

Formale Methoden der Softwaretechnik 1
Vorlesung vom 16.11.09:
Grundlage von Isabelle

Christoph Lüth, Lutz Schröder
Universität Bremen
Wintersemester 2009/10

1

Fahrplan

- ▶ Teil I: Grundlagen der Formalen Logik
- ▶ Teil II: Arbeiten mit Isabelle
 - ▶ Grundlagen von Isabelle
 - ▶ Logik höherer Ordnung
 - ▶ Isabelle/HOL
 - ▶ Beweise über funktionale Programme in Isabelle/HOL
 - ▶ Beweisen mit Isabelle: Simplifikation, Taktiken
- ▶ Teil III: Modellierung imperative Programme

2

Fahrplan

- ▶ Grundlagen:
 - ▶ der getypte λ -Kalkül
 - ▶ Unifikation und Resolution
- ▶ Beweisen in Isabelle

3

Der λ -Kalkül

- ▶ In den 30'ern von Alonzo Church als theoretisches Berechnungsmodell erfunden.
- ▶ In den 60'ern Basis von Lisp (John McCarthy)
- ▶ Mathematische Basis von funktionalen Sprachen (Haskell etc.)
- ▶ Hier: Grundlage der Syntax ("higher-order abstract syntax")
 - ▶ Typisierung
 - ▶ Gebundene Variablen
 - ▶ Warum?

4

Der polymorph getypte λ -Kalkül

- ▶ Typen *Type* gegeben durch
 - ▶ Typkonstanten: $c \in \mathcal{C}_{Type}$
 - ▶ Typvariablen: $\alpha \in \mathcal{V}_{Type}$ (Menge \mathcal{V}_{Type} gegeben)
 - ▶ Funktionen: $s, t \in Type$ dann $s \Rightarrow t$ in *Type*
- ▶ Terme *Term* gegeben durch
 - ▶ Konstanten: $c \in \mathcal{C}$
 - ▶ Variablen: $v \in \mathcal{V}$
 - ▶ Applikation: $s, t \in Term$ dann $s t \in Term$
 - ▶ Abstraktion: $x \in \mathcal{V}, \tau \in Type, t \in Term$ dann $\lambda x^\tau. t \in Term$
 - ▶ Typ τ kann manchmal berechnet werden.
- ▶ Signatur enthält $\mathcal{C}_{Type}, \mathcal{C}$

5

Typisierung

- ▶ Signatur: $\Sigma = (\mathcal{C}_{Type}, \mathcal{C}, ar), ar : \mathcal{C} \rightarrow Type$
 - ▶ Typisierung der Konstanten
 - ▶ Notation: $\Sigma = \{c_1 : \tau_1, \dots, c_n : \tau_n\}$ für $ar(c_i) = \tau_i$
- ▶ Term t hat Typ τ in einem Kontext Γ und einer Signatur Σ :

$$\Gamma \vdash_{\Sigma} t : \tau$$

- ▶ Kontext: $x_1 : \tau_1, \dots, x_n : \tau_n$ ($x_i \in \mathcal{V}$)
 - ▶ Typisierung von Variablen
- ▶ Vergleiche Haskell etc.

6

Typisierung

- ▶ Die Regeln:

$$\frac{c : \tau \in \Sigma}{\Gamma \vdash_{\Sigma} c : \tau} \text{CONST}$$

$$\frac{x : \tau \in \Gamma}{\Gamma \vdash_{\Sigma} x : \tau} \text{VAR}$$

$$\frac{\Gamma \vdash_{\Sigma} s : \sigma \Rightarrow \tau \quad \Gamma \vdash_{\Sigma} t : \sigma}{\Gamma \vdash_{\Sigma} s t : \tau} \text{APP}$$

$$\frac{\Gamma, x : \sigma \vdash_{\Sigma} t : \tau}{\Gamma \vdash_{\Sigma} \lambda x^\sigma. t : \sigma \Rightarrow \tau} \text{ABST}$$

7

Einbettungen

- ▶ Beispiel: Einbettung der Prädikatenlogik mit PA
- ▶ Basistypen:

$$\mathcal{C}_{Type} = \{i, o\}$$

- ▶ Konstanten:

$$\Sigma_T = \{0 : i, s : i \Rightarrow i, plus : i \Rightarrow i \Rightarrow i\}$$

$$\Sigma_A = \{eq : i \Rightarrow i \Rightarrow o, false : o, and : o \Rightarrow o \Rightarrow o, \dots, all : (i \Rightarrow o) \Rightarrow o, ex : (i \Rightarrow o) \Rightarrow o\}$$

$$\Sigma = \Sigma_T \cup \Sigma_A$$

- ▶ $\phi \in Form \iff \vdash_{\Sigma} t : o$
- ▶ Beispiel: $\forall x. \exists y. (x = y) \iff all (\lambda x. ex (\lambda y. (eq x) y))$

8

Substitution

- ▶ $s \left[\frac{t}{x} \right]$ ist Ersetzung von x durch t in s
- ▶ Definiert durch strukturelle Induktion:

$$\begin{aligned}
 c \left[\frac{t}{x} \right] &\stackrel{\text{def}}{=} c \\
 y \left[\frac{t}{x} \right] &\stackrel{\text{def}}{=} \begin{cases} t & x = y \\ y & x \neq y \end{cases} \\
 (r \ s) \left[\frac{t}{x} \right] &\stackrel{\text{def}}{=} r \left[\frac{t}{x} \right] s \left[\frac{t}{x} \right] \\
 (\lambda y. s) \left[\frac{t}{x} \right] &\stackrel{\text{def}}{=} \begin{cases} \lambda y. s & x = y \\ \lambda y. s \left[\frac{t}{x} \right] & x \neq y, y \notin FV(t) \\ \lambda z. (s \left[\frac{z}{y} \right]) \left[\frac{t}{x} \right] & x \neq y, y \in FV(t) \\ & \text{mit } z \notin FV(t) \text{ (z frisch)} \end{cases}
 \end{aligned}$$

9

Reduktion und Äquivalenzen

- ▶ β -Reduktion: $(\lambda x. s) t \rightarrow_{\beta} s \left[\frac{t}{x} \right]$
- ▶ β -Äquivalenz: $=_{\beta} \stackrel{\text{def}}{=} (\rightarrow_{\beta}^* \cup \leftarrow^*)^*$
 - ▶ $s =_{\beta} t$ iff. $\exists u. s \rightarrow_{\beta}^* u, t \rightarrow_{\beta}^* u$ (Church-Rosser)
- ▶ α -Äquivalenz: $\lambda x. t =_{\alpha} \lambda y. t \left[\frac{y}{x} \right], y \notin FV(t)$
 - ▶ Name von gebundenen Variablen unerheblich
 - ▶ In Isabelle Implizit (deBruijn-Indizes)
- ▶ η -Äquivalenz: $\lambda x. tx =_{\eta} t, x \notin FV(t)$
 - ▶ "punktfreie" Notation

10

Unifikation

- ▶ Für $s, t \in \text{Term}$ oder $s, t \in \text{Type}$ ist Substitution σ **Unifikator** wenn $s\sigma = t\sigma$
 - ▶ **Allgemeinster Unifikator** τ : alle anderen Unifikatoren sind Instanzen von τ
 - ▶ Allgemeinster Unifikator **eindeutig**, wenn er existiert
- ▶ Unifikationsalgorithmus:

$$\begin{aligned}
 \tau(f \ t, g \ s) &= \text{false} \\
 \tau(f \ t, f \ s) &= \tau(t, s) \\
 \tau(t, x) &= \left[\frac{t}{x} \right] \quad x \notin FVt \\
 \tau(y, s) &= \left[\frac{y}{s} \right] \quad y \notin FVs
 \end{aligned}$$

- ▶ **Matching**: einseitige Unifikation (σ mit $s\sigma = t$)

11

Grundlagen von Isabelle

- ▶ Grundlage: **getypter Λ -Kalkül**
- ▶ Unifikation und Matching (apply (rule r))
- ▶ **Äquivalenzen**:
 - ▶ β -Reduktion automatisch
 - ▶ α -Äquivalenz eingebaut (deBruijn-Indexing)
 - ▶ η -Äquivalenz automatisch
- ▶ Variablen, Formeln, Resolution

12

Variablen

- ▶ **Meta-Variablen**: können unifiziert und beliebig instanziiert werden
- ▶ **freie Variablen** (fixed): beliebig aber fest
- ▶ **Gebundene Variablen**: Name beliebig (α -Äquivalenz)
- ▶ **Meta-Quantoren**: Isabelles Eigenvariablen

$$\frac{\bigwedge x. P(x)}{\forall x. P(x)} \text{ all} \quad !!x. P \ x \implies \text{ALL } x. P \ x$$

- ▶ Beliebig instanziiierbar
- ▶ Gültigkeit auf diese (Teil)-Formel begrenzt

13

Formeln

- ▶ **Formeln** in Isabelle:

$$\frac{\phi_1, \dots, \phi_n}{\psi} \quad \llbracket \phi_1, \dots, \phi_n \rrbracket \implies \psi$$

- ▶ ϕ_1, \dots, ϕ_n Formeln, ψ atomar
- ▶ **Theoreme**: ableitbare Formeln
- ▶ **Ableitung** von Formeln: Resolution, Instantiierung, Gleichheit
- ▶ **Randbemerkung**:
 - ▶ $\implies, \bigwedge, \equiv$ formen Meta-Logik
 - ▶ Einbettung **anderer** Logiken als HOL möglich — generischer Theorembeweiser

14

Resolution

- ▶ Einfache Resolution (rule)
 - ▶ **Achtung**: **Lifting** von Meta-Quantoren und Bedingungen
 - ▶ Randbemerkung: Unifikation höherer Stufe **unentscheidbar**
- ▶ Eliminationsresolution (erule)
- ▶ Destruktionsresolution (drule)

15

Rückwärtsbeweis

- ▶ Ausgehend von **Beweisziel** ψ
- ▶ **Beweiszustand** ist $\llbracket \phi_1, \dots, \phi_n \rrbracket \implies \psi$
- ▶ ϕ_1, \dots, ϕ_n : **subgoals**
- ▶ **Beweisverfahren**: Resolution, Termersetzung, Beweissuche

16

Zusammenfassung

- ▶ Getypter λ -Kalkül als Grundlage der Metalogik, Modellierung von Logiken durch **Einbettung**
- ▶ Isabelle: **Korrektheit** durch **Systemarchitektur**
- ▶ **Beweis**: Rückwärts, Resolution
 - ▶ Anmerkung: auch **Vorwärtsbeweis** möglich
- ▶ Donnerstag: Logik **höherer Ordnung** in Isabelle