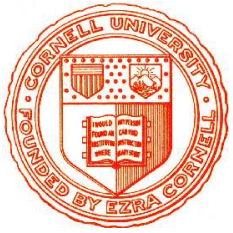
# Automatisierte Logik und Programmierung

## Einheit 7




## Abhängige Datentypen


1. Motivation
2. Syntax, Typzugehörigkeit, Beweisregeln
3. Ausdruckskraft
4. Universen und Abhängigkeit

# Erweitere Ausdruckskraft der Typentheorie

- **Einfache Typisierung von $\mathbb{B}$ ist zu restriktiv**

# Erweitere Ausdruckskraft der Typentheorie

- **Einfache Typisierung von $\mathbb{B}$ ist zu restriktiv**
  - Typschema für $\mathbb{B}$ ist $\mathtt{X} \to \mathtt{X} \to \mathtt{X}$,
    aber in $\textsf{if}\ b\ \textsf{then}\ s\ \textsf{else}\ t \equiv b\,s\,t$ hängt $\mathtt{X}$ von $s$ und $t$ ab

# Erweitere Ausdruckskraft der Typentheorie

- **Einfache Typisierung von $\mathbb{B}$ ist zu restriktiv**
  - Typschema für $\mathbb{B}$ ist $\mathtt{X} \to \mathtt{X} \to \mathtt{X}$,
    aber in $\textsf{if}\ b\ \textsf{then}\ s\ \textsf{else}\ t \equiv b\,s\,t$ hängt $\mathtt{X}$ von $s$ und $t$ ab
  - Einfache Typentheorie kann diese Abhängigkeit nicht beschreiben

# Erweitere Ausdruckskraft der Typentheorie

- **Einfache Typisierung von $\mathbb{B}$ ist zu restriktiv**
  - Typschema für $\mathbb{B}$ ist $\mathtt{X} \to \mathtt{X} \to \mathtt{X}$,
    aber in **if** $b$ **then** $s$ **else** $t \equiv b\,s\,t$ hängt $\mathtt{X}$ von $s$ und $t$ ab
  - Einfache Typentheorie kann diese Abhängigkeit nicht beschreiben
- **Church Numerals sind Funktionen und Daten**

# Erweitere Ausdruckskraft der Typentheorie

- **Einfache Typisierung von $\mathbb{B}$ ist zu restriktiv**
  - Typschema für $\mathbb{B}$ ist $\mathtt{X}\to\mathtt{X}\to\mathtt{X}$,
    aber in $\textsf{if}\ b\ \textsf{then}\ s\ \textsf{else}\ t \equiv b\,s\,t$ hängt $\mathtt{X}$ von $s$ und $t$ ab
  - Einfache Typentheorie kann diese Abhängigkeit nicht beschreiben
- **Church Numerals sind Funktionen und Daten**
  - Typschema für $\mathbb{N}$ ist $(\mathtt{X}\to\mathtt{X})\to\mathtt{X}\to\mathtt{X}$, aber
    in $\textsf{PRs}[\mathit{base},h] \equiv \lambda\mathtt{n}.\,\mathtt{n}\ \ h\ \ \mathit{base}$ muß $n \in \mathbb{N} = (\mathbb{N}\to\mathbb{N})\to\mathbb{N}\to\mathbb{N}$ sein

# Erweitere Ausdruckskraft der Typentheorie

- **Einfache Typisierung von $\mathbb{B}$ ist zu restriktiv**
  - Typschema für $\mathbb{B}$ ist $\mathtt{X}\to\mathtt{X}\to\mathtt{X}$,
    aber in $\textsf{if}\ b\ \textsf{then}\ s\ \textsf{else}\ t \equiv b\,s\,t$ hängt $\mathtt{X}$ von $s$ und $t$ ab
  - Einfache Typentheorie kann diese Abhängigkeit nicht beschreiben
- **Church Numerals sind Funktionen und Daten**
  - Typschema für $\mathbb{N}$ ist $(\mathtt{X}\to\mathtt{X})\to\mathtt{X}\to\mathtt{X}$, aber
    in $\textsf{PRs}[\mathit{base},h] \equiv \lambda\mathtt{n}.\,\mathtt{n}\ h\ \mathit{base}$ muß $n \in \mathbb{N} = (\mathbb{N}\to\mathbb{N})\to\mathbb{N}\to\mathbb{N}$ sein
  - Einfache Typentheorie kann diese Doppelrolle von $\mathbb{N}$ nicht beschreiben

# Erweitere Ausdruckskraft der Typentheorie

- **Einfache Typisierung von $\mathbb{B}$ ist zu restriktiv**
  - Typschema für $\mathbb{B}$ ist $\mathtt{X}\to\mathtt{X}\to\mathtt{X}$,
    aber in $\textsf{if}\ b\ \textsf{then}\ s\ \textsf{else}\ t \equiv b\,s\,t$ hängt $\mathtt{X}$ von $s$ und $t$ ab
  - Einfache Typentheorie kann diese Abhängigkeit nicht beschreiben
- **Church Numerals sind Funktionen und Daten**
  - Typschema für $\mathbb{N}$ ist $(\mathtt{X}\to\mathtt{X})\to\mathtt{X}\to\mathtt{X}$, aber
    in $\textsf{PRs}[\mathit{base},h] \equiv \lambda\mathtt{n}.\,\mathtt{n}\ h\ \mathit{base}$ muß $n\in\mathbb{N}=(\mathbb{N}\to\mathbb{N})\to\mathbb{N}\to\mathbb{N}$ sein
  - Einfache Typentheorie kann diese Doppelrolle von $\mathbb{N}$ nicht beschreiben
- **Funktionen wie $\lambda x.x$ sind polymorph**
  - Das Typschema *'für alle $T$ gilt $\lambda x.x \in T\to T$'* ist nicht beschreibbar

# Erweitere Ausdruckskraft der Typentheorie

- **Einfache Typisierung von $\mathbb{B}$ ist zu restriktiv**
  - Typschema für $\mathbb{B}$ ist $\mathtt{X}\to\mathtt{X}\to\mathtt{X}$,
    aber in $\textbf{if}\ b\ \textbf{then}\ s\ \textbf{else}\ t \equiv b\,s\,t$ hängt $\mathtt{X}$ von $s$ und $t$ ab
  - Einfache Typentheorie kann diese Abhängigkeit nicht beschreiben
- **Church Numerals sind Funktionen und Daten**
  - Typschema für $\mathbb{N}$ ist $(\mathtt{X}\to\mathtt{X})\to\mathtt{X}\to\mathtt{X}$, aber
    in $\textbf{PRs}[base,h] \equiv \lambda\mathtt{n}.\,\mathtt{n}\ h\ base$ muß $n\in\mathbb{N}=(\mathbb{N}\to\mathbb{N})\to\mathbb{N}\to\mathbb{N}$ sein
  - Einfache Typentheorie kann diese Doppelrolle von $\mathbb{N}$ nicht beschreiben
- **Funktionen wie $\lambda x.x$ sind polymorph**
  - Das Typschema '*für alle $T$ gilt $\lambda x.x \in T\to T$*' ist nicht beschreibbar
- **Formuliere Abhängigkeit in Datentypen**
  - Im Ausdruck $T[x]$ kann der Typ $T$ vom Wert eines $x\in S$ abhängen
  - Parameter $x$ macht den Typ $T$ polymorph

# Polymorphe Typisierung von $\mathbb{B}$ und $\mathbb{N}$

## Zieldatentyp als Parameter in Formalisierung

# Polymorphe Typisierung von $\mathbb{B}$ und $\mathbb{N}$

## Zieldatentyp als Parameter in Formalisierung

- **Neuformalisierung boolescher Operatoren**

# Polymorphe Typisierung von $\mathbb{B}$ und $\mathbb{N}$

## Zieldatentyp als Parameter in Formalisierung

- **Neuformalisierung boolescher Operatoren**
  - $\mathsf{T} \equiv \lambda \mathtt{X}.\lambda \mathtt{x}.\lambda \mathtt{y}.\mathtt{x}$
  - $\mathsf{F} \equiv \lambda \mathtt{X}.\lambda \mathtt{x}.\lambda \mathtt{y}.\mathtt{y}$
  - **if** $b$ **then** $s$ **else** $t : X \equiv b\,X\,s\,t$

# Polymorphe Typisierung von $\mathbb{B}$ und $\mathbb{N}$

## Zieldatentyp als Parameter in Formalisierung

- **Neuformalisierung boolescher Operatoren**
  - $\mathsf{T} \equiv \lambda \mathtt{X}.\lambda \mathtt{x}.\lambda \mathtt{y}.\mathtt{x}$
  - $\mathsf{F} \equiv \lambda \mathtt{X}.\lambda \mathtt{x}.\lambda \mathtt{y}.\mathtt{y}$
  - $\mathsf{if}\ b\ \mathsf{then}\ s\ \mathsf{else}\ t : X \equiv b\, X\, s\, t$
  - Führt zur Typisierung $\mathbb{B} \equiv \mathtt{X}{:}\mathbb{U} \to \mathtt{X} \to \mathtt{X} \to \mathtt{X}$
    wobei $\mathbb{U}$ Repräsentant der Menge aller Datentypen
  - Typisierung von $s$ und $t$ liefert Wert für den Parameter $\mathtt{X}$

# Polymorphe Typisierung von $\mathbb{B}$ und $\mathbb{N}$

## Zieldatentyp als Parameter in Formalisierung

- **Neuformalisierung boolescher Operatoren**
  - $\cdot$ **T** $\equiv \lambda \mathtt{X}.\lambda \mathtt{x}.\lambda \mathtt{y}.\mathtt{x}$
  - $\cdot$ **F** $\equiv \lambda \mathtt{X}.\lambda \mathtt{x}.\lambda \mathtt{y}.\mathtt{y}$
  - $\cdot$ **if** $b$ **then** $s$ **else** $t : X \equiv b\,X\,s\,t$

  – Führt zur Typisierung $\mathbb{B} \equiv \mathtt{X}:\mathbb{U} \to \mathtt{X} \to \mathtt{X} \to \mathtt{X}$
  wobei $\mathbb{U}$ Repräsentant der Menge aller Datentypen

  – Typisierung von $s$ und $t$ liefert Wert für den Parameter $\mathtt{X}$
- **Church-Numerals mit Abhängigkeiten**

# Polymorphe Typisierung von $\mathbb{B}$ und $\mathbb{N}$

## Zieldatentyp als Parameter in Formalisierung

- **Neuformalisierung boolescher Operatoren**
  - $\mathsf{T} \equiv \lambda \mathtt{X}.\lambda \mathtt{x}.\lambda \mathtt{y}.\mathtt{x}$
  - $\mathsf{F} \equiv \lambda \mathtt{X}.\lambda \mathtt{x}.\lambda \mathtt{y}.\mathtt{y}$
  - $\mathsf{if}\ b\ \mathsf{then}\ s\ \mathsf{else}\ t : X \equiv b\, X\, s\, t$
  - Führt zur Typisierung $\mathbb{B} \equiv \mathtt{X}{:}\mathbb{U} \to \mathtt{X} \to \mathtt{X} \to \mathtt{X}$
    wobei $\mathbb{U}$ Repräsentant der Menge aller Datentypen
  - Typisierung von $s$ und $t$ liefert Wert für den Parameter $\mathtt{X}$
- **Church-Numerals mit Abhängigkeiten**
  - $\overline{n} \equiv \lambda \mathtt{X}.\lambda \mathtt{f}.\lambda \mathtt{x}.\ \mathtt{f}^n\,\mathtt{x}$
  - $\mathsf{PRs}[base, h] \equiv \lambda \mathtt{n}.\ \mathtt{n}(\mathbb{N})\ h\ base$

# Polymorphe Typisierung von $\mathbb{B}$ und $\mathbb{N}$

## Zieldatentyp als Parameter in Formalisierung

- **Neuformalisierung boolescher Operatoren**
  - $\cdot$ **T** $\equiv \lambda \mathtt{X}.\lambda \mathtt{x}.\lambda \mathtt{y}.\mathtt{x}$
  - $\cdot$ **F** $\equiv \lambda \mathtt{X}.\lambda \mathtt{x}.\lambda \mathtt{y}.\mathtt{y}$
  - $\cdot$ **if** $b$ **then** $s$ **else** $t : X \equiv b\, X\, s\, t$
  - Führt zur Typisierung $\mathbb{B} \equiv \mathtt{X}{:}\mathbb{U} \to \mathtt{X} \to \mathtt{X} \to \mathtt{X}$ wobei $\mathbb{U}$ Repräsentant der Menge aller Datentypen
  - Typisierung von $s$ und $t$ liefert Wert für den Parameter $\mathtt{X}$
- **Church-Numerals mit Abhängigkeiten**
  - $\cdot$ $\overline{n} \equiv \lambda \mathtt{X}.\lambda \mathtt{f}.\lambda \mathtt{x}.\ \mathtt{f}^n\,\mathtt{x}$
  - $\cdot$ **PRs**$[base, h] \equiv \lambda \mathtt{n}.\,\mathtt{n}(\mathbb{N})\ h\ base$
  - Führt zu $\mathbb{N} \equiv \mathtt{X}{:}\mathbb{U} \to (\mathtt{X} \to \mathtt{X}) \to \mathtt{X} \to \mathtt{X}$
  - Instantiierung von $\mathtt{X}$ mit $\mathbb{N}$ führt zu kontrollierter Selbstreferenz

# Informatik benötigt Abhängige Datentypen

## Datentypen mit internen Querbezügen

# Informatik benötigt Abhängige Datentypen

## Datentypen mit internen Querbezügen

- **Verwendung in abstrakten Maschinenmodellen**

## Datentypen mit internen Querbezügen

- **Verwendung in abstrakten Maschinenmodellen**
  - Endliche Automaten $(Q, \Sigma, q_0, \delta, F)$ mit $q_0 \in Q$, $\delta : Q \times \Sigma \to Q$, $F \subseteq Q$
    - Datentyp von $q_0$ hängt ab vom Wert der ersten beiden Komponenten

# Informatik benötigt Abhängige Datentypen

## Datentypen mit internen Querbezügen

- **Verwendung in abstrakten Maschinenmodellen**
  - Endliche Automaten $(Q, \Sigma, q_0, \delta, F)$ mit $q_0 \in Q$, $\delta : Q \times \Sigma \to Q$, $F \subseteq Q$
    - Datentyp von $q_0$ hängt ab vom Wert der ersten beiden Komponenten
- **Verwendung in der Programmierung**

# Informatik benötigt Abhängige Datentypen

## Datentypen mit internen Querbezügen

- **Verwendung in abstrakten Maschinenmodellen**
  - Endliche Automaten $(Q, \Sigma, q_0, \delta, F)$ mit $q_0 \in Q$, $\delta : Q \times \Sigma \to Q$, $F \subseteq Q$
    - Datentyp von $q_0$ hängt ab vom Wert der ersten beiden Komponenten
- **Verwendung in der Programmierung**
  - Datentyp Record $[l_1 : T_1 ; \ \ldots ; \ f_n : T_n]$
    - Datentyp $T_i$ einer Komponente hängt vom Wert des Labels $l_i$ ab

# Informatik benötigt Abhängige Datentypen

## Datentypen mit internen Querbezügen

- **Verwendung in abstrakten Maschinenmodellen**
  - Endliche Automaten $(Q, \Sigma, q_0, \delta, F)$ mit $q_0 \in Q$, $\delta : Q \times \Sigma \to Q$, $F \subseteq Q$
    - Datentyp von $q_0$ hängt ab vom Wert der ersten beiden Komponenten
- **Verwendung in der Programmierung**
  - Datentyp Record $[l_1 : T_1 ; \ \ldots ; \ f_n : T_n]$
    - Datentyp $T_i$ einer Komponente hängt vom Wert des Labels $l_i$ ab
  - Variant record `type date = Jan of 1..31 | Feb of 1..28 |` ...
    - Datentyp der zweiten Komponente hängt vom Wert der ersten ab

# Informatik benötigt Abhängige Datentypen

## Datentypen mit internen Querbezügen

- **Verwendung in abstrakten Maschinenmodellen**
  - Endliche Automaten $(Q, \Sigma, q_0, \delta, F)$ mit $q_0 \in Q$, $\delta : Q \times \Sigma \to Q$, $F \subseteq Q$
    - Datentyp von $q_0$ hängt ab vom Wert der ersten beiden Komponenten
- **Verwendung in der Programmierung**
  - Datentyp Record $[l_1 : T_1;\ \ldots;\ f_n : T_n]$
    - Datentyp $T_i$ einer Komponente hängt vom Wert des Labels $l_i$ ab
  - Variant record `type date = Jan of 1..31 | Feb of 1..28 |` ...
    - Datentyp der zweiten Komponente hängt vom Wert der ersten ab
  - Modellierung einer optionalen Eingabe abhängig von Testwerten
    - `λx. if x=0 then 4 else λinput.5*input`
    - Datentyp ist je nach Wert von `x` eine Zahl oder eine Funktion

# Formalisierung von Abhängigkeiten

- **Abhängiges Produkt:** $x{:}S \times T[x]$

# Formalisierung von Abhängigkeiten

- **Abhängiges Produkt**: $\boldsymbol{x{:}S \times T[x]}$
  - Elemente sind Paare $(s, t)$
  - Typ der zweiten Komponente $t$ hängt vom Wert $s \in S$ der ersten ab

# Formalisierung von Abhängigkeiten

- **Abhängiges Produkt**: $x{:}S \times T[x]$
  - Elemente sind Paare $(s, t)$
  - Typ der zweiten Komponente $t$ hängt vom Wert $s \in S$ der ersten ab
- **Abhängiger Funktionenraum**: $x{:}S \to T[x]$

# Formalisierung von Abhängigkeiten

- **Abhängiges Produkt**: $\boldsymbol{x{:}S \times T[x]}$
  - Elemente sind Paare $(s, t)$
  - Typ der zweiten Komponente $t$ hängt vom Wert $s \in S$ der ersten ab
- **Abhängiger Funktionenraum**: $\boldsymbol{x{:}S \to T[x]}$
  - Elemente sind Funktionen $\lambda x.t$
  - Typ $T[x]$ des Bilds $t$ hängt vom Wert der Eingabe $s \in S$ ab

# Formalisierung von Abhängigkeiten

- **Abhängiges Produkt**: $\boldsymbol{x{:}S \times T[x]}$
  - Elemente sind Paare $(s, t)$
  - Typ der zweiten Komponente $t$ hängt vom Wert $s \in S$ der ersten ab
- **Abhängiger Funktionenraum**: $\boldsymbol{x{:}S \to T[x]}$
  - Elemente sind Funktionen $\lambda x.t$
  - Typ $T[x]$ des Bilds $t$ hängt vom Wert der Eingabe $s \in S$ ab
  - Einfachste Erweiterung des bisherigen Typsystems

# Formalisierung von Abhängigkeiten

- **Abhängiges Produkt**: $x{:}S \times T[x]$
  - Elemente sind Paare $(s, t)$
  - Typ der zweiten Komponente $t$ hängt vom Wert $s \in S$ der ersten ab
- **Abhängiger Funktionenraum**: $x{:}S \to T[x]$
  - Elemente sind Funktionen $\lambda x.t$
  - Typ $T[x]$ des Bilds $t$ hängt vom Wert der Eingabe $s \in S$ ab
  - Einfachste Erweiterung des bisherigen Typsystems
- **Komplexere Abhängigkeiten ebenfalls möglich**

# Formalisierung von Abhängigkeiten

- **Abhängiges Produkt: $x{:}S \times T[x]$**
  - Elemente sind Paare $(s, t)$
  - Typ der zweiten Komponente $t$ hängt vom Wert $s \in S$ der ersten ab
- **Abhängiger Funktionenraum: $x{:}S \to T[x]$**
  - Elemente sind Funktionen $\lambda x.t$
  - Typ $T[x]$ des Bilds $t$ hängt vom Wert der Eingabe $s \in S$ ab
  - Einfachste Erweiterung des bisherigen Typsystems
- **Komplexere Abhängigkeiten ebenfalls möglich**
  - Durchschnitt $\cap x : S \,.\, T[x]$ / Vereinigung $\cup x : S \,.\, T[x]$ von Typfamilien
    - Elemente müssen zu allen/einem $T[x]$ für $x \in S$ gehören

# Formalisierung von Abhängigkeiten

- **Abhängiges Produkt: $x{:}S \times T[x]$**
  - Elemente sind Paare $(s, t)$
  - Typ der zweiten Komponente $t$ hängt vom Wert $s \in S$ der ersten ab
- **Abhängiger Funktionenraum: $x{:}S \to T[x]$**
  - Elemente sind Funktionen $\lambda x.t$
  - Typ $T[x]$ des Bilds $t$ hängt vom Wert der Eingabe $s \in S$ ab
  - Einfachste Erweiterung des bisherigen Typsystems
- **Komplexere Abhängigkeiten ebenfalls möglich**
  - Durchschnitt $\cap x : S \,.\, T[x]$ / Vereinigung $\cup x : S \,.\, T[x]$ von Typfamilien
    - Elemente müssen zu allen/einem $T[x]$ für $x \in S$ gehören
  - Abhängiger Durchschnitt $x : S \cap T[x]$
    - Element $s$ muß zu $S$ und zu $T[s]$ gehören (Selbstreferenz!)
    - Gut für Formalisierung von Abstrakten Datentypen und Objekten

# Abhängige Funktionenräume $x{:}S \to T[x]$

- **Erweiterte Syntax von Typ-Ausdrücken**

# Abhängige Funktionenräume $x{:}S{\to}T[x]$

- **Erweiterte Syntax von Typ-Ausdrücken**
  - Typsymbole $T \in \mathcal{T}$
  - $x{:}S{\to}T[x]$ und $(T)$, wobei $S$ und $T$ Typen, $x \in \mathcal{V}$

# Abhängige Funktionenräume $x{:}S{\to}T[x]$

- **Erweiterte Syntax von Typ-Ausdrücken**
  - Typsymbole $T \in \mathcal{T}$
  - $x{:}S{\to}T[x]$ und $(T)$, wobei $S$ und $T$ Typen, $x \in \mathcal{V}$
  - $S{\to}T$ kürzt $x{:}S{\to}T[x]$ ab, wenn $x$ in $T$ nicht frei vorkommt

# Abhängige Funktionenräume $x{:}S{\to}T[x]$

- **Erweiterte Syntax von Typ-Ausdrücken**
  - Typsymbole $T \in \mathcal{T}$
  - $x{:}S{\to}T[x]$ und $(T)$, wobei $S$ und $T$ Typen, $x \in \mathcal{V}$
  - $S{\to}T$ kürzt $x{:}S{\to}T[x]$ ab, wenn $x$ in $T$ nicht frei vorkommt
- **Objekt-Ausdrücke und Prioritäten wie zuvor**

# Abhängige Funktionenräume $x{:}S{\to}T[x]$

- **Erweiterte Syntax von Typ-Ausdrücken**
  - Typsymbole $T \in \mathcal{T}$
  - $x{:}S{\to}T[x]$ und $(T)$, wobei $S$ und $T$ Typen, $x \in \mathcal{V}$
  - $S{\to}T$ kürzt $x{:}S{\to}T[x]$ ab, wenn $x$ in $T$ nicht frei vorkommt
- **Objekt-Ausdrücke und Prioritäten wie zuvor**
- **Modifizierte Typzugehörigkeitsrelation**

# Abhängige Funktionenräume $x{:}S{\to}T[x]$

- **Erweiterte Syntax von Typ-Ausdrücken**
  - Typsymbole $T \in \mathcal{T}$
  - $x{:}S{\to}T[x]$ und $(T)$, wobei $S$ und $T$ Typen, $x \in \mathcal{V}$
  - $S{\to}T$ kürzt $x{:}S{\to}T[x]$ ab, wenn $x$ in $T$ nicht frei vorkommt
- **Objekt-Ausdrücke und Prioritäten wie zuvor**
- **Modifizierte Typzugehörigkeitsrelation**
  - $x \in T$ falls $x \in \mathcal{V}$ und $T$ ein Typ

# Abhängige Funktionenräume $x{:}S{\to}T[x]$

- **Erweiterte Syntax von Typ-Ausdrücken**
  - Typsymbole $T \in \mathcal{T}$
  - $x{:}S{\to}T[x]$ und $(T)$, wobei $S$ und $T$ Typen, $x \in \mathcal{V}$
  - $S{\to}T$ kürzt $x{:}S{\to}T[x]$ ab, wenn $x$ in $T$ nicht frei vorkommt
- **Objekt-Ausdrücke und Prioritäten wie zuvor**
- **Modifizierte Typzugehörigkeitsrelation**
  - $x \in T$ falls $x \in \mathcal{V}$ und $T$ ein Typ
  - $\lambda y\,.\,t \in x{:}S{\to}T[x]$ falls $t[x/y] \in T[x]$ aus $x \in S$ folgt

# Abhängige Funktionenräume $x{:}S{\to}T[x]$

- **Erweiterte Syntax von Typ-Ausdrücken**
  - Typsymbole $T \in \mathcal{T}$
  - $x{:}S{\to}T[x]$ und $(T)$, wobei $S$ und $T$ Typen, $x \in \mathcal{V}$
  - $S{\to}T$ kürzt $x{:}S{\to}T[x]$ ab, wenn $x$ in $T$ nicht frei vorkommt
- **Objekt-Ausdrücke und Prioritäten wie zuvor**
- **Modifizierte Typzugehörigkeitsrelation**
  - $x \in T$ falls $x \in \mathcal{V}$ und $T$ ein Typ
  - $\lambda y\,.\,t \in x{:}S{\to}T[x]$ falls $t[x/y] \in T[x]$ aus $x \in S$ folgt
  - $f\ t \in T$ falls $f \in S{\to}T$ und $t \in S$ für einen Typ $S$

# Abhängige Funktionenräume $x{:}S{\to}T[x]$

- **Erweiterte Syntax von Typ-Ausdrücken**
  - Typsymbole $T \in \mathcal{T}$
  - $x{:}S{\to}T[x]$ und $(T)$, wobei $S$ und $T$ Typen, $x \in \mathcal{V}$
  - $S{\to}T$ kürzt $x{:}S{\to}T[x]$ ab, wenn $x$ in $T$ nicht frei vorkommt
- **Objekt-Ausdrücke und Prioritäten wie zuvor**
- **Modifizierte Typzugehörigkeitsrelation**
  - $x \in T$ falls $x \in \mathcal{V}$ und $T$ ein Typ
  - $\lambda y\,.\,t \in x{:}S{\to}T[x]$ falls $t[x/y] \in T[x]$ aus $x \in S$ folgt
  - $f\,t \in T$ falls $f \in S{\to}T$ und $t \in S$ für einen Typ $S$
  - $(t) \in T$, $t \in (T)$ falls $t \in T$

# Abhängige Funktionenräume $x{:}S{\to}T[x]$

- **Erweiterte Syntax von Typ-Ausdrücken**
  - Typsymbole $T \in \mathcal{T}$
  - $x{:}S{\to}T[x]$ und $(T)$, wobei $S$ und $T$ Typen, $x \in \mathcal{V}$
  - $S{\to}T$ kürzt $x{:}S{\to}T[x]$ ab, wenn $x$ in $T$ nicht frei vorkommt
- **Objekt-Ausdrücke und Prioritäten wie zuvor**
- **Modifizierte Typzugehörigkeitsrelation**
  - $x \in T$ falls $x \in \mathcal{V}$ und $T$ ein Typ
  - $\lambda y\,.\,t \in x{:}S{\to}T[x]$ falls $t[x/y] \in T[x]$ aus $x \in S$ folgt
  - $f\,t \in T$ falls $f \in S{\to}T$ und $t \in S$ für einen Typ $S$
  - $(t) \in T$, $t \in (T)$ falls $t \in T$
- **Typisierbarkeit und prinzipielles Typschema wie zuvor**

# Inferenzregeln für Abhängige Funktionenräume

## Verfeinerte Beweisregeln des einfachen Kalküls

# Inferenzregeln für Abhängige Funktionenräume

## Verfeinerte Beweisregeln des einfachen Kalküls

**applyEq** $S$

$$\frac{\Gamma \vdash f\,t = g\,u \in T \qquad \Gamma \vdash f = g \in S \to T \qquad \Gamma \vdash t = u \in S}{}$$

**lambdaEq** *

$$\frac{\Gamma \vdash \lambda x\,.\,t = \lambda y\,.\,u \in S \to T \qquad \Gamma, x' : S \vdash t[x'/x] = u[x'/y] \in T \qquad \Gamma \vdash S \in \mathbb{U}}{}$$

**reduction**

$$\frac{\Gamma \vdash (\lambda x\,.\,u)\,s = t \in T \qquad \Gamma \vdash u[s/x] = t \in T}{}$$

**declaration** $i$

$$\Gamma, x : T, \Delta \vdash x \in T \quad \hat{=} \quad x = x \in T$$

**functionI**

$$\frac{\Gamma \vdash S \to T \in \mathbb{U} \qquad \Gamma \vdash S \in \mathbb{U} \qquad \Gamma \vdash T \in \mathbb{U}}{}$$

**: Die Umbenennung $[x'/x]$ kann entfallen, wenn $x$ nicht frei in $\Gamma$ vorkommt*

# Inferenzregeln für Abhängige Funktionenräume

## Verfeinerte Beweisregeln des einfachen Kalküls

**applyEq** $S$

$$\frac{\Gamma \vdash f\,t = g\,u \in T[t/x] \qquad \Gamma \vdash f = g \in x:S \to T \qquad \Gamma \vdash t = u \in S}{}$$

**lambdaEq** *

$$\frac{\Gamma \vdash \lambda x.\,t = \lambda y.\,u \in S \to T \qquad \Gamma, x':S \vdash t[x'/x] = u[x'/y] \in T \qquad \Gamma \vdash S \in \mathbb{U}}{}$$

**reduction**

$$\frac{\Gamma \vdash (\lambda x.\,u)\,s = t \in T \qquad \Gamma \vdash u[s/x] = t \in T}{}$$

**declaration** $i$

$$\frac{\Gamma, x:T, \Delta \vdash x \in T}{} \quad \hat{=} \quad x{=}x \in T$$

**functionI**

$$\frac{\Gamma \vdash S \to T \in \mathbb{U} \qquad \Gamma \vdash S \in \mathbb{U} \qquad \Gamma \vdash T \in \mathbb{U}}{}$$

**: Die Umbenennung $[x'/x]$ kann entfallen, wenn $x$ nicht frei in $\Gamma$ vorkommt*

# Inferenzregeln für Abhängige Funktionenräume

## Verfeinerte Beweisregeln des einfachen Kalküls

**applyEq** $S$

$$\frac{\Gamma \vdash f\,t = g\,u \in T[t/x] \qquad \Gamma \vdash f = g \in x:S \to T \qquad \Gamma \vdash t = u \in S}{}$$

**lambdaEq** $^*$

$$\frac{\Gamma \vdash \lambda x.\,t = \lambda y.\,u \in x:S \to T \qquad \Gamma, x':S \vdash t[x'/x] = u[x'/y] \in T[x'/x] \qquad \Gamma \vdash S \in \mathbb{U}}{}$$

**reduction**

$$\frac{\Gamma \vdash (\lambda x.\,u)\,s = t \in T \qquad \Gamma \vdash u[s/x] = t \in T}{}$$

**declaration** $i$

$$\Gamma, x:T, \Delta \vdash x \in T \quad \hat{=} \quad x{=}x \in T$$

**functionI**

$$\frac{\Gamma \vdash S \to T \in \mathbb{U} \qquad \Gamma \vdash S \in \mathbb{U} \qquad \Gamma \vdash T \in \mathbb{U}}{}$$

$^*$: *Die Umbenennung* $[x'/x]$ *kann entfallen, wenn* $x$ *nicht frei in* $\Gamma$ *vorkommt*

# Inferenzregeln für Abhängige Funktionenräume

## Verfeinerte Beweisregeln des einfachen Kalküls

**applyEq** $S$

$$\frac{\Gamma \vdash f\,t = g\,u \in T[t/x] \qquad \Gamma \vdash f = g \in x:S \to T \qquad \Gamma \vdash t = u \in S}{}$$

**lambdaEq** $^*$

$$\frac{\Gamma \vdash \lambda x.\,t = \lambda y.\,u \in x:S \to T \qquad \Gamma, x':S \vdash t[x'/x] = u[x'/y] \in T[x'/x] \qquad \Gamma \vdash S \in \mathbb{U}}{}$$

**reduction**

$$\frac{\Gamma \vdash (\lambda x.\,u)\,s = t \in T \qquad \Gamma \vdash u[s/x] = t \in T}{}$$

**declaration** $i$

$$\Gamma, x:T, \Delta \vdash x \in T \quad \hat{=} \quad x{=}x \in T$$

**functionI**

$$\frac{\Gamma \vdash x:S \to T \in \mathbb{U} \qquad \Gamma \vdash S \in \mathbb{U} \qquad \Gamma, x':S \vdash T[x'/x] \in \mathbb{U}}{}$$

$^*$: *Die Umbenennung $[x'/x]$ kann entfallen, wenn $x$ nicht frei in $\Gamma$ vorkommt*

# Ausdruckskraft abhängiger Datentypen

**Einfache Repräsentation vieler Basiskonzepte**

# Ausdruckskraft abhängiger Datentypen

## Einfache Repräsentation vieler Basiskonzepte

- **Polymorphe Formulierung Boolescher Ausdrücke**
  - $\mathbb{B} \equiv \mathtt{X} : \mathbb{U} \to \mathtt{X} \to \mathtt{X} \to \mathtt{X}$

# Ausdruckskraft abhängiger Datentypen

**Einfache Repräsentation vieler Basiskonzepte**

- **Polymorphe Formulierung Boolescher Ausdrücke**
  - $\mathbb{B} \equiv \mathtt{X}:\mathbb{U} \to \mathtt{X} \to \mathtt{X} \to \mathtt{X}$
- **Korrekte Typisierung der Church Numerals**
  - $\mathbb{N} \equiv \mathtt{X}:\mathbb{U} \to (\mathtt{X} \to \mathtt{X}) \to \mathtt{X} \to \mathtt{X}$

# Ausdruckskraft abhängiger Datentypen

## Einfache Repräsentation vieler Basiskonzepte

- **Polymorphe Formulierung Boolescher Ausdrücke**
  - $\mathbb{B} \equiv \mathtt{X}:\mathbb{U} \to \mathtt{X} \to \mathtt{X} \to \mathtt{X}$
- **Korrekte Typisierung der Church Numerals**
  - $\mathbb{N} \equiv \mathtt{X}:\mathbb{U} \to (\mathtt{X} \to \mathtt{X}) \to \mathtt{X} \to \mathtt{X}$
- **Geschlossene Typisierung von Produkten**
  - $S \times T \equiv \mathtt{X}:\mathbb{U} \to (S \to T \to \mathtt{X}) \to \mathtt{X}$

# Ausdruckskraft abhängiger Datentypen

## Einfache Repräsentation vieler Basiskonzepte

- **Polymorphe Formulierung Boolescher Ausdrücke**
  - $\mathbb{B} \equiv \mathtt{X}{:}\mathbb{U} \to \mathtt{X} \to \mathtt{X} \to \mathtt{X}$
- **Korrekte Typisierung der Church Numerals**
  - $\mathbb{N} \equiv \mathtt{X}{:}\mathbb{U} \to (\mathtt{X} \to \mathtt{X}) \to \mathtt{X} \to \mathtt{X}$
- **Geschlossene Typisierung von Produkten**
  - $S \times T \equiv \mathtt{X}{:}\mathbb{U} \to (S \to T \to \mathtt{X}) \to \mathtt{X}$
- **Typisierung von Listen analog zu Zahlen**
  - $\mathtt{[]} \equiv \lambda \mathtt{X}.\lambda \mathtt{f}.\lambda \mathtt{x}.\mathtt{x}$
  - $t :: list \equiv \lambda \mathtt{X}.\lambda \mathtt{f}.\lambda \mathtt{x}.\,\mathtt{f}\ t\ (list(\mathtt{X})\ \mathtt{f}\ \mathtt{x})$
  - **list_ind**$[b,h] \equiv \lambda \mathtt{X}.\lambda \mathtt{list}.\ \mathtt{list}(\mathtt{X})\ h\ b$

# Ausdruckskraft abhängiger Datentypen

## Einfache Repräsentation vieler Basiskonzepte

- **Polymorphe Formulierung Boolescher Ausdrücke**
  - $\mathbb{B} \equiv \mathtt{X}{:}\mathbb{U} \to \mathtt{X} \to \mathtt{X} \to \mathtt{X}$
- **Korrekte Typisierung der Church Numerals**
  - $\mathbb{N} \equiv \mathtt{X}{:}\mathbb{U} \to (\mathtt{X} \to \mathtt{X}) \to \mathtt{X} \to \mathtt{X}$
- **Geschlossene Typisierung von Produkten**
  - $S \times T \equiv \mathtt{X}{:}\mathbb{U} \to (S \to T \to \mathtt{X}) \to \mathtt{X}$
- **Typisierung von Listen analog zu Zahlen**
  - $[] \equiv \lambda \mathtt{X}.\lambda \mathtt{f}.\lambda \mathtt{x}.\mathtt{x}$
  - $t::list \equiv \lambda \mathtt{X}.\lambda \mathtt{f}.\lambda \mathtt{x}.\,\mathtt{f}\ t\ (list(\mathtt{X})\ \mathtt{f}\ \mathtt{x})$
  - **list_ind**$[b,h] \equiv \lambda \mathtt{X}.\lambda \mathtt{list}.\ \mathtt{list}(\mathtt{X})\ h\ b$
  - $T$ list $\equiv \mathtt{X}{:}\mathbb{U} \to (T \to \mathtt{X} \to \mathtt{X}) \to \mathtt{X} \to \mathtt{X}$

# Die Curry-Howard Isomorphie setzt sich fort

$\Gamma \vdash \lambda y\,.\,t \in x{:}S{\to}T$ `BY lambdaI`

$\Gamma,\ z{:}S \vdash t[z/y] \in T[z/x]$

$\Gamma \vdash S \in \mathbb{U}$

# Die Curry-Howard Isomorphie setzt sich fort

$\Gamma \vdash \lambda y\,.\,t \in x\!:\!S \to T$ `BY lambdaI`

$\Gamma,\ z\!:\!S \vdash t[z/y] \in T[z/x]$

$\Gamma \vdash S \in \mathbb{U}$

$\Gamma \vdash \forall x\!:\!S\,.\,A$ `BY allI`

$\Gamma, z\!:\!S \vdash A[z/x]$

# Die Curry-Howard Isomorphie setzt sich fort

$\Gamma \vdash \lambda y\,.\,t \in x\!:\!S \to T$ `BY lambdaI`

$\Gamma,\ z\!:\!S \vdash t[z/y] \in T[z/x]$

$\Gamma \vdash S \in \mathbb{U}$

$\Gamma \vdash \forall x\!:\!S\,.\,A$ `BY allI`

$\Gamma, z\!:\!S \vdash A[z/x]$

$\Gamma \vdash f\,t \in T[t/x]$ `BY applyI` $x\!:\!S \to T$

$\Gamma \vdash f \in x\!:\!S \to T$

$\Gamma \vdash t \in S$

$\Gamma \vdash A[t/x]$ `BY InstUniv` $\forall x\!:\!S\,.\,A$

$\Gamma \vdash \forall x\!:\!S\,.\,A$

$\Gamma \vdash t \in S$

# Die Curry-Howard Isomorphie setzt sich fort

$\Gamma \vdash \lambda y\,.\,t \in x\!:\!S \to T$ `BY lambdaI`
$\quad \Gamma,\ z\!:\!S \vdash t[z/y] \in T[z/x]$
$\quad \Gamma \vdash S \in \mathbb{U}$

$\Gamma \vdash \forall x\!:\!S\,.\,A$ `BY allI`
$\quad \Gamma, z\!:\!S \vdash A[z/x]$

$\Gamma \vdash f\,t \in T[t/x]$ `BY applyI` $x\!:\!S \to T$
$\quad \Gamma \vdash f \in x\!:\!S \to T$
$\quad \Gamma \vdash t \in S$

$\Gamma \vdash A[t/x]$ `BY InstUniv` $\forall x\!:\!S\,.\,A$
$\quad \Gamma \vdash \forall x\!:\!S\,.\,A$
$\quad \Gamma \vdash t \in S$

**Analogien zwischen Typentheorie und Logik**

# Die Curry-Howard Isomorphie setzt sich fort

$\Gamma \vdash \lambda y\,.\,t \in x\!:\!S \to T$ `BY lambdaI`
$\quad \Gamma,\ z\!:\!S \vdash t[z/y] \in T[z/x]$
$\quad \Gamma \vdash S \in \mathbb{U}$

$\Gamma \vdash \forall x\!:\!S\,.\,A$ `BY allI`
$\quad \Gamma, z\!:\!S \vdash A[z/x]$

$\Gamma \vdash f\,t \in T[t/x]$ `BY applyI` $x\!:\!S \to T$
$\quad \Gamma \vdash f \in x\!:\!S \to T$
$\quad \Gamma \vdash t \in S$

$\Gamma \vdash A[t/x]$ `BY InstUniv` $\forall x\!:\!S\,.\,A$
$\quad \Gamma \vdash \forall x\!:\!S\,.\,A$
$\quad \Gamma \vdash t \in S$

## Analogien zwischen Typentheorie und Logik

| Typ | Formel |
|---|---|

# Die Curry-Howard Isomorphie setzt sich fort

$\Gamma \vdash \lambda y\,.\,t \in x\!:\!S \to T$ `BY lambdaI`
$\quad \Gamma,\ z\!:\!S \vdash t[z/y] \in T[z/x]$
$\quad \Gamma \vdash S \in \mathbb{U}$

$\Gamma \vdash \forall x\!:\!S\,.\,A$ `BY allI`
$\quad \Gamma, z\!:\!S \vdash A[z/x]$

$\Gamma \vdash f\,t \in T[t/x]$ `BY applyI` $x\!:\!S \to T$
$\quad \Gamma \vdash f \in x\!:\!S \to T$
$\quad \Gamma \vdash t \in S$

$\Gamma \vdash A[t/x]$ `BY InstUniv` $\forall x\!:\!S\,.\,A$
$\quad \Gamma \vdash \forall x\!:\!S\,.\,A$
$\quad \Gamma \vdash t \in S$

## Analogien zwischen Typentheorie und Logik

| Typ | Formel |
|---|---|
| Variable vom Typ $S$ | Annahme der Formel $S$ |

# Die Curry-Howard Isomorphie setzt sich fort

$\Gamma \vdash \lambda y\,.\,t \in x\!:\!S \to T$ `BY lambdaI`
$\Gamma,\ z\!:\!S \vdash t[z/y] \in T[z/x]$
$\Gamma \vdash S \in \mathbb{U}$

$\Gamma \vdash \forall x\!:\!S\,.\,A$ `BY allI`
$\Gamma, z\!:\!S \vdash A[z/x]$

$\Gamma \vdash f\,t \in T[t/x]$ `BY applyI` $x\!:\!S \to T$
$\Gamma \vdash f \in x\!:\!S \to T$
$\Gamma \vdash t \in S$

$\Gamma \vdash A[t/x]$ `BY InstUniv` $\forall x\!:\!S\,.\,A$
$\Gamma \vdash \forall x\!:\!S\,.\,A$
$\Gamma \vdash t \in S$

## Analogien zwischen Typentheorie und Logik

| | |
|---|---|
| Typ | Formel |
| Variable vom Typ $S$ | Annahme der Formel $S$ |
| Term vom Typ $T$ (freie Variablen aus $S_i$) | Beweis von $T$ (Annahmen $S_i$) |

# Die Curry-Howard Isomorphie setzt sich fort

$\Gamma \vdash \lambda y\,.\,t \in x\!:\!S \to T$ `BY lambdaI`
$\quad \Gamma,\ z\!:\!S \vdash t[z/y] \in T[z/x]$
$\quad \Gamma \vdash S \in \mathbb{U}$

$\Gamma \vdash \forall x\!:\!S\,.\,A$ `BY allI`
$\quad \Gamma, z\!:\!S \vdash A[z/x]$

$\Gamma \vdash f\,t \in T[t/x]$ `BY applyI` $x\!:\!S \to T$
$\quad \Gamma \vdash f \in x\!:\!S \to T$
$\quad \Gamma \vdash t \in S$

$\Gamma \vdash A[t/x]$ `BY InstUniv` $\forall x\!:\!S\,.\,A$
$\quad \Gamma \vdash \forall x\!:\!S\,.\,A$
$\quad \Gamma \vdash t \in S$

## Analogien zwischen Typentheorie und Logik

| | |
|---|---|
| Typ | Formel |
| Variable vom Typ $S$ | Annahme der Formel $S$ |
| Term vom Typ $T$ (freie Variablen aus $S_i$) | Beweis von $T$ (Annahmen $S_i$) |
| Abhängiger Funktionenraum | All-Quantor |

# Die Curry-Howard Isomorphie setzt sich fort

$\Gamma \vdash \lambda y\,.\,t \in x\!:\!S \to T$ `BY lambdaI`
$\quad \Gamma,\ z\!:\!S \vdash t[z/y] \in T[z/x]$
$\quad \Gamma \vdash S \in \mathbb{U}$

$\Gamma \vdash \forall x\!:\!S\,.\,A$ `BY allI`
$\quad \Gamma, z\!:\!S \vdash A[z/x]$

$\Gamma \vdash f\,t \in T[t/x]$ `BY applyI` $x\!:\!S \to T$
$\quad \Gamma \vdash f \in x\!:\!S \to T$
$\quad \Gamma \vdash t \in S$

$\Gamma \vdash A[t/x]$ `BY InstUniv` $\forall x\!:\!S\,.\,A$
$\quad \Gamma \vdash \forall x\!:\!S\,.\,A$
$\quad \Gamma \vdash t \in S$

## Analogien zwischen Typentheorie und Logik

| | |
|---|---|
| Typ | Formel |
| Variable vom Typ $S$ | Annahme der Formel $S$ |
| Term vom Typ $T$ (freie Variablen aus $S_i$) | Beweis von $T$ (Annahmen $S_i$) |
| Abhängiger Funktionenraum | All-Quantor |
| Unabhängiger Funktionenraum | Implikation |

# Die Curry-Howard Isomorphie setzt sich fort

$\Gamma \vdash \lambda y\,.\,t \in x\!:\!S \to T$ `BY lambdaI`
$\quad \Gamma,\ z\!:\!S \vdash t[z/y] \in T[z/x]$
$\quad \Gamma \vdash S \in \mathbb{U}$

$\Gamma \vdash \forall x\!:\!S\,.\,A$ `BY allI`
$\quad \Gamma, z\!:\!S \vdash A[z/x]$

$\Gamma \vdash f\,t \in T[t/x]$ `BY applyI` $x\!:\!S \to T$
$\quad \Gamma \vdash f \in x\!:\!S \to T$
$\quad \Gamma \vdash t \in S$

$\Gamma \vdash A[t/x]$ `BY InstUniv` $\forall x\!:\!S\,.\,A$
$\quad \Gamma \vdash \forall x\!:\!S\,.\,A$
$\quad \Gamma \vdash t \in S$

## Analogien zwischen Typentheorie und Logik

| | |
|---|---|
| Typ | Formel |
| Variable vom Typ $S$ | Annahme der Formel $S$ |
| Term vom Typ $T$ (freie Variablen aus $S_i$) | Beweis von $T$ (Annahmen $S_i$) |
| Abhängiger Funktionenraum | All-Quantor |
| Unabhängiger Funktionenraum | Implikation |

**Komplette Formalisierung der Logik möglich?**

# Logik ist durch abhängige Datentypen codierbar

- **Konjunktion** $A \wedge B$

# Logik ist durch abhängige Datentypen codierbar

- **Konjunktion $A \wedge B$**
  - $A \wedge B$ gilt, wenn jede Aussage, die aus $A$ und $B$ folgt, gültig ist

# Logik ist durch abhängige Datentypen codierbar

- **Konjunktion $A \wedge B$**
  - $A \wedge B$ gilt, wenn jede Aussage, die aus $A$ und $B$ folgt, gültig ist
  - $A \wedge B \equiv \forall \mathtt{P} : \mathbb{U} . (A \Rightarrow B \Rightarrow \mathtt{P}) \Rightarrow \mathtt{P}$

# Logik ist durch abhängige Datentypen codierbar

- **Konjunktion $A \wedge B$**
  - $A \wedge B$ gilt, wenn jede Aussage, die aus $A$ und $B$ folgt, gültig ist
  - $A \wedge B \equiv \forall \mathtt{P} : \mathbb{U} . (A \Rightarrow B \Rightarrow \mathtt{P}) \Rightarrow \mathtt{P}$
- **Disjunktion $A \vee B$**

# Logik ist durch abhängige Datentypen codierbar

- **Konjunktion $A \wedge B$**
  - $A \wedge B$ gilt, wenn jede Aussage, die aus $A$ und $B$ folgt, gültig ist
  - $A \wedge B \equiv \forall \mathtt{P} : \mathbb{U} . (A \Rightarrow B \Rightarrow \mathtt{P}) \Rightarrow \mathtt{P}$
- **Disjunktion $A \vee B$**
  - $A \vee B$ gilt, wenn jede Aussage gültig ist, die aus $A$ und auch aus $B$ folgt

# Logik ist durch abhängige Datentypen codierbar

- **Konjunktion $A \wedge B$**
  - $A \wedge B$ gilt, wenn jede Aussage, die aus $A$ und $B$ folgt, gültig ist
  - $A \wedge B \equiv \forall \mathtt{P} : \mathbb{U} . (A \Rightarrow B \Rightarrow \mathtt{P}) \Rightarrow \mathtt{P}$
- **Disjunktion $A \vee B$**
  - $A \vee B$ gilt, wenn jede Aussage gültig ist, die aus $A$ und auch aus $B$ folgt
  - $A \vee B \equiv \forall \mathtt{P} : \mathbb{U} . (A \Rightarrow \mathtt{P}) \Rightarrow (B \Rightarrow \mathtt{P}) \Rightarrow \mathtt{P}$

# Logik ist durch abhängige Datentypen codierbar

- **Konjunktion $A \wedge B$**
  - $A \wedge B$ gilt, wenn jede Aussage, die aus $A$ und $B$ folgt, gültig ist
  - $A \wedge B \equiv \forall \mathtt{P} : \mathbb{U} . (A \Rightarrow B \Rightarrow \mathtt{P}) \Rightarrow \mathtt{P}$
- **Disjunktion $A \vee B$**
  - $A \vee B$ gilt, wenn jede Aussage gültig ist, die aus $A$ und auch aus $B$ folgt
  - $A \vee B \equiv \forall \mathtt{P} : \mathbb{U} . (A \Rightarrow \mathtt{P}) \Rightarrow (B \Rightarrow \mathtt{P}) \Rightarrow \mathtt{P}$
- **Negation $\neg A$**

# Logik ist durch abhängige Datentypen codierbar

- **Konjunktion $A \wedge B$**
  - $A \wedge B$ gilt, wenn jede Aussage, die aus $A$ und $B$ folgt, gültig ist
  - $A \wedge B \equiv \forall \mathtt{P} : \mathbb{U} . (A \Rightarrow B \Rightarrow \mathtt{P}) \Rightarrow \mathtt{P}$
- **Disjunktion $A \vee B$**
  - $A \vee B$ gilt, wenn jede Aussage gültig ist, die aus $A$ und auch aus $B$ folgt
  - $A \vee B \equiv \forall \mathtt{P} : \mathbb{U} . (A \Rightarrow \mathtt{P}) \Rightarrow (B \Rightarrow \mathtt{P}) \Rightarrow \mathtt{P}$
- **Negation $\neg A$** ist eine Abkürzung: $\neg A \equiv A \Rightarrow \mathsf{ff}$

# Logik ist durch abhängige Datentypen codierbar

- **Konjunktion $A \wedge B$**
  - $A \wedge B$ gilt, wenn jede Aussage, die aus $A$ und $B$ folgt, gültig ist
  - $A \wedge B \equiv \forall \mathtt{P} : \mathbb{U} . (A \Rightarrow B \Rightarrow \mathtt{P}) \Rightarrow \mathtt{P}$
- **Disjunktion $A \vee B$**
  - $A \vee B$ gilt, wenn jede Aussage gültig ist, die aus $A$ und auch aus $B$ folgt
  - $A \vee B \equiv \forall \mathtt{P} : \mathbb{U} . (A \Rightarrow \mathtt{P}) \Rightarrow (B \Rightarrow \mathtt{P}) \Rightarrow \mathtt{P}$
- **Negation $\neg A$** ist eine Abkürzung: $\neg A \equiv A \Rightarrow \mathsf{ff}$
  - Ein fundamentaler Widerspruch ist die Aussage, daß jede Aussage gilt

# Logik ist durch abhängige Datentypen codierbar

- **Konjunktion $A \wedge B$**
  - $A \wedge B$ gilt, wenn jede Aussage, die aus $A$ und $B$ folgt, gültig ist
  - $A \wedge B \equiv \forall \mathtt{P} : \mathbb{U} . (A \Rightarrow B \Rightarrow \mathtt{P}) \Rightarrow \mathtt{P}$
- **Disjunktion $A \vee B$**
  - $A \vee B$ gilt, wenn jede Aussage gültig ist, die aus $A$ und auch aus $B$ folgt
  - $A \vee B \equiv \forall \mathtt{P} : \mathbb{U} . (A \Rightarrow \mathtt{P}) \Rightarrow (B \Rightarrow \mathtt{P}) \Rightarrow \mathtt{P}$
- **Negation $\neg A$** ist eine Abkürzung: $\neg A \equiv A \Rightarrow \mathsf{ff}$
  - Ein fundamentaler Widerspruch ist die Aussage, daß jede Aussage gilt
  - $\mathsf{ff} \equiv \forall \mathtt{P} : \mathbb{U} . \mathtt{P}$

# Logik ist durch abhängige Datentypen codierbar

- **Konjunktion $A \wedge B$**
  - $A \wedge B$ gilt, wenn jede Aussage, die aus $A$ und $B$ folgt, gültig ist
  - $A \wedge B \equiv \forall \mathtt{P}:\mathbb{U}.(A \Rightarrow B \Rightarrow \mathtt{P}) \Rightarrow \mathtt{P}$
- **Disjunktion $A \vee B$**
  - $A \vee B$ gilt, wenn jede Aussage gültig ist, die aus $A$ und auch aus $B$ folgt
  - $A \vee B \equiv \forall \mathtt{P}:\mathbb{U}.(A \Rightarrow \mathtt{P}) \Rightarrow (B \Rightarrow \mathtt{P}) \Rightarrow \mathtt{P}$
- **Negation $\neg A$** ist eine Abkürzung: $\neg A \equiv A \Rightarrow \mathsf{ff}$
  - Ein fundamentaler Widerspruch ist die Aussage, daß jede Aussage gilt
  - $\mathsf{ff} \equiv \forall \mathtt{P}:\mathbb{U}.\mathtt{P}$
- **Existenzquantor $\exists x{:}S.A$**

# Logik ist durch abhängige Datentypen codierbar

- **Konjunktion $A \wedge B$**
  - $A \wedge B$ gilt, wenn jede Aussage, die aus $A$ und $B$ folgt, gültig ist
  - $A \wedge B \equiv \forall \mathtt{P}:\mathbb{U}.(A \Rightarrow B \Rightarrow \mathtt{P}) \Rightarrow \mathtt{P}$
- **Disjunktion $A \vee B$**
  - $A \vee B$ gilt, wenn jede Aussage gültig ist, die aus $A$ und auch aus $B$ folgt
  - $A \vee B \equiv \forall \mathtt{P}:\mathbb{U}.(A \Rightarrow \mathtt{P}) \Rightarrow (B \Rightarrow \mathtt{P}) \Rightarrow \mathtt{P}$
- **Negation $\neg A$** ist eine Abkürzung: $\neg A \equiv A \Rightarrow \mathsf{ff}$
  - Ein fundamentaler Widerspruch ist die Aussage, daß jede Aussage gilt
  - $\mathsf{ff} \equiv \forall \mathtt{P}:\mathbb{U}.\mathtt{P}$
- **Existenzquantor $\exists x{:}S.A$**
  - $\exists x{:}S.A$ gilt, wenn jede Aussage gilt, die aus einem beliebigen $A[x]$ folgt

# Logik ist durch abhängige Datentypen codierbar

- **Konjunktion $A \wedge B$**
  - $A \wedge B$ gilt, wenn jede Aussage, die aus $A$ und $B$ folgt, gültig ist
  - $A \wedge B \;\equiv\; \forall \mathtt{P}:\mathbb{U}.\,(A \Rightarrow B \Rightarrow \mathtt{P}) \;\Rightarrow\; \mathtt{P}$
- **Disjunktion $A \vee B$**
  - $A \vee B$ gilt, wenn jede Aussage gültig ist, die aus $A$ und auch aus $B$ folgt
  - $A \vee B \;\equiv\; \forall \mathtt{P}:\mathbb{U}.\,(A \Rightarrow \mathtt{P}) \;\Rightarrow\; (B \Rightarrow \mathtt{P}) \;\Rightarrow\; \mathtt{P}$
- **Negation $\neg A$** ist eine Abkürzung: $\neg A \;\equiv\; A \Rightarrow \mathsf{ff}$
  - Ein fundamentaler Widerspruch ist die Aussage, daß jede Aussage gilt
  - $\mathsf{ff} \;\equiv\; \forall \mathtt{P}:\mathbb{U}.\mathtt{P}$
- **Existenzquantor $\exists x{:}S.A$**
  - $\exists x{:}S.A$ gilt, wenn jede Aussage gilt, die aus einem beliebigen $A[x]$ folgt
  - $\exists x:S\,.\,A \;\equiv\; \forall \mathtt{P}:\mathbb{U}.\,(\forall x:S\,.\,A \Rightarrow \mathtt{P}) \;\Rightarrow\; \mathtt{P}$

# Logik ist durch abhängige Datentypen codierbar

- **Konjunktion $A \wedge B$**
  - $A \wedge B$ gilt, wenn jede Aussage, die aus $A$ und $B$ folgt, gültig ist
  - $A \wedge B \;\equiv\; \forall \mathtt{P}:\mathbb{U}.\,(A \Rightarrow B \Rightarrow \mathtt{P}) \;\Rightarrow\; \mathtt{P}$
- **Disjunktion $A \vee B$**
  - $A \vee B$ gilt, wenn jede Aussage gültig ist, die aus $A$ und auch aus $B$ folgt
  - $A \vee B \;\equiv\; \forall \mathtt{P}:\mathbb{U}.\,(A \Rightarrow \mathtt{P}) \;\Rightarrow\; (B \Rightarrow \mathtt{P}) \;\Rightarrow\; \mathtt{P}$
- **Negation $\neg A$** ist eine Abkürzung: $\neg A \;\equiv\; A \Rightarrow \mathsf{ff}$
  - Ein fundamentaler Widerspruch ist die Aussage, daß jede Aussage gilt
  - $\mathsf{ff} \;\equiv\; \forall \mathtt{P}:\mathbb{U}.\mathtt{P}$
- **Existenzquantor $\exists x{:}S.A$**
  - $\exists x{:}S.A$ gilt, wenn jede Aussage gilt, die aus einem beliebigen $A[x]$ folgt
  - $\exists x:S.\,A \;\equiv\; \forall \mathtt{P}:\mathbb{U}.\,(\forall x:S.\,A \Rightarrow \mathtt{P}) \;\Rightarrow\; \mathtt{P}$
- **Gleichheit $x{=}y \in A$**

# Logik ist durch abhängige Datentypen codierbar

- **Konjunktion $A \wedge B$**
  - $A \wedge B$ gilt, wenn jede Aussage, die aus $A$ und $B$ folgt, gültig ist
  - $A \wedge B \;\equiv\; \forall \mathtt{P}:\mathbb{U}.\,(A \Rightarrow B \Rightarrow \mathtt{P}) \;\Rightarrow\; \mathtt{P}$
- **Disjunktion $A \vee B$**
  - $A \vee B$ gilt, wenn jede Aussage gültig ist, die aus $A$ und auch aus $B$ folgt
  - $A \vee B \;\equiv\; \forall \mathtt{P}:\mathbb{U}.\,(A \Rightarrow \mathtt{P}) \;\Rightarrow\; (B \Rightarrow \mathtt{P}) \;\Rightarrow\; \mathtt{P}$
- **Negation $\neg A$** ist eine Abkürzung: $\neg A \;\equiv\; A \Rightarrow \mathsf{ff}$
  - Ein fundamentaler Widerspruch ist die Aussage, daß jede Aussage gilt
  - $\mathsf{ff} \;\equiv\; \forall \mathtt{P}:\mathbb{U}.\mathtt{P}$
- **Existenzquantor $\exists x{:}S.A$**
  - $\exists x{:}S.A$ gilt, wenn jede Aussage gilt, die aus einem beliebigen $A[x]$ folgt
  - $\exists x:S.\,A \;\equiv\; \forall \mathtt{P}:\mathbb{U}.\,(\forall x:S.\,A \Rightarrow \mathtt{P}) \;\Rightarrow\; \mathtt{P}$
- **Gleichheit $x{=}y \in A$**
  - $x{=}y$ gilt, wenn kein Prädikat $x$ von $y$ unterscheiden kann

# Logik ist durch abhängige Datentypen codierbar

- **Konjunktion $A \wedge B$**
  - $A \wedge B$ gilt, wenn jede Aussage, die aus $A$ und $B$ folgt, gültig ist
  - $A \wedge B \;\equiv\; \forall \mathtt{P}:\mathbb{U}.\,(A \Rightarrow B \Rightarrow \mathtt{P}) \;\Rightarrow\; \mathtt{P}$
- **Disjunktion $A \vee B$**
  - $A \vee B$ gilt, wenn jede Aussage gültig ist, die aus $A$ und auch aus $B$ folgt
  - $A \vee B \;\equiv\; \forall \mathtt{P}:\mathbb{U}.\,(A \Rightarrow \mathtt{P}) \;\Rightarrow\; (B \Rightarrow \mathtt{P}) \;\Rightarrow\; \mathtt{P}$
- **Negation $\neg A$** ist eine Abkürzung: $\neg A \;\equiv\; A \Rightarrow \mathtt{ff}$
  - Ein fundamentaler Widerspruch ist die Aussage, daß jede Aussage gilt
  - $\mathtt{ff} \;\equiv\; \forall \mathtt{P}:\mathbb{U}.\mathtt{P}$
- **Existenzquantor $\exists x{:}S.A$**
  - $\exists x{:}S.A$ gilt, wenn jede Aussage gilt, die aus einem beliebigen $A[x]$ folgt
  - $\exists x:S.\,A \;\equiv\; \forall \mathtt{P}:\mathbb{U}.\,(\forall x:S.\,A \Rightarrow \mathtt{P}) \;\Rightarrow\; \mathtt{P}$
- **Gleichheit $x{=}y \in A$**
  - $x{=}y$ gilt, wenn kein Prädikat $x$ von $y$ unterscheiden kann
  - $x{=}y \in A \;\equiv\; \forall \mathtt{P}:A{\to}\mathbb{U}.\,\mathtt{P}(x) \Rightarrow \mathtt{P}(y)$

# Abhängige Typen sind eine sinnvolle Erweiterung

- **Zentrale Konzepte werden darstellbar**
  - Boolesche Ausdrücke, Natürliche Zahlen, Listen, Produkträume
  - Logischen Operatoren, Gleichheit

# Abhängige Typen sind eine sinnvolle Erweiterung

- **Zentrale Konzepte werden darstellbar**
  - Boolesche Ausdrücke, Natürliche Zahlen, Listen, Produkträume
  - Logischen Operatoren, Gleichheit
- **Informatik benötigt Abhängigkeitsstrukturen**

# Abhängige Typen sind eine sinnvolle Erweiterung

- **Zentrale Konzepte werden darstellbar**
  - Boolesche Ausdrücke, Natürliche Zahlen, Listen, Produkträume
  - Logischen Operatoren, Gleichheit
- **Informatik benötigt Abhängigkeitsstrukturen**
  - Endliche Automaten, (Variant) Records, optionale Argumente, ...

# Abhängige Typen sind eine sinnvolle Erweiterung

- **Zentrale Konzepte werden darstellbar**
  - Boolesche Ausdrücke, Natürliche Zahlen, Listen, Produkträume
  - Logischen Operatoren, Gleichheit
- **Informatik benötigt Abhängigkeitsstrukturen**
  - Endliche Automaten, (Variant) Records, optionale Argumente, ...
- **Unbegrenzte Selbstanwendung bleibt untypisierbar**

# Abhängige Typen sind eine sinnvolle Erweiterung

- **Zentrale Konzepte werden darstellbar**
  - Boolesche Ausdrücke, Natürliche Zahlen, Listen, Produkträume
  - Logischen Operatoren, Gleichheit
- **Informatik benötigt Abhängigkeitsstrukturen**
  - Endliche Automaten, (Variant) Records, optionale Argumente, ...
- **Unbegrenzte Selbstanwendung bleibt untypisierbar**
  - $\lambda x.x\ x$ ist nicht typisierbar, da $T = x{:}T \to T$ keine Lösung hat

# Abhängige Typen sind eine sinnvolle Erweiterung

- **Zentrale Konzepte werden darstellbar**
  - Boolesche Ausdrücke, Natürliche Zahlen, Listen, Produkträume
  - Logischen Operatoren, Gleichheit
- **Informatik benötigt Abhängigkeitsstrukturen**
  - Endliche Automaten, (Variant) Records, optionale Argumente, ...
- **Unbegrenzte Selbstanwendung bleibt untypisierbar**
  - $\lambda x.x\ x$ ist nicht typisierbar, da $T = x{:}T{\to}T$ keine Lösung hat
  - Parametrisierung wie bei $\mathbb{N}$ führt zu stark begrenzter Selbstanwendung
    - In $\lambda x.x(X)\,x$ hätte $x$ den Typ $T = X{:}\mathbb{U}{\to}X{\to}X$
    - Die Gleichung $T = T{:}\mathbb{U}{\to}T{\to}T$ enthält keine direkte Selbstreferenz

# Abhängige Typen sind eine sinnvolle Erweiterung

- **Zentrale Konzepte werden darstellbar**
  - Boolesche Ausdrücke, Natürliche Zahlen, Listen, Produkträume
  - Logischen Operatoren, Gleichheit
- **Informatik benötigt Abhängigkeitsstrukturen**
  - Endliche Automaten, (Variant) Records, optionale Argumente, ...
- **Unbegrenzte Selbstanwendung bleibt untypisierbar**
  - $\lambda x.x\ x$ ist nicht typisierbar, da $T = x{:}T{\to}T$ keine Lösung hat
  - Parametrisierung wie bei $\mathbb{N}$ führt zu stark begrenzter Selbstanwendung
    - In $\lambda x.x(X)\,x$ hätte $x$ den Typ $T$ = $X{:}\mathbb{U}{\to}X{\to}X$
    - Die Gleichung $T = T{:}\mathbb{U}{\to}T{\to}T$ enthält keine direkte Selbstreferenz
- **Es gibt einfache natürliche Modelle**

# Abhängige Typen sind eine sinnvolle Erweiterung

- **Zentrale Konzepte werden darstellbar**
  - Boolesche Ausdrücke, Natürliche Zahlen, Listen, Produkträume
  - Logischen Operatoren, Gleichheit
- **Informatik benötigt Abhängigkeitsstrukturen**
  - Endliche Automaten, (Variant) Records, optionale Argumente, ...
- **Unbegrenzte Selbstanwendung bleibt untypisierbar**
  - $\lambda x.x\ x$ ist nicht typisierbar, da $T = x{:}T{\to}T$ keine Lösung hat
  - Parametrisierung wie bei $\mathbb{N}$ führt zu stark begrenzter Selbstanwendung
    - In $\lambda x.x(X)\,x$ hätte $x$ den Typ $T\ \texttt{=}\ X{:}\mathbb{U}{\to}X{\to}X$
    - Die Gleichung $T = T{:}\mathbb{U}{\to}T{\to}T$ enthält keine direkte Selbstreferenz
- **Es gibt einfache natürliche Modelle**
  - $x{:}S{\to}T[x]$ entspricht Abbildungen in eine Familie von Bildbereichen

# Abhängige Typen sind eine sinnvolle Erweiterung

- **Zentrale Konzepte werden darstellbar**
  - Boolesche Ausdrücke, Natürliche Zahlen, Listen, Produkträume
  - Logischen Operatoren, Gleichheit
- **Informatik benötigt Abhängigkeitsstrukturen**
  - Endliche Automaten, (Variant) Records, optionale Argumente, ...
- **Unbegrenzte Selbstanwendung bleibt untypisierbar**
  - $\lambda x.x\ x$ ist nicht typisierbar, da $T = x{:}T{\to}T$ keine Lösung hat
  - Parametrisierung wie bei $\mathbb{N}$ führt zu stark begrenzter Selbstanwendung
    - In $\lambda x.x(X)\,x$ hätte $x$ den Typ $T$ = $X{:}\mathbb{U}{\to}X{\to}X$
    - Die Gleichung $T = T{:}\mathbb{U}{\to}T{\to}T$ enthält keine direkte Selbstreferenz
- **Es gibt einfache natürliche Modelle**
  - $x{:}S{\to}T[x]$ entspricht Abbildungen in eine Familie von Bildbereichen

**Gibt es Probleme?**

# Abhängigkeiten und Universen

- **Abhängigkeiten benötigen “höhere Typen”**

# Abhängigkeiten und Universen

- **Abhängigkeiten benötigen "höhere Typen"**
  - In $x{:}S \to T$ kommt $x$ frei in $T$ vor
    - $x$ muß durch jeden Wert $s \in S$ ersetzt werden können

# Abhängigkeiten und Universen

- **Abhängigkeiten benötigen "höhere Typen"**
  - In $x{:}S \to T$ kommt $x$ frei in $T$ vor
    - $x$ muß durch jeden Wert $s \in S$ ersetzt werden können
    - $T[s/x]$ ist ein Typ für beliebige $s \in S$, gehört also zum Universum $\mathbb{U}$

# Abhängigkeiten und Universen

- **Abhängigkeiten benötigen "höhere Typen"**
  - In $x{:}S \to T$ kommt $x$ frei in $T$ vor
    - $x$ muß durch jeden Wert $s \in S$ ersetzt werden können
    - $T[s/x]$ ist ein Typ für beliebige $s \in S$, gehört also zum Universum $\mathbb{U}$
  - Formale Beschreibung benötigt Funktion $\hat{T}$ mit $\hat{T}(s) \xrightarrow{\beta} T[s/x]$

# Abhängigkeiten und Universen

- **Abhängigkeiten benötigen "höhere Typen"**
  - In $x{:}S{\to}T$ kommt $x$ frei in $T$ vor
    - $x$ muß durch jeden Wert $s \in S$ ersetzt werden können
    - $T[s/x]$ ist ein Typ für beliebige $s \in S$, gehört also zum Universum $\mathbb{U}$
  - Formale Beschreibung benötigt Funktion $\hat{T}$ mit $\hat{T}(s) \xrightarrow{\beta} T[s/x]$
  - Was ist der Typ der Funktion $\hat{T}$?

# Abhängigkeiten und Universen

- **Abhängigkeiten benötigen "höhere Typen"**
  - In $x{:}S \to T$ kommt $x$ frei in $T$ vor
    - $x$ muß durch jeden Wert $s \in S$ ersetzt werden können
    - $T[s/x]$ ist ein Typ für beliebige $s \in S$, gehört also zum Universum $\mathbb{U}$
  - Formale Beschreibung benötigt Funktion $\hat{T}$ mit $\hat{T}(s) \xrightarrow{\beta} T[s/x]$
  - Was ist der Typ der Funktion $\hat{T}$?
    - $\hat{T}$ liefert für jedes $s \in S$ einen Typ, also $\hat{T} \in S \to \mathbb{U}$

# Abhängigkeiten und Universen

- **Abhängigkeiten benötigen "höhere Typen"**
  - In $x{:}S{\to}T$ kommt $x$ frei in $T$ vor
    - $x$ muß durch jeden Wert $s \in S$ ersetzt werden können
    - $T[s/x]$ ist ein Typ für beliebige $s \in S$, gehört also zum Universum $\mathbb{U}$
  - Formale Beschreibung benötigt Funktion $\hat{T}$ mit $\hat{T}(s) \xrightarrow{\beta} T[s/x]$
  - Was ist der Typ der Funktion $\hat{T}$?
    - $\hat{T}$ liefert für jedes $s \in S$ einen Typ, also $\hat{T} \in S{\to}\mathbb{U}$
    - $\boldsymbol{S{\to}\mathbb{U}}$ **muß ein Typ sein**

# Abhängigkeiten und Universen

- **Abhängigkeiten benötigen "höhere Typen"**
  - In $x{:}S{\to}T$ kommt $x$ frei in $T$ vor
    - $x$ muß durch jeden Wert $s \in S$ ersetzt werden können
    - $T[s/x]$ ist ein Typ für beliebige $s \in S$, gehört also zum Universum $\mathbb{U}$
  - Formale Beschreibung benötigt Funktion $\hat{T}$ mit $\hat{T}(s) \xrightarrow{\beta} T[s/x]$
  - Was ist der Typ der Funktion $\hat{T}$?
    - $\hat{T}$ liefert für jedes $s \in S$ einen Typ, also $\hat{T} \in S{\to}\mathbb{U}$
    - $\boldsymbol{S{\to}\mathbb{U}}$ **muß ein Typ sein**
- $\mathbb{U}$ **bekommt eine Doppelrolle**

# Abhängigkeiten und Universen

- **Abhängigkeiten benötigen "höhere Typen"**
  - In $x{:}S{\to}T$ kommt $x$ frei in $T$ vor
    - $x$ muß durch jeden Wert $s \in S$ ersetzt werden können
    - $T[s/x]$ ist ein Typ für beliebige $s \in S$, gehört also zum Universum $\mathbb{U}$
  - Formale Beschreibung benötigt Funktion $\hat{T}$ mit $\hat{T}(s) \xrightarrow{\beta} T[s/x]$
  - Was ist der Typ der Funktion $\hat{T}$?
    - $\hat{T}$ liefert für jedes $s \in S$ einen Typ, also $\hat{T} \in S{\to}\mathbb{U}$
    - $\boldsymbol{S{\to}\mathbb{U}}$ **muß ein Typ sein**
- $\mathbb{U}$ **bekommt eine Doppelrolle**
  - $\mathbb{U}$ ist ein Typ

# Abhängigkeiten und Universen

- **Abhängigkeiten benötigen "höhere Typen"**
  - In $x{:}S{\to}T$ kommt $x$ frei in $T$ vor
    - $x$ muß durch jeden Wert $s \in S$ ersetzt werden können
    - $T[s/x]$ ist ein Typ für beliebige $s \in S$, gehört also zum Universum $\mathbb{U}$
  - Formale Beschreibung benötigt Funktion $\hat{T}$ mit $\hat{T}(s) \xrightarrow{\beta} T[s/x]$
  - Was ist der Typ der Funktion $\hat{T}$?
    - $\hat{T}$ liefert für jedes $s \in S$ einen Typ, also $\hat{T} \in S{\to}\mathbb{U}$
    - $\boldsymbol{S{\to}\mathbb{U}}$ **muß ein Typ sein**
- $\mathbb{U}$ **bekommt eine Doppelrolle**
  - $\mathbb{U}$ ist ein Typ
  - $\mathbb{U}$ ist Repräsentant des Universums aller Typen

# Abhängigkeiten und Universen

- **Abhängigkeiten benötigen "höhere Typen"**
  - In $x{:}S{\to}T$ kommt $x$ frei in $T$ vor
    - $x$ muß durch jeden Wert $s \in S$ ersetzt werden können
    - $T[s/x]$ ist ein Typ für beliebige $s \in S$, gehört also zum Universum $\mathbb{U}$
  - Formale Beschreibung benötigt Funktion $\hat{T}$ mit $\hat{T}(s) \xrightarrow{\beta} T[s/x]$
  - Was ist der Typ der Funktion $\hat{T}$?
    - $\hat{T}$ liefert für jedes $s \in S$ einen Typ, also $\hat{T} \in S{\to}\mathbb{U}$
    - $\boldsymbol{S{\to}\mathbb{U}}$ **muß ein Typ sein**
- $\mathbb{U}$ **bekommt eine Doppelrolle**
  - $\mathbb{U}$ ist ein Typ
  - $\mathbb{U}$ ist Repräsentant des Universums aller Typen

**Wie formuliert man die Doppelrolle von $\mathbb{U}$?**

# Formalisierung der Rolle des Universums $\mathbb{U}$

**Elegantester Ansatz: definiere $\mathbb{U} \in \mathbb{U}$** (Martin-Löf, 1971)

# Formalisierung der Rolle des Universums $\mathbb{U}$

**Elegantester Ansatz: definiere $\mathbb{U} \in \mathbb{U}$** (Martin-Löf, 1971)

- **Einfacher, extrem ausdrucksstarker Kalkül**

# Formalisierung der Rolle des Universums $\mathbb{U}$

**Elegantester Ansatz: definiere $\mathbb{U} \in \mathbb{U}$** (Martin-Löf, 1971)

- **Einfacher, extrem ausdrucksstarker Kalkül**
- **Kein Problem mit Russel's Paradox**
  - Die Menge $\{S \mid S \notin S\}$ ist wegen der Typstruktur nicht formulierbar

# Formalisierung der Rolle des Universums $\mathbb{U}$

**Elegantester Ansatz: definiere $\mathbb{U} \in \mathbb{U}$** (Martin-Löf, 1971)

- **Einfacher, extrem ausdrucksstarker Kalkül**
- **Kein Problem mit Russel's Paradox**
  - Die Menge $\{S \mid S \notin S\}$ ist wegen der Typstruktur nicht formulierbar
- **Theorie scheitert aber an Girard's Paradox**
  - Eine Variante des Burali-Forti Paradox für Ordinalzahlen läßt sich in einer Theorie formalisieren, die abhängige Datentypen und das Axiom $\mathbb{U} \in \mathbb{U}$ enthält (Girard, 1972)

# Formalisierung der Rolle des Universums $\mathbb{U}$

**Elegantester Ansatz: definiere $\mathbb{U} \in \mathbb{U}$** (Martin-Löf, 1971)

- **Einfacher, extrem ausdrucksstarker Kalkül**
- **Kein Problem mit Russel's Paradox**
  - Die Menge $\{S \mid S \notin S\}$ ist wegen der Typstruktur nicht formulierbar
- **Theorie scheitert aber an Girard's Paradox**
  - Eine Variante des Burali-Forti Paradox für Ordinalzahlen läßt sich in einer Theorie formalisieren, die abhängige Datentypen und das Axiom $\mathbb{U} \in \mathbb{U}$ enthält (Girard, 1972)

$$\Downarrow$$

**Jede Theorie mit $x{:}S \to T$ und $\mathbb{U} \in \mathbb{U}$ ist inkonsistent**

# Girard's Paradox

- **Grundlage: Variante des Burali-Forti Paradox**

# Girard's Paradox

- **Grundlage: Variante des Burali-Forti Paradox**
  - Betrachte die Menge aller Wohlordnungen $(M, <_M)$ auf Mengen $M \in \mathbb{U}$
    - $<_M$ transitiv, ohne unendlich absteigende Kette $x_0 >_M x_1 >_M x_2 \ldots$

# Girard's Paradox

- **Grundlage: Variante des Burali-Forti Paradox**
  - Betrachte die Menge aller Wohlordnungen $(M,<_M)$ auf Mengen $M \in \mathbb{U}$

    $\cdot <_M$ transitiv, ohne unendlich absteigende Kette $x_0 >_M x_1 >_M x_2 \ldots$
  - Definiere eine Ordnung auf dieser Menge $(M,<_M) <_{\mathbb{U}} (M',<_{M'})$

    $\equiv \exists f{:}M \to M'.\ f$ ordnungsisomorph $\wedge\ \exists y{:}M'.\forall x{:}M.\ f(x) <_{M'} y$

# Girard's Paradox

- **Grundlage: Variante des Burali-Forti Paradox**
  - Betrachte die Menge aller Wohlordnungen $(M,<_M)$ auf Mengen $M \in \mathbb{U}$

    $\cdot <_M$ transitiv, ohne unendlich absteigende Kette $x_0 >_M x_1 >_M x_2 \ldots$
  - Definiere eine Ordnung auf dieser Menge $(M,<_M) <_{\mathbb{U}} (M',<_{M'})$

    $\equiv \exists f{:}M \to M'.\ f$ ordnungsisomorph $\wedge\ \exists y{:}M'.\forall x{:}M.\ f(x) <_{M'} y$
  - Dann ist $(\mathbb{U},<_{\mathbb{U}})$ eine Wohlordnung

# Girard's Paradox

- **Grundlage: Variante des Burali-Forti Paradox**
  - Betrachte die Menge aller Wohlordnungen $(M,<_M)$ auf Mengen $M \in \mathbb{U}$

    $\cdot <_M$ transitiv, ohne unendlich absteigende Kette $x_0 >_M x_1 >_M x_2 \ldots$
  - Definiere eine Ordnung auf dieser Menge $(M,<_M) <_{\mathbb{U}} (M',<_{M'})$

    $\equiv \exists f{:}M \to M'.\ f$ ordnungsisomorph $\wedge\ \exists y{:}M'.\forall x{:}M.\ f(x) <_{M'} y$
  - Dann ist $(\mathbb{U},<_{\mathbb{U}})$ eine Wohlordnung und $(M,<_M) <_{\mathbb{U}} (\mathbb{U},<_{\mathbb{U}})$

    gilt für jede Wohlordnung $(M,<_M)$ mit $M \in \mathbb{U}$

    d.h. $(\mathbb{U},<_{\mathbb{U}})$ ist ein maximales Element dieser Ordnung

# Girard's Paradox

- **Grundlage: Variante des Burali-Forti Paradox**
  - Betrachte die Menge aller Wohlordnungen $(M,<_M)$ auf Mengen $M \in \mathbb{U}$

    $\cdot <_M$ transitiv, ohne unendlich absteigende Kette $x_0 >_M x_1 >_M x_2 \ldots$
  - Definiere eine Ordnung auf dieser Menge $(M,<_M) <_{\mathbb{U}} (M',<_{M'})$

    $\equiv \exists f{:}M \to M'.\ f$ ordnungsisomorph $\wedge\ \exists y{:}M'.\forall x{:}M.\ f(x) <_{M'} y$
  - Dann ist $(\mathbb{U},<_{\mathbb{U}})$ eine Wohlordnung und $(M,<_M) <_{\mathbb{U}} (\mathbb{U},<_{\mathbb{U}})$

    gilt für jede Wohlordnung $(M,<_M)$ mit $M \in \mathbb{U}$

    d.h. $(\mathbb{U},<_{\mathbb{U}})$ ist ein maximales Element dieser Ordnung
  - Wegen $\mathbb{U} \in \mathbb{U}$ folgt dann auch $(\mathbb{U},<_{\mathbb{U}}) <_{\mathbb{U}} (\mathbb{U},<_{\mathbb{U}})$

# Girard's Paradox

- **Grundlage: Variante des Burali-Forti Paradox**
  - Betrachte die Menge aller Wohlordnungen $(M,<_M)$ auf Mengen $M \in \mathbb{U}$

    $\cdot <_M$ transitiv, ohne unendlich absteigende Kette $x_0 >_M x_1 >_M x_2 \ldots$
  - Definiere eine Ordnung auf dieser Menge $(M,<_M) <_{\mathbb{U}} (M',<_{M'})$

    $\equiv \exists f{:}M \to M'.\ f$ ordnungsisomorph $\wedge\ \exists y{:}M'.\forall x{:}M.\ f(x) <_{M'} y$
  - Dann ist $(\mathbb{U},<_{\mathbb{U}})$ eine Wohlordnung und $(M,<_M) <_{\mathbb{U}} (\mathbb{U},<_{\mathbb{U}})$

    gilt für jede Wohlordnung $(M,<_M)$ mit $M \in \mathbb{U}$

    d.h. $(\mathbb{U},<_{\mathbb{U}})$ ist ein maximales Element dieser Ordnung
  - Wegen $\mathbb{U} \in \mathbb{U}$ folgt dann auch $(\mathbb{U},<_{\mathbb{U}}) <_{\mathbb{U}} (\mathbb{U},<_{\mathbb{U}})$
  - Damit gibt es in $<_{\mathbb{U}}$ eine unendlich absteigende Kette

# Girard's Paradox

- **Grundlage: Variante des Burali-Forti Paradox**
  - Betrachte die Menge aller Wohlordnungen $(M,<_M)$ auf Mengen $M \in \mathbb{U}$

    $\cdot <_M$ transitiv, ohne unendlich absteigende Kette $x_0 >_M x_1 >_M x_2 \ldots$
  - Definiere eine Ordnung auf dieser Menge $(M,<_M) <_{\mathbb{U}} (M',<_{M'})$

    $\equiv \exists f{:}M \to M'.\ f$ ordnungsisomorph $\wedge\ \exists y{:}M'.\forall x{:}M.\ f(x) <_{M'} y$
  - Dann ist $(\mathbb{U},<_{\mathbb{U}})$ eine Wohlordnung und $(M,<_M) <_{\mathbb{U}} (\mathbb{U},<_{\mathbb{U}})$

    gilt für jede Wohlordnung $(M,<_M)$ mit $M \in \mathbb{U}$

    d.h. $(\mathbb{U},<_{\mathbb{U}})$ ist ein maximales Element dieser Ordnung
  - Wegen $\mathbb{U} \in \mathbb{U}$ folgt dann auch $(\mathbb{U},<_{\mathbb{U}}) <_{\mathbb{U}} (\mathbb{U},<_{\mathbb{U}})$
  - Damit gibt es in $<_{\mathbb{U}}$ eine unendlich absteigende Kette
- **Paradox formulierbar, falls Theorie $\mathbb{U} \in \mathbb{U}$ zuläßt**

# Girard's Paradox

- **Grundlage: Variante des Burali-Forti Paradox**
  - Betrachte die Menge aller Wohlordnungen $(M,<_M)$ auf Mengen $M \in \mathbb{U}$

    $\cdot <_M$ transitiv, ohne unendlich absteigende Kette $x_0 >_M x_1 >_M x_2 \ldots$
  - Definiere eine Ordnung auf dieser Menge $(M,<_M) <_{\mathbb{U}} (M',<_{M'})$

    $\equiv \exists f{:}M \to M'.\ f$ ordnungsisomorph $\wedge\ \exists y{:}M'.\forall x{:}M.\ f(x) <_{M'} y$
  - Dann ist $(\mathbb{U},<_{\mathbb{U}})$ eine Wohlordnung und $(M,<_M) <_{\mathbb{U}} (\mathbb{U},<_{\mathbb{U}})$

    gilt für jede Wohlordnung $(M,<_M)$ mit $M \in \mathbb{U}$

    d.h. $(\mathbb{U},<_{\mathbb{U}})$ ist ein maximales Element dieser Ordnung
  - Wegen $\mathbb{U} \in \mathbb{U}$ folgt dann auch $(\mathbb{U},<_{\mathbb{U}}) <_{\mathbb{U}} (\mathbb{U},<_{\mathbb{U}})$
  - Damit gibt es in $<_{\mathbb{U}}$ eine unendlich absteigende Kette
- **Paradox formulierbar, falls Theorie $\mathbb{U} \in \mathbb{U}$ zuläßt**
  - Alle Ordnungskonzepte sind mit Higher-Order Quantoren definierbar

# Girard's Paradox

- **Grundlage: Variante des Burali-Forti Paradox**
  - Betrachte die Menge aller Wohlordnungen $(M,<_M)$ auf Mengen $M \in \mathbb{U}$

    $\cdot <_M$ transitiv, ohne unendlich absteigende Kette $x_0 >_M x_1 >_M x_2 \ldots$
  - Definiere eine Ordnung auf dieser Menge $(M,<_M) <_{\mathbb{U}} (M',<_{M'})$

    $\equiv \exists f{:}M \to M'.\ f$ ordnungsisomorph $\wedge\ \exists y{:}M'.\forall x{:}M.\ f(x) <_{M'} y$
  - Dann ist $(\mathbb{U},<_{\mathbb{U}})$ eine Wohlordnung und $(M,<_M) <_{\mathbb{U}} (\mathbb{U},<_{\mathbb{U}})$

    gilt für jede Wohlordnung $(M,<_M)$ mit $M \in \mathbb{U}$

    d.h. $(\mathbb{U},<_{\mathbb{U}})$ ist ein maximales Element dieser Ordnung
  - Wegen $\mathbb{U} \in \mathbb{U}$ folgt dann auch $(\mathbb{U},<_{\mathbb{U}}) <_{\mathbb{U}} (\mathbb{U},<_{\mathbb{U}})$
  - Damit gibt es in $<_{\mathbb{U}}$ eine unendlich absteigende Kette
- **Paradox formulierbar, falls Theorie $\mathbb{U} \in \mathbb{U}$ zuläßt**
  - Alle Ordnungskonzepte sind mit Higher-Order Quantoren definierbar
  - $x{:}S \to T$ reicht zur Formalisierung der Quantoren

# Alternative Repräsentationen des Universums

- **Theorie benötigt Universum und Abhängigkeiten**

# Alternative Repräsentationen des Universums

- **Theorie benötigt Universum und Abhängigkeiten**
  - Universum $\mathbb{U}$ ist notwendig zur Formulierung des "Typseins"

# Alternative Repräsentationen des Universums

- **Theorie benötigt Universum und Abhängigkeiten**
  - Universum $\mathbb{U}$ ist notwendig zur Formulierung des "Typseins"
  - Abhängige Datentypen sind notwendig für die Ausdruckskraft

# Alternative Repräsentationen des Universums

- **Theorie benötigt Universum und Abhängigkeiten**
  - Universum $\mathbb{U}$ ist notwendig zur Formulierung des "Typseins"
  - Abhängige Datentypen sind notwendig für die Ausdruckskraft
  - Formulierung der Abhängigkeit benötigt Typen mit freien Variablen

# Alternative Repräsentationen des Universums

- **Theorie benötigt Universum und Abhängigkeiten**
  - Universum $\mathbb{U}$ ist notwendig zur Formulierung des "Typseins"
  - Abhängige Datentypen sind notwendig für die Ausdruckskraft
  - Formulierung der Abhängigkeit benötigt Typen mit freien Variablen
  - Der elegantest Weg ist versperrt: $\mathbb{U}$ darf selbst kein Typ sein

# Alternative Repräsentationen des Universums

- **Theorie benötigt Universum und Abhängigkeiten**
  - Universum $\mathbb{U}$ ist notwendig zur Formulierung des "Typseins"
  - Abhängige Datentypen sind notwendig für die Ausdruckskraft
  - Formulierung der Abhängigkeit benötigt Typen mit freien Variablen
  - Der elegantest Weg ist versperrt: $\mathbb{U}$ darf selbst kein Typ sein
- **Syntaktische Alternative** System F, Coq
  - Trennung: Variablen in Typausdrücken sind keine Objektvariablen
    - Funktionen $\hat{T} \in S \to \mathbb{U}$ werden für $x{:}S \to T[x]$ nicht benötigt

# Alternative Repräsentationen des Universums

- **Theorie benötigt Universum und Abhängigkeiten**
  - Universum $\mathbb{U}$ ist notwendig zur Formulierung des "Typseins"
  - Abhängige Datentypen sind notwendig für die Ausdruckskraft
  - Formulierung der Abhängigkeit benötigt Typen mit freien Variablen
  - Der elegantest Weg ist versperrt: $\mathbb{U}$ darf selbst kein Typ sein
- **Syntaktische Alternative** System F, Coq
  - Trennung: Variablen in Typausdrücken sind keine Objektvariablen
    - Funktionen $\hat{T} \in S \to \mathbb{U}$ werden für $x{:}S \to T[x]$ nicht benötigt
  - Alles andere kann i.w. beibehalten werden

# Alternative Repräsentationen des Universums

- **Theorie benötigt Universum und Abhängigkeiten**
  - Universum $\mathbb{U}$ ist notwendig zur Formulierung des "Typseins"
  - Abhängige Datentypen sind notwendig für die Ausdruckskraft
  - Formulierung der Abhängigkeit benötigt Typen mit freien Variablen
  - Der elegantest Weg ist versperrt: $\mathbb{U}$ darf selbst kein Typ sein
- **Syntaktische Alternative** System F, Coq
  - Trennung: Variablen in Typausdrücken sind keine Objektvariablen
    - Funktionen $\hat{T} \in S \to \mathbb{U}$ werden für $x{:}S \to T[x]$ nicht benötigt
  - Alles andere kann i.w. beibehalten werden
- **Semantische Alternative** Martin-Löf (1978), Nuprl

# Alternative Repräsentationen des Universums

- **Theorie benötigt Universum und Abhängigkeiten**
  - Universum $\mathbb{U}$ ist notwendig zur Formulierung des "Typseins"
  - Abhängige Datentypen sind notwendig für die Ausdruckskraft
  - Formulierung der Abhängigkeit benötigt Typen mit freien Variablen
  - Der elegantest Weg ist versperrt: $\mathbb{U}$ darf selbst kein Typ sein
- **Syntaktische Alternative** System F, Coq
  - Trennung: Variablen in Typausdrücken sind keine Objektvariablen
    - Funktionen $\hat{T} \in S \to \mathbb{U}$ werden für $x{:}S \to T[x]$ nicht benötigt
  - Alles andere kann i.w. beibehalten werden
- **Semantische Alternative** Martin-Löf (1978), Nuprl
  - Unterschied von Objekt- und Typausdrücken liegt in der Bedeutung
    - $\mathbb{U}$ und $S \to \mathbb{U}$ sind keine "einfache" Typen wie $\mathbb{N}$ oder $S \to T$

# Alternative Repräsentationen des Universums

- **Theorie benötigt Universum und Abhängigkeiten**
  - Universum $\mathbb{U}$ ist notwendig zur Formulierung des "Typseins"
  - Abhängige Datentypen sind notwendig für die Ausdruckskraft
  - Formulierung der Abhängigkeit benötigt Typen mit freien Variablen
  - Der elegantest Weg ist versperrt: $\mathbb{U}$ darf selbst kein Typ sein
- **Syntaktische Alternative** System F, Coq
  - Trennung: Variablen in Typausdrücken sind keine Objektvariablen
    - Funktionen $\hat{T} \in S \to \mathbb{U}$ werden für $x{:}S \to T[x]$ nicht benötigt
  - Alles andere kann i.w. beibehalten werden
- **Semantische Alternative** Martin-Löf (1978), Nuprl
  - Unterschied von Objekt- und Typausdrücken liegt in der Bedeutung
    - $\mathbb{U}$ und $S \to \mathbb{U}$ sind keine "einfache" Typen wie $\mathbb{N}$ oder $S \to T$
  - Universum gliedert sich in Stufen $\mathbb{U} \in \mathbb{U}_2 \in \mathbb{U}_3 \ldots$
    - Nur die Gesamtheit aller Stufen beschreibt das "Typsein"

# Alternative Repräsentationen des Universums

- **Theorie benötigt Universum und Abhängigkeiten**
  - Universum $\mathbb{U}$ ist notwendig zur Formulierung des "Typseins"
  - Abhängige Datentypen sind notwendig für die Ausdruckskraft
  - Formulierung der Abhängigkeit benötigt Typen mit freien Variablen
  - Der elegantest Weg ist versperrt: $\mathbb{U}$ darf selbst kein Typ sein
- **Syntaktische Alternative** System F, Coq
  - Trennung: Variablen in Typausdrücken sind keine Objektvariablen
    - Funktionen $\hat{T} \in S \to \mathbb{U}$ werden für $x{:}S \to T[x]$ nicht benötigt
  - Alles andere kann i.w. beibehalten werden
- **Semantische Alternative** Martin-Löf (1978), Nuprl
  - Unterschied von Objekt- und Typausdrücken liegt in der Bedeutung
    - $\mathbb{U}$ und $S \to \mathbb{U}$ sind keine "einfache" Typen wie $\mathbb{N}$ oder $S \to T$
  - Universum gliedert sich in Stufen $\mathbb{U} \in \mathbb{U}_2 \in \mathbb{U}_3 \ldots$
    - Nur die Gesamtheit aller Stufen beschreibt das "Typsein"

**Keine der Alternativen ist perfekt**

# Syntaktische Abtrennung des Universums

- **Spezielle Syntax vermeidet $\mathbb{U}$ in Typausdrücken**
  - Strikte syntaktische Trennung von Objekt- und Typausdrücken
  - In $x{:}S{\to}T[x]$ ist $x$ eine Spezies Variable
    und $T[x]$ ein Typ mit Parameter aus $S$

# Syntaktische Abtrennung des Universums

- **Spezielle Syntax vermeidet $\mathbb{U}$ in Typausdrücken**
  - Strikte syntaktische Trennung von Objekt- und Typausdrücken
  - In $x{:}S{\to}T[x]$ ist $x$ eine Spezies Variable
    und $T[x]$ ein Typ mit Parameter aus $S$
- **Girard's Paradox wird elegant umgangen**
  - Typisierte Objekte können selbst keine Typen enthalten
  - $\mathbb{U}$ ist selbst kein Typ — $\mathbb{U} \in \mathbb{U}$ ist nicht formulierbar
  - Unklar ob andere Paradoxien formulierbar sind (unwahrscheinlich)

# Syntaktische Abtrennung des Universums

- **Spezielle Syntax vermeidet $\mathbb{U}$ in Typausdrücken**
  - Strikte syntaktische Trennung von Objekt- und Typausdrücken
  - In $x{:}S{\to}T[x]$ ist $x$ eine Spezies Variable
    und $T[x]$ ein Typ mit Parameter aus $S$
- **Girard's Paradox wird elegant umgangen**
  - Typisierte Objekte können selbst keine Typen enthalten
  - $\mathbb{U}$ ist selbst kein Typ — $\mathbb{U} \in \mathbb{U}$ ist nicht formulierbar
  - Unklar ob andere Paradoxien formulierbar sind (unwahrscheinlich)
- **Theorie kann minimal bleiben**
  - Nur abhängiger Funktionenraum ("Π-Type") erforderlich
  - Andere Konzepte wie bisher repräsentierbar (unnatürlich?)

# Syntaktische Abtrennung des Universums

- **Spezielle Syntax vermeidet $\mathbb{U}$ in Typausdrücken**
  - Strikte syntaktische Trennung von Objekt- und Typausdrücken
  - In $x{:}S{\to}T[x]$ ist $x$ eine Spezies Variable
    und $T[x]$ ein Typ mit Parameter aus $S$
- **Girard's Paradox wird elegant umgangen**
  - Typisierte Objekte können selbst keine Typen enthalten
  - $\mathbb{U}$ ist selbst kein Typ — $\mathbb{U} \in \mathbb{U}$ ist nicht formulierbar
  - Unklar ob andere Paradoxien formulierbar sind (unwahrscheinlich)
- **Theorie kann minimal bleiben**
  - Nur abhängiger Funktionenraum ("Π-Type") erforderlich
  - Andere Konzepte wie bisher repräsentierbar (unnatürlich?)
- **Beweisformalismus hat relativ komplexe Syntax**
  - Formulierung von Coq verbessert System F signifikant

# Semantische Auftrennung des Universums

- **Semantisch der klarste Ansatz**
  - Abstraktes 'Typsein' ist syntaktisch nicht mit einem Konzept erfaßbar
    - Ein Universum von Typen ist selbst kein einfacher Typ

# Semantische Auftrennung des Universums

- **Semantisch der klarste Ansatz**
  - Abstraktes 'Typsein' ist syntaktisch nicht mit einem Konzept erfaßbar
    - Ein Universum von Typen ist selbst kein einfacher Typ
  - Das Universum einfacher Typen ist Element eines höheren Universums

# Semantische Auftrennung des Universums

- **Semantisch der klarste Ansatz**
  - Abstraktes 'Typsein' ist syntaktisch nicht mit einem Konzept erfaßbar
    - Ein Universum von Typen ist selbst kein einfacher Typ
  - Das Universum einfacher Typen ist Element eines höheren Universums
- **Typhierarchie** $\mathbb{U} \in \mathbb{U}_2 \in \mathbb{U}_3 \ldots$
  - Typsein ist nur durch Gesamtheit aller Universen repräsentierbar

# Semantische Auftrennung des Universums

- **Semantisch der klarste Ansatz**
  - Abstraktes 'Typsein' ist syntaktisch nicht mit einem Konzept erfaßbar
    - Ein Universum von Typen ist selbst kein einfacher Typ
  - Das Universum einfacher Typen ist Element eines höheren Universums
- **Typhierarchie** $\mathbb{U} \in \mathbb{U}_2 \in \mathbb{U}_3 \ldots$
  - Typsein ist nur durch Gesamtheit aller Universen repräsentierbar
  - Vermeide Komplikationen mit Mischtypen wie $S \to \mathbb{U}$ durch "kumulative" Universen $\mathbb{U} \subseteq \mathbb{U}_2 \subseteq \mathbb{U}_3 \ldots$ –

# Semantische Auftrennung des Universums

- **Semantisch der klarste Ansatz**
  - Abstraktes 'Typsein' ist syntaktisch nicht mit einem Konzept erfaßbar
    - Ein Universum von Typen ist selbst kein einfacher Typ
  - Das Universum einfacher Typen ist Element eines höheren Universums
- **Typhierarchie** $\mathbb{U} \in \mathbb{U}_2 \in \mathbb{U}_3 \ldots$
  - Typsein ist nur durch Gesamtheit aller Universen repräsentierbar
  - Vermeide Komplikationen mit Mischtypen wie $S \to \mathbb{U}$ durch "kumulative" Universen $\mathbb{U} \subseteq \mathbb{U}_2 \subseteq \mathbb{U}_3 \ldots$ –
- **Theorie wird sehr umfangreich**
  - Viele Konzepte nicht in voller Allgemeinheit simulierbar
    - Definitionen mit Universumsparameter wären nicht sinnvoll

# Semantische Auftrennung des Universums

- **Semantisch der klarste Ansatz**
  - Abstraktes 'Typsein' ist syntaktisch nicht mit einem Konzept erfaßbar
    - Ein Universum von Typen ist selbst kein einfacher Typ
  - Das Universum einfacher Typen ist Element eines höheren Universums
- **Typhierarchie** $\mathbb{U} \in \mathbb{U}_2 \in \mathbb{U}_3 \ldots$
  - Typsein ist nur durch Gesamtheit aller Universen repräsentierbar
  - Vermeide Komplikationen mit Mischtypen wie $S \to \mathbb{U}$ durch "kumulative" Universen $\mathbb{U} \subseteq \mathbb{U}_2 \subseteq \mathbb{U}_3 \ldots$ –
- **Theorie wird sehr umfangreich**
  - Viele Konzepte nicht in voller Allgemeinheit simulierbar
    - Definitionen mit Universumsparameter wären nicht sinnvoll
  - Nichtsimulierbare Konzepte müssen explizit formalisiert werden
    - Aufwendig für Theorieentwickler, sehr nützlich für Anwender

# Semantische Auftrennung des Universums

- **Semantisch der klarste Ansatz**
  - Abstraktes 'Typsein' ist syntaktisch nicht mit einem Konzept erfaßbar
    - Ein Universum von Typen ist selbst kein einfacher Typ
  - Das Universum einfacher Typen ist Element eines höheren Universums
- **Typhierarchie** $\mathbb{U} \in \mathbb{U}_2 \in \mathbb{U}_3 \ldots$
  - Typsein ist nur durch Gesamtheit aller Universen repräsentierbar
  - Vermeide Komplikationen mit Mischtypen wie $S \rightarrow \mathbb{U}$ durch "kumulative" Universen $\mathbb{U} \subseteq \mathbb{U}_2 \subseteq \mathbb{U}_3 \ldots$ –
- **Theorie wird sehr umfangreich**
  - Viele Konzepte nicht in voller Allgemeinheit simulierbar
    - Definitionen mit Universumsparameter wären nicht sinnvoll
  - Nichtsimulierbare Konzepte müssen explizit formalisiert werden
    - Aufwendig für Theorieentwickler, sehr nützlich für Anwender
  - Formulierung von Nuprl erweitert Martin-Löf's Theorie erheblich

# Formale Konstruktive Theorien – wohin?

- **Theorie abhängiger Typen ist ausdrucksstark**

# Formale Konstruktive Theorien – wohin?

- **Theorie abhängiger Typen ist ausdrucksstark**

  Umfaßt Prädikatenlogik (Schließen über logische Struktur)

  $\lambda$-Kalkül (Schließen über Wert von Ausdrücken)

  Typanalyse (Schließen über Eigenschaften von Ausdrücken)

# Formale Konstruktive Theorien – wohin?

- **Theorie abhängiger Typen ist ausdrucksstark**

  Umfaßt Prädikatenlogik (Schließen über logische Struktur)

  $\lambda$-Kalkül (Schließen über Wert von Ausdrücken)

  Typanalyse (Schließen über Eigenschaften von Ausdrücken)

- **Minimaler Kalkül elegant aber unnatürlich**
  - $\mathbb{B}, \mathbb{N}, \times, \wedge, \vee, \neg, \exists, =, \ldots$ sind fundamentale mathematische Konzepte

# Formale Konstruktive Theorien – wohin?

- **Theorie abhängiger Typen ist ausdrucksstark**

  Umfaßt Prädikatenlogik (Schließen über logische Struktur)

  $\lambda$-Kalkül (Schließen über Wert von Ausdrücken)

  Typanalyse (Schließen über Eigenschaften von Ausdrücken)

- **Minimaler Kalkül elegant aber unnatürlich**
  - $\mathbb{B}, \mathbb{N}, \times, \wedge, \vee, \neg, \exists, =, \ldots$ sind fundamentale mathematische Konzepte
  - Anwendungen benötigen Formalisierungen dieser Konzepte

# Formale Konstruktive Theorien – wohin?

- **Theorie abhängiger Typen ist ausdrucksstark**

  Umfaßt Prädikatenlogik (Schließen über logische Struktur)

  $\lambda$-Kalkül (Schließen über Wert von Ausdrücken)

  Typanalyse (Schließen über Eigenschaften von Ausdrücken)

- **Minimaler Kalkül elegant aber unnatürlich**
  - $\mathbb{B}, \mathbb{N}, \times, \wedge, \vee, \neg, \exists, =, \ldots$ sind fundamentale mathematische Konzepte
  - Anwendungen benötigen Formalisierungen dieser Konzepte
  - Eine Simulation erschwert formales Schließen in der Praxis

# Formale Konstruktive Theorien – wohin?

- **Theorie abhängiger Typen ist ausdrucksstark**

  Umfaßt Prädikatenlogik (Schließen über logische Struktur)

  $\lambda$-Kalkül (Schließen über Wert von Ausdrücken)

  Typanalyse (Schließen über Eigenschaften von Ausdrücken)

- **Minimaler Kalkül elegant aber unnatürlich**
  - $\mathbb{B}, \mathbb{N}, \times, \wedge, \vee, \neg, \exists, =, \ldots$ sind fundamentale mathematische Konzepte
  - Anwendungen benötigen Formalisierungen dieser Konzepte
  - Eine Simulation erschwert formales Schließen in der Praxis

- **Formalisiere wichtige Konzepte explizit**

# Formale Konstruktive Theorien – wohin?

- **Theorie abhängiger Typen ist ausdrucksstark**

  Umfaßt Prädikatenlogik (Schließen über logische Struktur)

  $\lambda$-Kalkül (Schließen über Wert von Ausdrücken)

  Typanalyse (Schließen über Eigenschaften von Ausdrücken)

- **Minimaler Kalkül elegant aber unnatürlich**
  - $\mathbb{B}, \mathbb{N}, \times, \wedge, \vee, \neg, \exists, =, \ldots$ sind fundamentale mathematische Konzepte
  - Anwendungen benötigen Formalisierungen dieser Konzepte
  - Eine Simulation erschwert formales Schließen in der Praxis

- **Formalisiere wichtige Konzepte explizit**
  - **Konstruktive (intuitionistische) Typentheorie** präzisiert syntaktische Repräsentation, Bedeutung und Inferenzregeln für Standardkonzepte aus Mathematik & Programmierung

# Formale Konstruktive Theorien – wohin?

- **Theorie abhängiger Typen ist ausdrucksstark**

  Umfaßt Prädikatenlogik (Schließen über logische Struktur)

  $\lambda$-Kalkül (Schließen über Wert von Ausdrücken)

  Typanalyse (Schließen über Eigenschaften von Ausdrücken)

- **Minimaler Kalkül elegant aber unnatürlich**
  - $\mathbb{B}, \mathbb{N}, \times, \wedge, \vee, \neg, \exists, =, \ldots$ sind fundamentale mathematische Konzepte
  - Anwendungen benötigen Formalisierungen dieser Konzepte
  - Eine Simulation erschwert formales Schließen in der Praxis

- **Formalisiere wichtige Konzepte explizit**
  - **Konstruktive (intuitionistische) Typentheorie** präzisiert syntaktische Repräsentation, Bedeutung und Inferenzregeln für Standardkonzepte aus Mathematik & Programmierung
  - Umfang des Formalismus erfordert Systematik für Aufbau des Kalküls

# Formale Konstruktive Theorien – wohin?

- **Theorie abhängiger Typen ist ausdrucksstark**

  Umfaßt Prädikatenlogik (Schließen über logische Struktur)

  $\lambda$-Kalkül (Schließen über Wert von Ausdrücken)

  Typanalyse (Schließen über Eigenschaften von Ausdrücken)

- **Minimaler Kalkül elegant aber unnatürlich**
  - $\mathbb{B}, \mathbb{N}, \times, \wedge, \vee, \neg, \exists, =, \ldots$ sind fundamentale mathematische Konzepte
  - Anwendungen benötigen Formalisierungen dieser Konzepte
  - Eine Simulation erschwert formales Schließen in der Praxis

- **Formalisiere wichtige Konzepte explizit**
  - **Konstruktive (intuitionistische) Typentheorie** präzisiert syntaktische Repräsentation, Bedeutung und Inferenzregeln für Standardkonzepte aus Mathematik & Programmierung
  - Umfang des Formalismus erfordert Systematik für Aufbau des Kalküls

# Formale Konstruktive Theorien – wohin?

- **Theorie abhängiger Typen ist ausdrucksstark**

  Umfaßt Prädikatenlogik (Schließen über logische Struktur)

  $\lambda$-Kalkül (Schließen über Wert von Ausdrücken)

  Typanalyse (Schließen über Eigenschaften von Ausdrücken)

- **Minimaler Kalkül elegant aber unnatürlich**
  - $\mathbb{B}, \mathbb{N}, \times, \wedge, \vee, \neg, \exists, =, \ldots$ sind fundamentale mathematische Konzepte
  - Anwendungen benötigen Formalisierungen dieser Konzepte
  - Eine Simulation erschwert formales Schließen in der Praxis

- **Formalisiere wichtige Konzepte explizit**
  - **Konstruktive (intuitionistische) Typentheorie** präzisiert syntaktische Repräsentation, Bedeutung und Inferenzregeln für Standardkonzepte aus Mathematik & Programmierung
  - Umfang des Formalismus erfordert Systematik für Aufbau des Kalküls