11. Übungsblatt zu Programmierung 1, WS 2012/13

Prof. Dr. Gert Smolka, Sigurd Schneider, B.Sc.
www.ps.uni-saarland.de/courses/prog-ws12/

Lesen Sie im Buch: Kapitel 12

Aufgabe 12.1 Geben Sie Deklarationen an (in Standard ML), die den Bezeichner *e* an die abstrakte Darstellung des Ausdrucks

$$fn\ f : int \to int \Rightarrow\ fn\ n : int \Rightarrow\ if\ n \leq 0\ then\ 1\ else\ n * f(n-1)$$

binden. Gehen Sie dabei schrittweise vor und beginnen Sie mit der Deklaration des Teilausdrucks $n \leq 0$:

```
val e1 = Opr(Leq, Id"n", Con(IC 0))
```

Aufgabe 12.2 In § 2.4 und § 3.8 haben wir definiert, was wir unter offenen und geschlossenen Ausdrücken und den freien Variablen eines Ausdrucks verstehen wollen.

a) Schreiben Sie eine Prozedur *closed* : *exp* → *bool*, die testet, ob ein Ausdruck geschlossen ist. Verwenden Sie dabei eine Hilfsprozedur *closed′* : *exp* → *id list* → *bool*, die testet, ob alle freien Bezeichner eines Ausdrucks in einer Liste vorkommen.
b) Schreiben Sie eine Prozedur *free* : *exp* → *id list*, die zu einem Ausdruck eine Liste liefert, die die in diesem Ausdruck frei auftretenden Bezeichner enthält. Die Liste darf denselben Bezeichner mehrfach enthalten. Verwenden Sie eine Hilfsprozedur *free′* : *id list* → *exp* → *id list*, die nur die frei auftretenden Bezeichner liefert, die nicht in einer Liste von "gebundenen" Bezeichnern enthalten sind.

Aufgabe 12.5 Geben Sie Typumgebungen an, für die der Ausdruck *if true then x else y* zulässig beziehungsweise unzulässig ist.

Aufgabe 12.6 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 12.7 Deklarieren Sie mithilfe der Prozedur *elab* eine Prozedur *test* : *exp* → *bool*, die testet, ob ein Ausdruck geschlossen und zulässig ist.

Aufgabe 12.9 Die Prozedur *elab* kann durch Werfen einer Ausnahme *Error s* einen Fehler *s* melden. Geben Sie möglichst einfache Ausdrücke an, für die die Prozedur *elab empty* die Fehler "*T Opr*", "*T If1*", "*T If2*", "*T App1*" und "*T App2*" meldet.

Aufgabe 12.11 Sei e eine Darstellung des Ausdrucks $fn\, x : int \Rightarrow y$. Überlegen Sie sich, welche Ergebnisse die folgenden Aufrufe von *elab* und *eval* liefern.

a) *elab empty e*
b) *eval empty e*
c) *eval empty* $(App(e,\ Con(IC\ 7)))$

Aufgabe 12.13 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 12.14 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 12.16 (Paare) Wir wollen F um Paare erweitern. Die abstrakte Syntax und die Menge der Werte erweitern wir wie folgt:

$$t \in Ty = \cdots \mid t * t$$

$$e \in Exp = \cdots \mid (e, e) \mid fst\ e \mid snd\ e$$

$$v \in Val = \mathbb{Z} \cup Pro \cup (Val \times Val)$$

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.