# 5. Übungsblatt „Programmiersprachen“ SS 99

Leif Kornstaedt, Prof. Gert Smolka

**Ausgabe:** 19. Mai 1999 **Abgabe:** 26. Mai 1999

**Aufgabe V.1:** Semantik von F1 mit Ausnahmen $(5 + 11$ Punkte$)$

In dieser Aufgabe wollen wir Ausnahmebehandlung in F1 formalisieren. Der Einfachheit halber beschränken wir uns auf eine Teilsprache F0 von F1, die weder Operationen noch rekursive Prozeduren hat:[1]

| | | | |
|---|---|---|---|
| $a$ | $::=$ | $c$ | Konstante |
| | $\mid$ | $x$ | Identifikator |
| | $\mid$ | **if** $a_1$ **then** $a_2$ **else** $a_3$ | Konditional |
| | $\mid$ | $a_1\ a_2$ | Applikation |
| | $\mid$ | **fn** $x$ **=>** $a$ | Abstraktion |

Die Sprache F erweitert F0 um eine primitive Ausnahmebehandlung:

| | | | |
|---|---|---|---|
| $a$ | $::=$ | $\cdots$ | |
| | $\mid$ | **raise** $a$ | Werfen einer Ausnahme |
| | $\mid$ | $a_1$ **handle** $a_2$ | Fangen einer Ausnahme |

Wir definieren die Ausnahmebehandlung durch Übersetzung von F nach SML. Der Ausnahmekonstruktor ist dabei durch

```
exception Ausnahme of int
```

deklariert.

| F | SML |
|---|---|
| **raise** $a$ | **raise** `Ausnahme` $a$ |
| $a_1$ **handle** $a_2$ | $a_1$ **handle** `Ausnahme x =>` $a_2$ `x` |

Beispielsweise wertet folgender Ausdruck zu `true` aus:

```
(raise 7) handle (fn x => true)
```

a. Definieren Sie die statische Semantik

$$T \vdash a \Rightarrow \tau$$

von F durch Inferenzregeln. Dabei sei

[1] Die Ergebnisse lassen sich direkt auf volles F1 mit Ausnahmen übertragen.

$$\begin{array}{lcl} \tau & ::= & \texttt{int} \\ & | & \texttt{bool} \\ & | & \tau_1 \texttt{ -> } \tau_2 \end{array}$$

b. Definieren Sie die dynamische Semantik

$$\begin{array}{ll} W \vdash a \Rightarrow w & \text{„Auswertung liefert Wert“} \\ W \vdash a \Rightarrow [w] & \text{„Auswertung wirft Ausnahme“} \end{array}$$

von F durch Inferenzregeln. Dabei sei

$$\begin{array}{lllll} w & \in & \text{Wert} & = & \mathbb{Z} \cup \mathbb{B} \cup \text{Pro} & \text{Werte} \\ & & \mathbb{Z} & & & \textit{Ganze Zahlen} \\ & & \mathbb{B} & = & \{\texttt{true}, \texttt{false}\} & \textit{Boole'sche Werte} \\ & & \text{Pro} & = & \text{Bez} \times \text{Aus} \times \text{WU} & \textit{Prozeduren} \\ W & \in & \text{WU} & = & \text{Bez} \xrightarrow{\text{fin}} \text{Wert} & \text{Wertumgebungen} \end{array}$$

## **Aufgabe V.2:** Semantik von F2 mit Ausnahmen $(3 + 6 + 4 + 4$ Punkte$)$

In dieser Aufgabe erweitern wir F2 um eine SML-artige Ausnahmebehandlung zu einer Sprache F3:

$$\begin{array}{lcll} a & ::= & \cdots & \\ & | & \textbf{exception } K \textbf{ in } a & \text{Ausnahmendeklaration} \\ & | & \textbf{raise } a & \text{Werfen einer Ausnahme} \\ & | & a \textbf{ handle } F & \text{Fangen einer Ausnahme} \end{array}$$

a. Erweitern Sie die statische Semantik von F2 (also die Relation $T \vdash a \Rightarrow \tau$) um Regeln für die 3 Konstrukte zur Ausnahmebehandlung.

b. Erweitern Sie die dynamische Semantik von F2 (also die Relation $W \vdash a \Rightarrow w$) um Regeln für die Konstrukte zur Ausnahmebehandlung. Dafür müssen Sie – wie in Aufgabe V.1b – eine neue Relation $W \vdash a \Rightarrow [w]$ für Ausnahmepakete einführen. Die dynamische Semantik von **exception** $K$ **in** $a$ sei durch die Regel

$$\frac{\vdash K \Rightarrow W' \quad W + W' \vdash a \Rightarrow w}{W \vdash \textbf{exception } K \textbf{ in } a \Rightarrow w}$$

vorgegeben. Geben Sie nur die Regeln für **raise** und **handle** an, die übrigen Regeln ergeben sich schematisch analog zu Aufgabe V.1b.

c. Zulässige F3-Ausdrücke sind nur dann typkonsistent, wenn sie denselben Bezeichner nicht mehr als einmal als einstelligen Ausnahmekonstruktor deklarieren. Geben Sie einen geschlossenen und zulässigen F3-Ausdruck vom Typ `*()` an, der ein Ergebnis vom Typ `*(*(), *())` liefert.

d. SML vermeidet das obige Konsistenzproblem dadurch, dass jede dynamische Auswertung einer Ausnahmedeklaration einen neuen Ausnahmekonstruktur liefert. Geben Sie einen geschlossenen und zulässigen F3-Ausdruck mit genau einer nullstelligen Ausnahmedeklaration an, dessen dynamische Semantik in F3 aufgrund dieser Tatsache verschieden von SML ist. Während F2 den Wert `()` liefert, soll SML eine Ausnahme werfen.