

Übungen zu „Prinzipien von Programmiersprachen“ Blatt 7

Aufgabe 21. Formalisiere die folgenden Konstrukte. Lege die statische Semantik jeweils mit Hilfe von Typregeln und die dynamische Semantik mit Hilfe von Auswertungsregeln fest. Verwende für die dynamische Semantik die abstrakte Maschine mit einem expliziten Kontrollstack (C-Maschine).

1. Alternative **if** e_1 **then** e_2 **else** e_3 ;
2. einseitige Alternative **when**: e_1 **when** $e_2 \cong$ **if** e_2 **then** e_1 **else** $\{ \}$;
3. einseitige Alternative **unless**: e_1 **unless** $e_2 \cong$ **if** e_2 **then** $\{ \}$ **else** e_1 .

Aufgabe 22. Formalisiere die dynamische Semantik der Schleifenkonstrukte aus Aufgabe 19 mit Hilfe der C-Maschine.

1. Endlosschleife der Form **forever** e **end**;
2. **repeat**-Schleife der Form **repeat** e_1 **until** e_2 ;
3. **for**-Schleife der Form **for** $x = e_1$ **to** e_2 **do** e_3 **end**;
4. **break**.

Aufgabe 23. Welches operationale Verhalten zeigt das folgende Programm?

```
continue (letcc k in continue k k end)
         (letcc k in continue k k end)
```

Ist das Programm wohlgetypt? Wenn ja, leite den Typ formal her. Wenn nein, wie muss das Programm modifiziert werden?

Aufgabe 24. Implementiere die folgenden Generatoren jeweils mit Hilfe eines lokalen Zustands und mit Hilfe einer Semi-Coroutine.

```
type Generator  $\alpha = \{ \} \rightarrow \alpha$ 
```

1. *elements* enumeriert alle Elemente einer gegebenen Liste:

```
elements :  $\forall \alpha. List \alpha \rightarrow Generator \alpha$ 
```

2. *sublists* generiert alle Teillisten einer gegebenen Liste:

```
sublists :  $\forall \alpha. List \alpha \rightarrow Generator (List \alpha)$ 
```

3. *permutations* generiert alle Permutationen einer gegebenen Liste:

```
permutations :  $\forall \alpha. List \alpha \rightarrow Generator (List \alpha)$ 
```