Material zur Vorlesung "Semantik von Programmiersprachen"

Prof. Gert Smolka und Martin Müller

1 Nicht-strikte Evaluierung

Idee. Evaluiere die Argumente von Prozeduren nicht vor dem Aufruf, sondern erst danach und nur falls erforderlich:

$$\frac{t_1 \to_l \lambda x.t_1' \qquad t_1'[t_2/x] \to_l c}{t_1\ t_2 \to_l c}$$

Konsequenzen.

- Konditional ist abstrahierbar

$$\begin{array}{ll} I & = \quad \lambda x.\lambda y.\lambda z.\mathbf{if}\ x\ \mathbf{then}\ y\ \mathbf{else}\ z \\ I\ t_1\ t_2\ t_3 & \text{verhält sich wie } \mathbf{if}\ t_1\ \mathbf{then}\ t_2\ \mathbf{else}\ t_3 \end{array}$$

- Kodierung der Booleschen Werte wird einfacher

$$\begin{array}{lcl} \mathbf{true} & \Rightarrow & \lambda x.\lambda y.x \\ \mathbf{false} & \Rightarrow & \lambda x.\lambda y.y \\ \mathbf{if}\ t_1\ \mathbf{then}\ t_2\ \mathbf{else}\ t_3 & \Rightarrow & t_1\ t_2\ t_3 \end{array}$$

- Abstraktion von Rekursion wird einfach

$$\mathbf{rec}\ x.t \ \sim_L\ R_L(\lambda x.t), \qquad R_L = \mathbf{rec}\ r.\lambda g.g\ (r\ g)$$

- Rekursion kann allgemeiner definiert werden

$$\mathbf{rec}\ x.1, x \quad \text{entspricht} \quad 1, (1, (1, \ldots))$$

Haskell und Miranda sind Programmiersprachen, die auf nicht-strikter Evaluierung beruhen.

2 Eine funktionale Programmiersprache mit nicht-strikter Evaluierung

LF ist eine funktionale Programmiersprache mit nicht-strikter Evaluierung ("lazy"). Das applikative Fragment LF_0 entspricht dem call-by-name oder lazy λ-Kalkül.

Die Terme **LTer** von LF sind dieselben wie die von F, bis auf das Rekursionskonstrukt **rec** $x.t$, das in LF für t einen beliebigen Term zuläßt. Die geschlossenen Terme von LF bezeichnen wir mit **GLTer**.

Kanonische Terme. Die Menge **CLTer** der kanonischen Terme von LF ist wie folgt definiert, wobei zusätzlich gefordert wird, daß jeder kanonischer Term geschlossen ist.

$$c \ ::= \ 0 \mid 1 \mid -1 \mid \ldots \mid \textbf{true} \mid \textbf{false} \mid () \mid t_1, t_2 \mid \textbf{inl}\ t \mid \textbf{inr}\ t \mid \lambda x.t$$

Evaluierung. Nicht-strikte Evaluierung ist eine binäre Relation $\rightarrow_l \subseteq \textbf{GLTer} \times \textbf{CLTer}$ auf geschlossenen Termen (Ergebnissemantik).

Regelname	Regel	Analoge Bildung
(kan-l)	$c \rightarrow_l c$	
(+-l)	$\dfrac{t_1 \rightarrow_l n_1 \qquad t_2 \rightarrow_l n_2}{t_1 + t_2 \rightarrow_l n} \quad n = n_1 + n_2$	$(*, -)$
($\leq$-false-l)	$\dfrac{t_1 \rightarrow_l n_1 \qquad t_2 \rightarrow_l n_2}{t_1 \leq t_2 \rightarrow_l \textbf{false}} \quad n_1 > n_2$	($\leq$-true-l)
(if-true-l)	$\dfrac{t_1 \rightarrow_l \textbf{true} \qquad t_2 \rightarrow_l c}{\textbf{if}\ t_1\ \textbf{then}\ t_2\ \textbf{else}\ t_3 \rightarrow_l c}$	(if-false-l)
(fst-l)	$\dfrac{t \rightarrow_l t_1, t_2 \qquad t_1 \rightarrow_l c_1}{\textbf{fst}\ t \rightarrow_l c_1}$	(snd-l)
(case-l-l)	$\dfrac{t \rightarrow_l \textbf{inl}\ t' \qquad t_1[t'/x_1] \rightarrow_l c}{\textbf{case}\ t\ \textbf{of}\ \textbf{inl}\ x_1.t_1\ []\ \textbf{inr}\ x_2.t_2 \rightarrow_l c}$	(case-r-l)
(appl)	$\dfrac{t_1 \rightarrow_l \lambda x.t_1' \qquad t_1'[t_2/x] \rightarrow_l c}{t_1\ t_2 \rightarrow_l c}$	
(rec)	$\dfrac{t[\textbf{rec}\ x.t/x] \rightarrow_l c}{\textbf{rec}\ x.t \rightarrow_l c}$	