Scheme 2

1 Kernsyntax

Namen: N

Konstanten:

$$K ::= \textit{Zahl} \mid \texttt{\#t} \mid \texttt{()} \mid \texttt{(}\textbf{quote}\ \textit{Symbol}\texttt{)}$$

Ausdrücke:

A	::=	K	*Konstante*
	|	N	*Name*
	|	**if** A_1 **then** A_2 **else** A_3 **end**	*Konditional*
	|	(`(`$A_1\ \ldots\ A_k$`)`)	*Applikation*, $k \geq 1$
	|	**lambda** (`(`$N_1\ \ldots\ N_k$`)`) A **end**	*Abstraktion*, $k \geq 0$
	|	N`:=`A	*Zuweisung*
	|	$A_1; A_2$	*Sequentialisierung*
	|	**new** N **in** A **end**	*Deklaration*

- Abstraktionen und Deklarationen sind *Variablenbinder*.
- Ausdrücke können eindeutig geparst werden, in dem man die Sequentialisierung nach rechts assoziieren läßt:

$$A_1; A_2; A_3 \qquad \Rightarrow \qquad A_1; \{A_2; A_3\}$$

- Die Sequentialisierung "$A_1; A_2$" wird oft als Juxtaposition "$A_1\ A_2$" geschrieben.

2 Werte, Umgebung, Zustände

- **Schreibweise:** $A \rightsquigarrow B$ bezeichnet eine endliche partielle Funktion von A nach B.
- **Umgebungen** (U) sind Funktionen: **Name** $\rightsquigarrow$ **Lokation**.
- **Zustände** (Z) sind Abbildungen: **Lokation** $\rightsquigarrow$ **Wert**.
- **Lokationen** (l) sind Elemente einer totalgeordneten unendlichen Menge **Lokation** (z.B. $\mathbb{N}$).
- **Werte** (W):

W	::=	*EchterWert*
	|	*DefinierteProzedur*

- **Echte Werte**:

$$\begin{aligned} \textit{EchterWert} \quad ::= \quad & \textit{Zahl} \mid \texttt{\#t} \mid \texttt{()} \mid \textit{Symbol} \\ \mid \quad & \boxed{\text{undef}} \\ \mid \quad & (l_1, l_2) \\ \mid \quad & \textit{ProzedurReferenz} \end{aligned}$$

- **Prozedur Referenzen**:

$$\textit{ProzedurReferenz} ::= \textit{PrimitiveProzedur} \mid l$$

- **Primitive Prozeduren**:

$$\textit{PrimitiveProzedur} ::= \boxed{+} \mid \boxed{=} \mid \boxed{\text{cons}} \mid \boxed{\text{car}} \mid \boxed{\text{set-car!}} \mid \boxed{\text{eq?}} \mid \cdots$$

- **Definierte Prozeduren**:

$$\textit{DefinierteProzedur} ::= (A, U)$$

  - A ist Abstraktion
  - U ist Umgebung, die genau auf den freien Variablen von A definiert ist.

Adjunktion:

- $$\textbf{adj}[Z, W] := (Z[W/l], l)$$
  wobei l die kleinste Lokation ist, auf der Z nicht definiert ist.
- $$\textbf{adj}[U, Z, N, W] := (U[l/N], Z')$$
  wobei $(Z', l) = \textbf{adj}[Z, W]$

3 Semantik

Eine Transition

$$A, U, Z \longrightarrow W, Z'$$

besagt, daß der Ausdruck A in der Umgebung U mit dem Anfangszustand Z zu dem Wert W mit dem Endzustand Z' evaluiert.
Für jede Transition $A, U, Z \longrightarrow W, Z'$ müssen die folgenden Bedingungen gelten:

1. U ist auf allen freien Namen von A definiert.
2. Wenn U auf N definiert ist, dann ist Z auf $U(N)$ definiert, und $Z(U(N))$ ist ein echter Wert.

3. Wenn $Z(l)$ eine Lokation ist, dann ist $Z(Z(l))$ eine definierte Prozedur.

4. Wenn $Z(l) = (A, U')$ und U' auf N' definiert ist, dann ist Z auf $U'(N')$ definiert, und $Z(U'(N'))$ ist ein echter Wert.

Konstanten

- $K, U, Z \longrightarrow K, Z$
  falls K eine Konstante, aber kein quotiertes Symbol ist.
- $\texttt{(quote}\ S\texttt{)}, U, Z \longrightarrow S, Z$

Namen

$$N, U, Z \longrightarrow Z(U(N)), Z$$

Zuweisung

$$\frac{A, U, Z \longrightarrow W, Z'}{N\texttt{:=}A, U, Z \longrightarrow \boxed{\text{undef}}, Z'[W/U(N)]}$$

Sequentialisierung

$$\frac{A_1, U, Z \longrightarrow W_1, Z_1 \qquad A_2, U, Z_1 \longrightarrow W_2, Z_2}{A_1\texttt{;}\ A_2, U, Z \longrightarrow W_2, Z_2}$$

Deklaration

$$\frac{A, U', Z' \longrightarrow W, Z''}{\texttt{new}\ N\ \texttt{in}\ A\ \texttt{end}, U, Z \longrightarrow W, Z''} \qquad U', Z' = \textbf{adj}[U, Z, N, \boxed{\text{undef}}]$$

Konditional

- $$\frac{A_1, U, Z \longrightarrow \texttt{()}, Z_1 \qquad A_3, U, Z_1 \longrightarrow W_3, Z_3}{\texttt{if}\ A_1\ \texttt{then}\ A_2\ \texttt{else}\ A_3\ \texttt{end}, U, Z \longrightarrow W_3, Z_3}$$
- $$\frac{A_1, U, Z \longrightarrow W_1, Z_1 \qquad A_2, U, Z_1 \longrightarrow W_2, Z_2}{\texttt{if}\ A_1\ \texttt{then}\ A_2\ \texttt{else}\ A_3\ \texttt{end}, U, Z \longrightarrow W_2, Z_2} \qquad W_1 \neq \texttt{()}$$

Abstraktion

$$A, U, Z \longrightarrow l, Z'$$

falls A Abstraktion, $P = (A, U|_{\mathrm{F}[A]})$, $(Z', l) = \textbf{adj}[Z, P]$.

Applikation

$$\frac{\begin{array}{rcl} A_1, U, Z & \longrightarrow & W_1, Z_1 \\ A_2, U, Z_1 & \longrightarrow & W_2, Z_2 \\ & \vdots & \\ A_k, U, Z_{k-1} & \longrightarrow & W_k, Z_k \end{array} \qquad \textbf{apply}[W_1, \ldots, W_k], U, Z_k \longrightarrow W, Z'}{\texttt{(}A_1\ \ \ldots\ \ A_k\texttt{)}, U, Z \longrightarrow W, Z'}$$

Primitive Prozeduren

- **apply**$[\boxed{\text{cons}}, W_1, W_2], U, Z \longrightarrow (l_1, l_2), Z_2$
  falls $(Z_1, l_1) =$ **adj**$[Z, W_1]$ und $(Z_2, l_2) =$ **adj**$[Z_1, W_2]$.
- **apply**$[\boxed{\text{car}}, (l_1, l_2)], U, Z \longrightarrow Z(l_1), Z$
- **apply**$[\boxed{\text{set-car!}}, (l_1, l_2), W], U, Z \longrightarrow \boxed{\text{undef}}, Z[W/l_1]$
- **apply**$[\boxed{\text{eq?}}, W_1, W_2], U, Z \longrightarrow W, Z$
  wobei $W = \begin{cases} \texttt{\#t} & \text{falls } W_1 = W_2 \\ \texttt{()} & \text{sonst} \end{cases}$

Definierte Prozeduren

$$\frac{A, U', Z' \longrightarrow W', Z''}{\textbf{apply}[l, W], U, Z \longrightarrow W', Z''} \qquad \begin{array}{l} Z(l) = (\texttt{lambda}\ \ (N)\ \ A\ \ \texttt{end}, U_A) \\ U', Z' = \textbf{adj}[U_A, Z, N, W] \end{array}$$

Die Regel für nichteinstellige definierte Prozeduren ist entsprechend.