

Polymorpher Lambda-Kalkül [Girard 1972, Reynolds 1974]

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

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

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

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

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

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

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

Das Parametritäts-Theorem [Reynolds 1983, Wadler 1989]

Gegeben τ und Environments θ_1, θ_2, ρ mit $\rho(\alpha) \subseteq \theta_1(\alpha) \times \theta_2(\alpha)$,
definiere $\Delta_{\tau, \rho} \subseteq \llbracket \tau \rrbracket_{\theta_1} \times \llbracket \tau \rrbracket_{\theta_2}$ wie folgt:

$$\Delta_{\alpha, \rho} = \rho(\alpha)$$

$$\Delta_{\tau_1 \rightarrow \tau_2, \rho} = \{(f_1, f_2) \mid \forall (a_1, a_2) \in \Delta_{\tau_1, \rho}. (f_1 a_1, f_2 a_2) \in \Delta_{\tau_2, \rho}\}$$

$$\Delta_{\forall \alpha. \tau, \rho} = \{(g_1, g_2) \mid \forall \mathcal{R} \subseteq S_1 \times S_2. (g_1 S_1, g_2 S_2) \in \Delta_{\tau, \rho[\alpha \mapsto \mathcal{R}]}\}$$

Dann gilt für jeden geschlossenen Term t geschlossenen Typs τ :

$$(\llbracket t \rrbracket_{\emptyset, \emptyset}, \llbracket t \rrbracket_{\emptyset, \emptyset}) \in \Delta_{\tau, \emptyset}.$$

Beweis-Skizze

Beweise folgende allgemeinere Aussage:

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

vorausgesetzt $(\sigma_1(x), \sigma_2(x)) \in \Delta_{\tau', \rho}$ für jedes $x : \tau'$ in Γ

per Induktion über die Struktur von Typableitungen.

Beweis-Skizze

Beweise folgende allgemeinere Aussage:

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

vorausgesetzt $(\sigma_1(x), \sigma_2(x)) \in \Delta_{\tau', \rho}$ für jedes $x : \tau'$ in Γ

per Induktion über die Struktur von Typableitungen.

Der Basis-Fall ist einfach.

Beweis-Skizze

Beweise folgende allgemeinere Aussage:

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

vorausgesetzt $(\sigma_1(x), \sigma_2(x)) \in \Delta_{\tau', \rho}$ für jedes $x : \tau'$ in Γ

per Induktion über die Struktur von Typableitungen.

Der Basis-Fall ist einfach. In den anderen Fällen:

$$\frac{\Gamma, x : \tau_1 \vdash t : \tau_2}{\Gamma \vdash (\lambda x : \tau_1. t) : \tau_1 \rightarrow \tau_2}$$

Beweis-Skizze

Beweise folgende allgemeinere Aussage:

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

vorausgesetzt $(\sigma_1(x), \sigma_2(x)) \in \Delta_{\tau', \rho}$ für jedes $x : \tau'$ in Γ

per Induktion über die Struktur von Typableitungen.

Der Basis-Fall ist einfach. In den anderen Fällen:

$$\frac{\Gamma, x : \tau_1 \vdash t : \tau_2}{(\llbracket \lambda x : \tau_1. t \rrbracket_{\theta_1, \sigma_1}, \llbracket \lambda x : \tau_1. t \rrbracket_{\theta_2, \sigma_2}) \in \Delta_{\tau_1 \rightarrow \tau_2, \rho}}$$

Beweis-Skizze

Beweise folgende allgemeinere Aussage:

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

vorausgesetzt $(\sigma_1(x), \sigma_2(x)) \in \Delta_{\tau', \rho}$ für jedes $x : \tau'$ in Γ

per Induktion über die Struktur von Typableitungen.

Der Basis-Fall ist einfach. In den anderen Fällen:

$$\frac{\forall (a_1, a_2) \in \Delta_{\tau_1, \rho}. (\llbracket t \rrbracket_{\theta_1, \sigma_1[x \mapsto a_1]}, \llbracket t \rrbracket_{\theta_2, \sigma_2[x \mapsto a_2]}) \in \Delta_{\tau_2, \rho}}{(\llbracket \lambda x : \tau_1. t \rrbracket_{\theta_1, \sigma_1}, \llbracket \lambda x : \tau_1. t \rrbracket_{\theta_2, \sigma_2}) \in \Delta_{\tau_1 \rightarrow \tau_2, \rho}}$$

Beweis-Skizze

Beweise folgende allgemeinere Aussage:

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

vorausgesetzt $(\sigma_1(x), \sigma_2(x)) \in \Delta_{\tau', \rho}$ für jedes $x : \tau'$ in Γ

per Induktion über die Struktur von Typableitungen.

Der Basis-Fall ist einfach. In den anderen Fällen:

$$\frac{\forall (a_1, a_2) \in \Delta_{\tau_1, \rho}. (\llbracket t \rrbracket_{\theta_1, \sigma_1[x \mapsto a_1]}, \llbracket t \rrbracket_{\theta_2, \sigma_2[x \mapsto a_2]}) \in \Delta_{\tau_2, \rho}}{(\llbracket \lambda x : \tau_1. t \rrbracket_{\theta_1, \sigma_1}, \llbracket \lambda x : \tau_1. t \rrbracket_{\theta_2, \sigma_2}) \in \Delta_{\tau_1 \rightarrow \tau_2, \rho}}$$
$$\frac{\Gamma \vdash t : \tau_1 \rightarrow \tau_2 \quad \Gamma \vdash u : \tau_1}{\Gamma \vdash (t \ u) : \tau_2}$$

Beweis-Skizze

Beweise folgende allgemeinere Aussage:

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

vorausgesetzt $(\sigma_1(x), \sigma_2(x)) \in \Delta_{\tau', \rho}$ für jedes $x : \tau'$ in Γ

per Induktion über die Struktur von Typableitungen.

Der Basis-Fall ist einfach. In den anderen Fällen:

$$\frac{\forall (a_1, a_2) \in \Delta_{\tau_1, \rho}. (\llbracket t \rrbracket_{\theta_1, \sigma_1[x \mapsto a_1]}, \llbracket t \rrbracket_{\theta_2, \sigma_2[x \mapsto a_2]}) \in \Delta_{\tau_2, \rho}}{(\llbracket \lambda x : \tau_1. t \rrbracket_{\theta_1, \sigma_1}, \llbracket \lambda x : \tau_1. t \rrbracket_{\theta_2, \sigma_2}) \in \Delta_{\tau_1 \rightarrow \tau_2, \rho}}$$
$$\frac{\Gamma \vdash t : \tau_1 \rightarrow \tau_2 \quad \Gamma \vdash u : \tau_1}{(\llbracket t \ u \rrbracket_{\theta_1, \sigma_1}, \llbracket t \ u \rrbracket_{\theta_2, \sigma_2}) \in \Delta_{\tau_2, \rho}}$$

Beweis-Skizze

Beweise folgende allgemeinere Aussage:

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

vorausgesetzt $(\sigma_1(x), \sigma_2(x)) \in \Delta_{\tau', \rho}$ für jedes $x : \tau'$ in Γ

per Induktion über die Struktur von Typableitungen.

Der Basis-Fall ist einfach. In den anderen Fällen:

$$\frac{\forall (a_1, a_2) \in \Delta_{\tau_1, \rho}. (\llbracket t \rrbracket_{\theta_1, \sigma_1[x \mapsto a_1]}, \llbracket t \rrbracket_{\theta_2, \sigma_2[x \mapsto a_2]}) \in \Delta_{\tau_2, \rho}}{(\llbracket \lambda x : \tau_1. t \rrbracket_{\theta_1, \sigma_1}, \llbracket \lambda x : \tau_1. t \rrbracket_{\theta_2, \sigma_2}) \in \Delta_{\tau_1 \rightarrow \tau_2, \rho}}$$
$$\frac{(\llbracket t \rrbracket_{\theta_1, \sigma_1}, \llbracket t \rrbracket_{\theta_2, \sigma_2}) \in \Delta_{\tau_1 \rightarrow \tau_2, \rho} \quad (\llbracket u \rrbracket_{\theta_1, \sigma_1}, \llbracket u \rrbracket_{\theta_2, \sigma_2}) \in \Delta_{\tau_1, \rho}}{(\llbracket t \ u \rrbracket_{\theta_1, \sigma_1}, \llbracket t \ u \rrbracket_{\theta_2, \sigma_2}) \in \Delta_{\tau_2, \rho}}$$

Beweis-Skizze

Beweise folgende allgemeinere Aussage:

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

vorausgesetzt $(\sigma_1(x), \sigma_2(x)) \in \Delta_{\tau', \rho}$ für jedes $x : \tau'$ in Γ

per Induktion über die Struktur von Typableitungen.

Der Basis-Fall ist einfach. In den anderen Fällen:

$$\frac{\forall (a_1, a_2) \in \Delta_{\tau_1, \rho}. (\llbracket t \rrbracket_{\theta_1, \sigma_1[x \mapsto a_1]}, \llbracket t \rrbracket_{\theta_2, \sigma_2[x \mapsto a_2]}) \in \Delta_{\tau_2, \rho}}{(\llbracket \lambda x : \tau_1. t \rrbracket_{\theta_1, \sigma_1}, \llbracket \lambda x : \tau_1. t \rrbracket_{\theta_2, \sigma_2}) \in \Delta_{\tau_1 \rightarrow \tau_2, \rho}}$$
$$\frac{(\llbracket t \rrbracket_{\theta_1, \sigma_1}, \llbracket t \rrbracket_{\theta_2, \sigma_2}) \in \Delta_{\tau_1 \rightarrow \tau_2, \rho} \quad (\llbracket u \rrbracket_{\theta_1, \sigma_1}, \llbracket u \rrbracket_{\theta_2, \sigma_2}) \in \Delta_{\tau_1, \rho}}{(\llbracket t \ u \rrbracket_{\theta_1, \sigma_1}, \llbracket t \ u \rrbracket_{\theta_2, \sigma_2}) \in \Delta_{\tau_2, \rho}}$$
$$\frac{\alpha, \Gamma \vdash t : \tau}{\Gamma \vdash (\Lambda \alpha. t) : \forall \alpha. \tau}$$

Beweis-Skizze

Beweise folgende allgemeinere Aussage:

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

vorausgesetzt $(\sigma_1(x), \sigma_2(x)) \in \Delta_{\tau', \rho}$ für jedes $x : \tau'$ in Γ

per Induktion über die Struktur von Typableitungen.

Der Basis-Fall ist einfach. In den anderen Fällen:

$$\frac{\forall (a_1, a_2) \in \Delta_{\tau_1, \rho}. (\llbracket t \rrbracket_{\theta_1, \sigma_1[x \mapsto a_1]}, \llbracket t \rrbracket_{\theta_2, \sigma_2[x \mapsto a_2]}) \in \Delta_{\tau_2, \rho}}{(\llbracket \lambda x : \tau_1. t \rrbracket_{\theta_1, \sigma_1}, \llbracket \lambda x : \tau_1. t \rrbracket_{\theta_2, \sigma_2}) \in \Delta_{\tau_1 \rightarrow \tau_2, \rho}}$$
$$\frac{(\llbracket t \rrbracket_{\theta_1, \sigma_1}, \llbracket t \rrbracket_{\theta_2, \sigma_2}) \in \Delta_{\tau_1 \rightarrow \tau_2, \rho} \quad (\llbracket u \rrbracket_{\theta_1, \sigma_1}, \llbracket u \rrbracket_{\theta_2, \sigma_2}) \in \Delta_{\tau_1, \rho}}{(\llbracket t \ u \rrbracket_{\theta_1, \sigma_1}, \llbracket t \ u \rrbracket_{\theta_2, \sigma_2}) \in \Delta_{\tau_2, \rho}}$$
$$\frac{\alpha, \Gamma \vdash t : \tau}{(\llbracket \Lambda \alpha. t \rrbracket_{\theta_1, \sigma_1}, \llbracket \Lambda \alpha. t \rrbracket_{\theta_2, \sigma_2}) \in \Delta_{\forall \alpha. \tau, \rho}}$$

Beweis-Skizze

Beweise folgende allgemeinere Aussage:

$\Gamma \vdash t : \tau$ impliziert $(\llbracket t \rrbracket_{\theta_1, \sigma_1}, \llbracket t \rrbracket_{\theta_2, \sigma_2}) \in \Delta_{\tau, \rho}$,
vorausgesetzt $(\sigma_1(x), \sigma_2(x)) \in \Delta_{\tau', \rho}$ für jedes $x : \tau'$ in Γ

per Induktion über die Struktur von Typableitungen.

Der Basis-Fall ist einfach. In den anderen Fällen:

$$\frac{\forall (a_1, a_2) \in \Delta_{\tau_1, \rho}. (\llbracket t \rrbracket_{\theta_1, \sigma_1[x \mapsto a_1]}, \llbracket t \rrbracket_{\theta_2, \sigma_2[x \mapsto a_2]}) \in \Delta_{\tau_2, \rho}}{(\llbracket \lambda x : \tau_1. t \rrbracket_{\theta_1, \sigma_1}, \llbracket \lambda x : \tau_1. t \rrbracket_{\theta_2, \sigma_2}) \in \Delta_{\tau_1 \rightarrow \tau_2, \rho}}$$
$$\frac{(\llbracket t \rrbracket_{\theta_1, \sigma_1}, \llbracket t \rrbracket_{\theta_2, \sigma_2}) \in \Delta_{\tau_1 \rightarrow \tau_2, \rho} \quad (\llbracket u \rrbracket_{\theta_1, \sigma_1}, \llbracket u \rrbracket_{\theta_2, \sigma_2}) \in \Delta_{\tau_1, \rho}}{(\llbracket t \ u \rrbracket_{\theta_1, \sigma_1}, \llbracket t \ u \rrbracket_{\theta_2, \sigma_2}) \in \Delta_{\tau_2, \rho}}$$
$$\frac{\forall \mathcal{R} \subseteq \mathcal{S}_1 \times \mathcal{S}_2. (\llbracket t \rrbracket_{\theta_1[\alpha \mapsto \mathcal{S}_1], \sigma_1}, \llbracket t \rrbracket_{\theta_2[\alpha \mapsto \mathcal{S}_2], \sigma_2}) \in \Delta_{\tau, \rho[\alpha \mapsto \mathcal{R}]}}{(\llbracket \Lambda \alpha. t \rrbracket_{\theta_1, \sigma_1}, \llbracket \Lambda \alpha. t \rrbracket_{\theta_2, \sigma_2}) \in \Delta_{\forall \alpha. \tau, \rho}}$$

Beweis-Skizze

Beweise folgende allgemeinere Aussage:

$\Gamma \vdash t : \tau$ impliziert $(\llbracket t \rrbracket_{\theta_1, \sigma_1}, \llbracket t \rrbracket_{\theta_2, \sigma_2}) \in \Delta_{\tau, \rho}$,
vorausgesetzt $(\sigma_1(x), \sigma_2(x)) \in \Delta_{\tau', \rho}$ für jedes $x : \tau'$ in Γ

per Induktion über die Struktur von Typableitungen.

Der Basis-Fall ist einfach. In den anderen Fällen:

$$\frac{\forall (a_1, a_2) \in \Delta_{\tau_1, \rho}. (\llbracket t \rrbracket_{\theta_1, \sigma_1[x \mapsto a_1]}, \llbracket t \rrbracket_{\theta_2, \sigma_2[x \mapsto a_2]}) \in \Delta_{\tau_2, \rho}}{(\llbracket \lambda x : \tau_1. t \rrbracket_{\theta_1, \sigma_1}, \llbracket \lambda x : \tau_1. t \rrbracket_{\theta_2, \sigma_2}) \in \Delta_{\tau_1 \rightarrow \tau_2, \rho}}$$
$$\frac{(\llbracket t \rrbracket_{\theta_1, \sigma_1}, \llbracket t \rrbracket_{\theta_2, \sigma_2}) \in \Delta_{\tau_1 \rightarrow \tau_2, \rho} \quad (\llbracket u \rrbracket_{\theta_1, \sigma_1}, \llbracket u \rrbracket_{\theta_2, \sigma_2}) \in \Delta_{\tau_1, \rho}}{(\llbracket t \ u \rrbracket_{\theta_1, \sigma_1}, \llbracket t \ u \rrbracket_{\theta_2, \sigma_2}) \in \Delta_{\tau_2, \rho}}$$
$$\frac{\forall \mathcal{R} \subseteq \mathcal{S}_1 \times \mathcal{S}_2. (\llbracket t \rrbracket_{\theta_1[\alpha \mapsto \mathcal{S}_1], \sigma_1}, \llbracket t \rrbracket_{\theta_2[\alpha \mapsto \mathcal{S}_2], \sigma_2}) \in \Delta_{\tau, \rho[\alpha \mapsto \mathcal{R}]}}{(\llbracket \Lambda \alpha. t \rrbracket_{\theta_1, \sigma_1}, \llbracket \Lambda \alpha. t \rrbracket_{\theta_2, \sigma_2}) \in \Delta_{\forall \alpha. \tau, \rho}}$$
$$\frac{\Gamma \vdash t : \forall \alpha. \tau}{\Gamma \vdash (t \ \tau') : \tau[\tau' / \alpha]}$$

Beweis-Skizze

Beweise folgende allgemeinere Aussage:

$\Gamma \vdash t : \tau$ impliziert $(\llbracket t \rrbracket_{\theta_1, \sigma_1}, \llbracket t \rrbracket_{\theta_2, \sigma_2}) \in \Delta_{\tau, \rho}$,
vorausgesetzt $(\sigma_1(x), \sigma_2(x)) \in \Delta_{\tau', \rho}$ für jedes $x : \tau'$ in Γ

per Induktion über die Struktur von Typableitungen.

Der Basis-Fall ist einfach. In den anderen Fällen:

$$\frac{\forall (a_1, a_2) \in \Delta_{\tau_1, \rho}. (\llbracket t \rrbracket_{\theta_1, \sigma_1[x \mapsto a_1]}, \llbracket t \rrbracket_{\theta_2, \sigma_2[x \mapsto a_2]}) \in \Delta_{\tau_2, \rho}}{(\llbracket \lambda x : \tau_1. t \rrbracket_{\theta_1, \sigma_1}, \llbracket \lambda x : \tau_1. t \rrbracket_{\theta_2, \sigma_2}) \in \Delta_{\tau_1 \rightarrow \tau_2, \rho}}$$
$$\frac{(\llbracket t \rrbracket_{\theta_1, \sigma_1}, \llbracket t \rrbracket_{\theta_2, \sigma_2}) \in \Delta_{\tau_1 \rightarrow \tau_2, \rho} \quad (\llbracket u \rrbracket_{\theta_1, \sigma_1}, \llbracket u \rrbracket_{\theta_2, \sigma_2}) \in \Delta_{\tau_1, \rho}}{(\llbracket t \ u \rrbracket_{\theta_1, \sigma_1}, \llbracket t \ u \rrbracket_{\theta_2, \sigma_2}) \in \Delta_{\tau_2, \rho}}$$
$$\frac{\forall \mathcal{R} \subseteq \mathcal{S}_1 \times \mathcal{S}_2. (\llbracket t \rrbracket_{\theta_1[\alpha \mapsto \mathcal{S}_1], \sigma_1}, \llbracket t \rrbracket_{\theta_2[\alpha \mapsto \mathcal{S}_2], \sigma_2}) \in \Delta_{\tau, \rho[\alpha \mapsto \mathcal{R}]}}{(\llbracket \Lambda \alpha. t \rrbracket_{\theta_1, \sigma_1}, \llbracket \Lambda \alpha. t \rrbracket_{\theta_2, \sigma_2}) \in \Delta_{\forall \alpha. \tau, \rho}}$$
$$\frac{\Gamma \vdash t : \forall \alpha. \tau}{(\llbracket t \ \tau' \rrbracket_{\theta_1, \sigma_1}, \llbracket t \ \tau' \rrbracket_{\theta_2, \sigma_2}) \in \Delta_{\tau[\tau'/\alpha], \rho}}$$

Beweis-Skizze

Beweise folgende allgemeinere Aussage:

$\Gamma \vdash t : \tau$ impliziert $(\llbracket t \rrbracket_{\theta_1, \sigma_1}, \llbracket t \rrbracket_{\theta_2, \sigma_2}) \in \Delta_{\tau, \rho}$,
vorausgesetzt $(\sigma_1(x), \sigma_2(x)) \in \Delta_{\tau', \rho}$ für jedes $x : \tau'$ in Γ

per Induktion über die Struktur von Typableitungen.

Der Basis-Fall ist einfach. In den anderen Fällen:

$$\frac{\forall (a_1, a_2) \in \Delta_{\tau_1, \rho}. (\llbracket t \rrbracket_{\theta_1, \sigma_1[x \mapsto a_1]}, \llbracket t \rrbracket_{\theta_2, \sigma_2[x \mapsto a_2]}) \in \Delta_{\tau_2, \rho}}{(\llbracket \lambda x : \tau_1. t \rrbracket_{\theta_1, \sigma_1}, \llbracket \lambda x : \tau_1. t \rrbracket_{\theta_2, \sigma_2}) \in \Delta_{\tau_1 \rightarrow \tau_2, \rho}}$$
$$\frac{(\llbracket t \rrbracket_{\theta_1, \sigma_1}, \llbracket t \rrbracket_{\theta_2, \sigma_2}) \in \Delta_{\tau_1 \rightarrow \tau_2, \rho} \quad (\llbracket u \rrbracket_{\theta_1, \sigma_1}, \llbracket u \rrbracket_{\theta_2, \sigma_2}) \in \Delta_{\tau_1, \rho}}{(\llbracket t \ u \rrbracket_{\theta_1, \sigma_1}, \llbracket t \ u \rrbracket_{\theta_2, \sigma_2}) \in \Delta_{\tau_2, \rho}}$$
$$\frac{\forall \mathcal{R} \subseteq S_1 \times S_2. (\llbracket t \rrbracket_{\theta_1[\alpha \mapsto S_1], \sigma_1}, \llbracket t \rrbracket_{\theta_2[\alpha \mapsto S_2], \sigma_2}) \in \Delta_{\tau, \rho[\alpha \mapsto \mathcal{R}]}}{(\llbracket \Lambda \alpha. t \rrbracket_{\theta_1, \sigma_1}, \llbracket \Lambda \alpha. t \rrbracket_{\theta_2, \sigma_2}) \in \Delta_{\forall \alpha. \tau, \rho}}$$
$$\frac{(\llbracket t \rrbracket_{\theta_1, \sigma_1}, \llbracket t \rrbracket_{\theta_2, \sigma_2}) \in \Delta_{\forall \alpha. \tau, \rho}}{(\llbracket t \ \tau' \rrbracket_{\theta_1, \sigma_1}, \llbracket t \ \tau' \rrbracket_{\theta_2, \sigma_2}) \in \Delta_{\tau[\tau'/\alpha], \rho}}$$

Hinzufügen von Datentypen

Typen: $\tau := \dots \mid \text{Bool} \mid [\tau]$

Terme: $t := \dots \mid \text{True} \mid \text{False} \mid []_{\tau} \mid t : t \mid \text{case } t \text{ of } \{\dots\}$

Hinzufügen von Datentypen

Typen: $\tau := \dots \mid \text{Bool} \mid [\tau]$

Terme: $t := \dots \mid \text{True} \mid \text{False} \mid []_{\tau} \mid t : t \mid \text{case } t \text{ of } \{\dots\}$

$\Gamma \vdash \text{True} : \text{Bool}$, $\Gamma \vdash \text{False} : \text{Bool}$, $\Gamma \vdash []_{\tau} : [\tau]$

$$\frac{\Gamma \vdash t : \tau \quad \Gamma \vdash u : [\tau]}{\Gamma \vdash (t : u) : [\tau]}$$

$$\frac{\Gamma \vdash t : \text{Bool} \quad \Gamma \vdash u : \tau \quad \Gamma \vdash v : \tau}{\Gamma \vdash (\text{case } t \text{ of } \{\text{True} \rightarrow u; \text{False} \rightarrow v\}) : \tau}$$

$$\frac{\Gamma \vdash t : [\tau'] \quad \Gamma \vdash u : \tau \quad \Gamma, x_1 : \tau', x_2 : [\tau'] \vdash v : \tau}{\Gamma \vdash (\text{case } t \text{ of } \{[] \rightarrow u; (x_1 : x_2) \rightarrow v\}) : \tau}$$

Hinzufügen von Datentypen

Typen: $\tau := \dots \mid \text{Bool} \mid [\tau]$

Terme: $t := \dots \mid \text{True} \mid \text{False} \mid []_{\tau} \mid t : t \mid \text{case } t \text{ of } \{\dots\}$

$\Gamma \vdash \text{True} : \text{Bool}$, $\Gamma \vdash \text{False} : \text{Bool}$, $\Gamma \vdash []_{\tau} : [\tau]$

$$\frac{\Gamma \vdash t : \tau \quad \Gamma \vdash u : [\tau]}{\Gamma \vdash (t : u) : [\tau]}$$

$$\frac{\Gamma \vdash t : \text{Bool} \quad \Gamma \vdash u : \tau \quad \Gamma \vdash v : \tau}{\Gamma \vdash (\text{case } t \text{ of } \{\text{True} \rightarrow u; \text{False} \rightarrow v\}) : \tau}$$

$$\frac{\Gamma \vdash t : [\tau'] \quad \Gamma \vdash u : \tau \quad \Gamma, x_1 : \tau', x_2 : [\tau'] \vdash v : \tau}{\Gamma \vdash (\text{case } t \text{ of } \{[] \rightarrow u; (x_1 : x_2) \rightarrow v\}) : \tau}$$

Mit „offensichtlichen“ Erweiterungen der Semantik und mit

$$\Delta_{\text{Bool}, \rho} = \{(\text{True}, \text{True}), (\text{False}, \text{False})\}$$

$$\Delta_{[\tau], \rho} = \{([x_1, \dots, x_n], [y_1, \dots, y_n]) \mid n \geq 0, (x_i, y_i) \in \Delta_{\tau, \rho}\},$$

ist das Parametritäts-Theorem immer noch erfüllt.

Typklassen

Für jedes

$$g :: [\alpha] \rightarrow [\alpha]$$

hatten wir

$$g (\text{map } f \ l) = \text{map } f (g \ l)$$

für beliebige f und l .

Typklassen

Für jedes

$$g :: [\alpha] \rightarrow [\alpha]$$

hatten wir

$$g (\text{map } f \ l) = \text{map } f \ (g \ l)$$

für beliebige f und l .

Was ist mit

$$g :: \text{Eq } \alpha \Rightarrow [\alpha] \rightarrow [\alpha] \ ?$$

Typklassen

Für jedes

$$g :: [\alpha] \rightarrow [\alpha]$$

hatten wir

$$g (\text{map } f \ l) = \text{map } f (g \ l)$$

für beliebige f und l .

Was ist mit

$$g :: \text{Eq } \alpha \Rightarrow [\alpha] \rightarrow [\alpha] \ ?$$

Obiges freies Theorem schlägt fehl!

Gegenbeispiel: $g = \text{nub}$, $f = \text{const } 1$ und $l = [1, 2]$.

Warum $g (\text{map } f \ l) = \text{map } f \ (g \ l)$, intuitiv gesehen

- $g :: [\alpha] \rightarrow [\alpha]$ muss für jede mögliche Instanziierung von α einheitlich arbeiten.
- Die Ausgabeliste kann nur Elemente der Eingabe l enthalten.
- Welche, und in welcher Reihenfolge/Vielfachheit, kann lediglich von l abhängen.
- Die einzig mögliche Grundlage zur Entscheidung ist die Länge von l .
- Die Listen $(\text{map } f \ l)$ und l haben stets die selbe Länge.
- Also wählt g stets „die selben“ Elemente aus $(\text{map } f \ l)$ wie es dies aus l tut, außer dass im ersten Fall die entsprechenden Abbilder unter f ausgegeben werden.
- Also ist $(g (\text{map } f \ l))$ gleich $(\text{map } f \ (g \ l))$.
- Argumentation gelungen!

Warum $g (\text{map } f \ l) = \text{map } f (g \ l)$, intuitiv gesehen

- $g :: \text{Eq } \alpha \Rightarrow [\alpha] \rightarrow [\alpha]$ muss für jede mögliche Instanziierung von α **einheitlich** arbeiten.

Warum $g (\text{map } f \ l) = \text{map } f (g \ l)$, intuitiv gesehen

- $g :: \text{Eq } \alpha \Rightarrow [\alpha] \rightarrow [\alpha]$ muss für jede mögliche Instanziierung von α einheitlich arbeiten.
- Die Ausgabeliste kann nur **Elemente der Eingabe l** enthalten.

Warum $g (\text{map } f \ l) = \text{map } f (g \ l)$, intuitiv gesehen

- $g :: \text{Eq } \alpha \Rightarrow [\alpha] \rightarrow [\alpha]$ muss für jede mögliche Instanziierung von α einheitlich arbeiten.
- Die Ausgabeliste kann nur **Elemente der Eingabe l** enthalten.
- Welche, und in welcher Reihenfolge/Vielfachheit, kann lediglich **von l abhängen**.

Warum $g (\text{map } f \ l) = \text{map } f \ (g \ l)$, intuitiv gesehen

- $g :: \text{Eq } \alpha \Rightarrow [\alpha] \rightarrow [\alpha]$ muss für jede mögliche Instanziierung von α einheitlich arbeiten.
- Die Ausgabeliste kann nur Elemente der Eingabe l enthalten.
- Welche, und in welcher Reihenfolge/Vielfachheit, kann lediglich **von l abhängen**.
- Die einzig mögliche Grundlage zur Entscheidung ist die **Länge von l** .

Warum $g (\text{map } f \ l) = \text{map } f \ (g \ l)$, intuitiv gesehen

- $g :: \text{Eq } \alpha \Rightarrow [\alpha] \rightarrow [\alpha]$ muss für jede mögliche Instanziierung von α einheitlich arbeiten.
- Die Ausgabeliste kann nur Elemente der Eingabe l enthalten.
- Welche, und in welcher Reihenfolge/Vielfachheit, kann lediglich **von l abhängen**.
- Die einzig mögliche Grundlage zur Entscheidung ist die **Länge von l** .

⚡ Falsch! Auch möglich: Elemente von l auf Gleichheit testen.

Warum $g (\text{map } f \ l) = \text{map } f \ (g \ l)$, intuitiv gesehen

- $g :: \text{Eq } \alpha \Rightarrow [\alpha] \rightarrow [\alpha]$ muss für jede mögliche Instanziierung von α einheitlich arbeiten.
 - Die Ausgabeliste kann nur Elemente der Eingabe l enthalten.
 - Welche, und in welcher Reihenfolge/Vielfachheit, kann lediglich von l abhängen.
 - Die einzig mögliche Grundlage zur Entscheidung ist die **Länge von l** .
- ⚡ Falsch! Auch möglich: Elemente von l auf Gleichheit testen.
- Die Listen $(\text{map } f \ l)$ und l haben stets **die selbe Länge**.

Warum $g (\text{map } f \ l) = \text{map } f \ (g \ l)$, intuitiv gesehen

- $g :: \text{Eq } \alpha \Rightarrow [\alpha] \rightarrow [\alpha]$ muss für jede mögliche Instanziierung von α einheitlich arbeiten.
- Die Ausgabeliste kann nur Elemente der Eingabe l enthalten.
- Welche, und in welcher Reihenfolge/Vielfachheit, kann lediglich von l abhängen.
- Die einzig mögliche Grundlage zur Entscheidung ist die Länge von l .

⚡ Falsch! Auch möglich: Elemente von l auf Gleichheit testen.

- Die Listen $(\text{map } f \ l)$ und l haben stets die selbe Länge.

Aber Gleichheitstests innerhalb beider Listen liefern nicht notwendigerweise immer das selbe Ergebnis!

Warum $g (\text{map } f \ l) = \text{map } f \ (g \ l)$, intuitiv gesehen

- $g :: \text{Eq } \alpha \Rightarrow [\alpha] \rightarrow [\alpha]$ muss für jede mögliche Instanziierung von α einheitlich arbeiten.
- Die Ausgabeliste kann nur Elemente der Eingabe l enthalten.
- Welche, und in welcher Reihenfolge/Vielfachheit, kann lediglich von l abhängen.
- Die einzig mögliche Grundlage zur Entscheidung ist die Länge von l .

⚡ Falsch! Auch möglich: Elemente von l auf Gleichheit testen.

- Die Listen $(\text{map } f \ l)$ und l haben stets die selbe Länge.

Aber Gleichheitstests innerhalb beider Listen liefern nicht notwendigerweise immer das selbe Ergebnis!

Sie tun es, wenn f „injektiv“ ist.

Warum $g (\text{map } f \ l) = \text{map } f \ (g \ l)$, intuitiv gesehen

- $g :: \text{Eq } \alpha \Rightarrow [\alpha] \rightarrow [\alpha]$ muss für jede mögliche Instanziierung von α einheitlich arbeiten.
- Die Ausgabeliste kann nur Elemente der Eingabe l enthalten.
- Welche, und in welcher Reihenfolge/Vielfachheit, kann lediglich von l abhängen.
- Die einzig mögliche Grundlage zur Entscheidung ist die Länge von l .

⚡ Falsch! Auch möglich: Elemente von l auf Gleichheit testen.

- Die Listen $(\text{map } f \ l)$ und l haben stets die selbe Länge.

Aber Gleichheitstests innerhalb beider Listen liefern nicht notwendigerweise immer das selbe Ergebnis!

Sie tun es, wenn f „injektiv“ ist.

- Also wählt g stets „die selben“ Elemente aus $(\text{map } f \ l)$ wie es dies aus l tut,

Warum $g (\text{map } f \ l) = \text{map } f \ (g \ l)$, intuitiv gesehen

- $g :: \text{Eq } \alpha \Rightarrow [\alpha] \rightarrow [\alpha]$ muss für jede mögliche Instanziierung von α einheitlich arbeiten.
- Die Ausgabeliste kann nur Elemente der Eingabe l enthalten.
- Welche, und in welcher Reihenfolge/Vielfachheit, kann lediglich von l abhängen.
- Die einzig mögliche Grundlage zur Entscheidung ist die Länge von l .

⚡ Falsch! Auch möglich: Elemente von l auf Gleichheit testen.

- Die Listen $(\text{map } f \ l)$ und l haben stets die selbe Länge.

Aber Gleichheitstests innerhalb beider Listen liefern nicht notwendigerweise immer das selbe Ergebnis!

Sie tun es, wenn f „injektiv“ ist.

- Also wählt g stets „die selben“ Elemente aus $(\text{map } f \ l)$ wie es dies aus l tut, außer dass im ersten Fall die entsprechenden Abbilder unter f ausgegeben werden.

Warum $g (\text{map } f \ l) = \text{map } f (g \ l)$, intuitiv gesehen

- $g :: \text{Eq } \alpha \Rightarrow [\alpha] \rightarrow [\alpha]$ muss für jede mögliche Instanziierung von α einheitlich arbeiten.
- Die Ausgabeliste kann nur Elemente der Eingabe l enthalten.
- Welche, und in welcher Reihenfolge/Vielfachheit, kann lediglich von l abhängen.
- Die einzig mögliche Grundlage zur Entscheidung ist die Länge von l .

⚡ Falsch! Auch möglich: Elemente von l auf Gleichheit testen.

- Die Listen $(\text{map } f \ l)$ und l haben stets die selbe Länge.

Aber Gleichheitstests innerhalb beider Listen liefern nicht notwendigerweise immer das selbe Ergebnis!

Sie tun es, wenn f „injektiv“ ist.

- Also wählt g stets „die selben“ Elemente aus $(\text{map } f \ l)$ wie es dies aus l tut, außer dass im ersten Fall die entsprechenden **Abbilder unter f** ausgegeben werden.
- Also ist $(g (\text{map } f \ l))$ gleich $(\text{map } f (g \ l))$.

Warum $g (\text{map } f \ l) = \text{map } f (g \ l)$, intuitiv gesehen

- $g :: \text{Eq } \alpha \Rightarrow [\alpha] \rightarrow [\alpha]$ muss für jede mögliche Instanziierung von α einheitlich arbeiten.
- Die Ausgabeliste kann nur Elemente der Eingabe l enthalten.
- Welche, und in welcher Reihenfolge/Vielfachheit, kann lediglich von l abhängen.
- Die einzig mögliche Grundlage zur Entscheidung ist die Länge von l .

⚡ Falsch! Auch möglich: Elemente von l auf Gleichheit testen.

- Die Listen $(\text{map } f \ l)$ und l haben stets die selbe Länge.

Aber Gleichheitstests innerhalb beider Listen liefern nicht notwendigerweise immer das selbe Ergebnis!

Sie tun es, wenn f „injektiv“ ist.

- Also wählt g stets „die selben“ Elemente aus $(\text{map } f \ l)$ wie es dies aus l tut, außer dass im ersten Fall die entsprechenden Abbilder unter f ausgegeben werden.
- Also ist $(g (\text{map } f \ l))$ gleich $(\text{map } f (g \ l))$.
- Dies liefert ein verfeinertes freies Theorem!

Formaler: Dictionary Translation

Jedes

$$g :: \text{Eq } \alpha \Rightarrow [\alpha] \rightarrow [\alpha]$$

kann als ein

$$g' :: (\alpha \rightarrow \alpha \rightarrow \text{Bool}) \rightarrow [\alpha] \rightarrow [\alpha]$$

angesehen werden

Formaler: Dictionary Translation

Jedes

$$g :: \text{Eq } \alpha \Rightarrow [\alpha] \rightarrow [\alpha]$$

kann als ein

$$g' :: (\alpha \rightarrow \alpha \rightarrow \text{Bool}) \rightarrow [\alpha] \rightarrow [\alpha]$$

angesehen werden, wobei für jeden Instanztyp τ von Eq,

$$g_\tau = g'_\tau (==)_\tau$$

Formaler: Dictionary Translation

Jedes

$$g :: \text{Eq } \alpha \Rightarrow [\alpha] \rightarrow [\alpha]$$

kann als ein

$$g' :: (\alpha \rightarrow \alpha \rightarrow \text{Bool}) \rightarrow [\alpha] \rightarrow [\alpha]$$

angesehen werden, wobei für jeden Instanztyp τ von Eq,

$$g_\tau = g'_\tau (==)_\tau$$

Das freie Theorem für g' besagt, dass

$$g' p (\text{map } f l) = \text{map } f (g' q l),$$

vorausgesetzt, dass für alle x und y , $q x y = p (f x) (f y)$.

Formaler: Dictionary Translation

Jedes

$$g :: \text{Eq } \alpha \Rightarrow [\alpha] \rightarrow [\alpha]$$

kann als ein

$$g' :: (\alpha \rightarrow \alpha \rightarrow \text{Bool}) \rightarrow [\alpha] \rightarrow [\alpha]$$

angesehen werden, wobei für jeden Instanztyp τ von `Eq`,

$$g_\tau = g'_\tau (==)_\tau$$

Das freie Theorem für g' besagt, dass

$$g' p (\text{map } f l) = \text{map } f (g' q l),$$

vorausgesetzt, dass für alle x und y , $q x y = p (f x) (f y)$.

Dies bedeutet, dass

$$g' (==) (\text{map } f l) = \text{map } f (g' (==) l),$$

vorausgesetzt, dass für alle x und y , $x == y$ gdw. $(f x) == (f y)$.

Formaler: Dictionary Translation

Jedes

$$g :: \text{Eq } \alpha \Rightarrow [\alpha] \rightarrow [\alpha]$$

kann als ein

$$g' :: (\alpha \rightarrow \alpha \rightarrow \text{Bool}) \rightarrow [\alpha] \rightarrow [\alpha]$$

angesehen werden, wobei für jeden Instanztyp τ von Eq,

$$g_\tau = g'_\tau (==)_\tau$$

Das freie Theorem für g' besagt, dass

$$g' p (\text{map } f l) = \text{map } f (g' q l),$$

vorausgesetzt, dass für alle x und y , $q x y = p (f x) (f y)$.

Dies bedeutet, dass

$$g (\text{map } f l) = \text{map } f (g l),$$

vorausgesetzt, dass für alle x und y , $x == y$ gdw. $(f x) == (f y)$.

Weiteres wichtiges Konstrukt: Allgemeine Rekursion

Wir hatten für jedes

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

behauptet, dass

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

für beliebige p , h und l .

Weiteres wichtiges Konstrukt: Allgemeine Rekursion

Wir hatten für jedes

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

behauptet, dass

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

für beliebige p , h und l .

Was ist mit

$$\begin{aligned} g &:: (a \rightarrow \text{Bool}) \rightarrow [a] \rightarrow [a] \\ g \ p \ l &= [\text{head } (g \ p \ l)] \quad ? \end{aligned}$$

Weiteres wichtiges Konstrukt: Allgemeine Rekursion

Wir hatten für jedes

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

behauptet, dass

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

für beliebige p , h und l .

Was ist mit

$$\begin{aligned} g &:: (a \rightarrow \text{Bool}) \rightarrow [a] \rightarrow [a] \\ g\ p\ l &= [\text{head}\ (g\ p\ l)] \quad ? \end{aligned}$$

Obiges freies Theorem schlägt fehl!

Gegenbeispiel: $p = \text{id}$, $h = \text{const True}$ und $l = []$.

Versuch einer Argumentation

- $g :: (a \rightarrow \text{Bool}) \rightarrow [a] \rightarrow [a]$ muss für jede mögliche Instanziierung von a einheitlich arbeiten.
- Die Ausgabeliste kann nur Elemente der Eingabe l enthalten.
- Welche, und in welcher Reihenfolge/Vielfachheit, kann lediglich von l und dem Eingabepredikat p abhängen.
- Die einzig möglichen Grundlagen zur Entscheidung sind die Länge von l und die Ergebnisse von p auf Elementen von l .
- 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 .
- 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.
- Also ist $(g \ p \ (\text{map } h \ l))$ gleich $(\text{map } h \ (g \ (p \circ h) \ l))$.
- **Genau das wollten wir beweisen!**

Versuch einer Argumentation

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

Versuch einer Argumentation

- $g :: (a \rightarrow \text{Bool}) \rightarrow [a] \rightarrow [a]$ muss für jede mögliche Instanziierung von a einheitlich arbeiten.
- Die Ausgabeliste kann nur **Elemente der Eingabe /** enthalten.

Versuch einer Argumentation

- $g :: (a \rightarrow \text{Bool}) \rightarrow [a] \rightarrow [a]$ muss für jede mögliche Instanziierung von a einheitlich arbeiten.
- Die Ausgabeliste kann nur **Elemente der Eingabe /** enthalten.

⚡ **Falsch! Auch möglich: \perp .**

Versuch einer Argumentation

- $g :: (a \rightarrow \text{Bool}) \rightarrow [a] \rightarrow [a]$ muss für jede mögliche Instanziierung von a einheitlich arbeiten.
- Die Ausgabeliste kann nur **Elemente der Eingabe I** enthalten.

⚡ Falsch! **Auch möglich: \perp .**

- Welche, und in welcher Reihenfolge/Vielfachheit, kann lediglich **von I und dem Eingabepredikat p** abhängen.

Versuch einer Argumentation

- $g :: (a \rightarrow \text{Bool}) \rightarrow [a] \rightarrow [a]$ muss für jede mögliche Instanziierung von a einheitlich arbeiten.
- Die Ausgabeliste kann nur Elemente der Eingabe I enthalten.

⚡ Falsch! Auch möglich: \perp .

- Welche, und in welcher Reihenfolge/Vielfachheit, kann lediglich **von I und dem Eingabepredikat p abhängen**.
- Die einzig möglichen Grundlagen zur Entscheidung sind die **Länge von I** und die **Ergebnisse von p auf Elementen von I** .

Versuch einer Argumentation

- $g :: (a \rightarrow \text{Bool}) \rightarrow [a] \rightarrow [a]$ muss für jede mögliche Instanziierung von a einheitlich arbeiten.
- Die Ausgabeliste kann nur Elemente der Eingabe l enthalten.

⚡ Falsch! Auch möglich: \perp .

- Welche, und in welcher Reihenfolge/Vielfachheit, kann lediglich von l und dem Eingabepredikat p abhängen.
- Die einzig möglichen Grundlagen zur Entscheidung sind die Länge von l und die Ergebnisse von p auf Elementen von l .

⚡ Falsch! Weitere mögliche Grundlage: Ergebnis von p auf \perp .

Versuch einer Argumentation

- $g :: (a \rightarrow \text{Bool}) \rightarrow [a] \rightarrow [a]$ muss für jede mögliche Instanziierung von a einheitlich arbeiten.
- Die Ausgabeliste kann nur Elemente der Eingabe l enthalten.

⚡ Falsch! Auch möglich: \perp .

- Welche, und in welcher Reihenfolge/Vielfachheit, kann lediglich von l und dem Eingabepredikat p abhängen.
- Die einzig möglichen Grundlagen zur Entscheidung sind die **Länge von l** und die Ergebnisse von p auf Elementen von l .

⚡ Falsch! Weitere mögliche Grundlage: Ergebnis von p auf \perp .

- Aber, die Listen $(\text{map } h \ l)$ und l haben stets **die selbe Länge**.

Versuch einer Argumentation

- $g :: (a \rightarrow \text{Bool}) \rightarrow [a] \rightarrow [a]$ muss für jede mögliche Instanziierung von a einheitlich arbeiten.
- Die Ausgabeliste kann nur Elemente der Eingabe l enthalten.

⚡ Falsch! Auch möglich: \perp .

- Welche, und in welcher Reihenfolge/Vielfachheit, kann lediglich von l und dem Eingabepredikat p abhängen.
- Die einzig möglichen Grundlagen zur Entscheidung sind die Länge von l und die **Ergebnisse von p auf Elementen von l** .

⚡ Falsch! Weitere mögliche Grundlage: Ergebnis von p 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 .

Versuch einer Argumentation

- $g :: (a \rightarrow \text{Bool}) \rightarrow [a] \rightarrow [a]$ muss für jede mögliche Instanziierung von a einheitlich arbeiten.
- Die Ausgabeliste kann nur Elemente der Eingabe l enthalten.

⚡ Falsch! Auch möglich: \perp .

- Welche, und in welcher Reihenfolge/Vielfachheit, kann lediglich von l und dem Eingabepredikat p abhängen.
- Die einzig möglichen Grundlagen zur Entscheidung sind die Länge von l und die Ergebnisse von p auf Elementen von l .

⚡ Falsch! **Weitere mögliche Grundlage: Ergebnis von p 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 $(p \circ h)$ auf \perp , vorausgesetzt h ist strikt ($h \ \perp = \perp$).

Versuch einer Argumentation

- Die Ausgabeliste kann nur Elemente der Eingabe l enthalten.

⚡ Falsch! Auch möglich: \perp .

- Welche, und in welcher Reihenfolge/Vielfachheit, kann lediglich von l und dem Eingabepredikat p abhängen.
- Die einzig möglichen Grundlagen zur Entscheidung sind die Länge von l und die Ergebnisse von p auf Elementen von l .

⚡ Falsch! Weitere mögliche Grundlage: Ergebnis von p 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 $(p \circ h)$ auf \perp , vorausgesetzt h ist strikt ($h \ \perp = \perp$).

- Also wählt g mit p stets „die selben“ Elemente aus $(\text{map } h \ l)$ wie es g mit $(p \circ h)$ aus l tut,

Versuch einer Argumentation

- Die Ausgabeliste kann nur Elemente der Eingabe l enthalten.

⚡ Falsch! Auch möglich: \perp .

- Welche, und in welcher Reihenfolge/Vielfachheit, kann lediglich von l und dem Eingabepredikat p abhängen.
- Die einzig möglichen Grundlagen zur Entscheidung sind die Länge von l und die Ergebnisse von p auf Elementen von l .

⚡ Falsch! Weitere mögliche Grundlage: Ergebnis von p 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 $(p \circ h)$ auf \perp , vorausgesetzt h ist strikt ($h \ \perp = \perp$).

- 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.

Versuch einer Argumentation

⚡ Falsch! Auch möglich: \perp .

- Welche, und in welcher Reihenfolge/Vielfachheit, kann lediglich von l und dem Eingabepredikat p abhängen.
- Die einzig möglichen Grundlagen zur Entscheidung sind die Länge von l und die Ergebnisse von p auf Elementen von l .

⚡ Falsch! Weitere mögliche Grundlage: Ergebnis von p auf \perp .

- Aber, die Listen (`map h l`) und l haben stets die selbe Länge.
- Und Anwendung von p auf ein Element von (`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 $(p \circ h)$ auf \perp , vorausgesetzt h ist strikt ($h \perp = \perp$).

- Also wählt g mit p stets „die selben“ Elemente aus (`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.

Aber sie könnten auch, an gleichen Positionen, \perp ausgeben.

Versuch einer Argumentation

- Welche, und in welcher Reihenfolge/Vielfachheit, kann lediglich von l und dem Eingabepredikat p abhängen.
- Die einzig möglichen Grundlagen zur Entscheidung sind die Länge von l und die Ergebnisse von p auf Elementen von l .

⚡ Falsch! Weitere mögliche Grundlage: Ergebnis von p 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 $(p \circ h)$ auf \perp , vorausgesetzt h ist strikt ($h \ \perp = \perp$).

- 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.

Aber sie könnten auch, an gleichen Positionen, \perp ausgeben.

- Also ist $(g \ p \ (\text{map } h \ l))$ gleich $(\text{map } h \ (g \ (p \circ h) \ l))$,

Versuch einer Argumentation

- Die einzig möglichen Grundlagen zur Entscheidung sind die Länge von l und die Ergebnisse von p auf Elementen von l .

⚡ Falsch! Weitere mögliche Grundlage: Ergebnis von p 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 $(p \circ h)$ auf \perp , vorausgesetzt h ist strikt ($h \ \perp = \perp$).

- 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.

Aber sie könnten auch, an gleichen Positionen, \perp ausgeben.

- Also ist $(g \ p \ (\text{map } h \ l))$ gleich $(\text{map } h \ (g \ (p \circ h) \ l))$, wenn h strikt ist.

Versuch einer Argumentation

- Die einzig möglichen Grundlagen zur Entscheidung sind die Länge von l und die Ergebnisse von p auf Elementen von l .

⚡ Falsch! Weitere mögliche Grundlage: Ergebnis von p 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 $(p \circ h)$ auf \perp , **vorausgesetzt h ist strikt** ($h \ \perp = \perp$).

- 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.

Aber sie könnten auch, an gleichen Positionen, \perp ausgeben.

- Also ist $(g \ p \ (\text{map } h \ l))$ gleich $(\text{map } h \ (g \ (p \circ h) \ l))$,
wenn h strikt ist.

- Die Erweiterung des Kalküls um **vordefinierte Funktionen** stellte keine Veränderung der Ausdrucksmächtigkeit dar, da diese Funktionen auch simuliert werden konnten.
- Um die Äquivalenz des Kalküls zur Turing-Maschine zu zeigen, fehlt aber noch die Möglichkeit **Rekursion** auszudrücken.
- Problem: Die Funktionen im λ -Kalkül haben eigtl. keinen Namen.

$(\lambda \text{ id } . \dots) (\lambda x . x)$

Im Funktionskörper ist unter dem Variablennamen *id* die Funktion $(\lambda x . x)$ bekannt.

Der Bezeichnung der Funktion $(\lambda x . x)$ mit *id* ist aber nur lokal und somit in anderen Funktionen nicht bekannt.

- Wegen der **fehlenden** globalen Funktionsbezeichner kann z.B. die rekursive Fakultätsfunktion nicht direkt als λ -Ausdruck definiert werden:

$$\text{fak } n := \text{if } (n = 0) \text{ then } 1 \text{ else } n \times \text{fak } (n-1)$$

- Statt der obigen Definition betrachten wir nun eine leicht modifizierte **nicht-rekursive** Variante fak' :

$$\text{fak}' \text{ f } n := \text{if } (n = 0) \text{ then } 1 \text{ else } n \times \text{f } (n-1)$$

es gilt:

$$\text{fak}' (\text{fak}) = \text{fak}$$

$$\text{fak}' \text{ f } n \dots \equiv \lambda \text{ f } n . \dots$$

Da fak' nicht-rekursiv ist,
gibt es einen dazugehörigen
 λ -Ausdruck.

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.

Zusammenfassung:

- Um eine rekursive **Funktion H** zu definieren, betrachten wir stattdessen die nicht-rekursive **Variante H'** .
- Die Funktionalität von H liefert dann der λ -Ausdruck **$Y(H')$** .

\Rightarrow Zahlen, Arithmetik, Logik, Rekursion im λ -Kalkül **ausdrückbar!**
- Turing vollständig -

Mit dem typisierten λ -Kalkül wird jetzt eine echte Steigerung der Ausdrucksmächtigkeit vorgestellt....