Inferenzmethoden

Einheit 17

Logik höherer Stufe

1. Syntax und Semantik
2. Simulation mathematischer Konstrukte
3. Beweisführung in Logik höherer Stufe

Logik höherer Stufe

- **Logik erster Stufe hat nur einfache Variablen**
  – Keine Quantifizierung über Funktions- oder Prädikatensymbole erlaubt
- **Konstrukte höherer Stufe kommen überall vor**
  – Funktionen (2. Stufe): "Bestimme x+2 bei Eingabe x" $\lambda x.x{+}2$
  – Induktionsprinzip: $\forall P.P(0) \land (\forall y{:}\mathbb{N}.P(y) \Rightarrow P(y{+}1)) \Rightarrow \forall x{:}\mathbb{N}.P(x)$
  – Zwischenwertsatz: $\forall f{:}\mathbb{R}{\rightarrow}\mathbb{R}.\forall a < b{:}\mathbb{R}.(f(a){>}0 \land f(b){<}0) \Rightarrow \exists x{:}\mathbb{R}.f(x){=}0$
  – Funktionale (3. Stufe): Ableitungsoperator $\lambda f.df/dx$
  – Quantifizierung über Funktionale, ...
- **Logik höherer Stufe hat keine Einschränkungen**
  – Extrem einfache Syntax: x, $\lambda x.t$, $f(a)$, $\forall x\, P$, $P \Rightarrow Q$
  – Fester Auswertungsmechanismus (β-)Reduktion: $(\lambda x.t)(a) \longrightarrow t[a/x]$
- **Logik höherer Stufe ist minimale Grundlagentheorie**
  – Keine Abstützung auf Mengentheorie erforderlich
  – Reduktion erklärt Wert von Ausdrücken
  – Mathematische Konzepte werden nicht über Axiome erklärt sondern als definitorische Abkürzung für komplexe Terme (logizistischer Ansatz)

Logik höherer Stufe – Syntax

- **Alphabet für erlaubte Symbole**
  - $\mathcal{V}$: Variablensymbole
- **Terme**
  - Variablen $\boldsymbol{x} \in \mathcal{V}$
  - $\boldsymbol{\lambda x.t}$, wobei $x \in \mathcal{V}$ und t Term — λ-Abstraktion
  - $\boldsymbol{f\ t}$, wobei t und f Terme — Applikation
  - $(\boldsymbol{t})$, wobei t Term
  - $(\boldsymbol{P \Rightarrow Q})$, wobei P,Q Terme
  - $(\forall \boldsymbol{x}\ \boldsymbol{P})$, wobei $x \in \mathcal{V}$ und P Term
- **Konventionen**
  - Applikation bindet stärker als λ-Abstraktion
  - Applikation ist links-assoziativ: $\boldsymbol{f\ t_1\ t_2} \mathrel{\hat{=}} (\boldsymbol{f\ t_1})\ \boldsymbol{t_2}$
  - Notation $f(t_1, \ldots t_n)$ entspricht iterierter Applikation $f\ \ t_1 \ldots t_n$

Logik höherer Stufe – (Wert-)Semantik

- **α-Konversion:** Umbenennung gebundener Variablen
  - Ersetze Teilterm der Gestalt $\lambda x.t$ durch $\lambda z.t[z/x]$ (z neue Variable)
  - Ersetze Teilterm der Gestalt $\forall x\ P$ durch $\forall z\ P[z/x]$
  - Terme t und u sind kongruent (α-konvertibel), wenn sie auseinander durch endlich viele Umbenennungen gebundener Variablen entstehen
- **(β-)Reduktion:** Auswertung von Termen
  - Ersetze Teilterm der Gestalt $(\lambda x.t)(s)$ (Redex) durch $t[s/x]$ (Kontraktum)
  - t ist reduzierbar auf u ($\boldsymbol{t \xrightarrow{*} u}$), wenn u aus t durch endlich viele Reduktionen und Umbenennungen entsteht ($\hat{=}$ Termersetzung)
- **(Semantische) Gleichheit $\boldsymbol{t = s}$**
  - Es gibt einen Term u mit $t \xrightarrow{*} u$ und $s \xrightarrow{*} u$
- **Normalform von $\boldsymbol{t}$:** Wert eines Terms
  - Irreduzibler (Redex-freier) Term u mit $t = u$
  - Der Wert eines Terms ist eindeutig ($\xrightarrow{*}$ ist konfluent)
  - Nicht jeder Term hat einen Wert (keine starke Normalisierbarkeit)
- **Semantik von $\forall$ und $\Rightarrow$ analog zur Prädikatenlogik**

Definition logischer Standardkonzepte

Junktoren werden zu benutzerdefinierten Funktionen

Konjunktion	$\wedge$	$\equiv$	$\lambda A.\lambda B.\forall P\,(A \Rightarrow (B \Rightarrow P)) \Rightarrow P$
Disjunktion	$\vee$	$\equiv$	$\lambda A.\lambda B.\forall P\,((A \Rightarrow P) \Rightarrow (B \Rightarrow P)) \Rightarrow P$
Falschheit	$\bot$	$\equiv$	$\forall P\; P$ $\qquad$ $(intuitionistisch)$
Negation	$\neg$	$\equiv$	$\lambda A.\forall P\,(A \Rightarrow P)$
Existenzquantor	$\exists \boldsymbol{x}\; \boldsymbol{A}$	$\equiv$	$\forall P\,(\forall x\,(A \Rightarrow P)) \Rightarrow P$
Gleichheit	$\doteq$	$\equiv$	$\lambda x.\lambda y.\forall P\,(\,P(x) \Rightarrow P(y)\,)$

Definition von Zahlen: Church Numerals

$$\overline{n} \equiv \lambda f.\lambda x.\ f^n\,x \equiv \lambda f.\lambda x.\ \underbrace{f\,(f \ldots (f\,x)\ \ldots)}_{n\text{-}mal}$$

$$\mathsf{s} \equiv \lambda n.\lambda f.\lambda x.\ n\ f\ (f\ x)$$

$$\mathsf{add} \equiv \lambda m.\lambda n.\lambda f.\lambda x.\ m\ f\ (n\ f\ x)$$

$$\mathsf{mul} \equiv \lambda m.\lambda n.\lambda f.\lambda x.\ m\ (n\ f)\ x$$

$$\mathsf{exp} \equiv \lambda m.\lambda n.\lambda f.\lambda x.\ n\ m\ f\ x$$

$$\mathsf{zero} \equiv \lambda n.\ n\ (\lambda n.\mathsf{F})\ \mathsf{T}$$

$$\mathsf{p} \equiv \lambda n.\ (n\,(\lambda fx.\ \langle \mathsf{s}, \mathsf{let}\ \ \langle \mathtt{f}\,,\mathtt{x}\rangle = \mathtt{fx}\ \mathsf{in}\ \ \mathtt{f}\ \ \mathtt{x}\rangle)\,)\,\langle \lambda z.\overline{0}, \overline{0}\rangle\,).2$$

$$N \equiv \lambda x.\forall P\ (\forall y\,P(y) \Rightarrow P(\mathsf{s}\ y)) \Rightarrow P(\overline{0}) \Rightarrow P(x) \qquad \boxed{N(x) \hat{=} x \in \mathbb{N}}$$

Abkürzungen

$$\mathsf{T} \equiv \lambda x.\lambda y.x$$

$$\mathsf{F} \equiv \lambda x.\lambda y.y$$

$$\mathsf{if}\ b\ \ \mathsf{then}\ s\ \ \mathsf{else}\ t \equiv b\,s\,t$$

$$\langle s,t\rangle \equiv \lambda p.\ p\,s\,t$$

$$\mathsf{let}\ \ \langle x\,,y\rangle = pair\ \ \mathsf{in}\ \ t \equiv pair\ \ (\lambda x.\lambda y.t)$$

Church Numerals: Auswertung von Funktionen

$$
\begin{array}{lcll}
\mathsf{s}\ \overline{n} & \equiv & (\lambda n.\lambda f.\lambda x.\ n\ f\ (f\ x))\ (\lambda f.\lambda x.\ f^n\ x) & \\
& \longrightarrow & \lambda f.\lambda x.\ (\lambda f.\lambda x.\ f^n\ x)\ f\ (f\ x) & \\
& \longrightarrow & \lambda f.\lambda x.\ (\lambda x.\ f^n\ x)\ (f\ x) & \\
& \longrightarrow & \lambda f.\lambda x.\ f^n\ (f\ x) & \\
& \longrightarrow & \lambda f.\lambda x.\ f^{n+1}\ x & \equiv\ \overline{n+1} \\
\mathsf{add}\ \overline{m}\ \overline{n} & \equiv & (\lambda m.\lambda n.\lambda f.\lambda x.\ m\ f\ (n\ f\ x))\ \overline{m}\ \overline{n} & \\
& \longrightarrow & (\lambda n.\lambda f.\lambda x.\ \overline{m}\ f\ (n\ f\ x))\ \overline{n} & \\
& \longrightarrow & \lambda f.\lambda x.\ \overline{m}\ f\ (\overline{n}\ f\ x) & \\
& \equiv & \lambda f.\lambda x.\ (\lambda f.\lambda x.\ f^m\ x)\ f\ (\overline{n}\ f\ x) & \\
& \longrightarrow & \lambda f.\lambda x.\ (\lambda x.\ f^m\ x)\ (\overline{n}\ f\ x) & \\
& \longrightarrow & \lambda f.\lambda x.\ f^m\ (\overline{n}\ f\ x) & \\
& \equiv & \lambda f.\lambda x.\ f^m\ ((\lambda f.\lambda x.\ f^n\ x)\ f\ x) & \\
& \longrightarrow & \lambda f.\lambda x.\ f^m\ ((\lambda x.\ f^n\ x)\ x) & \\
& \longrightarrow & \lambda f.\lambda x.\ f^m\ (f^n\ x) & \\
& \longrightarrow & \lambda f.\lambda x.\ f^{m+n}\ x & \equiv\ \overline{m+n}
\end{array}
$$

Beweisführung in Logik höherer Stufe

Extensionsverfahren ähnlich wie bei Prädikatenlogik

- **Erweiterte Matrixcharakterisierung der Gültigkeit**
  - **F ist gültig gdw. alle Pfade durch F komplementär**
  - Komplementarität mit erweitertem Substitutionsbegriff
    - Prädikats- und Funktionssymbole dürfen ersetzt werden
    - Konnektierte Terme müssen nur semantisch gleich sein
- **Erweitertes Beweissuchverfahren** → TPS (Andrews)
  - Konnektionen-orientiertes Pfadüberprüfungsverfahren
  - Komplementaritätstest mit Unifikation höherer Stufe
    - i.A. unentscheidbar (!) und erheblich komplizierteres Verfahren → Huet
- **Beweise sind i.A. sehr kurz und elegant**
  - Aber erheblich schwerer zu finden

Beweis der Grundeigenschaften der Gleichheit

$$\doteq \quad \equiv \quad \lambda x.\lambda y.\forall P(P(x)\Rightarrow P(y))$$

- **Reflexivität:** $a\doteq a$

$$\begin{bmatrix} Pa^T & Pa^F \end{bmatrix}$$

- **Kommutativität:** $a\doteq b \Rightarrow b\doteq a$

$$\begin{bmatrix} Xa^F & Pb^T & Pa^F \\ Xb^T & & \end{bmatrix} \qquad [\lambda z.\neg Pz/X]$$

- **Transitivität:** $a\doteq b \wedge b\doteq c \Rightarrow a\doteq c$

$$\begin{bmatrix} Xa^F & Yb^F & Pa^T & Pc^F \\ Xb^T & Yc^T & & \end{bmatrix} \qquad [P/X, P/Y]$$

- **Substitutivität:** $Pa \wedge a\doteq b \Rightarrow Pb$

$$\begin{bmatrix} Pa^T & Xa^F & Pb^F \\ & Xb^T & \end{bmatrix} \qquad [P/X]$$

Intuitionistische Logik höherer Stufe

- **Unterstützt formale Manipulation von Algorithmen**
  - Synthese aus Spezifikationen, Optimierung, Verifikation, ...
  - Einheitliche Sprache für Spezifikation, Programmierung, Deduktion...
- **(Zur Zeit noch) viele verschiedene Formulierungen**
  - Martin-Löf'sche Typentheorie (Computational Type Theory)
  - Kalkül der Konstruktionen
  - System F
  - LCF (Logik berechenbarer Funktionen)

    ⋮
- **Beweissysteme interaktiv mit taktischer Steuerung**
  - AUTOMATH (historischer Vorläufer)
  - Nuprl, MetaPRL (Computational Type Theory)
  - Alf / Agda, ... (Martin-Löf Typentheorie)
  - Lego, Coq (Kalkül der Konstruktionen)
  - Cambridge LCF
- **Viele erfolgreiche Anwendungen**