

Formale Methoden der Softwaretechnik
Vorlesung vom 30.05.11: Logik Höherer Stufe

Till Mossakowski & Christoph Lüth

Universität Bremen

Sommersemester 2011

Fahrplan

- ▶ Alles über Logik höherer Stufe (Higher-order Logic, HOL):
 - ▶ Typen und Terme
 - ▶ Die Basis-Axiome
 - ▶ Definierte Operatoren

Fahrplan

- ▶ Aussagenlogik
- ▶ Prädikatenlogik
- ▶ Isabelle I: Grundlagen
 - ▶ Aussagenlogik und natürliches Schließen
 - ▶ Prädikatenlogik und Quantoren
 - ▶ Logik höherer Stufe
 - ▶ Definitionen und konservative Erweiterung
 - ▶ Automatische Beweisprozeduren
- ▶ Isabelle II: Anwendungen

Logik höherer Stufe

- ▶ **Ziel:** Formalisierung von Mathematik
 - ▶ “Logik für Erwachsene”
- ▶ **Problem:** Mögliche Inkonsistenz (Russel’s Paradox)
- ▶ **Lösung:** Restriktion vs. Ausdruckstärke
- ▶ Alternative **Grundlagen:**
 - ▶ Andere Typtheorien (Martin-Löf, Calculus of Constructions)
 - ▶ Ungetypte Mengenlehre (ZFC)
- ▶ **HOL:** guter Kompromiss, weit verbreitet.
 - ▶ Klassische Logik höherer Stufe nach Church
 - ▶ Schwächer als ZFC, stärker als Typtheorien

Warum Logik höherer Stufe?

- ▶ Aussagenlogik: keine Quantoren
- ▶ Logik 1. Stufe: Quantoren über Terme

$$\forall x y. x = y \longrightarrow y = x$$

Warum Logik höherer Stufe?

- ▶ **Aussagenlogik**: keine Quantoren
- ▶ Logik **1. Stufe**: Quantoren über Terme

$$\forall x y. x = y \longrightarrow y = x$$

- ▶ Logik **2. Stufe**: Quantoren über **Prädikaten** und **Funktionen**

$$\forall P. (P 0 \wedge \forall x. P x \longrightarrow P (S x)) \longrightarrow \forall x. P x$$

Warum Logik höherer Stufe?

- ▶ **Aussagenlogik**: keine Quantoren
- ▶ Logik **1. Stufe**: Quantoren über Terme

$$\forall x y. x = y \longrightarrow y = x$$

- ▶ Logik **2. Stufe**: Quantoren über **Prädikaten** und **Funktionen**

$$\forall P. (P 0 \wedge \forall x. P x \longrightarrow P (S x)) \longrightarrow \forall x. P x$$

- ▶ Logik **3. Stufe**: Quantoren über Argumenten von Prädikaten

Warum Logik höherer Stufe?

- ▶ **Aussagenlogik**: keine Quantoren
- ▶ Logik **1. Stufe**: Quantoren über Terme

$$\forall x y. x = y \longrightarrow y = x$$

- ▶ Logik **2. Stufe**: Quantoren über **Prädikaten** und **Funktionen**

$$\forall P. (P 0 \wedge \forall x. P x \longrightarrow P (S x)) \longrightarrow \forall x. P x$$

- ▶ Logik **3. Stufe**: Quantoren über Argumenten von Prädikaten
- ▶ **Logik höherer Stufe (HOL)**: alle endlichen Quantoren
 - ▶ Keine **wesentlichen Vorteile** von Logik 2. Ordnung

Warum Logik höherer Stufe?

- ▶ **Aussagenlogik**: keine Quantoren
- ▶ Logik **1. Stufe**: Quantoren über Terme

$$\forall x y. x = y \longrightarrow y = x$$

- ▶ Logik **2. Stufe**: Quantoren über **Prädikaten** und **Funktionen**

$$\forall P. (P 0 \wedge \forall x. P x \longrightarrow P (S x)) \longrightarrow \forall x. P x$$

- ▶ Logik **3. Stufe**: Quantoren über Argumenten von Prädikaten
- ▶ **Logik höherer Stufe (HOL)**: alle endlichen Quantoren
 - ▶ Keine **wesentlichen Vorteile** von Logik 2. Ordnung

Vermeidung von Inkonsistenzen

- ▶ Russell's Paradox

- ▶ $R = \{X \mid X \notin X\}$

- ▶ Abhilfe: Typen

- ▶ Gödel's 2. Unvollständigkeitssatz:

- ▶ Jede Logik, die ihre eigene Konsistenz beweist, ist inkonsistent.

- ▶ Unterscheidung zwischen Termen und Aussagen

- ▶ Dadurch in HOL keine Aussage über HOL

Typen

- ▶ Typen $Type$ gegeben durch
 - ▶ **Typkonstanten**: $c \in \mathcal{C}_{Type}$ (Menge $\mathcal{C}_{Type}$ durch Signatur gegeben)
 - ▶ $Prop, Bool \in \mathcal{C}_{Type}$: $Prop$ alle Terme, $Bool$ alle Aussagen
 - ▶ **Typvariablen**: $\alpha \in \mathcal{V}_{Type}$ (Menge $\mathcal{V}_{Type}$ fest)
 - ▶ **Funktionen**: $s, t \in Type$ dann $s \Rightarrow t$ in $Type$
- ▶ **Konvention**: Funktionsraum nach rechts geklammert
 $\alpha \Rightarrow \beta \Rightarrow \gamma$ für $\alpha \Rightarrow (\beta \Rightarrow \gamma)$

Terme

- ▶ Terme *Term* gegeben durch
 - ▶ **Konstanten**: $c \in \mathcal{C}$ (Menge $\mathcal{C}$ durch Signatur gegeben)
 - ▶ **Variablen**: $v \in \mathcal{V}$
 - ▶ **Applikation**: $s, t \in \text{Term}$ dann $s\ t \in \text{Term}$
 - ▶ **Abstraktion**: $x \in \mathcal{V}, t \in \text{Term}$ dann $\lambda x. t \in \text{Term}$
- ▶ Konventionen: **Applikation** links geklammert, mehrfache **Abstraktion**
 $\lambda x\ y\ z. f\ x\ y\ z$ für $\lambda x. \lambda y. \lambda z. ((f\ x)\ y)\ z$

Basis-Syntax

$= \quad :: \alpha \Rightarrow \alpha \Rightarrow Bool$
 $\longrightarrow \quad :: Bool \Rightarrow Bool \Rightarrow Bool$
 $\iota \quad :: (\alpha \Rightarrow Bool) \Rightarrow \alpha$

$\neg \quad :: Bool \Rightarrow Bool$

true $:: Bool$

false $:: Bool$

if $:: Bool \Rightarrow \alpha \Rightarrow \alpha \Rightarrow \alpha$

$\forall \quad :: (\alpha \Rightarrow Bool) \Rightarrow Bool$

$\exists \quad :: (\alpha \Rightarrow Bool) \Rightarrow Bool$

$\wedge \quad :: Bool \Rightarrow Bool \Rightarrow Bool$

$\vee \quad :: Bool \Rightarrow Bool \Rightarrow Bool$

► Einbettung (wird weggelassen)
trueprop $:: Bool \Rightarrow Prop$

► **Basis-Operatoren:** $=, \longrightarrow, \iota$

► **Syntaktische Konventionen:**

► **Bindende Operatoren:** $\forall, \exists, \iota$
 $\forall x. P \equiv \forall (\lambda x. P)$

► **Infix-Operatoren:** $\wedge, \vee, \longrightarrow, =$

► **Mixfix-Operator:**
if b then p else q $\equiv$ *if b p q*

Basis-Axiome I: Gleichheit

- Reflexivität:

$$\overline{t = t} \text{ refl}$$

- Substitutivität:

$$\frac{s = t \quad P(s)}{P(t)} \text{ subst}$$

- Extensionalität:

$$\frac{\forall x. fx = gx}{(\lambda x. fx) = (\lambda x. gx)} \text{ ext}$$

- Einführungsregel:

$$\overline{(P \longrightarrow Q) \longrightarrow (Q \longrightarrow P) \longrightarrow (P = Q)} \text{ iff}$$

Basis-Axiome II: Implikation und Auswahl

- ▶ Einführungsregel **Implikation**:

$$\frac{\begin{array}{c} [P] \\ \vdots \\ Q \end{array}}{P \longrightarrow Q} \text{ impl}$$

- ▶ Eliminationsregel **Implikation**:

$$\frac{P \longrightarrow Q \quad P}{Q} \text{ mp}$$

- ▶ Eliminationsregel
Auswahloperator:

$$\frac{}{(\lambda x. x = a) = a} \text{ the_eq}$$

- ▶ HOL ist **klassisch**:

$$\frac{}{(P = \text{true}) \vee (P = \text{false})} \text{ true_or_fal}$$

Die Basis-Axiome (Isabelle-Syntax)

refl : $t = t$

subst : $\llbracket s = t; P(s) \rrbracket \Longrightarrow P(t)$

ext : $\llbracket \bigwedge x. fx = gx \rrbracket \Longrightarrow (\lambda x. fx) = (\lambda x. gx)$

impl : $\llbracket P \Longrightarrow Q \rrbracket \Longrightarrow P \longrightarrow Q$

mp : $\llbracket P \longrightarrow Q; P \rrbracket \Longrightarrow Q$

iff : $(P \longrightarrow Q) \longrightarrow (Q \longrightarrow P) \longrightarrow (P = Q)$

the_eq : $(\iota x. x = a) = a$

true_or_false : $(P = true) \vee (P = false)$

Abgeleitete Operatoren

$$\text{true} \equiv (\lambda x.x) = (\lambda x.x)$$

$$\forall P \equiv (P = \lambda x.\text{true})$$

$$\exists P \equiv \forall Q.(\forall x.Px \longrightarrow Q) \longrightarrow Q$$

$$\text{false} \equiv \forall P.P$$

$$\neg P \equiv P \longrightarrow \text{false}$$

$$P \wedge Q \equiv \forall R.(P \longrightarrow Q \longrightarrow R) \longrightarrow R$$

$$P \vee Q \equiv \forall R.(P \longrightarrow R) \longrightarrow (Q \longrightarrow R) \longrightarrow R$$

$$\text{if } P \text{ then } x \text{ else } y \equiv \iota z.(P = \text{true} \longrightarrow z = x) \wedge (P = \text{false} \longrightarrow z = y)$$

Erweiterungen

- ▶ Weitere Operatoren
- ▶ Weitere Typen: natürliche Zahlen, Datentypen
- ▶ Axiomatisch (vgl. Peano/Presburger in FOL)
 - ▶ Mögliche Inkonsistenzen
- ▶ Konservative Erweiterung
 - ▶ Logik konsistentzerhaltend erweitern

Zusammenfassung

Logik **höherer Stufe** (HOL):

- ▶ Syntax basiert auf dem **einfach getypten λ -Kalkül**
- ▶ **Drei** Basis-Operatoren, **acht** Basis-Axiome
- ▶ **Rest** folgt durch **konservative Erweiterung** — nächstes Mal