Klausur Programmierung WS 2002/03

Prof. Dr. Gert Smolka, Dipl. Inf. Thorsten Brunklaus

14. Dezember 2002

Leo Schlau	45
Vor- und Nachname	Sitz-Nr.
4711	007
Matrikelnummer	Code

- Bitte öffnen Sie das Klausurheft erst dann, wenn Sie dazu aufgefordert werden (gleiche Bearbeitungszeit für alle).
- Sie können die Klausur nur auf dem für Sie vorgesehen Platz mitschreiben. Sie müssen das mit Ihrem Namen und Ihrer Matrikelnummer versehene Klausurheft verwenden.
- Hilfsmittel sind nicht zugelassen. Am Arbeitsplatz dürfen nur Schreibgeräte, Getränke, Speisen sowie Ausweise mitgeführt werden. Taschen, Jacken und Mäntel müssen an den Wänden des Klausursaals zurückgelassen werden.
- Verlassen des Saals ohne vorherige Abgabe des Klausurhefts gilt als Täuschungsversuch.
- Wenn Sie während der Bearbeitung auf die Toilette müssen, geben Sie bitte Ihr Klausurheft bei der Aufsicht ab. Es kann immer nur eine Person auf die Toilette.
- Alle Lösungen müssen auf den bedruckten rechten Seiten des Klausurhefts notiert werden. Die leeren linken Seiten dienen als Platz für Skizzen und werden **nicht korrigiert**. Notizpapier ist nicht zugelassen. Sie können mit Bleistift schreiben.
- Für die Bearbeitung der Klausur stehen 150 Minuten zur Verfügung. Insgesamt sind 150 Punkte erreichbar. Die für jede Aufgabe angebene Punktzahl gibt Ihnen also einen Anhaltspunkt, wieviel Zeit Sie auf die Bearbeitung der Aufgabe verwenden sollten. Zum Bestehen der Klausur genügen 75 Punkte.
- Bitte legen Sie Ihren Personalausweis oder Reisepass sowie Ihren Studierendenausweis zur Identifikation neben sich.
- Viel Erfolg!

1	2	3	4	5	6	7	$\sum$	Note
14	12	22	40	12	24	26	150	

Aufgabe 1: Bindungs- und Typanalyse (14 = 4 + 4 + 6)

(a) x, h, f, g

(b) (i) 10

(ii) $\forall\alpha, \beta, \gamma, \delta, \eta\ \ (\alpha \to \beta \to (\alpha \to \gamma) \to (\beta \to \delta) \to (\gamma \to \delta \to \eta) \to \eta)$

Aufgabe 2: Falten von Listen (12 = 2 ∗ 6)

(a)
```
fun p' f s g xs = foldr (fn (x, a) => f(g x, a)) s xs
```

(b)
```
fun q' f g xs = foldl (fn (x, a) => f(g x):: a) nil xs
```

Aufgabe 3: Sortieren durch Mischen (22 = 3 ∗ 6 + 4)

(a)
```
fun split xs = foldl (fn (x, (ys, zs)) => (zs, x::ys)) ([],[]) xs
```

(b)
```
fun merge f [] ys           = ys
  | merge f xs []           = xs
  | merge f (x::xr) (y::yr) =
 if (f(x,y) = LESS) then x::(merge f xr (y::yr))
 else y::(merge f (x::xr) yr)
```

(c)
```
fun sort f []  = []
  | sort f [x] = [x]
  | sort f xs  = let
                   val (ys,zs) = split xs
                 in
                   merge f (sort f ys) (sort f zs)
                 end
```

(d) $\Theta(n \log n)$, wobei n die Länge der Eingabeliste ist.

Aufgabe 4: Arithmetische Ausdrücke (40 = 2 ∗ 5 + 8 + 2 ∗ 6 + 10)

(a)
```
fun eval (C c)      = c
  | eval (V _)      = raise Var
  | eval (A(e,e')) = eval e + eval e'
  | eval (M(e,e')) = eval e * eval e'
```

(b)
```
fun var e = (eval e; false) handle Var => true
```

(c)
```
fun vars (C _)      = []
  | vars (V x)      = [x]
  | vars (A(e,e')) = vars e @ vars e'
  | vars (M(e,e')) = vars e @ vars e'
```

(d)
```
fun adjoin env x z y = if y=x then z else env y
```

(e)
```
fun eval' env (C c)      = c
  | eval' env (V x)      = env x
  | eval' env (A(e,e')) = eval' env e + eval' env e'
  | eval' env (M(e,e')) = eval' env e * eval' env e'
```

(f)
```
fun subexp e []                = e
  | subexp (A(e,e')) (1::ns) = subexp e ns
  | subexp (A(e,e')) (2::ns) = subexp e' ns
  | subexp (M(e,e')) (1::ns) = subexp e ns
  | subexp (M(e,e')) (2::ns) = subexp e' ns
  | subexp _ _               = raise Subscript
```

Aufgabe 5: Grenze von Bäumen mit Akkumulatorargument ($12 = 2 * 6$)

(a)
```
fun frontier' (T(x,[])) ys  = x::ys
  | frontier' (T(_, ts)) ys =
  foldr (fn (t, xs) => frontier' t xs) ys ts
```

(b)
```
fun frontier'' (T(x,[])) ys  = x::ys
  | frontier'' (T(_, ts)) ys =
  foldl (fn (t, xs) => frontier'' t xs) ys ts
```

Aufgabe 6: Iteration und Induktion ($24 = 4 + 9 + 7 + 4$)

(a) $count'(nil, y, a) = a$

$count'(x::xr, y, a) = count'(xr, y,\ a + (\text{if } x = y \text{ then } 1 \text{ else } 0))$

(b) Beweis durch strukturelle Induktion über $xs \in \mathcal{L}(X)$. Sei $y \in X$ und $a \in \mathbb{Z}$. Wir unterscheiden drei Fälle.

$xs = nil$. Dann $count'(xs, a, y) = a = count(xs, y) + a$ mit den Definitionen von $count'$ und $count$.

$xs = x::xr$ und $x = y$. Dann:

$$\begin{aligned} count'(xs, y, a) &= count'(xr, y, a + 1) && \text{Definition } count' \\ &= count(xr, y) + a + 1 && \text{Induktionsannahme } (xr \prec xs) \\ &= count(xs, y) + a && \text{Definition } count. \end{aligned}$$

$xs = x::xr$ und $x \neq y$. Dann:

$$\begin{aligned} count'(xs, y, a) &= count'(xr, y, a + 1) && \text{Definition } count' \\ &= count(xr, y) + a + 1 && \text{Induktionsannahme } (xr \prec xs) \\ &= count(xs, y) + a && \text{Definition } count. \end{aligned}$$

(c) Grundmenge: $\mathcal{L}(X)$

Aussagemenge: $\{ xs \in \mathcal{L}(X) \mid \forall y \in X\ \forall a \in \mathbb{Z}\colon\ count'(xs, y, a) = count(xs, y) + a \}$

Induktionsrelation: $SO(\mathcal{L}(X))$

Induktionsannahme für $zs \in \mathcal{L}(X)$:
$\forall xs \in \mathcal{L}(X)\colon\ xs \prec zs \ \Rightarrow\ (\forall y \in X\ \forall a \in \mathbb{Z}\colon\ count'(xs, y, a) = count(xs, y) + a)$

(d) `fun count'' xs x = foldl (fn (y, a)=> a+(if x=y then 1 else 0)) 0 xs`

Aufgabe 7: Laufzeiten $(26 = 3 * 2 + 2 * (4 + 4 + 2))$

(a) $f(n) = \frac{n}{2}(n+1)$

(b) $\phi_f(n) = n + 1$

(c) $\Theta(n)$

(d) $g(n) = n\ (n+1)$

(e) $\phi_g(n) \ = \ (\sum_{i=0}^{n+2} i) - 2 \ = \ \dfrac{(n+2)(n+3)}{2} - 2 \ = \ \dfrac{n^2}{2} + \dfrac{5}{2}n + 1$

(f) $\Theta(n^2)$

(g) $h(n) = 2^n$

(h) $\phi_h(n) = 2^{n+1} - 1$

(i) $\Theta(2^n)$