Praktische Informatik 3: Funktionale Programmierung
Vorlesung 9 vom 13.12.2016: Spezifikation und Beweis

Christoph Lüth

Universität Bremen

Wintersemester 2016/17

Fahrplan

- Teil I: Funktionale Programmierung im Kleinen
- Teil II: Funktionale Programmierung im Großen
  - Abstrakte Datentypen
  - Signaturen und Eigenschaften
  - Spezifikation und Beweis
- Teil III: Funktionale Programmierung im richtigen Leben

Formaler Beweis

- Warum?
  - Formaler Beweis zeigt Korrektheit
- Wie?
  - Formale Notation
  - Beweisformen (Schließregeln)
- Wozu?
  - Formaler Beweis zur Analyse des Algorithmus
  - Haskell als Modellierungssprache

Eigenschaften

- Prädikate:
  - Haskell-Ausdrücke vom Typ Bool
  - Quantifizierte Aussagen:
    Wenn P :: $\alpha \to$ Bool, dann ist ALL x. P x :: Bool ein Prädikat und
    EX x. P x :: Bool ein Prädikat
- Sonderfall Gleichungen `s == t` und transitive Relationen
- Prädikate müssen nicht ausführbar sein

Wie beweisen?

- Beweis $\longleftrightarrow$ Programmdefinition

Gleichungsumformung	Funktionsdefinition
Fallunterscheidung	Fallunterscheidung (Guards)
Induktion	Rekursive Funktionsdefinition

- Wichtig: formale Notation

Notation

Allgemeine Form:

Lemma (1) P
$\Leftrightarrow$ P_1 — Begründung
$\Leftrightarrow$ P_2 — Begründung
$\Leftrightarrow$ ...
$\Leftrightarrow$ True
□

Sonderfall Gleichungen:

Lemma (2) a= b

```
  a
= x1   — Begründung
= x2   — Begründung
= ...
= b
```

□

Beweis durch vollständige Induktion

Zu zeigen:

Für alle natürlichen Zahlen x gilt $P(x)$.

Beweis:

- Induktionsbasis: $P(0)$
- Induktionsschritt:
  - Induktionsvoraussetzung $P(x)$
  - zu zeigen $P(x+1)$

Beweis durch strukturelle Induktion (Listen)

Zu zeigen:

Für alle endlichen Listen xs gilt P xs

Beweis:

- Induktionsbasis: P []
- Induktionsschritt:
  - Induktionsvoraussetzung P xs
  - zu zeigen P (x:xs)

Beweis durch strukturelle Induktion (Allgemein)

Zu zeigen:

Für alle x in T gilt $P(x)$

Beweis:

- Für jeden Konstruktor C_i:
  - Voraussetzung: für alle $t_{i,j}$ gilt $P(t_{i,j})$
  - zu zeigen $P(C_i\ t_{i,1}\ \ldots\ t_{i,k_i})$

Beweisstrategien

- Fold-Unfold:
  - Im Induktionsschritt Funktionsdefinition auffalten
  - Ausdruck umformen, bis Induktionsvoraussetzung anwendbar
  - Funktionsdefinitionen zusammenfalten
- Induktionsvoraussetzung stärken:
  - Stärkere Behautung $\Longrightarrow$ stärkere Induktionsvoraussetzung, daher:
  - um Behauptung P zu zeigen, stärkere Behauptung P' zeigen, dann P als Korollar

Zusammenfassung

- Beweise beruhen auf:
  - Gleichungs- und Äquivalenzumformung
  - Fallunterscheidung
  - Induktion
- Beweisstrategien:
  - Sinnvolle Lemmata
  - Fold/Unfold
  - Induktionsvoraussetzung stärken
- Warum Beweisen?
  - Korrektheit von Haskell-Programmen
  - Haskell als Modellierungssprache