Formale Modellierung
Vorlesung 10 vom 10.06.13: Modellierung von Programmen

Serge Autexier & Christoph Lüth

Universität Bremen

Sommersemester 2013

Fahrplan

- Teil I: Formale Logik
- Teil II: Spezifikation und Verifikation
  - Modellierung von Programmen
  - Die Z-Notation
  - Formale Modellierung mit der UML und OCL
- Teil III: Schluß

Das Tagesmenü

- Bis jetzt:
  1. Grundlagen — formale Logik
  2. Statische Datentypen
- Heute: Programme
  1. Funktional,
  2. imperativ,
  3. objektorientiert.

Funktionale Programme

- Wir haben:
  1. Logik höherer Stufe
  2. terminierende rekursive Funktionen
  3. induktive (generierte) Datentypen
- Funktionale Programme:
  1. Funktionen höherer Ordnung
  2. Beliebige Rekursion
  3. Beliebige Datentypen

Modellierung beliebiger Rekursion

Definition 1 (Partielle Ordnung)

$(X, \sqsubseteq)$ mit $\sqsubseteq$ antisymmetrisch, reflexiv und transitiv.

Definition 2 (CPO)

$(X, \sqsubseteq)$ mit $\sqsubseteq$ partielle Ordnung und jede Kette C

$$x_1 \sqsubseteq x_2 \sqsubseteq \ldots x_n \sqsubseteq \ldots$$

hat kleinste obere Schranke $\bigsqcup_{i<\omega} x_i$

Definition 3 (Stetige Funktion)

Funktion $f : (X, \sqsubseteq) \to (Y, \preccurlyeq)$ ist Scott-stetig, wenn f die Ordnung $\sqsubseteq$ und $\bigsqcup$ bewahrt.

Eigenschaften von CPOs

Theorem 4 (Kleene-Tarski-Fixpunktsatz)

Sei $f : (X, \sqsubseteq) \to (X, \sqsubseteq)$ Scott-stetig und $\bot \in X$ kleinstes Element, dann hat f einen kleinsten Fixpunkt lfp_f mit $f(lfp_f) = lfp_f$.

- CPOs und stetige Funktionen modellieren funktionale Programme (Logic of computable functions, LCF)
- CPOs modellieren Domänen (Datentypen für funktionale Programme)
- Möglich: konservative Einbettung in HOL (HOLCF)

Ein Beispiel

Beispiel: Flugbuchungssystem

- Ein Flug hat eine eindeutige Flugnummer, Start und Ziel, sowie die Anzahl Sitze (verfügbar und insgesamt).
- Das Flugbuchungssystem identifiziert Flüge anhand ihrer Flugnummer.
- Das Flugbuchungsystem soll folgende Operationen unterstützen:
  - Anfrage nach Verbindung (Start und Ziel) sowie Anzahl Plätze;
  - Buchung mit Flugnummer, Anzahl Plätze

- Modellierung funktional (Isabelle/HOL): axiomatisch vs. konservativ

- Modellierung imperativ: Zustandsübergang

Die Z Notation

- Basiert auf getypter Mengenlehre
- Entwickelt seit 1980 (Jean-Claude Abrial, Oxford PRG)
- Industriell genutzt (IBM, Altran Praxis (vorm. Praxis Critical Systems))
- LaTeX-Notation und Werkzeugunterstützung (Community Z Tools, HOL-Z, ProofPower)

Modellierung in Z (1)

$[FLIGHTID, STRING]$

Flight

$$
\begin{array}{l}
flightid : FLIGHTID \\
dept : STRING \\
arr : STRING \\
avail : \mathbb{N} \\
total : \mathbb{N}
\end{array}
$$

FlightDB

$$
\begin{array}{l}
flights : \mathbb{P}\, FLIGHTID \\
data : FLIGHTID \pfun Flight \\
\hline
flights = \mathrm{dom}\, data \\
\forall\, i : FLIGHTID \bullet (data(i)).flightid = i
\end{array}
$$

Modellierung in Z (2)

Lookup

$$
\begin{array}{l}
\mathit{FlightDB} \\
\mathit{flid}? : \mathit{FLIGHTID} \\
\mathit{flight}! : \mathit{Flight} \\
\hline
\mathit{flid}? \in \mathit{flights} \\
\mathit{flight}! = \mathit{data}(\mathit{flid}?)
\end{array}
$$

Modellierung in Z (3)

SuccessfulQuery

$$
\begin{array}{l}
\Xi\, FlightDB \\
from? : STRING \\
to? : STRING \\
seats? : \mathbb{N} \\
fid! : FLIGHTID \\
\hline
fid! \in flights \\
(data(fid!)).avail \geq seats? \\
(data(fid!)).dept = from? \\
(data(fid!)).arr = to?
\end{array}
$$

Modellierung in Z (4)

FailedQuery

$FlightDB$

$from? : STRING$

$to? : STRING$

$seats? : \mathbb{N}$

$fid! : FLIGHTID$

$fid! \notin flights$

Zusammenfassung

- Modellierung funktionaler Programme: cpos (LCF) für beliebige Rekursion
  - Datenmodellierung: erzeugte Datentypen
- Modellierung imperativer Programme: Zustandsübergang
  - Datenmodellierung: als Mengen (Z) oder als Klassen (UML/OCL)
- Nächste Woche: Z im Gesamtüberblick