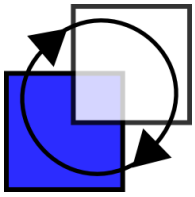
REACTIVE SYSTEMS GROUP

Universität des Saarlandes

Prof. Bernd Finkbeiner, Ph.D.
Markus Rabe, M.Sc.



Programmierung 1 (SS 2010) - 12. Übungsblatt

http://react.cs.uni-saarland.de/prog1/

Lesen Sie im Buch bis 13.4

Aufgabe 12.1

Übersetzen Sie die Gleichungen für freie Bezeichner aus Abbildung 12.3 im Buch gemäß den Typdeklarationen in Abbildung 12.1 nach Standard ML. Realisieren Sie Mengen als Listen.

Aufgabe 12.2

Geben Sie eine Ableitung für die folgende Aussage an:

$$[x := int] \vdash fn\ f : int \to bool \Rightarrow fn\ y : int \Rightarrow f(2 * x + y) : (int \to bool) \to (int \to bool)$$

Aufgabe 12.3

Geben Sie einen möglichst einfachen Ausdruck e an, so dass bei der Ausführung von *eval empty* e jede der 6 Regeln der Prozedur *eval* zum Einsatz kommt.

Aufgabe 12.4

Die Prozedur *eval* kann durch Werfen einer Ausnahme *Error s* einen Fehler s melden. Geben Sie möglichst einfache Ausdrücke an, für die die Prozedur *eval empty* die Fehler "*R Opr*", "*R If*" und "*R App*" meldet. Überprüfen Sie Ihre Antworten mit einem Interpreter.

Aufgabe 12.5

Die Prozedur *eval empty* liefert für viele Ausdrücke Ergebnisse, für die *elab empty* Fehler meldet. Geben Sie für jeden der von *elab* behandelten Fehler einen entsprechenden Ausdruck an.

Aufgabe 12.6

Wir wollen F um Paare erweitern. Die abstrakte Syntax und die Menge der Werte erweitern wir wie folgt:

$$
\begin{aligned}
t \in \mathit{Ty} &= \cdots \mid t * t \\
e \in \mathit{Exp} &= \cdots \mid (e, e) \mid \mathit{fst}\ e \mid \mathit{snd}\ e \\
v \in \mathit{Val} &= \mathbb{Z} \cup \mathit{Pro} \cup (\mathit{Val} \times \mathit{Val})
\end{aligned}
$$

a) Geben Sie die Inferenzregeln für die statische Semantik von Paaren an.

b) Geben Sie die Inferenzregeln für die dynamische Semantik von Paaren an.

c) Erweitern Sie die Deklarationen der Typen *ty*, *exp* und *value* um Konstruktoren für Paare.

d) Erweitern Sie die Deklaration der Prozedur *elab* um Regeln für Paare.

e) Erweitern Sie die Deklaration der Prozedur *eval* um Regeln für Paare.

Aufgabe 12.7

Geben Sie passend zu den Gleichungen in Abbildung 12.3 auf S. 244 eine Gleichung an, die die freien Bezeichner rekursiver Abstraktionen beschreibt.

Aufgabe 12.8

Warum ist die rekursive Abstraktion $\mathit{rfn}\ f\ (f : \mathit{int}) : \mathit{int} \Rightarrow f$ gemäß der Regel Srabs semantisch zulässig?

Aufgabe 12.9

a) Erweitern Sie die Deklarationen der Typen *exp* und *value* um einen Konstruktor für rekursive Abstraktionen.

b) Erweitern Sie die Prozedur *elab* um eine Regel für rekursive Abstraktionen.

c) Erweitern Sie die Prozedur *eval* um eine Regel für rekursive Abstraktionen. Erweitern Sie zudem die Fallunterscheidung in der Regel für Prozeduranwendungen um die Anwendung rekursiver Prozeduren. Erproben Sie Ihre erweiterte Prozedur *eval* mit einer rekursiven Prozedur, die Fakultäten berechnet.

d) Geben Sie einen zulässigen Ausdruck e an, so dass die Auswertung von *eval empty e* divergiert.

Aufgabe 12.10

Deklarieren Sie eine *endrekursive* Prozedur

$$lex' : token\ list \to char\ list \to token\ list,$$

so dass $lex'\ nil\ xs$ für jede Zeichenkette xs dasselbe Ergebnis liefert wie $lex\ xs$ (siehe Buch S. 260).

Aufgabe 12.11

Zeichnen Sie Syntaxbäume für ty und die folgenden Wortfolgen:

a) $int \to (bool \to int)$

b) $(bool \to bool) \to int$

c) $int \to (bool \to bool) \to int$

Aufgabe 12.12

Zeigen Sie mit einem Beispiel, dass die Grammatik

$$exp ::= \text{"}\mathtt{x}\text{"} \ \mid \ exp\ exp$$

nicht eindeutig ist. Geben Sie eine eindeutige Grammatik an, die dieselben Wortfolgen darstellt.