Institut für Informatik
der Universität München
Dr. Andreas Abel

SS 2007

22. Juni 2007

Übungen zur Vorlesung
Typsysteme

Blatt 8

Aufgabe P-22 (Powerlists): Powerlisten $\mathsf{PList}\,A$ sind A-Listen, deren Länge eine Zweiterpotenz (engl. *power of two*) ist. Gegeben sind sie durch die Konstruktoren

$$\begin{array}{lcl} \mathsf{PList} & : & * \to * \\ \mathsf{zero} & : & \forall A.\, A \to \mathsf{PList}\,A \\ \mathsf{succ} & : & \forall A.\, \mathsf{PList}\,(A \times A) \to \mathsf{PList}\,A \end{array}$$

Z.B. ist $p_0 := \mathsf{succ}\,(\mathsf{succ}\,(\mathsf{succ}\,(\mathsf{zero}\,(((1,2),(3,4)),((5,6),(7,8))))))$ eine Powerlist der Länge 2^3. In F^ω kann man Powerlisten imprädikativ kodieren:

$$\begin{array}{lcl} \mathsf{PList} & := & \lambda B.\, \forall X : * \to *.\, (\forall A.\, A \to X\,A) \to (\forall A.\, X\,(A \times A) \to X\,A) \to X\,B \\ \mathsf{zero} & := & \lambda a.\, \lambda z \lambda s.\, z\,a \\ \mathsf{succ} & := & \lambda p.\, \lambda z \lambda s.\, s\,(p\,z\,s) \end{array}$$

Schreiben Sie die Definitionen von zero und succ im Church-Stil. Definieren Sie die Funktionen

$$\begin{array}{lcl} \mathsf{map} & : & \forall A.\, \mathsf{PList}\,A \to \forall B.\, (A \to B) \to \mathsf{PList}\,B \\ \mathsf{sum} & : & \forall A.\, \mathsf{PList}\,A \to (A \to \mathsf{Int}) \to \mathsf{Int} \end{array}$$

unter Zuhilfenahme von $+ : \mathsf{Int} \to \mathsf{Int} \to \mathsf{Int}$ und berechnen Sie $\mathsf{sum}\,p_0\,(\lambda x x)$.

Aufgabe P-23 (Lesemonade): Die Lesemonade $\mathsf{Reader}\,R := \lambda A.\, R \to A : * \to *$ encapsuliert Berechnungen vom Typ A, die auf einen Speicher R nurlesend zugreifen. Geben Sie λ-Terme für die allgemeinen Monadenoperationen

$$\begin{array}{lcl} \mathsf{return} & : & \forall R \forall A.\, A \to \mathsf{Reader}\,R\,A \\ \mathsf{bind} & : & \forall R \forall A.\, \mathsf{Reader}\,R\,A \to (A \to \mathsf{Reader}\,R\,B) \to \mathsf{Reader}\,R\,B \end{array}$$

an, und zeigen Sie, dass die Monadengesetze (siehe unten) erfüllt sind. Geben Sie auch λ-Terme an für die Reader-speziellen Operationen

$$\begin{array}{lcl} \mathsf{ask} & : & \forall R.\, \mathsf{Reader}\,R\,R \\ \mathsf{local} & : & \forall R \forall A.\, (R \to R) \to \mathsf{Reader}\,R\,A \to \mathsf{Reader}\,R\,A \end{array}$$

Dabei fragt ask den Speicher ab, und $\mathsf{local}\,f\,m$ führt die Berechnung m in dem durch f modifizierten Speicher aus. (Dies ist keine Verletzung von *nur-lesend*, da die Modifikation nur lokal für m gilt und danach wieder ungeschehen ist.)

Aufgabe H-24 (Programmieren mit der Zustandsmonade, 6 Punkte): Die Zustandsmonade

$$\begin{array}{lcl} \mathsf{State} & : & * \to * \to * \\ \mathsf{State} & := & \lambda S \lambda A.\, S \to A \times S \end{array}$$

wird in Haskell dazu verwendet, Berechungen mit veränderbaren Speicher vom Typ S (also eine "große" Zelle vom Typ S) zu kodieren.

Implementieren Sie in SML die Monadenoperationen für State,

$$\begin{array}{lcl}
\mathsf{return} & : & \forall S \forall A.\, A \to \mathsf{State}\, S\, A \\
\mathsf{bind} & : & \forall S \forall A.\, \mathsf{State}\, S\, A \to (A \to \mathsf{State}\, S\, B) \to \mathsf{State}\, S\, B \\
\mathsf{get} & : & \forall S.\, \mathsf{State}\, S\, S \\
\mathsf{set} & : & \forall S.\, S \to \mathsf{State}\, S\, \mathsf{Unit}
\end{array}$$

und eine Beispielanwendung. Vervollständigen Sie dazu die Datei `state-stub.sml`. Bemerkung: $\mathsf{return}\, a : \mathsf{State}\, S\, A$ bettet einen speicherunabhängigen Wert a in die Monade ein, $\mathsf{bind}\, m\, (\lambda a.\, f\, a)$ schleift den Speicher durch die Berechnung m, die einen Wert a liefert, und die folgende Berechnung $f\, a$ durch. Mit get wird der aktuelle Speicherinhalt gelesen, und mit $\mathsf{set}\, s$ auf s gesetzt; set ist reiner Effekt, liefert also kein Ergebnis (Typ Unit).

Aufgabe H-25 (Monadengesetze, 6 Punkte): Beweisen Sie informell, dass die folgenden 3 Monadengesetze für die Zustandsmonade gelten:

$$\begin{array}{lllcl}
(\beta) & \mathsf{bind}\, (\mathsf{return}\, a)\, f & = & f\, a \\
(\eta) & \mathsf{bind}\, m\, \mathsf{return} & = & m \\
(assoc) & \mathsf{bind}\, (\mathsf{bind}\, m\, (\lambda a.\, f\, a)\, g & = & \mathsf{bind}\, m\, (\lambda a.\, \mathsf{bind}\, (f\, a)\, g)
\end{array}$$

Aufgabe H-26 (Zulässigkeit von η in der algorithmischen Gleichheit, 4 Punkte): Zeigen Sie: Wenn $\Gamma \vdash F = F' \leftleftarrows \kappa \to \kappa'$ und $X \notin \mathsf{FV}(F)$, dann auch $\Gamma \vdash (\lambda X.\, F\, X) = F' \leftleftarrows \kappa \to \kappa'$. Dafür dürfen Sie ohne Beweis benutzen, dass $\mathsf{whnf}((\lambda X F)\, G\, \vec{G}) = \mathsf{whnf}(F[G/X]\, \vec{G})$.

Aufgabe H-27 (Produktarten, 4 Punkte + 2 Sonderpunkte): Folgende Regeln erweitern F^ω um *product kinds*.

$$\frac{\Gamma \vdash F_1 : \kappa_1 \qquad \Gamma \vdash F_2 : \kappa_2}{\Gamma \vdash \langle F_1, F_2 \rangle : \kappa_1 \times \kappa_2} \qquad \frac{\Gamma \vdash F : \kappa_1 \times \kappa_2}{\Gamma \vdash \pi_1 F : \kappa_1} \qquad \frac{\Gamma \vdash F : \kappa_1 \times \kappa_2}{\Gamma \vdash \pi_2 F : \kappa_2}$$

$$\frac{\Gamma \vdash F_1 : \kappa_1 \qquad \Gamma \vdash F_2 : \kappa_2}{\Gamma \vdash \pi_i \langle F_1, F_2 \rangle = F_i : \kappa_i} \qquad \frac{\Gamma \vdash F : \kappa_1 \times \kappa_2}{\Gamma \vdash \langle \pi_1 F, \pi_2 F \rangle = F : \kappa_1 \times \kappa_2}$$

Neue neutrale Terme sind $\pi_i N$, und $\langle F_1, F_2 \rangle$ ist eine neue schwache Kopf-Normalform.

Erweitern Sie die Definitionen von whnf und der algorithmischen Gleichheit, d.h., fügen Sie die nötigen Regeln zu den Urteilen $\Gamma \vdash F = F' \leftleftarrows \kappa$ und $\Gamma \vdash N = N' \rightrightarrows \kappa$ hinzu.

Abgabe der Hausaufgaben H-X zum Beginn der nächsten Übungsstunde.