$$t \doteq t' \quad :\Longleftrightarrow \quad \text{für alle } f : A \to \mathsf{Bool} \text{ gilt } f\,t \simeq f\,t'.$$

Interpretation von Typen

Eine Typumgebung für einen Typkontext Δ ist eine Abbildung

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

wobei wir $\mathsf{Types}_{\widetilde{\Delta}} := \{\tau \mid \widetilde{\Delta} \vdash \tau\}$ für alle Typkontexte $\widetilde{\Delta}$ definieren.

Interpretation von Typen

Eine Typumgebung für einen Typkontext Δ ist eine Abbildung

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

wobei wir $\mathsf{Types}_{\widetilde{\Delta}} := \{\tau \mid \widetilde{\Delta} \vdash \tau\}$ für alle Typkontexte $\widetilde{\Delta}$ definieren.

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

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

Relationen zwischen Typen

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

$$\text{aus} \quad t_1 \cong t_2, \;\; t_1' \cong t_2' \qquad \text{folgt} \qquad \mathcal{A}(t_1, t_1') \iff \mathcal{A}(t_2, t_2').$$

Relationen zwischen Typen

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

$$\text{aus} \quad t_1 \cong t_2,\;\; t_1' \cong t_2' \qquad \text{folgt} \qquad \mathcal{A}(t_1, t_1') \iff \mathcal{A}(t_2, t_2').$$

Wichtiges Beispiel: Seien A und B Typen und $f : A \to B$ eine Funktion. Dann definiert

$$x\,|f\rangle\,y \quad :\Longleftrightarrow \quad f\,x \cong y$$

eine Relation $|f\rangle : A \Leftrightarrow B$.

Relationen zwischen Typen

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

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

Relationen zwischen Typen

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

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

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:

$$\begin{array}{rcl}
[\![\mathsf{Bool}]\!]_{\vec{\mathcal{A}}} & := & (\simeq) \\
[\![X]\!]_{\vec{\mathcal{A}}} & := & \mathcal{A}_X \\
[\![\tau_1 \to \tau_2]\!]_{\vec{\mathcal{A}}} & := & [\![\tau_1]\!]_{\vec{\mathcal{A}}} \to [\![\tau_2]\!]_{\vec{\mathcal{A}}} \\
[\![\forall X.\, \tau]\!]_{\vec{\mathcal{A}}} & := & \sim \ \text{mit } g \sim g' \text{ genau dann wenn} \\
& & \quad \text{für alle } A, A' \in \mathsf{Types}_\bullet \\
& & \quad \text{und Relationen } \mathcal{A} : A \Leftrightarrow A' \\
& & \quad \text{gilt } [\![\tau]\!]_{(\vec{\mathcal{A}} \cup \{X \mapsto \mathcal{A}\})}(g\, A, g'\, A'),
\end{array}$$

wobei wir definieren:

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

Satz (Parametrizität)

Für jeden Typ $\tau \in$ *Types*$_\bullet$ *und jeden Term* $\mathrm{t} : \tau$ *gilt* $[\![\tau]\!]_\bullet(\mathrm{t}, \mathrm{t})$

Satz (Parametrizität)

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

Satz

Für jeden Typ $\tau \in$ *Types*$_\bullet$ *und Terme* $t, t' : \tau$ *gilt*

$$t \cong t' \iff [\![\tau]\!]_\bullet(t, t').$$

Ein Isomorphismus

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

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

Ein Isomorphismus

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

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

Es gilt $j\,(i\,x) \rightsquigarrow x$, also $j\,(i\,x) \stackrel{\frown}{=} x$, aber stimmt auch $i\,(j\,h) \stackrel{\frown}{=} h$?

Ein Isomorphismus

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

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

Es gilt $j\,(i\,x) \rightsquigarrow x$, also $j\,(i\,x) \cong x$, aber stimmt auch $i\,(j\,h) \cong h$?
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

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

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

$$[\![\widehat{A}]\!]_{\bullet}(h, h)$$

Ein Isomorphismus

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

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

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

$$\begin{aligned} & [\![\widehat{A}]\!]_\bullet(h, h) \\ \iff \quad & \text{für alle } S, S' \text{ und } \mathcal{S} : S \Leftrightarrow S' \text{ gilt } [\![(A \to Y) \to Y]\!]_{\{Y \mapsto \mathcal{S}\}}(h\,S, h\,S') \end{aligned}$$

Ein Isomorphismus

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

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

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

$$\begin{aligned} & [\![\widehat{A}]\!]_\bullet(h, h) \\ \iff \quad & \text{für alle } S, S' \text{ und } \mathcal{S} : S \Leftrightarrow S' \text{ gilt } [\![(A \to Y) \to Y]\!]_{\{Y \mapsto \mathcal{S}\}}(h\,S, h\,S') \\ \implies \quad & [\![(A \to Y) \to Y]\!]_{\{Y \mapsto \langle b \rangle\}}(h\,B, h\,B') \end{aligned}$$

Ein Isomorphismus

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

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

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

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

Ein Isomorphismus

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

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

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

$$
\begin{aligned}
& [\![\widehat{A}]\!]_\bullet(h, h) \\
\iff\quad & \text{für alle } S, S' \text{ und } \mathcal{S} : S \Leftrightarrow S' \text{ gilt } [\![(A \to Y) \to Y]\!]_{\{Y \mapsto \mathcal{S}\}}(h\,S, h\,S') \\
\implies\quad & [\![(A \to Y) \to Y]\!]_{\{Y \mapsto |b\rangle\}}(h\,B, h\,B') \\
\iff\quad & \text{für alle } g : A \to B,\ g' : A \to B' \text{ mit } b\,(g\,a) = g'\,a \text{ f. a. } a \in A \\
& \text{gilt } [\![Y]\!]_{\{Y \mapsto |b\rangle\}}(h\,B\,g, h\,B'\,g')
\end{aligned}
$$

Ein Isomorphismus

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

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

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

$$\begin{aligned} & [\![\widehat{A}]\!]_\bullet(h, h) \\ \iff\quad & \text{für alle } S, S' \text{ und } \mathcal{S} : S \Leftrightarrow S' \text{ gilt } [\![(A \to Y) \to Y]\!]_{\{Y \mapsto \mathcal{S}\}}(h\, S, h\, S') \\ \implies\quad & [\![(A \to Y) \to Y]\!]_{\{Y \mapsto \langle b \rangle\}}(h\, B, h\, B') \\ \iff\quad & \text{für alle } g : A \to B,\ g' : A \to B' \text{ mit } b\,(g\, a) = g'\, a \text{ f. a. } a \in A \\ & \text{gilt } [\![Y]\!]_{\{Y \mapsto \langle b \rangle\}}(h\, B\, g, h\, B'\, g') \\ \implies\quad & [\![Y]\!]_{\{Y \mapsto \langle b \rangle\}}(h\, B\, f, h\, B'\, (b \circ f)) \end{aligned}$$

Ein Isomorphismus

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

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

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

$$
\begin{aligned}
& [\![\widehat{A}]\!]_\bullet(h, h) \\
\iff \quad & \text{für alle } S, S' \text{ und } \mathcal{S} : S \Leftrightarrow S' \text{ gilt } [\![(A \to Y) \to Y]\!]_{\{Y \mapsto \mathcal{S}\}}(h\,S, h\,S') \\
\implies \quad & [\![(A \to Y) \to Y]\!]_{\{Y \mapsto \langle b \rangle\}}(h\,B, h\,B') \\
\iff \quad & \text{für alle } g : A \to B,\ g' : A \to B' \text{ mit } b\,(g\,a) = g'\,a \text{ f. a. } a \in A \\
& \text{gilt } [\![Y]\!]_{\{Y \mapsto \langle b \rangle\}}(h\,B\,g, h\,B'\,g') \\
\implies \quad & [\![Y]\!]_{\{Y \mapsto \langle b \rangle\}}(h\,B\,f, h\,B'\,(b \circ f)) \\
\iff \quad & b\,(h\,B\,f) \cong h\,B'\,(b \circ f)
\end{aligned}
$$

Ein Isomorphismus

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

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

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

$$b\,(h\,B\,f) \cong h\,B'\,(b \circ f).$$

Ein Isomorphismus

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

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

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

$$b\,(h\,B\,f) \cong h\,B'\,(b \circ f).$$

Es folgt mit $B = A$, $B' = X$, $b = g$ und $f = \mathsf{id}\ A$:

$$\begin{aligned} i\,(j\,h) &\cong \Lambda X.\,\lambda g{:}A \to X.\,g\,(h\,A\,(\mathsf{id}\ A)) \\ &\cong \Lambda X.\,\lambda g{:}A \to X.\,h\,X\,(g \circ (\mathsf{id}\ A)) \end{aligned}$$

Ein Isomorphismus

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

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

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

$$b\,(h\,B\,f) \cong h\,B'\,(b \circ f).$$

Es folgt mit $B = A$, $B' = X$, $b = g$ und $f = \mathsf{id}\ A$:

$$\begin{aligned} i\,(j\,h) &\cong \Lambda X.\,\lambda g{:}A \to X.\,g\,(h\,A\,(\mathsf{id}\ A)) \\ &\cong \Lambda X.\,\lambda g{:}A \to X.\,h\,X\,(g \circ (\mathsf{id}\ A)) \\ &\cong \Lambda X.\,\lambda g{:}A \to X.\,h\,X\,g \end{aligned}$$

Ein Isomorphismus

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

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

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

$$b\,(h\,B\,f) \cong h\,B'\,(b \circ f).$$

Es folgt mit $B = A$, $B' = X$, $b = g$ und $f = \mathsf{id}\ A$:

$$\begin{aligned} i\,(j\,h) &\cong \Lambda X.\,\lambda g{:}A \to X.\,g\,(h\,A\,(\mathsf{id}\ A)) \\ &\cong \Lambda X.\,\lambda g{:}A \to X.\,h\,X\,(g \circ (\mathsf{id}\ A)) \\ &\cong \Lambda X.\,\lambda g{:}A \to X.\,h\,X\,g \\ &\cong h \end{aligned}$$

Das zweite Funktoraxiom

Sei $F(X)$ ein über X parametrisierter Typ, (d.h. $X \vdash F(X)$) und gelte, dass X in $F(X)$ nur kovariant vorkommt.

Das zweite Funktoraxiom

Sei $F(X)$ ein über X parametrisierter Typ, (d.h. $X \vdash F(X)$) und gelte, dass X in $F(X)$ nur kovariant vorkommt.

Lemma

Für alle $A, B \in$ *Types*$_\bullet$ *und* $f : A \to B$ *gibt es eine Funktion* $f_* : F(A) \to F(B)$, *sodass für alle* $a : F(A)$ *und* $b : F(B)$ *gilt:*

$$[\![F(X)]\!]_{\{X \mapsto |f\rangle\}}(a, b) \quad \Longleftrightarrow \quad f_*(a) \cong b$$

Das zweite Funktoraxiom

Sei $F(X)$ ein über X parametrisierter Typ, (d.h. $X \vdash F(X)$) und gelte, dass X in $F(X)$ nur kovariant vorkommt.

Lemma

Für alle $A, B \in$ *Types*$_\bullet$ *und* $f : A \to B$ *gibt es eine Funktion* $f_* : F(A) \to F(B)$*, sodass für alle* $a : F(A)$ *und* $b : F(B)$ *gilt:*

$$[\![F(X)]\!]_{\{X \mapsto |f\rangle\}}(a, b) \quad \Longleftrightarrow \quad f_*(a) \cong b$$

Seien $\mathrm{fmap} : \forall X.\, \forall Y.\, (X \to Y) \to (F(X) \to F(Y))$, Typen A, A', B, B' und Funktionen $f_1 : A \to A'$, $f_2 : A' \to B'$, $g_1 : A \to B$, $g_2 : B \to B'$ mit $f_2 \circ f_1 \cong g_2 \circ g_1$ sowie $p : F(A)$ gegeben. Dann gilt:

Das zweite Funktoraxiom

Sei $F(X)$ ein über X parametrisierter Typ, (d.h. $X \vdash F(X)$) und gelte, dass X in $F(X)$ nur kovariant vorkommt.

Lemma

Für alle $A, B \in$ *Types*$_\bullet$ *und* $f : A \to B$ *gibt es eine Funktion* $f_* : F(A) \to F(B)$, *sodass für alle* $a : F(A)$ *und* $b : F(B)$ *gilt:*

$$[\![F(X)]\!]_{\{X \mapsto \langle f \rangle\}}(a, b) \quad \Longleftrightarrow \quad f_*(a) \cong b$$

Seien $\mathrm{fmap} : \forall X.\, \forall Y.\, (X \to Y) \to (F(X) \to F(Y))$, Typen A, A', B, B' und Funktionen $f_1 : A \to A'$, $f_2 : A' \to B'$, $g_1 : A \to B$, $g_2 : B \to B'$ mit $f_2 \circ f_1 \cong g_2 \circ g_1$ sowie $p : F(A)$ gegeben. Dann gilt:

$$[\![\forall X.\, \forall Y.\, (X \to Y) \to (F(X) \to F(Y)]\!]_\bullet(\mathrm{fmap}, \mathrm{fmap})$$

Das zweite Funktoraxiom

Sei $F(X)$ ein über X parametrisierter Typ, (d.h. $X \vdash F(X)$) und gelte, dass X in $F(X)$ nur kovariant vorkommt.

Lemma

Für alle $A, B \in$ *Types*$_\bullet$ *und* $f : A \to B$ *gibt es eine Funktion* $f_* : F(A) \to F(B)$, *sodass für alle* $a : F(A)$ *und* $b : F(B)$ *gilt:*

$$[\![F(X)]\!]_{\{X \mapsto |f\rangle\}}(a, b) \iff f_*(a) \cong b$$

Seien $\mathsf{fmap} : \forall X.\, \forall Y.\, (X \to Y) \to (F(X) \to F(Y))$, Typen A, A', B, B' und Funktionen $f_1 : A \to A'$, $f_2 : A' \to B'$, $g_1 : A \to B$, $g_2 : B \to B'$ mit $f_2 \circ f_1 \cong g_2 \circ g_1$ sowie $p : F(A)$ gegeben. Dann gilt:

$$\begin{aligned} & [\![\forall X.\, \forall Y.\, (X \to Y) \to (F(X) \to F(Y)]\!]_\bullet(\mathsf{fmap}, \mathsf{fmap}) \\ \implies \quad & [\![(X \to Y) \to (F(X) \to F(Y)]\!]_{\{X \mapsto |f_1\rangle, Y \mapsto |g_2\rangle\}}(\mathsf{fmap}\ A\ B, \mathsf{fmap}\ A'\ B') \end{aligned}$$

Das zweite Funktoraxiom

Sei $F(X)$ ein über X parametrisierter Typ, (d.h. $X \vdash F(X)$) und gelte, dass X in $F(X)$ nur kovariant vorkommt.

Lemma

Für alle $A, B \in$ *Types*$_\bullet$ *und* $f : A \to B$ *gibt es eine Funktion* $f_* : F(A) \to F(B)$, *sodass für alle* $a : F(A)$ *und* $b : F(B)$ *gilt:*

$$[\![F(X)]\!]_{\{X \mapsto |f\rangle\}}(a, b) \iff f_*(a) \cong b$$

Seien $\mathsf{fmap} : \forall X. \forall Y. (X \to Y) \to (F(X) \to F(Y))$, Typen A, A', B, B' und Funktionen $f_1 : A \to A'$, $f_2 : A' \to B'$, $g_1 : A \to B$, $g_2 : B \to B'$ mit $f_2 \circ f_1 \cong g_2 \circ g_1$ sowie $p : F(A)$ gegeben. Dann gilt:

$$\begin{aligned}
& [\![\forall X. \forall Y. (X \to Y) \to (F(X) \to F(Y)]\!]_\bullet(\mathsf{fmap}, \mathsf{fmap}) \\
\Longrightarrow \quad & [\![(X \to Y) \to (F(X) \to F(Y)]\!]_{\{X \mapsto |f_1\rangle, Y \mapsto |g_2\rangle\}}(\mathsf{fmap}\ A\ B, \mathsf{fmap}\ A'\ B') \\
\Longrightarrow \quad & [\![F(X) \to F(Y)]\!]_{\{X \mapsto |f_1\rangle, Y \mapsto |g_2\rangle\}}(\mathsf{fmap}\ A\ B\ g_1, \mathsf{fmap}\ A'\ B'\ f_2)
\end{aligned}$$

Das zweite Funktoraxiom

Sei $F(X)$ ein über X parametrisierter Typ, (d.h. $X \vdash F(X)$) und gelte, dass X in $F(X)$ nur kovariant vorkommt.

Lemma

Für alle $A, B \in$ *Types*$_\bullet$ *und* $f : A \to B$ *gibt es eine Funktion* $f_* : F(A) \to F(B)$, *sodass für alle* $a : F(A)$ *und* $b : F(B)$ *gilt:*

$$[\![F(X)]\!]_{\{X \mapsto |f\rangle\}}(a, b) \quad \Longleftrightarrow \quad f_*(a) \cong b$$

Seien $\mathrm{fmap} : \forall X. \forall Y. (X \to Y) \to (F(X) \to F(Y))$, Typen A, A', B, B' und Funktionen $f_1 : A \to A'$, $f_2 : A' \to B'$, $g_1 : A \to B$, $g_2 : B \to B'$ mit $f_2 \circ f_1 \cong g_2 \circ g_1$ sowie $p : F(A)$ gegeben. Dann gilt:

$$\begin{aligned}
& [\![\forall X. \forall Y. (X \to Y) \to (F(X) \to F(Y)]\!]_\bullet(\mathrm{fmap}, \mathrm{fmap}) \\
\Longrightarrow \quad & [\![(X \to Y) \to (F(X) \to F(Y)]\!]_{\{X \mapsto |f_1\rangle, Y \mapsto |g_2\rangle\}}(\mathrm{fmap}\ A\ B, \mathrm{fmap}\ A'\ B') \\
\Longrightarrow \quad & [\![F(X) \to F(Y)]\!]_{\{X \mapsto |f_1\rangle, Y \mapsto |g_2\rangle\}}(\mathrm{fmap}\ A\ B\ g_1, \mathrm{fmap}\ A'\ B'\ f_2) \\
\Longrightarrow \quad & [\![F(Y)]\!]_{\{X \mapsto |f_1\rangle, Y \mapsto |g_2\rangle\}}(\mathrm{fmap}\ A\ B\ g_1\ p, \mathrm{fmap}\ A'\ B'\ f_2\ ((f_1)_*\ p))
\end{aligned}$$

Das zweite Funktoraxiom

Sei $F(X)$ ein über X parametrisierter Typ, (d.h. $X \vdash F(X)$) und gelte, dass X in $F(X)$ nur kovariant vorkommt.

Lemma

Für alle $A, B \in$ *Types*$_\bullet$ *und* $f : A \to B$ *gibt es eine Funktion* $f_* : F(A) \to F(B)$, *sodass für alle* $a : F(A)$ *und* $b : F(B)$ *gilt:*

$$[\![F(X)]\!]_{\{X \mapsto |f\rangle\}}(a, b) \iff f_*(a) \cong b$$

Seien $\mathrm{fmap} : \forall X.\, \forall Y.\, (X \to Y) \to (F(X) \to F(Y))$, Typen A, A', B, B' und Funktionen $f_1 : A \to A'$, $f_2 : A' \to B'$, $g_1 : A \to B$, $g_2 : B \to B'$ mit $f_2 \circ f_1 \cong g_2 \circ g_1$ sowie $p : F(A)$ gegeben. Dann gilt:

$$\begin{aligned}
& [\![\forall X.\, \forall Y.\, (X \to Y) \to (F(X) \to F(Y)]\!]_\bullet(\mathrm{fmap}, \mathrm{fmap}) \\
\implies\; & [\![(X \to Y) \to (F(X) \to F(Y)]\!]_{\{X \mapsto |f_1\rangle, Y \mapsto |g_2\rangle\}}(\mathrm{fmap}\ A\ B, \mathrm{fmap}\ A'\ B') \\
\implies\; & [\![F(X) \to F(Y)]\!]_{\{X \mapsto |f_1\rangle, Y \mapsto |g_2\rangle\}}(\mathrm{fmap}\ A\ B\ g_1, \mathrm{fmap}\ A'\ B'\ f_2) \\
\implies\; & [\![F(Y)]\!]_{\{X \mapsto |f_1\rangle, Y \mapsto |g_2\rangle\}}(\mathrm{fmap}\ A\ B\ g_1\ p, \mathrm{fmap}\ A'\ B'\ f_2\ ((f_1)_*\ p)) \\
\iff\; & (g_2)_*\ (\mathrm{fmap}\ A\ B\ g_1\ p) \cong \mathrm{fmap}\ A'\ B'\ f_2\ ((f_1)_*\ p)
\end{aligned}$$

Das zweite Funktoraxiom

Seien $\text{fmap} : \forall X.\, \forall Y.\, (X \to Y) \to (F(X) \to F(Y))$, Typen A, A', B, B' und Funktionen $f_1 : A \to A'$, $f_2 : A' \to B'$, $g_1 : A \to B$, $g_2 : B \to B'$ mit $f_2 \circ f_1 \cong g_2 \circ g_1$ sowie $p : F(A)$ gegeben. Dann gilt:

$$(g_2)_* \circ \text{fmap}\ A\, B\, g_1 \cong \text{fmap}\ A'\, B'\, f_2 \circ (f_1)_*$$

Das zweite Funktoraxiom

Seien $\mathsf{fmap} : \forall X.\, \forall Y.\, (X \to Y) \to (F(X) \to F(Y))$, Typen A, A', B, B' und Funktionen $f_1 : A \to A'$, $f_2 : A' \to B'$, $g_1 : A \to B$, $g_2 : B \to B'$ mit $f_2 \circ f_1 \cong g_2 \circ g_1$ sowie $p : F(A)$ gegeben. Dann gilt:

$$(g_2)_* \circ \mathsf{fmap}\ A\, B\, g_1 \cong \mathsf{fmap}\ A'\, B'\, f_2 \circ (f_1)_*$$

Das zweite Funktoraxiom

Seien $\mathsf{fmap} : \forall X.\, \forall Y.\, (X \to Y) \to (F(X) \to F(Y))$, Typen A, A', B, B' und Funktionen $f_1 : A \to A'$, $f_2 : A' \to B'$, $g_1 : A \to B$, $g_2 : B \to B'$ mit $f_2 \circ f_1 \cong g_2 \circ g_1$ sowie $p : F(A)$ gegeben. Dann gilt:

$$(g_2)_* \circ \mathsf{fmap}\ A\, B\, g_1 \cong \mathsf{fmap}\ A'\, B'\, f_2 \circ (f_1)_*$$

Angenommen, für alle Typen A gilt $\mathsf{fmap}\ A\, A\, (\mathsf{id}\ A) \cong \mathsf{id}\ F(A)$.

Das zweite Funktoraxiom

Seien $\mathrm{fmap} : \forall X.\, \forall Y.\, (X \to Y) \to (F(X) \to F(Y))$, Typen A, A', B, B' und Funktionen $f_1 : A \to A'$, $f_2 : A' \to B'$, $g_1 : A \to B$, $g_2 : B \to B'$ mit $f_2 \circ f_1 \cong g_2 \circ g_1$ sowie $p : F(A)$ gegeben. Dann gilt:

$$(g_2)_* \circ \mathrm{fmap}\ A\, B\, g_1 \cong \mathrm{fmap}\ A'\, B'\, f_2 \circ (f_1)_*$$

Angenommen, für alle Typen A gilt $\mathrm{fmap}\ A\, A\, (\mathrm{id}\ A) \cong \mathrm{id}\ F(A)$.
Dann gilt für alle Funktionen $f : A \to B$:

Das zweite Funktoraxiom

Seien $\mathsf{fmap} : \forall X.\, \forall Y.\, (X \to Y) \to (F(X) \to F(Y))$, Typen A, A', B, B' und Funktionen $f_1 : A \to A'$, $f_2 : A' \to B'$, $g_1 : A \to B$, $g_2 : B \to B'$ mit $f_2 \circ f_1 \cong g_2 \circ g_1$ sowie $p : F(A)$ gegeben. Dann gilt:

$$(g_2)_* \circ \mathsf{fmap}\ A\, B\, g_1 \cong \mathsf{fmap}\ A'\, B'\, f_2 \circ (f_1)_*$$

Angenommen, für alle Typen A gilt $\mathsf{fmap}\ A\, A\, (\mathsf{id}\ A) \cong \mathsf{id}\ F(A)$.
Dann gilt für alle Funktionen $f : A \to B$:

$$\mathsf{fmap}\ A\, B\, f \cong (\mathsf{id}\ B)_* \circ \mathsf{fmap}\ A\, B\, f \cong \mathsf{fmap}\ B\, B\, (\mathsf{id}\ B) \circ f_* \cong f_*$$

Für Typen C, D, E und Funktionen $k : C \to D$ und $h : D \to E$ folgt:

Das zweite Funktoraxiom

Seien $\mathsf{fmap} : \forall X.\, \forall Y.\, (X \to Y) \to (F(X) \to F(Y))$, Typen A, A', B, B' und Funktionen $f_1 : A \to A'$, $f_2 : A' \to B'$, $g_1 : A \to B$, $g_2 : B \to B'$ mit $f_2 \circ f_1 \cong g_2 \circ g_1$ sowie $p : F(A)$ gegeben. Dann gilt:

$$(g_2)_* \circ \mathsf{fmap}\ A\, B\, g_1 \cong \mathsf{fmap}\ A'\, B'\, f_2 \circ (f_1)_*$$

Angenommen, für alle Typen A gilt $\mathsf{fmap}\ A\, A\, (\mathsf{id}\ A) \cong \mathsf{id}\ F(A)$.
Dann gilt für alle Funktionen $f : A \to B$:

$$\mathsf{fmap}\ A\, B\, f \cong (\mathsf{id}\ B)_* \circ \mathsf{fmap}\ A\, B\, f \cong \mathsf{fmap}\ B\, B\, (\mathsf{id}\ B) \circ f_* \cong f_*$$

Für Typen C, D, E und Funktionen $k : C \to D$ und $h : D \to E$ folgt:

$$\mathsf{fmap}\ D\, E\, h \circ \mathsf{fmap}\ C\, D\, k$$

Das zweite Funktoraxiom

Seien $\mathsf{fmap} : \forall X. \forall Y. (X \to Y) \to (F(X) \to F(Y))$, Typen A, A', B, B' und Funktionen $f_1 : A \to A'$, $f_2 : A' \to B'$, $g_1 : A \to B$, $g_2 : B \to B'$ mit $f_2 \circ f_1 \cong g_2 \circ g_1$ sowie $p : F(A)$ gegeben. Dann gilt:

$$(g_2)_* \circ \mathsf{fmap}\ A\, B\, g_1 \cong \mathsf{fmap}\ A'\, B'\, f_2 \circ (f_1)_*$$

Angenommen, für alle Typen A gilt $\mathsf{fmap}\ A\, A\, (\mathsf{id}\ A) \cong \mathsf{id}\ F(A)$.
Dann gilt für alle Funktionen $f : A \to B$:

$$\mathsf{fmap}\ A\, B\, f \cong (\mathsf{id}\ B)_* \circ \mathsf{fmap}\ A\, B\, f \cong \mathsf{fmap}\ B\, B\, (\mathsf{id}\ B) \circ f_* \cong f_*$$

Für Typen C, D, E und Funktionen $k : C \to D$ und $h : D \to E$ folgt:

$$\mathsf{fmap}\ D\, E\, h \circ \mathsf{fmap}\ C\, D\, k \quad \cong \quad h_* \circ \mathsf{fmap}\ C\, D\, k$$

Das zweite Funktoraxiom

Seien $\mathsf{fmap} : \forall X.\, \forall Y.\, (X \to Y) \to (F(X) \to F(Y))$, Typen A, A', B, B' und Funktionen $f_1 : A \to A'$, $f_2 : A' \to B'$, $g_1 : A \to B$, $g_2 : B \to B'$ mit $f_2 \circ f_1 \cong g_2 \circ g_1$ sowie $p : F(A)$ gegeben. Dann gilt:

$$(g_2)_* \circ \mathsf{fmap}\ A\, B\, g_1 \cong \mathsf{fmap}\ A'\, B'\, f_2 \circ (f_1)_*$$

Angenommen, für alle Typen A gilt $\mathsf{fmap}\ A\, A\, (\mathsf{id}\ A) \cong \mathsf{id}\ F(A)$.
Dann gilt für alle Funktionen $f : A \to B$:

$$\mathsf{fmap}\ A\, B\, f \cong (\mathsf{id}\ B)_* \circ \mathsf{fmap}\ A\, B\, f \cong \mathsf{fmap}\ B\, B\, (\mathsf{id}\ B) \circ f_* \cong f_*$$

Für Typen C, D, E und Funktionen $k : C \to D$ und $h : D \to E$ folgt:

$$\begin{aligned} \mathsf{fmap}\ D\, E\, h \circ \mathsf{fmap}\ C\, D\, k &\cong h_* \circ \mathsf{fmap}\ C\, D\, k \\ &\cong \mathsf{fmap}\ E\, E\, (\mathsf{id}\ E) \circ (h \circ k)_* \end{aligned}$$

Das zweite Funktoraxiom

Seien $\mathsf{fmap} : \forall X.\, \forall Y.\, (X \to Y) \to (F(X) \to F(Y))$, Typen A, A', B, B' und Funktionen $f_1 : A \to A'$, $f_2 : A' \to B'$, $g_1 : A \to B$, $g_2 : B \to B'$ mit $f_2 \circ f_1 \cong g_2 \circ g_1$ sowie $p : F(A)$ gegeben. Dann gilt:

$$(g_2)_* \circ \mathsf{fmap}\ A\, B\, g_1 \cong \mathsf{fmap}\ A'\, B'\, f_2 \circ (f_1)_*$$

Angenommen, für alle Typen A gilt $\mathsf{fmap}\ A\, A\, (\mathsf{id}\ A) \cong \mathsf{id}\ F(A)$.
Dann gilt für alle Funktionen $f : A \to B$:

$$\mathsf{fmap}\ A\, B\, f \cong (\mathsf{id}\ B)_* \circ \mathsf{fmap}\ A\, B\, f \cong \mathsf{fmap}\ B\, B\, (\mathsf{id}\ B) \circ f_* \cong f_*$$

Für Typen C, D, E und Funktionen $k : C \to D$ und $h : D \to E$ folgt:

$$\begin{aligned} \mathsf{fmap}\ D\, E\, h \circ \mathsf{fmap}\ C\, D\, k &\cong h_* \circ \mathsf{fmap}\ C\, D\, k \\ &\cong \mathsf{fmap}\ E\, E\, (\mathsf{id}\ E) \circ (h \circ k)_* \\ &\cong (h \circ k)_* \end{aligned}$$

Das zweite Funktoraxiom

Seien $\mathsf{fmap} : \forall X. \forall Y. (X \to Y) \to (F(X) \to F(Y))$, Typen A, A', B, B' und Funktionen $f_1 : A \to A'$, $f_2 : A' \to B'$, $g_1 : A \to B$, $g_2 : B \to B'$ mit $f_2 \circ f_1 \cong g_2 \circ g_1$ sowie $p : F(A)$ gegeben. Dann gilt:

$$(g_2)_* \circ \mathsf{fmap}\ A\, B\, g_1 \cong \mathsf{fmap}\ A'\, B'\, f_2 \circ (f_1)_*$$

Angenommen, für alle Typen A gilt $\mathsf{fmap}\ A\, A\, (\mathsf{id}\ A) \cong \mathsf{id}\ F(A)$.
Dann gilt für alle Funktionen $f : A \to B$:

$$\mathsf{fmap}\ A\, B\, f \cong (\mathsf{id}\ B)_* \circ \mathsf{fmap}\ A\, B\, f \cong \mathsf{fmap}\ B\, B\, (\mathsf{id}\ B) \circ f_* \cong f_*$$

Für Typen C, D, E und Funktionen $k : C \to D$ und $h : D \to E$ folgt:

$$\begin{aligned}
\mathsf{fmap}\ D\, E\, h \circ \mathsf{fmap}\ C\, D\, k &\cong h_* \circ \mathsf{fmap}\ C\, D\, k \\
&\cong \mathsf{fmap}\ E\, E\, (\mathsf{id}\ E) \circ (h \circ k)_* \\
&\cong (h \circ k)_* \\
&\cong \mathsf{fmap}\ C\, E\, (h \circ k)
\end{aligned}$$

Das zweite Funktoraxiom

Seien $\mathsf{fmap} : \forall X.\, \forall Y.\, (X \to Y) \to (F(X) \to F(Y))$, Typen A, A', B, B' und Funktionen $f_1 : A \to A'$, $f_2 : A' \to B'$, $g_1 : A \to B$, $g_2 : B \to B'$ mit $f_2 \circ f_1 \cong g_2 \circ g_1$ sowie $p : F(A)$ gegeben. Dann gilt:

$$(g_2)_* \circ \mathsf{fmap}\ A\, B\, g_1 \cong \mathsf{fmap}\ A'\, B'\, f_2 \circ (f_1)_*$$

Angenommen, für alle Typen A gilt $\mathsf{fmap}\ A\, A\, (\mathsf{id}\ A) \cong \mathsf{id}\ F(A)$. Dann gilt für alle Funktionen $f : A \to B$:

$$\mathsf{fmap}\ A\, B\, f \cong (\mathsf{id}\ B)_* \circ \mathsf{fmap}\ A\, B\, f \cong \mathsf{fmap}\ B\, B\, (\mathsf{id}\ B) \circ f_* \cong f_*$$

Für Typen C, D, E und Funktionen $k : C \to D$ und $h : D \to E$ folgt:

$$\begin{aligned}
\mathsf{fmap}\ D\, E\, h \circ \mathsf{fmap}\ C\, D\, k &\cong h_* \circ \mathsf{fmap}\ C\, D\, k \\
&\cong \mathsf{fmap}\ E\, E\, (\mathsf{id}\ E) \circ (h \circ k)_* \\
&\cong (h \circ k)_* \\
&\cong \mathsf{fmap}\ C\, E\, (h \circ k)
\end{aligned}$$

Fazit: Das zweite Funktoraxiom ist kostenlos!

Sortierfunktionen

Angenommen, wir haben $s : \forall X.\, (X \to X \to \mathsf{Bool}) \to \mathsf{List}(X) \to \mathsf{List}(X)$, $c' : B' \to B' \to \mathsf{Bool}$, $f : B \to B'$ und $bs : \mathsf{List}(B)$.
Definiere $c := (\lambda x{:}B.\, \lambda y{:}B.\, c\,(f\,x)\,(f\,y)) : B \to B \to \mathsf{Bool}$. Dann gilt

$$[\![\forall X.\, (X \to X \to \mathsf{Bool}) \to \mathsf{List}(X) \to \mathsf{List}(X)]\!]_{\bullet}(s, s)$$

Sortierfunktionen

Angenommen, wir haben $s : \forall \mathsf{X}.\,(\mathsf{X} \to \mathsf{X} \to \mathsf{Bool}) \to \mathsf{List}(X) \to \mathsf{List}(X)$, $c' : B' \to B' \to \mathsf{Bool}$, $f : B \to B'$ und $bs : \mathsf{List}(B)$.
Definiere $c := (\lambda x{:}B.\, \lambda y{:}B.\, c\,(f\,x)\,(f\,y)) : B \to B \to \mathsf{Bool}$. Dann gilt

$$
\begin{array}{ll}
 & [\![\forall \mathsf{X}.\,(\mathsf{X} \to \mathsf{X} \to \mathsf{Bool}) \to \mathsf{List}(X) \to \mathsf{List}(X)]\!]_\bullet(s, s) \\
\iff & \text{für alle } S, S', \mathcal{S} : S \Leftrightarrow S' \text{ gilt} \\
 & [\![(\mathsf{X} \to \mathsf{X} \to \mathsf{Bool}) \to \mathsf{List}(X) \to \mathsf{List}(X)]\!]_{\{\mathsf{X} \mapsto \mathcal{S}\}}(s\,S, s\,S')
\end{array}
$$

Sortierfunktionen

Angenommen, wir haben $s : \forall \mathsf{X}.\, (\mathsf{X} \to \mathsf{X} \to \mathsf{Bool}) \to \mathsf{List}(X) \to \mathsf{List}(X)$,
$c' : B' \to B' \to \mathsf{Bool}$, $f : B \to B'$ und $bs : \mathsf{List}(B)$.
Definiere $c := (\lambda x{:}B.\, \lambda y{:}B.\, c\,(f\,x)\,(f\,y)) : B \to B \to \mathsf{Bool}$. Dann gilt

$$\begin{array}{rl}
& [\![\forall \mathsf{X}.\, (\mathsf{X} \to \mathsf{X} \to \mathsf{Bool}) \to \mathsf{List}(X) \to \mathsf{List}(X)]\!]_{\bullet}(s, s) \\
\iff & \text{für alle } S, S', \mathcal{S} : S \Leftrightarrow S' \text{ gilt} \\
& [\![(\mathsf{X} \to \mathsf{X} \to \mathsf{Bool}) \to \mathsf{List}(X) \to \mathsf{List}(X)]\!]_{\{\mathsf{X} \mapsto \mathcal{S}\}}(s\,S, s\,S') \\
\implies & [\![(\mathsf{X} \to \mathsf{X} \to \mathsf{Bool}) \to \mathsf{List}(X) \to \mathsf{List}(X)]\!]_{\{\mathsf{X} \mapsto |f\rangle\}}(s\,B, s\,B')
\end{array}$$

Sortierfunktionen

Angenommen, wir haben $s : \forall X.\, (X \to X \to \mathsf{Bool}) \to \mathsf{List}(X) \to \mathsf{List}(X)$, $c' : B' \to B' \to \mathsf{Bool}$, $f : B \to B'$ und $bs : \mathsf{List}(B)$.
Definiere $c := (\lambda x{:}B.\, \lambda y{:}B.\, c\,(f\,x)\,(f\,y)) : B \to B \to \mathsf{Bool}$. Dann gilt

$$\begin{array}{ll}
& [\![\forall X.\, (X \to X \to \mathsf{Bool}) \to \mathsf{List}(X) \to \mathsf{List}(X)]\!]_{\bullet}(s, s) \\
\iff & \text{für alle } S, S', \mathcal{S} : S \Leftrightarrow S' \text{ gilt} \\
& [\![(X \to X \to \mathsf{Bool}) \to \mathsf{List}(X) \to \mathsf{List}(X)]\!]_{\{X \mapsto \mathcal{S}\}}(s\,S, s\,S') \\
\implies & [\![(X \to X \to \mathsf{Bool}) \to \mathsf{List}(X) \to \mathsf{List}(X)]\!]_{\{X \mapsto \langle f \rangle\}}(s\,B, s\,B') \\
\iff & \text{für alle } g : A \to A \to \mathsf{Bool},\ g' : B' \to B' \to \mathsf{Bool} \\
& \text{mit } [\![X \to X \to \mathsf{Bool}]\!]_{\{X \mapsto \langle f \rangle\}}(g, g') \\
& \text{gilt } [\![\mathsf{List}(X) \to \mathsf{List}(X)]\!]_{\{X \mapsto \langle f \rangle\}}(s\,B\,g, s\,B'\,g')
\end{array}$$

Sortierfunktionen

Angenommen, wir haben $s : \forall X.\,(X \to X \to \mathsf{Bool}) \to \mathsf{List}(X) \to \mathsf{List}(X)$, $c' : B' \to B' \to \mathsf{Bool}$, $f : B \to B'$ und $bs : \mathsf{List}(B)$.
Definiere $c := (\lambda x{:}B.\,\lambda y{:}B.\,c\,(f\,x)\,(f\,y)) : B \to B \to \mathsf{Bool}$. Dann gilt

$$
\begin{array}{cl}
 & [\![\forall X.\,(X \to X \to \mathsf{Bool}) \to \mathsf{List}(X) \to \mathsf{List}(X)]\!]_{\bullet}(s, s) \\
\iff & \text{für alle } S, S', \mathcal{S} : S \Leftrightarrow S' \text{ gilt} \\
 & [\![(X \to X \to \mathsf{Bool}) \to \mathsf{List}(X) \to \mathsf{List}(X)]\!]_{\{X \mapsto \mathcal{S}\}}(s\,S, s\,S') \\
\implies & [\![(X \to X \to \mathsf{Bool}) \to \mathsf{List}(X) \to \mathsf{List}(X)]\!]_{\{X \mapsto \langle f \rangle\}}(s\,B, s\,B') \\
\iff & \text{für alle } g : A \to A \to \mathsf{Bool},\ g' : B' \to B' \to \mathsf{Bool} \\
 & \text{mit } [\![X \to X \to \mathsf{Bool}]\!]_{\{X \mapsto \langle f \rangle\}}(g, g') \\
 & \text{gilt } [\![\mathsf{List}(X) \to \mathsf{List}(X)]\!]_{\{X \mapsto \langle f \rangle\}}(s\,B\,g, s\,B'\,g') \\
\implies & [\![\mathsf{List}(X) \to \mathsf{List}(X)]\!]_{\{X \mapsto \langle f \rangle\}}(s\,B\,c, s\,B'\,c')
\end{array}
$$

Sortierfunktionen

Angenommen, wir haben $s : \forall X.\, (X \to X \to \mathsf{Bool}) \to \mathsf{List}(X) \to \mathsf{List}(X)$, $c' : B' \to B' \to \mathsf{Bool}$, $f : B \to B'$ und $bs : \mathsf{List}(B)$.
Definiere $c := (\lambda x{:}B.\, \lambda y{:}B.\, c\,(f\,x)\,(f\,y)) : B \to B \to \mathsf{Bool}$. Dann gilt

$$\begin{array}{cl}
 & [\![\forall X.\, (X \to X \to \mathsf{Bool}) \to \mathsf{List}(X) \to \mathsf{List}(X)]\!]_{\bullet}(s, s) \\
\iff & \text{für alle } S, S', \mathcal{S} : S \Leftrightarrow S' \text{ gilt} \\
 & [\![(X \to X \to \mathsf{Bool}) \to \mathsf{List}(X) \to \mathsf{List}(X)]\!]_{\{X \mapsto \mathcal{S}\}}(s\,S, s\,S') \\
\implies & [\![(X \to X \to \mathsf{Bool}) \to \mathsf{List}(X) \to \mathsf{List}(X)]\!]_{\{X \mapsto |f\rangle\}}(s\,B, s\,B') \\
\iff & \text{für alle } g : A \to A \to \mathsf{Bool},\ g' : B' \to B' \to \mathsf{Bool} \\
 & \text{mit } [\![X \to X \to \mathsf{Bool}]\!]_{\{X \mapsto |f\rangle\}}(g, g') \\
 & \text{gilt } [\![\mathsf{List}(X) \to \mathsf{List}(X)]\!]_{\{X \mapsto |f\rangle\}}(s\,B\,g, s\,B'\,g') \\
\implies & [\![\mathsf{List}(X) \to \mathsf{List}(X)]\!]_{\{X \mapsto |f\rangle\}}(s\,B\,c, s\,B'\,c') \\
\iff & \text{für alle } xs : \mathsf{List}(B),\ ys : \mathsf{List}(B') \text{ mit } [\![\mathsf{List}(X)]\!]_{\{X \mapsto |f\rangle\}}(xs, ys) \\
 & \text{gilt } [\![\mathsf{List}(X)]\!]_{\{X \mapsto |f\rangle\}}(s\,B\,c\,xs, s\,B'\,c'\,ys)
\end{array}$$

Sortierfunktionen

Angenommen, wir haben $s : \forall X.\,(X \to X \to \mathsf{Bool}) \to \mathsf{List}(X) \to \mathsf{List}(X)$, $c' : B' \to B' \to \mathsf{Bool}$, $f : B \to B'$ und $bs : \mathsf{List}(B)$.
Definiere $c := (\lambda x{:}B.\,\lambda y{:}B.\,c\,(f\,x)\,(f\,y)) : B \to B \to \mathsf{Bool}$. Dann gilt

$$
\begin{array}{rl}
 & [\![\forall X.\,(X \to X \to \mathsf{Bool}) \to \mathsf{List}(X) \to \mathsf{List}(X)]\!]_\bullet(s, s) \\
\iff & \text{für alle } S, S', \mathcal{S} : S \Leftrightarrow S' \text{ gilt} \\
 & [\![(X \to X \to \mathsf{Bool}) \to \mathsf{List}(X) \to \mathsf{List}(X)]\!]_{\{X \mapsto \mathcal{S}\}}(s\,S, s\,S') \\
\implies & [\![(X \to X \to \mathsf{Bool}) \to \mathsf{List}(X) \to \mathsf{List}(X)]\!]_{\{X \mapsto \langle f \rangle\}}(s\,B, s\,B') \\
\iff & \text{für alle } g : A \to A \to \mathsf{Bool},\ g' : B' \to B' \to \mathsf{Bool} \\
 & \text{mit } [\![X \to X \to \mathsf{Bool}]\!]_{\{X \mapsto \langle f \rangle\}}(g, g') \\
 & \text{gilt } [\![\mathsf{List}(X) \to \mathsf{List}(X)]\!]_{\{X \mapsto \langle f \rangle\}}(s\,B\,g, s\,B'\,g') \\
\implies & [\![\mathsf{List}(X) \to \mathsf{List}(X)]\!]_{\{X \mapsto \langle f \rangle\}}(s\,B\,c, s\,B'\,c') \\
\iff & \text{für alle } xs : \mathsf{List}(B),\ ys : \mathsf{List}(B') \text{ mit } [\![\mathsf{List}(X)]\!]_{\{X \mapsto \langle f \rangle\}}(xs, ys) \\
 & \text{gilt } [\![\mathsf{List}(X)]\!]_{\{X \mapsto \langle f \rangle\}}(s\,B\,c\,xs, s\,B'\,c'\,ys) \\
\implies & [\![\mathsf{List}(X)]\!]_{\{X \mapsto \langle f \rangle\}}(s\,B\,c\,bs, s\,B'\,c'\,(\mathsf{map}\,B\,B'\,f\,bs))
\end{array}
$$

Sortierfunktionen

Angenommen, wir haben $s : \forall \mathsf{X}.\, (\mathsf{X} \to \mathsf{X} \to \mathsf{Bool}) \to \mathsf{List}(\mathsf{X}) \to \mathsf{List}(\mathsf{X})$, $c' : B' \to B' \to \mathsf{Bool}$, $f : B \to B'$ und $bs : \mathsf{List}(B)$.
Definiere $c := (\lambda x{:}B.\, \lambda y{:}B.\, c\,(f\,x)\,(f\,y)) : B \to B \to \mathsf{Bool}$. Dann gilt

$$
\begin{array}{ll}
& [\![\forall \mathsf{X}.\, (\mathsf{X} \to \mathsf{X} \to \mathsf{Bool}) \to \mathsf{List}(\mathsf{X}) \to \mathsf{List}(\mathsf{X})]\!]_{\bullet}(s, s) \\
\iff & \text{für alle } S, S', \mathcal{S} : S \Leftrightarrow S' \text{ gilt} \\
& [\![(\mathsf{X} \to \mathsf{X} \to \mathsf{Bool}) \to \mathsf{List}(\mathsf{X}) \to \mathsf{List}(\mathsf{X})]\!]_{\{\mathsf{X} \mapsto \mathcal{S}\}}(s\,S, s\,S') \\
\implies & [\![(\mathsf{X} \to \mathsf{X} \to \mathsf{Bool}) \to \mathsf{List}(\mathsf{X}) \to \mathsf{List}(\mathsf{X})]\!]_{\{\mathsf{X} \mapsto \langle f \rangle\}}(s\,B, s\,B') \\
\iff & \text{für alle } g : A \to A \to \mathsf{Bool},\ g' : B' \to B' \to \mathsf{Bool} \\
& \text{mit } [\![\mathsf{X} \to \mathsf{X} \to \mathsf{Bool}]\!]_{\{\mathsf{X} \mapsto \langle f \rangle\}}(g, g') \\
& \text{gilt } [\![\mathsf{List}(\mathsf{X}) \to \mathsf{List}(\mathsf{X})]\!]_{\{\mathsf{X} \mapsto \langle f \rangle\}}(s\,B\,g, s\,B'\,g') \\
\implies & [\![\mathsf{List}(\mathsf{X}) \to \mathsf{List}(\mathsf{X})]\!]_{\{\mathsf{X} \mapsto \langle f \rangle\}}(s\,B\,c, s\,B'\,c') \\
\iff & \text{für alle } xs : \mathsf{List}(B),\ ys : \mathsf{List}(B') \text{ mit } [\![\mathsf{List}(\mathsf{X})]\!]_{\{\mathsf{X} \mapsto \langle f \rangle\}}(xs, ys) \\
& \text{gilt } [\![\mathsf{List}(\mathsf{X})]\!]_{\{\mathsf{X} \mapsto \langle f \rangle\}}(s\,B\,c\,xs, s\,B'\,c'\,ys) \\
\implies & [\![\mathsf{List}(\mathsf{X})]\!]_{\{\mathsf{X} \mapsto \langle f \rangle\}}(s\,B\,c\,bs, s\,B'\,c'\,(\mathsf{map}\,B\,B'\,f\,bs)) \\
\iff & \mathsf{map}\,B\,B'\,f\,(s\,B\,c\,bs) \cong s\,B'\,c'\,(\mathsf{map}\,B\,B'\,f\,bs)
\end{array}
$$

Natürlichkeit

Seien $F(X)$ und $G(X)$ über X parametrisierte Typen (wie eben) und $eta : \forall Y.\, F(Y) \to G(Y)$ eine Funktion.

Natürlichkeit

Seien $F(X)$ und $G(X)$ über X parametrisierte Typen (wie eben) und $eta : \forall Y.\, F(Y) \to G(Y)$ eine Funktion.
Beispiele in Haskell: maybeToList, reverse, maybeHead

Natürlichkeit

Seien $F(X)$ und $G(X)$ über X parametrisierte Typen (wie eben) und $eta : \forall Y.\, F(Y) \to G(Y)$ eine Funktion.
Beispiele in Haskell: maybeToList, reverse, maybeHead
Für alle Typen A, A' und Funktionen $f : A \to A'$ folgt aus Parametrizität:

$$[\![\forall Y.\, F(Y) \to G(Y)]\!]_{\bullet}(eta, eta)$$

Natürlichkeit

Seien $F(X)$ und $G(X)$ über X parametrisierte Typen (wie eben) und $eta : \forall Y.\, F(Y) \to G(Y)$ eine Funktion.
Beispiele in Haskell: maybeToList, reverse, maybeHead
Für alle Typen A, A' und Funktionen $f : A \to A'$ folgt aus Parametrizität:

$$\begin{array}{ll} & [\![\forall Y.\, F(Y) \to G(Y)]\!]_{\bullet}(eta, eta) \\ \implies & [\![F(Y) \to G(Y)]\!]_{\{Y \mapsto \langle f \rangle\}}(eta\,A, eta\,A') \end{array}$$

Natürlichkeit

Seien $F(X)$ und $G(X)$ über X parametrisierte Typen (wie eben) und $eta : \forall Y.\, F(Y) \to G(Y)$ eine Funktion.
Beispiele in Haskell: maybeToList, reverse, maybeHead
Für alle Typen A, A' und Funktionen $f : A \to A'$ folgt aus Parametrizität:

$$
\begin{array}{ll}
 & [\![\forall Y.\, F(Y) \to G(Y)]\!]_{\bullet}(eta, eta) \\
\Longrightarrow & [\![F(Y) \to G(Y)]\!]_{\{Y \mapsto \langle f \rangle\}}(eta\,A, eta\,A') \\
\Longleftrightarrow & \text{für alle } a : A \text{ und } a' : A' \text{ mit } [\![F(Y)]\!]_{\{Y \mapsto \langle f \rangle\}}(a, a') \\
 & \text{gilt } [\![G(Y)]\!]_{\{Y \mapsto \langle f \rangle\}}(eta\,A\,a, eta\,A'\,a')
\end{array}
$$

Natürlichkeit

Seien $F(X)$ und $G(X)$ über X parametrisierte Typen (wie eben) und $eta : \forall Y.\, F(Y) \to G(Y)$ eine Funktion.
Beispiele in Haskell: maybeToList, reverse, maybeHead
Für alle Typen A, A' und Funktionen $f : A \to A'$ folgt aus Parametrizität:

$$\begin{array}{ll}
& [\![\forall Y.\, F(Y) \to G(Y)]\!]_\bullet(eta, eta) \\
\Longrightarrow & [\![F(Y) \to G(Y)]\!]_{\{Y \mapsto \langle f \rangle\}}(eta\, A, eta\, A') \\
\Longleftrightarrow & \text{für alle } a : A \text{ und } a' : A' \text{ mit } [\![F(Y)]\!]_{\{Y \mapsto \langle f \rangle\}}(a, a') \\
& \text{gilt } [\![G(Y)]\!]_{\{Y \mapsto \langle f \rangle\}}(eta\, A\, a, eta\, A'\, a') \\
\Longleftrightarrow & \text{für alle } a : A \text{ und } a' : A' \text{ mit } f_*\, a \cong a' \\
& \text{gilt } f_*\, (eta\, A\, a) \cong eta\, A'\, a'
\end{array}$$

Natürlichkeit

Seien $F(X)$ und $G(X)$ über X parametrisierte Typen (wie eben) und $eta : \forall Y.\, F(Y) \to G(Y)$ eine Funktion.
Beispiele in Haskell: maybeToList, reverse, maybeHead
Für alle Typen A, A' und Funktionen $f : A \to A'$ folgt aus Parametrizität:

$$
\begin{array}{ll}
 & [\![\forall Y.\, F(Y) \to G(Y)]\!]_{\bullet}(eta, eta) \\
\Longrightarrow & [\![F(Y) \to G(Y)]\!]_{\{Y \mapsto |f\rangle\}}(eta\, A, eta\, A') \\
\Longleftrightarrow & \text{für alle } a : A \text{ und } a' : A' \text{ mit } [\![F(Y)]\!]_{\{Y \mapsto |f\rangle\}}(a, a') \\
 & \text{gilt } [\![G(Y)]\!]_{\{Y \mapsto |f\rangle\}}(eta\, A\, a, eta\, A'\, a') \\
\Longleftrightarrow & \text{für alle } a : A \text{ und } a' : A' \text{ mit } f_*\, a \cong a' \\
 & \text{gilt } f_*\, (eta\, A\, a) \cong eta\, A'\, a' \\
\Longleftrightarrow & f_* \circ eta\, A \cong eta\, A' \circ f_*
\end{array}
$$

Natürlichkeit

Seien $F(X)$ und $G(X)$ über X parametrisierte Typen (wie eben) und $eta : \forall Y.\, F(Y) \to G(Y)$ eine Funktion.
Beispiele in Haskell: maybeToList, reverse, maybeHead
Für alle Typen A, A' und Funktionen $f : A \to A'$ folgt aus Parametrizität:

$$
\begin{array}{ll}
 & [\![\forall Y.\, F(Y) \to G(Y)]\!]_{\bullet}(eta, eta) \\
\Longrightarrow & [\![F(Y) \to G(Y)]\!]_{\{Y \mapsto \langle f \rangle\}}(eta\, A, eta\, A') \\
\Longleftrightarrow & \text{für alle } a : A \text{ und } a' : A' \text{ mit } [\![F(Y)]\!]_{\{Y \mapsto \langle f \rangle\}}(a, a') \\
 & \text{gilt } [\![G(Y)]\!]_{\{Y \mapsto \langle f \rangle\}}(eta\, A\, a, eta\, A'\, a') \\
\Longleftrightarrow & \text{für alle } a : A \text{ und } a' : A' \text{ mit } f_*\, a \cong a' \\
 & \text{gilt } f_*\, (eta\, A\, a) \cong eta\, A'\, a' \\
\Longleftrightarrow & f_* \circ eta\, A \cong eta\, A' \circ f_*
\end{array}
$$

Fazit: Natürlichkeit ist kostenlos!

Beweis von Parametrizität: Interpretation von Termen

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

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

von Termen.

Beweis von Parametrizität: 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 Interpretationsabbildung

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

rekursiv definiert durch

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

Beweis von 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

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

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

Beweis von 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* $\Delta; \Gamma \models t : \tau$

Beweis von 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

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

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

Satz (Parametrizität')

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

Kurz: Interpretationen in relatierten Umgebungen sind relatiert.

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

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

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

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

Links und Quellen

- `http://twitter.com/parametricity`
- Free Theorems Web-UI: `http://www-ps.iai.uni-bonn.de/cgi-bin/free-theorems-webui.cgi`
- 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
- Edward Kmett, *The free theorem for fmap*, `https://www.schoolofhaskell.com/user/edwardk/snippets/fmap`