Programmierung und Modellierung mit Haskell

Denotationelle Semantik

Martin Hofmann Steffen Jost

LFE Theoretische Informatik, Institut für Informatik,
Ludwig-Maximilians Universität, München

6. Juli 2015

Denotationelle Semantik

Die Vorlesungen am 6. und 13. Juli 2015 folgte dem Kapitel über denotationelle Semantik aus dem Wikibook über Haskell:
http://en.wikibooks.org/wiki/Haskell/Denotational_semantics

Behandelt wurde der Text von Beginn an bis zum Unterabschnitt "Convergence" in der ersten Vorlesung und in der zweiten Vorlesung bis inklusive Unterabschnitt "Recursive Data Types and Infinite Lists".

Auf den folgenden Folien folgt ein Transskript des Tafelanschriebs.

Hinweis:
In den folgenden mathematischen Definition lesen wir : wie :: in Haskell.

Tafelanschrieb 06.07.2015

```
shaves :: Integer -> Integer -> Bool
1 `shaves` 1 = True
2 `shaves` 2 = False
0 `shaves` x = not (x `shaves` x)
_ `shaves` _ = False

g :: (Integer -> Integer) -> (Integer -> Integer)
g x = \n -> if n == 0 then 1 else n * x (n-1)

x0 :: Integer -> Integer
x0 = undefined

(f0:f1:f2:f3:f4:fs) = iterate g x0

fix :: (a -> a) -> a
fix f = let x = f x in x
factorial = fix g
```

Partiell geordnete Menge

Definition (poset)

Eine **partiell geordnete Menge** (engl. partially ordered set, oder kurz **poset**), geschrieben $(M, \preceq)$, besteht aus einer Menge M und einer partiellen Ordnungsrelation darauf $\preceq \subseteq M \times M$.

In einer partiell geordneten Menge kann es also Elemente geben, welche wir *nicht* miteinander vergleichen können.

Definition (partielle Ordnung)

Eine **partielle Ordnungsrelation** $\preceq$ ist eine binäre Relation, welche folgendende Eigenschaften hat:

Reflexiv $\forall x \,.\, x \preceq x$

Antisymmetrisch $\forall x, y \,.\, x \preceq y$ und $y \preceq x$ impliziert $x = y$

Transitiv $\forall x, y, z \,.\, x \preceq y$ und $y \preceq z$ impliziert $x \preceq z$

Hasse Diagramm

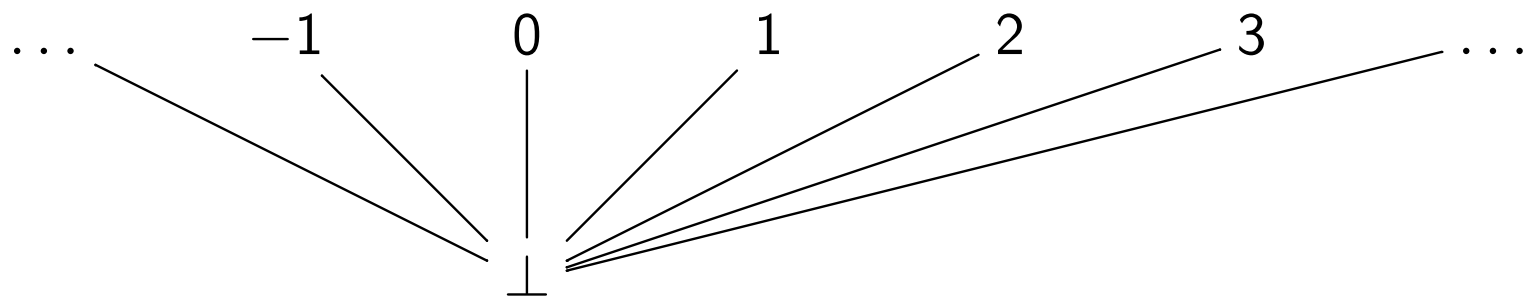
Mit einem **Hasse Diagramm** kann man partiell geordnete Mengen anschaulich darstellen: Zwei Elemente sind genau dann miteinander vergleichbar, wenn man im Diagramm über ein oder mehrere Linien von einem Element zu anderen gelangen kann, ohne dabei die Richtung zu wechseln. Das obere Element ist dabei das größere.

Beispiel: Links ist das Hasse Diagramm des posets $(\mathbb{Z}, \leq)$ (natürliche Ordnung auf den ganzen Zahlen).

$0 < 2$ und im Diagramm erreichen wir von der 0 ausgehend die 2 durch zweimaliges aufsteigen entlang der Linie.

Hasse Diagramm

Hasse Diagramm des posets $(\mathbb{Z} \cup \{\bot\}, \sqsubseteq)$



Informelle Interpretation:

Jede ganze Zahl ist "gleich-gut" bezüglich $\sqsubseteq$, in dem Sinne, dass jede Zahl vollständig definiert ist. Eine nicht-terminierende oder fehlerhafte Berechnung, also $\bot$, ist dagegen völlig undefiniert und damit gilt $\bot \sqsubseteq z$ für alle Zahlen $z \in \mathbb{Z}$.

Monotone Funktionen

Satz

Sei $h : \mathbb{Z} \to \mathbb{Z}$ monoton und $h(\bot) = 1$. Dann gilt schon $h(n) = 1$ für alle $n \in \mathbb{Z}$.

Beweis: Sei $n \in \mathbb{Z}$ vorgegeben. Da $\bot \sqsubseteq n$ folgt mit der Monotonie, dass $h(\bot) \sqsubseteq h(n)$, also $1 \sqsubseteq h(n)$, also $h(n) = 1$. □

Beachte:
$\sqsubseteq$ bezeichnet die semantische Approximationsordnung auf $\mathbb{Z}$.

Folgerung

Jede nicht-konstante Funktion $h : \mathbb{Z} \to \mathbb{Z}$ hat die Eigenschaft, dass $h(\bot) = \bot$ ist.

Monotone Funktionen

Satz

Seien $(A, \sqsubseteq)$ und $(B, \sqsubseteq)$ partielle Ordnungen (posets), dann sind die monotonen Funktionen von A nach B, also die Menge $\{f : A \to B \mid \forall x, y \in A \,.\, x \sqsubseteq y \implies f(x) \sqsubseteq f(y)\}$ mit der punktweisen Ordnung $f \sqsubseteq g \iff \forall x \in A \,.\, f(x) \sqsubseteq g(x)$, eine partiell geordnete Menge.

Beweis: Reflexiv: $f \sqsubseteq f$ bedeutet: $\forall x \in A \,.\, f(x) \sqsubseteq f(x)$ i.O.
Anti-Symmetrisch: $f \sqsubseteq g$, $g \sqsubseteq f$. Für ein beliebiges $x \in A$ folgt dann $f(x) \sqsubseteq g(x)$ und $g(x) \sqsubseteq f(x)$ nach Definition. Da $\sqsubseteq$ auf B anti-Symmetrisch ist, folgt $f(x) = g(x)$.
Transitiv: $f \sqsubseteq g$, $g \sqsubseteq h$. Sei $x \in A \,.\, f(x) \sqsubseteq g(x)$ und $g(x) \sqsubseteq h(x)$, also $f(x) \sqsubseteq h(x)$. Also $f \sqsubseteq h$.

Monotone Funktionen

Rechnen mit Monotonie:

Falls g monoton ist, so kann man aus $u \sqsubseteq v$ auf $g(u) \sqsubseteq g(v)$ schliessen ("bottom up mode").
Umgekehrt, kann man, falls $g(u) \sqsubseteq g(v)$ zu zeigen ist, stattdessen $u \sqsubseteq v$ zu zeigen versuchen ("top down mode").

Programmierung und Modellierung mit Haskell

Denotationelle Semantik

Martin Hofmann Steffen Jost

LFE Theoretische Informatik, Institut für Informatik,
Ludwig-Maximilians Universität, München

13. Juli 2015

Suprema und dcpo

Sei $(X, \sqsubseteq)$ ein poset und $x_i \in X$. Eine Folge $(x_i)_{i \in \mathbb{N}}$, welche monton aufsteigend ist, also $x_0 \sqsubseteq x_1 \sqsubseteq x_2 \sqsubseteq \cdots$, nennen wir **Kette**.

Definition (Supremum)

Falls für einen Wert $x \in X$ und einer Kette $(x_i)_{i \in \mathbb{N}}$ gilt:

1. Für alle $i \in \mathbb{N}$ gilt $x_i \sqsubseteq x$
2. Für alle $y \in X$, welche ebenfalls für alle $i \in \mathbb{N}$ die Eigenschaft $x_i \sqsubseteq y$ haben, muss auch $x \subseteq y$ gelten

So nennen wir x das **Supremum** der Kette, geschrieben $\sup_{i \in \mathbb{N}} x_i$.

Ein Supremum ist die *kleinste obere Schranke*: es ist das kleinste Element, welches größer als jedes Element der Folge ist.

Suprema sind eindeutig bestimmt, falls sie existieren. Ein poset in dem jede Kette ein Supremum hat, heisst **dcpo**.

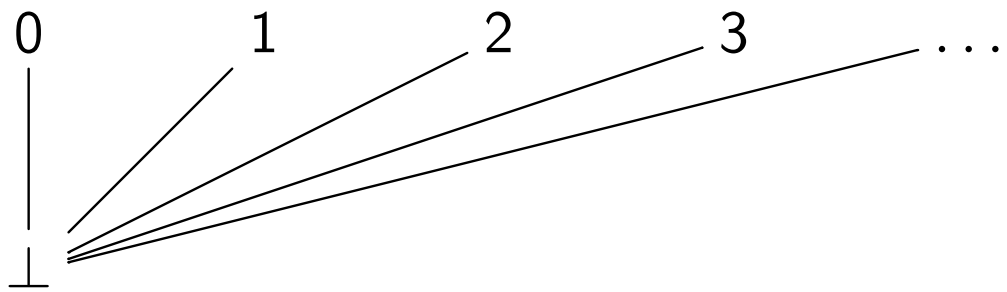
vollständige Halbordnung, engl. directed-complete partial order

Beispiel 1: dcpo

$\mathbb{N}_\perp$ ist eine dcpo:

Mögliche Kette	Supremum
$17, 17, 17, 17, \ldots$	17
$\perp, 17, 17, 17, 17, \ldots$	17
$\perp, \perp, \perp, 37, 37, 37, \ldots$	37
$\perp, \perp, \perp, \perp, \perp, \perp, \perp, \ldots$	$\perp$

Hasse Diagramm von $\mathbb{N}_\perp$ zur Erinnerung:



Beispiel 2: dcpo

Funktionen $\mathbb{N}_\bot \to \mathbb{N}_\bot$ bilden auch eine dcpo:
$\sup\big(f_0 \sqsubseteq f_1 \sqsubseteq f_2 \sqsubseteq \cdots\big) = \backslash n \to sub\big(f_0(n) \sqsubseteq f_1(n) \sqsubseteq f_2(n) \sqsubseteq \cdots\big)$

Beispiel Element dieser dcpo

	$\bot$	0	1	2	3	4	5	6	7	8	$\cdots$
f_0	$\bot$	$\bot$	$\bot$	$\bot$	$\bot$	$\bot$	$\bot$	$\bot$	$\bot$	$\bot$	$\cdots$
f_1	$\bot$	10	$\bot$	$\bot$	$\bot$	$\bot$	$\bot$	$\bot$	$\bot$	$\bot$	$\cdots$
f_2	$\bot$	10	$\bot$	12	$\bot$	$\bot$	$\bot$	$\bot$	$\bot$	$\bot$	$\cdots$
f_3	$\bot$	10	$\bot$	12	$\bot$	14	$\bot$	$\bot$	$\bot$	$\bot$	$\cdots$
f	$\bot$	10	$\bot$	12	$\bot$	14	$\bot$	16	$\bot$	18	$\cdots$

Das Suprema $f := \sup\big(f_0 \sqsubseteq f_1 \sqsubseteq f_2 \sqsubseteq \cdots\big)$ könnten wir in Haskell definieren durch: `f n | even n = 10+n`
Der ungerade Fall bleibt undefiniert.

Eine andere obere Schranke, aber nicht die kleinste, wäre $g(n) = 10 + n$. Es gilt $f \sqsubseteq g$, aber $g \not\sqsubseteq f$.

Scott Stetigkeit

Definition (Scott Stetigkeit)

Seien X und Y dcpos. Eine Funktion $f : X \to Y$ ist **stetig** (engl. continuous), wenn sie monoton ist und für jede Kette $x_0 \sqsubseteq x_1 \sqsubseteq x_2 \sqsubseteq \cdots$ in X mit Supremum $x \in X$ folgt, dass

$$f\big(\sup(x_0 \sqsubseteq x_1 \sqsubseteq x_2 \sqsubseteq \cdots)\big) = \sup\big(f(x_0) \sqsubseteq f(x_1) \sqsubseteq f(x_2) \sqsubseteq \cdots\big)$$

gilt.

Benannt nach dem amerikanischen Mathematiker Dana Scott

Semantik für logisches Oder

```
or True _    = True
or _    True = True
or _    _    = False
```

Implementation des logischen ODER hat folgende Semantik:

		Zweites Argument		
	or	⊥	True	False
1. Arg.	⊥	⊥	⊥	⊥
	True	True	True	True
	False	⊥	True	False

Eine Funktion mit folgender Semantik ist in Haskell (ohne Nebenläufigkeit) nicht programmierbar:

por	⊥	True	False
⊥	⊥	True	⊥
True	True	True	True
False	⊥	True	False

Supremum einer Kette von Listen

Supremum einer Kette von Listen:

$$\bot \sqsubseteq () : \bot \sqsubseteq () : () : \bot \sqsubseteq () : () : () : \bot \sqsubseteq () : () : () : () : \bot \sqsubseteq \cdots$$

Diese Kette ist von der Form $x_0 \sqsubseteq g(x_0) \sqsubseteq g(g(x_0)) \sqsubseteq \cdots$ wobei $x_0 = \bot$ und $g(l) = () : l$.

Das Supremum ist die unendliche Liste von Unit-Elementen:

$$() : () : () : () : () : \cdots$$

Zusammenfassung Denotationelle Semantik

- Denotationelle Semantik interpretiert Datentypen als Mengen: Funktionen als mathematische Funktionen zwischen diesen Mengen
- Sinn der denotationellen Semantik: Beschreibung des Programmverhaltens, Spezifikation des Auswertemechanismus, Gleichungsschließen für die Programmverifikation.
- Beliebige Mengen, beliebige Funktionen funktioniert nicht, da Rekursion nicht interpretierbar. Daher zusätzliche Struktur mitführen (dcpo, Monotonie, Stetigkeit, s.u.)
- Partielle Ordnung, Hasse Diagramm, kleinstes Element $\bot$
- Monotone Funktionen
- Rekursive Definition als Supremum einer Kette, Fixpunktsemantik
- Gerichtet vollständige partielle Ordnung (dcpo), stetige Funktionen
- Unterschied: strikte, nicht-strikte Funktion
- Interpretation algebraischer Datentypen als dcpos. Insbesondere Hasse Diagramme von `Maybe Bool` und `[()]` Listen über `()`