Funktionales Programmieren

Teil 8

Carl Philipp Reh

Universität Siegen

24. November 2023

Semantik der Fakultät

Sei wieder fact: $\mathbb{Z}_\perp \to \mathbb{Z}_\perp$ definiert als

$$\text{fact}(x) = \begin{cases} \perp & \text{falls } x = \perp, \\ 1 & \text{falls } x < 0, \\ x! & \text{falls } x \geq 0. \end{cases}$$

Eine *rekursive* Implementierung von fact in Haskell könnte folgendermaßen aussehen:

```
fact :: Int -> Int
fact x = if x <= 0 then 1
         else x * fact (x - 1)
```

Wegen dem rekursiven Aufruf hängt die Semantik von `fact` von sich selbst ab. Wie gehen wir damit um?

Semantik der Fakultät

Betrachten wir wieder die Kette $c(i) = \mathsf{fact}_i$, wobei $\mathsf{fact}_i : \mathbb{Z}_\perp \to \mathbb{Z}_\perp$ für $i \in \mathbb{N}$ definiert ist als

$$\mathsf{fact}_0(x) = \perp$$

$$\mathsf{fact}_{i+1}(x) = \begin{cases} \perp & \text{falls } x = \perp \text{ oder } x > i, \\ 1 & \text{falls } x < 0, \\ x! & \text{falls } 0 \leq x \leq i. \end{cases}$$

In Haskell könnte man für endlich viele i jeweils fact_i *ohne Rekursion* über `fact_i :: Int -> Int` implementieren mit

```
fact_0 x = undefined
fact_{i+1} x = if x <= 0 then 1
               else x * fact_i (x - 1)
```

Semantik der Fakultät

Wir haben zum Beispiel, dass

```
fact_1 x = if x <= 0 then 1
           else x * fact_0 (x - 1)
fact_2 x = if x <= 0 then 1
           else x * fact_1 (x - 1)
```

`fact_{i+1}` ist also die Funktion, wo die Rekursion i mal aufgefaltet wurde. Deshalb berechnet `fact_{i+1}` die Fakultät „bis i“, also fact_{i+1}. Solch eine Implementierung würde natürlich unendlich viel Code benötigen.
Stattdessen betrachten wir die Funktion `rfact`, die es uns erlaubt, von `fact_i` zu `fact_{i+1}` zu gelangen, also die Rekursion einmal aufzufalten. Diese Funktion erhält im Gegensatz zu `fact` die rekursiv aufzurufende Funktion als Parameter.

Semantik der Fakultät

```
rfact :: (Int -> Int) -> (Int -> Int)
rfact f x = if x <= 0 then 1
            else x * f (x - 1)
```

Wir erhalten für $[\![\mathrm{rfact}]\!] : (\mathbb{Z}_\perp \to \mathbb{Z}_\perp) \to (\mathbb{Z}_\perp \to \mathbb{Z}_\perp)$, dass

$$[\![\mathrm{rfact}]\!](f)(x) = \begin{cases} \perp & \text{falls } x = \perp, \\ 1 & \text{falls } x \leq 0, \\ x \cdot f(x-1) & \text{falls } x > 0. \end{cases}$$

Wir können nun `fact_i` erhalten, indem wir `rfact` i mal auf **`undefined`** anwenden, also

```
fact_0 = undefined
fact_1 = rfact undefined
fact_2 = rfact (rfact undefined)
...
```

Fixpunkte

Die Idee ist nun, dass man immer weiter `rfact` anwendet, also die Rekursion immer weiter auffaltet, und so irgendwann bei `fact` ankommt. Intuitiv ist man mit dem Prozess erst fertig, wenn man eine Funktion f erhält, sodass $[\![\mathrm{rfact}]\!](f) = f$ gilt. Das heißt, dass `f` ein *Fixpunkt* von $[\![\mathrm{rfact}]\!]$ ist. In dem Fall kann nämlich ein weiteres Anwenden von `rfact` nichts „Neues“ mehr bringen.
Die Fakultät ist ein solcher Fixpunkt von $[\![\mathrm{rfact}]\!]$, da sie folgende Gleichung erfüllt:

$$[\![\mathsf{rfact}]\!](\mathsf{fact})(x) = \begin{cases} \bot & \text{falls } x = \bot, \\ 1 & \text{falls } x \leq 0, \\ x \cdot \mathsf{fact}(x-1) & \text{falls } x > 0. \end{cases}$$

Dies kann man nachrechnen, indem man die Definition von fact einsetzt.

Fixpunkte

Funktionen können aber mehrere Fixpunkte haben. Zum Beispiel betrachten wir

```
nonterm :: Int -> Int
nonterm x = nonterm (x + 1)
```

Wir können auch hier die Rekursion als Parameter übergeben und erhalten:

```
rnonterm :: (Int -> Int) -> (Int -> Int)
rnonterm f x = f (x + 1)
```

Die Forderung für einen Fixpunkt f von $[\![\mathrm{rnonterm}]\!]$ ist also, dass $f(x) = f(x+1)$ für alle $x \in \mathbb{Z}$ gilt. Das heißt, es muss $f(y) = f(z)$ für alle $y, z \in \mathbb{Z}$ gelten. Außerdem gilt $f(\bot) = f(\bot + 1) = f(\bot)$, also wird über $f(\bot)$ nichts gefordert.

Fixpunkte

Wir wollen nun, dass `fact` der „am wenigsten definierte“ Fixpunkt von `rfact` ist. Intuitiv heißt das, dass der Fixpunkt, den wir wählen, nichts definiert, was nicht aus der Definition von `rfact` kommt. Im Fall von `nonterm` ist dies also die überall undefinierte Funktion.

Formal ist ein *Fixpunkt* einer Funktion $f : D \to D$ ein Element $x \in D$ mit $f(x) = x$. Sei $(D, \sqsubseteq_D)$ eine partielle Ordnung. x heißt *kleinster Fixpunkt von f*, wenn er kleiner als alle (anderen) Fixpunkte von f ist, also wenn für jedes $y \in D$ mit $f(y) = y$ gilt, dass $x \sqsubseteq_D y$. Wir bezeichnen den kleinsten Fixpunkt von f bezüglich $\sqsubseteq_D$ (falls dieser existiert) mit μf.

Fixpunktsatz (Teil 1)

Der folgende Satz ist das zentrale Ergebnis über stetige Funktionen. Er besagt, dass jede stetige Funktion einen kleinsten Fixpunkt hat und zeigt sogar, wie man diesen erhält.

Satz 11

Sei $(D, \sqsubseteq_D)$ eine CPO. Jede stetige Funktion $f : [D \to D]$ hat einen kleinsten Fixpunkt. Dieser ist $\mu f = \sqcup c$, wobei $c : \mathbb{N} \to D$ die Kette $c(i) = f^i(\bot_D)$ ist.

Beweis.

Da $\bot_D$ kleinstes Element ist, gilt $\bot_D \sqsubseteq f(\bot_D)$. Per Induktion folgt aus der Monotonie von f, dass $f^i(\bot_D) \sqsubseteq f^{i+1}(\bot_D)$ für alle $i \in \mathbb{N}$. Somit ist c eine Kette. Wir wollen nun zeigen, dass $\sqcup c = \mu f$.

Fixpunktsatz (Teil 2)

Beweis.

Um zu zeigen, dass $\sqcup c$ überhaupt ein Fixpunkt von f ist, müssen wir zeigen, dass $f(\sqcup c) = \sqcup c$. Zunächst gilt, dass $f(\sqcup c) = \sqcup(f \circ c)$, weil f stetig ist. Es gilt $\lambda i.(f \circ c)(i) = \lambda i.f^{i+1}(\bot_D) = \lambda i.c(i+1)$. Dann ist $c' : \mathbb{N} \to D$ mit $c'(i) = c(i+1) = (f \circ c)(i)$ auch eine Kette (nach Folgerung 5). Des Weiteren gilt $\sqcup c = \sqcup c'$ (Beweis: Übung), woraus wir schließen, dass $\sqcup(f \circ c) = \sqcup c$.
Sei d ein (anderer) Fixpunkt von f, also $f(d) = d$. Zu zeigen ist noch, dass $\sqcup c \sqsubseteq d$. Es gilt $\bot_D \sqsubseteq d$. Per Induktion folgt aus der Monotonie von f, dass $f^i(\bot_D) \sqsubseteq f^i(d)$ für alle $i \in \mathbb{N}$. Weil d Fixpunkt ist, gilt auch $f^i(d) = d$ für alle $i \in \mathbb{N}$, also $f^i(\bot_D) \sqsubseteq d$. Damit haben wir, dass $\mathrm{Img}(c) = \{f^i(\bot_D) \mid i \in \mathbb{N}\} \sqsubseteq d$ und somit $\sqcup c \sqsubseteq d$. □

Semantik der Fakultät

Wir wollen nun noch zeigen, dass $\mu[\![\mathrm{rfact}]\!] = \mathsf{fact}$. Dazu zeigen wir zunächst per Induktion, dass $[\![\mathrm{rfact}]\!]^i(\bot) = \mathsf{fact}_i$ für alle $i \in \mathbb{N}$. Es gilt $[\![\mathrm{rfact}]\!]^0(\bot) = \bot = \mathsf{fact}_0$. Sei $i \in \mathbb{N}$. Dann gilt $[\![\mathrm{rfact}]\!](\mathsf{fact}_i) = \mathsf{fact}_{i+1}$ (Beweis: Übung). Aus $[\![\mathrm{rfact}]\!]^i(\bot) = \mathsf{fact}_i$ folgt, dass

$$[\![\mathrm{rfact}]\!]^{i+1}(\bot) = [\![\mathrm{rfact}]\!]([\![\mathrm{rfact}]\!]^i(\bot)) = [\![\mathrm{rfact}]\!](\mathsf{fact}_i) = [\![\mathsf{fact}_{i+1}]\!].$$

Wir erhalten mit dem Fixpunktsatz, dass $\mu[\![\mathrm{rfact}]\!] = \sqcup\lambda i.[\![\mathrm{rfact}]\!]^i(\bot)$. Außerdem wissen wir bereits, dass $\sqcup\lambda i.\,\mathsf{fact}_i = \mathsf{fact}$. Insgesamt erhalten wir also, dass

$$\mu[\![\mathrm{rfact}]\!] = \sqcup\lambda i.[\![\mathrm{rfact}]\!]^i(\bot) = \sqcup\lambda i.\,\mathsf{fact}_i = \mathsf{fact}\,.$$

Man müsste noch zeigen, dass $[\![\mathrm{rfact}]\!]$ stetig ist. Dies erhält man aus der Tatsache, dass Fallunterscheidung, Applikation und die Basisfunktionen wie Vergleiche, Multiplikation, usw. stetig sind.

Fixpunkte

Den kleinsten Fixpunkt von `rfact` kann man *nicht* ausrechnen, da man `rfact` unendlich oft anwenden müsste. Warum ist dies aber kein Problem?
Erstens wollen wir über diesen Prozess eine mathematische Funktion definieren. Dazu müssen wir diese nicht von einem Programm ausrechnen lassen.
Zweitens genügt es für das Ausrechnen einer Stelle der Fakultät, `rfact` nur *endlich* oft anzuwenden, also eine endliche Approximation von $\mu[\![\mathrm{rfact}]\!]$ auszurechnen.