

Polymorpher Lambda-Kalkül [Girard 1972, Reynolds 1974]

Typen: $\tau := \alpha \mid \tau \rightarrow \tau \mid \forall \alpha. \tau$

Terme: $t := x \mid \lambda x : \tau. t \mid t t \mid \Lambda \alpha. t \mid t \tau$

$$\Gamma, x : \tau \vdash x : \tau \qquad \llbracket x \rrbracket_{\theta, \sigma} = \sigma(x)$$

$$\frac{\Gamma, x : \tau_1 \vdash t : \tau_2}{\Gamma \vdash (\lambda x : \tau_1. t) : \tau_1 \rightarrow \tau_2} \qquad \llbracket \lambda x : \tau_1. t \rrbracket_{\theta, \sigma} a = \llbracket t \rrbracket_{\theta, \sigma[x \mapsto a]}$$

$$\frac{\Gamma \vdash t : \tau_1 \rightarrow \tau_2 \quad \Gamma \vdash u : \tau_1}{\Gamma \vdash (t u) : \tau_2} \qquad \llbracket t u \rrbracket_{\theta, \sigma} = \llbracket t \rrbracket_{\theta, \sigma} \llbracket u \rrbracket_{\theta, \sigma}$$

$$\frac{\alpha, \Gamma \vdash t : \tau}{\Gamma \vdash (\Lambda \alpha. t) : \forall \alpha. \tau} \qquad \llbracket \Lambda \alpha. t \rrbracket_{\theta, \sigma} S = \llbracket t \rrbracket_{\theta[\alpha \mapsto S], \sigma}$$

$$\frac{\Gamma \vdash t : \forall \alpha. \tau}{\Gamma \vdash (t \tau') : \tau[\tau'/\alpha]} \qquad \llbracket t \tau' \rrbracket_{\theta, \sigma} = \llbracket t \rrbracket_{\theta, \sigma} \llbracket \tau' \rrbracket_{\theta}$$

Rekursion im λ -Kalkül (3)

- Die Transformation von fak zu fak' war systematisch – zu jeder rekursiven Funktion H kann ein entsprechendes (nicht-rekursives) **Funktional** H' gefunden werden.
- Wir suchen jetzt eine (nicht-rekursive) Funktion Y , für die gilt:

$$Y(H') = H \quad (H \text{ rekursiv, } Y \text{ und } H' \text{ nicht-rekursiv})$$

- Da generell $H'(H) = H$ gilt, ist H ein **Fixpunkt** der Funktion H' .
- Weil für die gesuchte Funktion $Y(H') \rightarrow H$ gilt, wird sie auch als **Fixpunktkombinator** bezeichnet:

$$H = Y(H') = H'(H) = H'(Y H')$$

insbes. gilt:

$$Y(H') = H'(Y H')$$

- Die Funktion **Y** wird definiert als:

$$\lambda h x . h (x x) (\lambda x . h (x x))$$

- Dass diese Funktion das gesuchte Y ist, wird durch folgenden Zusammenhang belegt:

$$\begin{aligned} \mathbf{Y H'} &= (\lambda h x . h (x x) (\lambda x . h (x x))) H' \\ &\rightarrow (\lambda x . H' (x x)) (\lambda x . H' (x x)) \\ &\rightarrow H' ((\lambda x . H' (x x)) (\lambda x . H' (x x))) \\ &= \mathbf{H' (Y H')} \end{aligned}$$

Ein vollständiger Beweis ist das aber nicht – dieser kann z.B. bei Meyer oder Stoy nachgelesen werden.

Allgemeine Rekursion im Polymorphen Lambda-Kalkül

Terme: $t := \dots \mid \mathbf{fix} \ t$

Allgemeine Rekursion im Polymorphen Lambda-Kalkül

Terme: $t := \dots \mid \mathbf{fix} \ t$

$$\frac{\Gamma \vdash t : \tau \rightarrow \tau}{\Gamma \vdash (\mathbf{fix} \ t) : \tau}$$

Allgemeine Rekursion im Polymorphen Lambda-Kalkül

Terme: $t ::= \dots \mid \mathbf{fix} \ t$

$$\frac{\Gamma \vdash t : \tau \rightarrow \tau}{\Gamma \vdash (\mathbf{fix} \ t) : \tau}$$

Um Semantik anzugeben, werden Typen als vollständige Halbordnungen mit kleinstem Element interpretiert, und:

$$\llbracket \mathbf{fix} \ t \rrbracket_{\theta, \sigma} = \bigsqcup_{i \geq 0} (\llbracket t \rrbracket_{\theta, \sigma}^i \perp).$$

Verwendung im Beispiel

Aus:

```
filter :: ( $\alpha \rightarrow \text{Bool}$ )  $\rightarrow$  [ $\alpha$ ]  $\rightarrow$  [ $\alpha$ ]  
filter p [] = []  
filter p (a : as) = if p a then a : filter p as  
                    else filter p as
```

wird:

```
fix ( $\lambda f : (\forall \alpha. (\alpha \rightarrow \text{Bool}) \rightarrow [\alpha] \rightarrow [\alpha]).$   
     $\Lambda \alpha. \lambda p : (\alpha \rightarrow \text{Bool}). \lambda l : [\alpha].$   
    case l of { []  $\rightarrow$  [] $_{\alpha}$  ;  
              (a : as)  $\rightarrow$  case p a of  
                            { True  $\rightarrow$  a : (f  $\alpha$  p as) ;  
                              False  $\rightarrow$  f  $\alpha$  p as } })
```

Allgemeine Rekursion im Polymorphen Lambda-Kalkül

Terme: $t ::= \dots \mid \mathbf{fix} \ t$

$$\frac{\Gamma \vdash t : \tau \rightarrow \tau}{\Gamma \vdash (\mathbf{fix} \ t) : \tau}$$

Um Semantik anzugeben, werden Typen als vollständige Halbordnungen mit kleinstem Element interpretiert, und:

$$\llbracket \mathbf{fix} \ t \rrbracket_{\theta, \sigma} = \bigsqcup_{i \geq 0} (\llbracket t \rrbracket_{\theta, \sigma}^i \perp).$$

Und was ist mit dem Parametritäts-Theorem?

Allgemeine Rekursion im Polymorphen Lambda-Kalkül

Terme: $t ::= \dots \mid \mathbf{fix} \ t$

$$\frac{\Gamma \vdash t : \tau \rightarrow \tau}{\Gamma \vdash (\mathbf{fix} \ t) : \tau}$$

Um Semantik anzugeben, werden Typen als vollständige Halbordnungen mit kleinstem Element interpretiert, und:

$$\llbracket \mathbf{fix} \ t \rrbracket_{\theta, \sigma} = \bigsqcup_{i \geq 0} (\llbracket t \rrbracket_{\theta, \sigma}^i \perp).$$

Und was ist mit dem Parametritäts-Theorem?

Der relevante Induktions-Fall ist:

$$\frac{\Gamma \vdash t : \tau \rightarrow \tau}{\Gamma \vdash (\mathbf{fix} \ t) : \tau}$$

Allgemeine Rekursion im Polymorphen Lambda-Kalkül

Terme: $t := \dots \mid \mathbf{fix} \ t$

$$\frac{\Gamma \vdash t : \tau \rightarrow \tau}{\Gamma \vdash (\mathbf{fix} \ t) : \tau}$$

Um Semantik anzugeben, werden Typen als vollständige Halbordnungen mit kleinstem Element interpretiert, und:

$$\llbracket \mathbf{fix} \ t \rrbracket_{\theta, \sigma} = \bigsqcup_{i \geq 0} (\llbracket t \rrbracket_{\theta, \sigma}^i \perp).$$

Und was ist mit dem Parametritäts-Theorem?

Der relevante Induktions-Fall ist:

$$\frac{\Gamma \vdash t : \tau \rightarrow \tau}{(\llbracket \mathbf{fix} \ t \rrbracket_{\theta_1, \sigma_1}, \llbracket \mathbf{fix} \ t \rrbracket_{\theta_2, \sigma_2}) \in \Delta_{\tau, \rho}}$$

Allgemeine Rekursion im Polymorphen Lambda-Kalkül

Terme: $t := \dots \mid \mathbf{fix} \ t$

$$\frac{\Gamma \vdash t : \tau \rightarrow \tau}{\Gamma \vdash (\mathbf{fix} \ t) : \tau}$$

Um Semantik anzugeben, werden Typen als vollständige Halbordnungen mit kleinstem Element interpretiert, und:

$$\llbracket \mathbf{fix} \ t \rrbracket_{\theta, \sigma} = \bigsqcup_{i \geq 0} (\llbracket t \rrbracket_{\theta, \sigma}^i \perp).$$

Und was ist mit dem Parametritäts-Theorem?

Der relevante Induktions-Fall ist:

$$\frac{(\llbracket t \rrbracket_{\theta_1, \sigma_1}, \llbracket t \rrbracket_{\theta_2, \sigma_2}) \in \Delta_{\tau \rightarrow \tau, \rho}}{(\llbracket \mathbf{fix} \ t \rrbracket_{\theta_1, \sigma_1}, \llbracket \mathbf{fix} \ t \rrbracket_{\theta_2, \sigma_2}) \in \Delta_{\tau, \rho}}$$

Allgemeine Rekursion im Polymorphen Lambda-Kalkül

Terme: $t ::= \dots \mid \mathbf{fix} \ t$

$$\frac{\Gamma \vdash t : \tau \rightarrow \tau}{\Gamma \vdash (\mathbf{fix} \ t) : \tau}$$

Um Semantik anzugeben, werden Typen als vollständige Halbordnungen mit kleinstem Element interpretiert, und:

$$\llbracket \mathbf{fix} \ t \rrbracket_{\theta, \sigma} = \bigsqcup_{i \geq 0} (\llbracket t \rrbracket_{\theta, \sigma}^i \perp).$$

Und was ist mit dem Parametritäts-Theorem?

Der relevante Induktions-Fall ist:

$$\frac{\forall (a_1, a_2) \in \Delta_{\tau, \rho}. (\llbracket t \rrbracket_{\theta_1, \sigma_1} a_1, \llbracket t \rrbracket_{\theta_2, \sigma_2} a_2) \in \Delta_{\tau, \rho}}{(\llbracket \mathbf{fix} \ t \rrbracket_{\theta_1, \sigma_1}, \llbracket \mathbf{fix} \ t \rrbracket_{\theta_2, \sigma_2}) \in \Delta_{\tau, \rho}}$$

Allgemeine Rekursion im Polymorphen Lambda-Kalkül

Terme: $t ::= \dots \mid \mathbf{fix} \ t$

$$\frac{\Gamma \vdash t : \tau \rightarrow \tau}{\Gamma \vdash (\mathbf{fix} \ t) : \tau}$$

Um Semantik anzugeben, werden Typen als vollständige Halbordnungen mit kleinstem Element interpretiert, und:

$$\llbracket \mathbf{fix} \ t \rrbracket_{\theta, \sigma} = \bigsqcup_{i \geq 0} (\llbracket t \rrbracket_{\theta, \sigma}^i \perp).$$

Und was ist mit dem Parametritäts-Theorem?

Der relevante Induktions-Fall ist:

$$\frac{\forall (a_1, a_2) \in \Delta_{\tau, \rho}. (\llbracket t \rrbracket_{\theta_1, \sigma_1} a_1, \llbracket t \rrbracket_{\theta_2, \sigma_2} a_2) \in \Delta_{\tau, \rho}}{(\llbracket \mathbf{fix} \ t \rrbracket_{\theta_1, \sigma_1}, \llbracket \mathbf{fix} \ t \rrbracket_{\theta_2, \sigma_2}) \in \Delta_{\tau, \rho}}$$

Das Parametritäts-Theorem gilt weiter, vorausgesetzt alle Relationen sind strikt und stetig.

Anwendung dieser Erkenntnis

Für jedes $g :: (a \rightarrow \text{Bool}) \rightarrow [a] \rightarrow [a]$,

$$g \ p \ (\text{map } h \ l) = \text{map } h \ (g \ (p \circ h) \ l)$$

wenn h strikt ($h \ \perp = \perp$).

Anwendung dieser Erkenntnis

Für jedes $g :: (a \rightarrow \text{Bool}) \rightarrow [a] \rightarrow [a]$,

$$g \ p \ (\text{map } h \ l) \ = \ \text{map } h \ (g \ (p \circ h) \ l)$$

wenn h strikt ($h \ \perp = \perp$).

Analoge Auswirkungen ergeben sich für andere freie Theoreme.

Anwendung dieser Erkenntnis

Für jedes $g :: (a \rightarrow \text{Bool}) \rightarrow [a] \rightarrow [a]$,

$$g \ p \ (\text{map } h \ l) = \text{map } h \ (g \ (p \circ h) \ l)$$

wenn h strikt ($h \ \perp = \perp$).

Analoge Auswirkungen ergeben sich für andere freie Theoreme.

Übrigens sind solche Nebenbedingungen im Allgemeinen nur hinreichend, nicht unbedingt notwendig.

Hinzufügen Selektiver Striktheit

Terme: $t := \dots \mid \mathbf{seq} \ t \ t$

Hinzufügen Selektiver Striktheit

Terme: $t := \dots \mid \mathbf{seq} \ t \ t$

$$\frac{\Gamma \vdash t_1 : \tau_1 \quad \Gamma \vdash t_2 : \tau_2}{\Gamma \vdash (\mathbf{seq} \ t_1 \ t_2) : \tau_2}$$

Hinzufügen Selektiver Striktheit

Terme: $t := \dots \mid \mathbf{seq} \ t \ t$

$$\frac{\Gamma \vdash t_1 : \tau_1 \quad \Gamma \vdash t_2 : \tau_2}{\Gamma \vdash (\mathbf{seq} \ t_1 \ t_2) : \tau_2}$$

Semantik:

$$\llbracket \mathbf{seq} \ t_1 \ t_2 \rrbracket_{\theta, \sigma} = \begin{cases} \perp & \text{wenn } \llbracket t_1 \rrbracket_{\theta, \sigma} = \perp \\ \llbracket t_2 \rrbracket_{\theta, \sigma} & \text{wenn } \llbracket t_1 \rrbracket_{\theta, \sigma} \neq \perp. \end{cases}$$

Hinzufügen Selektiver Striktheit

Terme: $t := \dots \mid \mathbf{seq} \ t \ t$

$$\frac{\Gamma \vdash t_1 : \tau_1 \quad \Gamma \vdash t_2 : \tau_2}{\Gamma \vdash (\mathbf{seq} \ t_1 \ t_2) : \tau_2}$$

Semantik:

$$\llbracket \mathbf{seq} \ t_1 \ t_2 \rrbracket_{\theta, \sigma} = \begin{cases} \perp & \text{wenn } \llbracket t_1 \rrbracket_{\theta, \sigma} = \perp \\ \llbracket t_2 \rrbracket_{\theta, \sigma} & \text{wenn } \llbracket t_1 \rrbracket_{\theta, \sigma} \neq \perp. \end{cases}$$

Das Parametritäts-Theorem steht wieder in Zweifel!

Ohne **seq**, $g \ p \ (\text{map } h \ l) = \text{map } h \ (g \ (p \circ h) \ l)$

- $g :: (a \rightarrow \text{Bool}) \rightarrow [a] \rightarrow [a]$ muss einheitlich arbeiten.
- Die Ausgabeliste kann nur Elemente aus l , sowie \perp enthalten.
- Welche, und in welcher Reihenfolge/Vielfachheit, kann lediglich von l und dem Eingabepredikat p abhängen.
- Einzig mögliche Grundlagen zur Entscheidung sind die Länge von l und Ergebnisse von p auf Elementen von l und auf \perp .
- Aber, die Listen $(\text{map } h \ l)$ und l haben stets die selbe Länge.
- Und Anwendung von p auf ein Element von $(\text{map } h \ l)$ hat stets das selbe Ergebnis wie Anwendung von $(p \circ h)$ auf das entsprechende Element von l .
- Und Anwendung von p auf \perp hat das selbe Ergebnis wie Anwendung von $(p \circ h)$, **vorausgesetzt h ist strikt**.
- Also wählt g mit p stets „die selben“ Elemente aus $(\text{map } h \ l)$ wie es g mit $(p \circ h)$ aus l tut, außer dass im ersten Fall die entsprechenden Abbilder unter h ausgegeben werden, und dass sie auch, an gleichen Positionen, \perp ausgeben können.
- Also $(g \ p \ (\text{map } h \ l)) = (\text{map } h \ (g \ (p \circ h) \ l))$, **wenn h strikt ist**.

Mit `seq`, $g \ p \ (\text{map } h \ l) = \text{map } h \ (g \ (p \circ h) \ l) \ ?$

- $g :: (a \rightarrow \text{Bool}) \rightarrow [a] \rightarrow [a]$ muss **einheitlich** arbeiten.

Mit `seq`, $g \ p \ (\text{map } h \ l) = \text{map } h \ (g \ (p \circ h) \ l) \ ?$

- $g :: (a \rightarrow \text{Bool}) \rightarrow [a] \rightarrow [a]$ muss einheitlich arbeiten.
- Die Ausgabeliste kann nur **Elemente aus l , sowie \perp** enthalten.

Mit `seq`, g p $(\text{map } h \ l) = \text{map } h \ (g \ (p \circ h) \ l)$?

- $g :: (a \rightarrow \text{Bool}) \rightarrow [a] \rightarrow [a]$ muss einheitlich arbeiten.
- Die Ausgabeliste kann nur **Elemente aus l , sowie \perp** enthalten.
- Welche, und in welcher Reihenfolge/Vielfachheit, kann lediglich **von l und dem Eingabepredikat p abhängen**.

Mit **seq**, $g \ p \ (\text{map } h \ l) = \text{map } h \ (g \ (p \circ h) \ l) \ ?$

- $g :: (a \rightarrow \text{Bool}) \rightarrow [a] \rightarrow [a]$ muss einheitlich arbeiten.
- Die Ausgabeliste kann nur Elemente aus l , sowie \perp enthalten.
- Welche, und in welcher Reihenfolge/Vielfachheit, kann lediglich **von l und dem Eingabepredikat p abhängen**.
- Einzig mögliche Grundlagen zur Entscheidung sind die **Länge von l und Ergebnisse von p auf Elementen von l und auf \perp** .

Mit `seq`, $g \ p \ (\text{map } h \ l) = \text{map } h \ (g \ (p \circ h) \ l) \ ?$

- $g :: (a \rightarrow \text{Bool}) \rightarrow [a] \rightarrow [a]$ muss einheitlich arbeiten.
- Die Ausgabeliste kann nur Elemente aus l , sowie \perp enthalten.
- Welche, und in welcher Reihenfolge/Vielfachheit, kann lediglich von l und dem Eingabepredikat p abhängen.
- Einzig mögliche Grundlagen zur Entscheidung sind die **Länge von l** und **Ergebnisse von p auf Elementen von l und auf \perp** .

⚡ Falsch! Auch möglich:

- **Elemente von l auf \perp prüfen**
- **p auf \perp prüfen**

Mit `seq`, $g \ p \ (\text{map } h \ l) = \text{map } h \ (g \ (p \circ h) \ l) \ ?$

- $g :: (a \rightarrow \text{Bool}) \rightarrow [a] \rightarrow [a]$ muss einheitlich arbeiten.
- Die Ausgabeliste kann nur Elemente aus l , sowie \perp enthalten.
- Welche, und in welcher Reihenfolge/Vielfachheit, kann lediglich von l und dem Eingabepredikat p abhängen.
- Einzig mögliche Grundlagen zur Entscheidung sind die Länge von l und Ergebnisse von p auf Elementen von l und auf \perp .

⚡ Falsch! Auch möglich:

- Elemente von l auf \perp prüfen
- p auf \perp prüfen

... ???

Anpassungen Freier Theoreme

[Wadler 1989] : für jedes $g :: (a \rightarrow \text{Bool}) \rightarrow [a] \rightarrow [a]$,

$$g \ p \ (\text{map } h \ l) = \text{map } h \ (g \ (p \circ h) \ l)$$

- wenn h strikt.

Anpassungen Freier Theoreme

[Wadler 1989] : für jedes $g :: (a \rightarrow \text{Bool}) \rightarrow [a] \rightarrow [a]$,

$$g \ p \ (\text{map } h \ l) = \text{map } h \ (g \ (p \circ h) \ l)$$

- wenn h strikt.

[Johann & V. 2004] : in Gegenwart von `seq`, wenn zusätzlich:

- $p \neq \perp$ und
- h total ($\forall x \neq \perp. h \ x \neq \perp$).

Anpassungen Freier Theoreme

[Wadler 1989] : für jedes $g :: (a \rightarrow \text{Bool}) \rightarrow [a] \rightarrow [a]$,

$$g \ p \ (\text{map } h \ l) = \text{map } h \ (g \ (p \circ h) \ l)$$

- wenn h strikt.

[Johann & V. 2004] : in Gegenwart von `seq`, wenn zusätzlich:

- $p \neq \perp$ und
- h total ($\forall x \neq \perp. h \ x \neq \perp$).

[Johann & V. 2009] : Betrachtung von Laufzeitfehlern

[Stenger & V. 2009] : Betrachtung von „impräziser“ Fehlersemantik