
Grundlagen von Programmiersprachen
<http://www.informatik.uni-freiburg.de/proglang/teaching/ws2003-2004/gvps>

Übungsblatt 9

Abgabe: 22.12.2003

Aufgabe 1 (IMP mit Listen).

Implementiere die imperative Programmiersprache IMP mit einer Erweiterung um Listen.

Um Listen erzeugen und manipulieren zu können wird IMP um Anweisungen zur Erzeugung von leeren Listen (`make-null`) und von Paaren (`make-cons`) und zur Mutation der ersten bzw. der zweiten Paarkomponente eines Paares (`set-car!` bzw. `set-cdr!`) erweitert. Zusätzlich gibt es Primitiva für Typprädikate für leere Listen (`null?`) und Paare (`pair?`) und für den Zugriff auf die erste bzw. zweite Paarkomponente (`car` bzw. `cdr`):

$$\begin{aligned}
 e & ::= \dots \mid \text{null? } e \mid \text{pair? } e \mid \text{car } e \mid \text{cdr } e \\
 s & ::= \dots \mid v := \text{make-null} \mid v := \text{cons } e \ e' \mid \text{set-car! } e \ e' \mid \text{set-cdr! } e \ e'
 \end{aligned}$$

Die Werte werden um die leere Liste und Paare erweitert

$$y \ni \text{Val} = \mathbf{Z} + \text{Loc} + \{\text{nil}\} + (\text{Val} \times \text{Val}) + \dots$$

und die Auswertungsregeln der Big-Step-Semantik zu diesen Operationen sehen wie folgt aus:

$$\begin{array}{c}
 \frac{\sigma, \rho \vdash e \hookrightarrow l \quad \sigma(l) = \text{nil}}{\sigma, \rho \vdash \text{null? } e \hookrightarrow 1} \\
 \\
 \frac{\sigma, \rho \vdash e \hookrightarrow l \quad \sigma(l) \neq \text{nil}}{\sigma, \rho \vdash \text{null? } e \hookrightarrow 0} \\
 \\
 \frac{\sigma, \rho \vdash e \hookrightarrow l \quad \sigma(l) = (y, y')}{\sigma, \rho \vdash \text{pair? } e \hookrightarrow 1} \\
 \\
 \frac{\sigma, \rho \vdash e \hookrightarrow l \quad \sigma(l) \neq (y, y')}{\sigma, \rho \vdash \text{pair? } e \hookrightarrow 0} \\
 \\
 \frac{\sigma, \rho \vdash e \hookrightarrow l \quad \sigma(l) = (y, y')}{\sigma, \rho \vdash \text{car } e \hookrightarrow y} \\
 \\
 \frac{l = \rho(v) \quad l' \notin \text{dom}(\sigma)}{\sigma, \rho \vdash v := \text{make-null} \rightsquigarrow \sigma[l \mapsto l', l' \mapsto \text{nil}], \rho} \\
 \\
 \frac{\sigma, \rho \vdash e \hookrightarrow y \quad \sigma, \rho \vdash e' \hookrightarrow y' \quad l = \rho(v) \quad l' \notin \text{dom}(\sigma)}{\sigma, \rho \vdash v := \text{cons } e \ e' \rightsquigarrow \sigma[l \mapsto l', l' \mapsto (y, y')], \rho} \\
 \\
 \frac{\sigma, \rho \vdash e_1 \hookrightarrow l \quad \sigma, \rho \vdash e_2 \hookrightarrow y \quad \sigma(l) = (y', y'')}{\sigma, \rho \vdash \text{set-car! } e_1 \ e_2 \rightsquigarrow \sigma[l \mapsto (y, y'')], \rho}
 \end{array}$$

Teste Deine Implementierung an mehreren kleinen Beispielprogrammen und gib diese mit an.

Aufgabe 2 (IMP — Parameterübergabe).

Erweitere IMP mit Listen aus Aufgabe 1 um einfache Top-Level-Prozeduren.

- (i) Implementiere diese Erweiterung von IMP, wobei Parameter als Referenzen (*call-by-reference*) übergeben werden sollen.
- (ii) Formalisiere die alternative Parameterübergabestrategie *call-by-value-result* für diese Sprache in Form einer Big-Step-Semantik.
- (iii) Implementiere diese alternative Parameterübergabestrategie.