Institut für Informatik
der Universität München
Prof. Dr. M. Hofmann
Dr. J. Johannsen, Dr. M. Lange

WS 03/04

20.11.03

Übungen zur Vorlesung
Informatik I

Blatt 5

Abgabe der Hausaufgaben spätestens am 1.12.03, 11:00 Uhr. Programmieraufgaben über
`http://miles.tcs.informatik.uni-muenchen.de/info1/abgabe.php`,
schriftliche Aufgaben auf Papier im Briefkasten in der Theresienstraße 39, 1. Stock. Notieren Sie Namen, Matrikelnummern und Ihre Übungsgruppe auf den Blättern. Bearbeitung in Gruppen zu max. 3 Personen ist zulässig. Besprechung der Aufgaben in den Übungen ab 8.12.03.

Schriftliche Aufgabe S-21: **5 Punkte**

Bestimmen Sie, welche der folgenden Zeichenketten syntaktisch korrekte Ocaml-Ausdrücke sind, und geben Sie eine kurze Begründung an bei den Fällen, wo Sie der Meinung sind, sie seien nicht korrekt.

a) `2 + let f = function x -> x * let y = 2 in g y + 1 in`
   `f 1 + 3 ;;`

b) `if a>10 then let y = 2*a else let y = a*a in f (y+3) ;;`

c) `let x = let y = f 0 and z = g 1 in g y <= f z ;;`

d) `function a -> let x = a*a in let z = x+a in x*z-3*z*z`
   `and y = a*a*a in y-13 ;;`

e) `let y = let rec x n = x 0 in x ( x 1 ) in y*y/2 ;;`

Hinweis: natürlich könnten Sie den Ocaml-Interpreter die Syntax überprüfen lassen, aber der wird diese Ausdrücke nicht ohne Weiteres akzeptieren, da sie freie Variable enthalten. Zum besseren Verständnis sollten Sie die Korrektheit selbst manuell überprüfen, dazu genügt das Fragment der Syntaxdefinition auf den Vorlesungsfolien, Seite 131.

Schriftliche Aufgabe S-22: **8 Punkte**

Betrachten Sie die folgende Grammatik in BNF.

$$
\begin{array}{rcl}
\langle \mathit{Natexpr} \rangle & ::= & \mathtt{0} \mid \mathtt{1} \mid \langle \mathit{Variable} \rangle \mid (\,\langle \mathit{Natexpr} \rangle + \langle \mathit{Natexpr} \rangle\,) \mid \langle \mathit{Function} \rangle\ \langle \mathit{Natexpr} \rangle \\
& & \mid \mathtt{if}\ \langle \mathit{Boolexpr} \rangle\ \mathtt{then}\ \langle \mathit{Natexpr} \rangle\ \mathtt{else}\ \langle \mathit{Natexpr} \rangle \\
& & \mid \mathtt{let}\ \langle \mathit{Variable} \rangle\ =\ \langle \mathit{Natexpr} \rangle\ \mathtt{in}\ \langle \mathit{Natexpr} \rangle \\
\langle \mathit{Boolexpr} \rangle & ::= & (\,\langle \mathit{Natexpr} \rangle = \langle \mathit{Natexpr} \rangle\,) \mid \mathtt{not}\ \langle \mathit{Boolexpr} \rangle \\
\langle \mathit{Function} \rangle & ::= & \mathtt{function}\ \langle \mathit{Variable} \rangle\ .\ \langle \mathit{Natexpr} \rangle \\
\langle \mathit{Variable} \rangle & ::= & \{\mathtt{a} \mid \ldots \mid \mathtt{z}\}^{+}
\end{array}
$$

Geben Sie zu den folgenden Ausdrücken jeweils an, ob sie sich aus $\langle \mathit{Natexpr} \rangle$ ableiten lassen. Falls ja, dann zeichnen Sie einen Ableitungsbaum. Falls nein, dann geben Sie alle syntaktischen Fehler in dem jeweiligen Ausdruck an.

– bitte wenden –

a) `let zwei = (eins+eins) in function x.x (1+1) + zwei`

b) `1 + function x . let 2 = (1+1) in (2+x)`

c) `let q = function n.(n+n) function m.(m+m) (1+1) in (q+q)`

d) `function f. function n. if not not (n=0) then 0 else f n (n+1)`

Programmieraufgabe P-23 (`landkarte.ml`): **7 Punkte**

Wir stellen uns die Welt zwei-dimensional und unterteilt in Parzellen vor (so wie das Schachbrett auf dem ersten Übungsblatt). Parzellen lassen sich durch Werte vom Typ $\mathbb{N} \times \mathbb{N}$ modellieren. Eine Parzelle ist entweder belegt durch eine *Wüste* (W), einen *Berg* (B) oder einen *See* (S). Der untere Rand ist belegt durch WBSWBSWBS... (nach rechts), der linke Rand ist belegt durch WSBWSBWSB... (nach oben). Die Belegung einer Parzelle (n, m), die nicht am Rand liegt, hängt von den Parzellen links daneben $(n-1, m)$, darunter $(n, m-1)$ und schräg links darunter $(n-1, m-1)$ ab:

- Sind alle drei gleich belegt, dann ist (n, m) mit einer Wüste belegt.
- Sind alle drei verschieden belegt, dann wächst auf (n, m) ein Berg.
- Sind genau zwei dieser drei Nachbarparzellen gleich belegt, dann befindet sich auf (n, m) ein See.

Die Welt bis zur Parzelle $(5, 5)$ sieht somit so aus:

```
 ⋮
BSSSBS
SBSWBS
WBSSBS ...
BSSWBS
SBSSBS
WBSWBS
```

Schreiben Sie drei Funktionen `wueste`, `berg` und `see`, die jeweils vom Typ `int * int -> bool` sind und zu einer Parzelle (n, m) angeben, ob diese mit einer Wüste, einem Berg bzw. einem See belegt ist. Also müssen z.B. `berg(1,1)` und `see(2,1)=true` sein, aber `see(1,1)=false`. Halten Sie sich bitte unbedingt an die angegebenen Funktionennamen und -typen!

Hinweis: Für die Definition eines Sees bietet es sich an, eine Hilfsfunktion `f` vom Typ `('a -> bool) * 'a * 'a *'a -> bool` zu schreiben, welche testet, ob eine gegebene Funktion auf genau zwei von drei gegebenen Werten zu `true` ausgewertet wird. Testet also z.B. `u` vom Typ `int->bool`, ob eine Zahl ungerade ist, so ist `f(u,2,1,4) = false`, aber `f(u,3,1,2) = true`.