11. Übungsblatt zu Programmierung

Prof. Gert Smolka, Thorsten Brunklaus
www.ps.uni-sb.de/courses/prog-ws00/

Abgabe: 2. Februar 2001 vor der Vorlesung (teils per Email und teils auf Papier)

Allgemeine Hinweise: Die Übungsblätter sollen in Zweiergruppen bearbeitet werden. Die Lösungen der Aufgaben 11.1(b) und 11.2 geben Sie bitte auf Papier ab. Alle anderen Lösungen schicken Sie bitte bis Freitag vor der Vorlesung per Email an Ihren Übungsgruppenleiter. Jede Gruppe soll nur eine Lösung einreichen, versehen mit den Namen und den Matrikelnummern der Gruppenmitglieder.

Aufgabe 11.1: F/FR Ausdrücke (1+1+1+1)

(a) Geben Sie einen Ausdruck von F an, der weder reduzierbar noch kanonisch ist.

(b) Stellen Sie die lexikalischen Bindungen der FR Ausdrücke

```
rec f (n:int) : int => if n<=1 then 1 else n * f (n-1)

rec x (x:int) : int->int => fn y:int => z x y
```

durch Überstreichen und Indizieren von Bezeichnerauftreten dar.

(c) Beschreiben Sie die zwei obigen FR Ausdrücke durch zwei Standard ML Ausdrücke des Typs `exp`.

(d) Geben Sie den Ausdruck an, den die folgende Substitution liefert:

$(x\ 5) + (x\ 3)\,[\texttt{fn}\ x : \texttt{int} \Rightarrow x\ /\ x]$

Aufgabe 11.2: Erweiterung von F um Andalso (4+4+4) Sei die abstrakte Syntax von F um Ausdrücke der Form

e_1 `andalso` e_2

erweitert. Diese Ausdrücke sollen in F dieselbe Semantik wie in Standard ML haben. Beachten Sie, dass e_2 nur dann ausgewertet wird, wenn e_1 zu `true` auswertet.

(a) Geben Sie die für die statische Semantik zusätzlich erforderlichen Inferenzregeln an.

(b) Geben Sie die für die dynamische Semantik zusätzlich erforderlichen Inferenzregeln an.

(c) Geben Sie die für die Einschritt-Semantik zusätzlich erforderlichen Inferenzregeln an.

Aufgabe 11.3: Freie Bezeichner (6) Schreiben Sie eine Prozedur

```
freeIds : exp -> id list
```

die zu einem Ausdruck eine Liste der frei auftretenden Bezeichner liefert. Die Liste darf denselben Bezeichner mehrfach enthalten. Verwenden Sie eine Hilfsprozedur

```
freeIds' : id list -> exp -> id list
```

die nur die frei auftretenden Bezeichner liefert, die nicht in einer Liste von gebundenen Bezeichnern enthalten sind. Legen Sie die Ausdrücke von FR zugrunde.

Aufgabe 11.4: F mit Rekursion (9) Sie sollen die Implementierung der statischen und der dynamischen Sematik von F um rekursive Abstraktionen erweitern. Geben Sie die entsprechenden Erweiterungen der Prozeduren `elab` und `eval` an.

Aufgabe 11.5: Einschritt-Semantik (3+4+4+4+4) Sie sollen die Einschritt-Semantik für FR implementieren. Schreiben Sie dazu Prozeduren

```
canonical : exp -> bool
subst     : exp -> exp -> id -> exp
reduceOp  : con -> ops -> con -> exp  (* Error *)
reduce    : exp -> exp  (* Error *)
normalize : exp -> exp  (* Error *)
```

wie folgt:

(a) `canonical` testet, ob ein Ausdruck kanonisch ist.

(b) `subst` liefert zu e, e' und x den Ausdruck $e\,[e'/x]$.

(c) `reduceOp` liefert zu c_1, o und c_2 einen kanonischen Ausdruck, der das Ergebnis der Operationsanwendung $c_1\ o\ c_2$ beschreibt.

(d) `reduce` versucht, einen Ausdruck einmal mit $\rightarrow$ zu reduzieren. Wenn der Ausdruck nicht reduzierbar ist, wird die Ausnahme `Error` geworfen.

(e) `normalize` versucht, die Normalform eines Ausdrucks zu berechnen. Wenn keine Normalform existiert, aber der Ausdruck nur endlich oft reduziert werden kann, wird die Ausnahme `Error` geworfen.

Aufgabe 11.6: Challenge: Fakultät ohne Rekursion Geben Sie in F einen geschlossenen Ausdruck *fak* an, der zu einer Prozedur evaluiert, die die Fakultätsfunktion berechnet. Sie sollen dabei keine rekursive Abstraktion benutzen. Stattdessen soll *fak* Rekursion mithilfe von Selbstapplikation $x\ x$ simulieren. Daher wird es keinen Typ t mit $\emptyset \vdash \mathit{fak} \Rightarrow t$ geben. Beginnen Sie mit dem Ausdruck

```
fn f:int->int => fn n:int =>
  if n<=1 then 1 else n * f (n-1)
```

Überzeugen Sie sich davon, dass Ihr Ausdruck *fak* tut was er soll, indem Sie *fak n* für einige $n \in \mathbb{N}$ mit `eval` und `normalize` (siehe Aufgabe 11.5) auswerten.