

## Übungen zu Informatik I

### Aufgabe 13-1

### Quicksort

(keine Abgabe)

```
a) fun filter pred nil = nil
    | filter pred (x::xs) = if (pred x) then x::(filter pred xs)
                          else filter pred xs;;
fun append3 nil y ys = y::ys
  | append3 (x::xs) y ys = x::(append3 xs y ys);;
fun quicksort l =
  if null l orelse null(tl l) then l                (* (1) *)
  else let val l1 = filter (fn x => x <= (hd l)) (tl l)
          val l2 = filter (fn x => x > (hd l)) (tl l)   (* (2) *)
          val l1' = (quicksort l1)
          val l2' = (quicksort l2)                   (* (3) *)
          in append3 l1' (hd l) l2' end;;           (* (4) *)

quicksort [];
quicksort [1,2,3,4,5,6,7];
quicksort [4,7,2,3,1,5,6];
quicksort [1,2,3,1,2,3,1,2,3];
```

- b) Induktion (mit Werteverlaufsrekursion) nach der Länge der Eingabe. Sei  $n \in \mathbb{N}_0$ . Induktionsvoraussetzung: *quicksort* terminiert für alle Eingaben der Länge kleiner  $n$ . Zu zeigen: *quicksort* terminiert für Eingaben der Länge  $n$ . Falls  $n$  gleich 0 oder 1 ist, terminiert *quicksort* in Schritt (1). Sei  $n > 1$ . Dann wird Schritt (2) ausgeführt. Die Längen von  $l_1$  und  $l_2$  sind beide kleiner  $n$ , somit terminieren die rekursiven Aufrufe nach Induktionsvoraussetzung; damit terminiert auch *quicksort*.
- c) Induktion (mit Werteverlaufsrekursion) nach der Länge der Eingabliste. Sei  $n \in \mathbb{N}_0$ . Induktionsvoraussetzung: Das Ergebnis von *quicksort* ist für alle Eingaben mit Länge kleiner  $n$  aufsteigend sortiert. Zu zeigen: Das Ergebnis von *quicksort* ist für alle Eingaben der Länge  $n$  aufsteigend sortiert. Falls  $n = 0$  oder  $n = 1$  ist das klar. Sei  $n > 1$ . Dann wird Schritt (2) ausgeführt. Nach Induktionsvoraussetzung sind  $l_1$  und  $l_2$  aufsteigens sortiert. Nach Definition von  $l_1$  sind alle Elemente von  $l_1$  kleiner oder gleich  $hd(l)$ ; ebenso sind alle Elemente von  $l_2$  größer als  $hd(l)$ . Damit ist auch *append3*  $l_1$   $hd(l)$   $l_2$  aufsteigend sortiert.
- d) Sei  $l$  eine aufsteigend sortierte Liste der Länge  $n > 1$  mit lauter verschiedenen Elementen. Falls die Funktion *pred* in konstanter Zeit berechnet werden kann hat *filter* asymptotische Komplexität  $O(n)$ , da die Eingabe einmal durchlaufen wird. Ebenso hat *append3* asymptotische Komplexität  $O(n_1)$ , wobei  $n_1$  die Länge des ersten Arguments ist.

Sei  $T(n)$ , die Anzahl der Schritte, die zur Berechnung von *quicksort* für eine sortierte Liste der Länge  $n$  benötigt werden. Dann gilt für  $n > 1$ :

$$\begin{aligned}T(n) &= c_1 + 2c_2n + T(0) + T(n-1) + c_3 \\T(1) &= c_1 \\T(0) &= c_1 \quad .\end{aligned}$$

Dabei kommt der Faktor  $c_1$  von der Berechnung des Tests,  $2c_2n$  ist die zur Berechnung der Listen  $l_1$  und  $l_2$  benötigte Anzahl von Schritten. Da die Eingabeliste sortiert ist, ist  $l_1$  leer,  $l_2$  hat die Länge  $n-1$ , also benötigt man  $T(0)$  Schritte zur Berechnung von  $l_1'$ ,  $T(n-1)$  zur Berechnung von  $l_2'$ . Zur Konkatenation des Ergebnisses benötigt man  $c_3$  Schritte, da das erste Argument immer eine leere Liste ist. Damit erhält man

$$\begin{aligned}T(n) &= \sum_{i=0}^n (2c_2i + (c_1 + T(0) + c_3)) \\&= 2c_2(n + (n-1) + \dots + 1) + (n+1)(2c_1 + c_3) \\&= 2c_2 \cdot \frac{(n+1)n}{2} + (n+1)(2c_1 + c_3) \\&= O(n^2) \quad .\end{aligned}$$

Für sortierte Eingabelisten hat *quicksort* also quadratische Komplexität.

- e) Sei  $T(n)$  die Anzahl der Schritte, die zur Berechnung von *quicksort* benötigt werden, falls die Eingabe  $l$  den Bedingungen der Aufgabe genügt und Länge  $n$  hat. Dann erhält man für  $n > 1$  die Abschätzung

$$\begin{aligned} T(n) &\approx c_1 + 2c_2n + 2T((n-1)/2) + c_3n \\ &\approx c'(n) + 2T((n-1)/2) \\ &\approx \sum_{i=0}^{\lfloor \log n \rfloor} c'2^i \\ &< \sum_{i=0}^{\lfloor \log n \rfloor} c'n \\ &= O(n \log n) \quad . \end{aligned}$$

### Aufgabe 13-2

### Folgen als cpos

(keine Abgabe)

Sei  $A$  eine Menge.

- a) Wir zeigen nur, dass  $(A^\infty, \sqsubseteq)$  eine partielle Ordnung ist.

- (Reflexivität) klar.
- (Transitivität) Seien  $a = (a_n)_{n < k}$ ,  $b = (b_n)_{n < l}$  und  $c = (c_n)_{n < m}$  Folgen aus  $A^\infty$  mit  $a \sqsubseteq b$  und  $b \sqsubseteq c$ . Zu zeigen:  $a \sqsubseteq c$ .  
Wegen  $a \sqsubseteq b$  und  $b \sqsubseteq c$  gilt  $k \leq l$  und  $l \leq m$ , also auch  $k \leq m$ . Weiter gilt  $a_i = b_i$  für alle  $i < k$  und  $b_i = c_i$  für alle  $i < m$ . Da  $k \leq m$  ist, gilt also  $a_i = c_i$  für alle  $i < k$ , d. h.  $a \sqsubseteq c$ .
- (Antisymmetrie) Seien  $a = (a_n)_{n < k}$  und  $b = (b_n)_{n < l}$  Folgen aus  $A^\infty$  mit  $a \sqsubseteq b$  und  $b \sqsubseteq a$ . Zu zeigen:  $a = b$ .  
Wegen  $a \sqsubseteq b$  und  $b \sqsubseteq a$  gilt  $k \leq l$  und  $l \leq k$ , also  $k = l$ . Ausserdem gilt  $a_i = b_i$  für alle  $i < k$  ( $= l$ ), also  $(a_n)_{n < k} = (b_n)_{n < l}$ .

- b) *Bem:* für  $B \subseteq A^\infty$  gilt:  $B$  gerichtet  $\Leftrightarrow B$  Kette. (gilt nicht allgemein!)

- $A^*$  ist nicht vollständig für  $A \neq \emptyset$ :  
Sei  $x \in A$  beliebig und  $B := \{(a_n)_{n < k} \mid k \in \mathbb{N}_0, a_n = x \text{ für alle } n \in \mathbb{N}_0\} \subset A^*$ . Offenbar ist  $B$  eine Kette, also insbesondere gerichtet. Sie hat aber kein Supremum in  $A^*$ , da die Länge einer oberen Schranke unendlich sein müsste und  $A^*$  besteht aus endlichen Folgen.
- $A^\infty$  ist vollständig:  
Sei  $B \subseteq A^\infty$  gerichtet, also nach obiger Bemerkung eine Kette. Wir unterscheiden drei Fälle:  
Fall 1: es gibt  $a \in B$  mit maximaler, endlicher Länge, d. h.  $\text{lg}(a) = k \in \mathbb{N}_0$  und für alle  $b \in B$  gilt:  $\text{lg}(b) \leq k$ . Dann gilt  $a = \sup B$ . Denn sei  $b \in B$  beliebig. Da  $B$  eine Kette ist, gilt  $a \sqsubseteq b$  oder  $b \sqsubseteq a$ . Da  $\text{lg}(b) \leq \text{lg}(a)$  ist, muss  $b \sqsubseteq a$  gelten. Wegen  $a \in B$  ist  $a$  damit die kleinste obere Schranke von  $B$ .  
Fall 2: alle Folgen in  $B$  haben endliche Länge, aber es gibt keine mit maximaler Länge, d. h. für alle  $m \in \mathbb{N}_0$  gibt es  $a^{(m)} \in B$  mit  $\text{lg}(a^{(m)}) = n_m > m$ . Da  $B$  eine Kette ist, (vgl. *Bem.*), muss für alle  $(b_n)_{n < k} \in B$  und  $(c_n)_{n < l} \in B$  gelten:  $b_i = c_i$  für alle  $i < \min(k, l)$ . Wir nennen dieses Element  $a_i$ . Sei nun  $a := (a_n)_{n < \infty}$ . Dann gilt offenbar  $b \sqsubseteq a$  für alle  $b \in B$ . Man sieht leicht, dass  $a$  die kleinste obere Schranke von  $B$  ist.  
Fall 3: es gibt  $a \in B$  mit  $\text{lg}(a) = \infty$ . Dadurch ist  $a$  eindeutig bestimmt. Dieses  $a$  ist gerade das Supremum von  $B$ .

- c) *Bem:* für alle  $a, b \in A^\infty$  gilt:  $p_x(a) \sqsubseteq p_x(b) \Leftrightarrow a \sqsubseteq b$ .

Sei  $B \subseteq A^\infty$  eine Kette. Es ist folgendes zu zeigen:

- 1): es existiert  $\sup p_x(B)$  in  $A^\infty$
- 2): für alle  $b \in p_x(B)$ :  $b \sqsubseteq p_x(\sup B)$
- 3):  $b \sqsubseteq c$  für alle  $b \in p_x(B) \Rightarrow p_x(\sup B) \sqsubseteq c$ .

Zu 1): Aus der Bemerkung oben folgt, dass mit  $B$  auch  $p_x(B)$  eine Kette in  $A^\infty$  ist. Da  $A^\infty$  vollständig ist, existiert  $\sup p_x(B)$  in  $A^\infty$ .

Zu 2): Sei  $b \in p_x(B)$ . Dann gibt es  $b' \in B$  mit  $b = p_x(b')$ . Wegen  $b' \sqsubseteq \sup B$  folgt die Behauptung aus der obigen Bemerkung.

Zu 3): Sei  $c \in A^\infty$  so, dass für alle  $b \in p_x(B)$  gilt:  $b \sqsubseteq c$ . Dann gibt es  $c' \in A^\infty$  mit  $p_x(c') = c$ . Nach der obigen Bemerkung gilt  $b' \sqsubseteq c'$  für alle  $b' \in B$ . Folglich ist  $c'$  eine obere Schranke von  $B$  und damit gilt  $\sup B \sqsubseteq c'$ . Nach der Bemerkung von oben gilt also  $p_x(\sup B) \sqsubseteq p_x(c') = c$ .

- d) Argumentation im wesentlichen wie bei c).

e) Mit c) und d) genügt es folgendes zu zeigen:

$$f : A \rightarrow B, g : B \rightarrow C \text{ stetig} \Rightarrow g \circ f : A \rightarrow C, x \mapsto g(f(x)) \text{ stetig}$$

Beweis: Sei  $K \subseteq A$  eine Kette. Da  $f$  stetig ist, existiert  $\sup f(K)$  und es gilt:  $f(\sup K) = \sup f(K)$ . Ausserdem ist  $f$  monoton (folgt aus der Stetigkeit) und damit ist  $f(K)$  eine Kette in  $B$ . Also existiert wegen der Stetigkeit von  $g$  auch  $\sup g(f(K))$  und es gilt  $\sup g(f(K)) = g(\sup f(K))$ , also folgt insgesamt, dass  $\sup g(f(K)) = g(\sup f(K))$  gilt. Damit ist die Stetigkeit von  $g \circ f$  gezeigt.

f) Wir zeigen folgendes:  $f((a_n)_{n < k}) = (a_n)_{n < k} \Rightarrow k = \infty$  und  $a_i = 2^i$  für alle  $i \in \mathbb{N}_0$ .

Sei also  $a = (a_n)_{n < k}$  ein Fixpunkt von  $f$ . Dass  $a$  dann unendliche Länge haben muss ist klar. Wir zeigen also noch folgendes:

Für alle  $i \in \mathbb{N}_0$ :  $a_i = 2^i$ .

Beweis durch Induktion nach  $i$ .

$i = 0$ : Nach der Definition von  $f$  gilt  $f(a)_0 = 1 = 2^0$ . Da  $a$  ein Fixpunkt von  $f$  ist, gilt  $a_0 = f(a)_0 = 2^0$ .

$i \mapsto i + 1$ : Nach der Definition von  $f$  gilt  $f(a)_{i+1} = 2 \cdot a_i$ . Da nach I.V.  $a_i = 2^i$  ist, gilt offenbar  $f(a)_{i+1} = 2^{i+1}$ . Andererseits ist  $a$  ein Fixpunkt von  $f$  und folglich gilt  $a_{i+1} = f(a)_{i+1} = 2^{i+1}$ .

### Aufgabe 13-3

### Permutationen

(3 Punkte)

a)  $O(n^2)$ : Offensichtlich haben *member* und *remove* Komplexität  $O(n)$ . Seien  $l, m$  Listen der Länge  $n$ . Falls  $m$  eine Permutation von  $l$  ist, wird jedes Element von  $l$  während der Abarbeitung betrachtet. Dabei muss für das  $i$ -te Element von  $l$  eine Liste der Länge  $n - i$  mit *member* durchsucht werden; ebenso muss aus einer Liste der Länge  $n - i$  ein Element mit *remove* gelöscht werden. Damit benötigt man für das  $i$ -te Element  $c_1(n - i) + c_2(n - i) = c(n - i)$  Schritte, insgesamt also:

$$\begin{aligned} \sum_{i=0}^n c(n - i) &= c \sum_{i=0}^n (n - i) \\ &= c \sum_{i=0}^n i \\ &= c \frac{n(n + 1)}{2} \\ &= O(n^2) \end{aligned}$$

b) (\* perm2 benötigt quicksort \*)

```
fun perm2 l m =
  let val l' = quicksort l
      val m' = quicksort m
      fun loop nil nil = true
        | loop (x::xs) (y::ys) = x = y andalso loop xs ys
        | loop _ _ = false
      in loop l' m' end
```

### Aufgabe 13-4

### Kleinstes gemeinsames Vielfaches

(3 Punkte)

a) Falls *kgv* nicht schon vorher terminiert hat, wird nach  $(m - 1)n$  Schritten *loop(mn)* aufgerufen, und das Programm terminiert.

b) Für voneinander verschiedene Primzahlen  $m, n$  ist  $mn$  das kleinste gemeinsame Vielfache, also hat *kgv* die asymptotische Komplexität  $O(mn \log(m) \log(n))$ .

c) fun euclid m n =

```
  let fun loop k l = if l = 0
                    then k
                    else loop l (k mod l)
      in if m < n then loop n m else loop m n end
  fun kgv2(m,n) = let val ggt = euclid m n
                  val m' = m div ggt
                  val n' = n div ggt
                  in ggt * m' * n' end
```

*Anmerkung:* Im Gegensatz zur Definition der Stetigkeit aus der Vorlesung verlangt man üblicherweise das Erhalten aller Suprema gerichteter Mengen. Die Aussagen 13-5 (b) und 13-5 (c) gelten auch für die übliche Definition der Stetigkeit, was hier jedoch nicht bewiesen wird.

Seien  $(D, \sqsubseteq_D)$  und  $(E, \sqsubseteq_E)$  cpos.

a)  $D \times E$  ist partielle Ordnung:

- (Reflexivität)  $d \sqsubseteq_D d, e \sqsubseteq_E e \implies (d, e) \sqsubseteq_{D \times E} (d, e)$
- (Transitivität) Gelte  $(d_0, e_0) \sqsubseteq_{D \times E} (d_1, e_1)$  und  $(d_1, e_1) \sqsubseteq_{D \times E} (d_2, e_2)$ . Nach Def.  $d_0 \sqsubseteq_D d_1 \sqsubseteq_D d_2$ , also  $d_0 \sqsubseteq d_2$ . Aus dem gleichen Grunde  $e_0 \sqsubseteq_E e_2$ , also  $(d_0, e_0) \sqsubseteq_{D \times E} (d_2, e_2)$ .
- (Antisymmetrie) Gelte  $(d_0, e_0) \sqsubseteq_{D \times E} (d_1, e_1)$  und  $(d_1, e_1) \sqsubseteq_{D \times E} (d_0, e_0)$ . Nach Def.  $d_0 \sqsubseteq_D d_1 \sqsubseteq_D d_0$ , also  $d_0 = d_1$ . Aus dem gleichen Grunde  $e_0 = e_1$ , also  $(d_0, e_0) = (d_1, e_1)$ .

$D \times E$  ist vollständig: Sei  $A \subseteq D \times E$  gerichtet. Setze  $A_D = \{d \in D \mid \exists e \in E. (d, e) \in A\}$  und  $A_E = \{e \in E \mid \exists d \in D. (d, e) \in A\}$ . Dann sind  $A_D$  und  $A_E$  gerichtet(!), folglich existiert  $d = \bigsqcup A_D$  und  $e = \bigsqcup A_E$ . Wir zeigen  $(d, e) = \bigsqcup A$ :

- $(d, e)$  ist obere Schranke von  $A$ : Sei  $(d_0, e_0) \in A$ . Dann  $d_0 \in A_D$  und  $e_0 \in A_E$ , also  $d_0 \sqsubseteq d$  und  $e_0 \sqsubseteq e$  nach Def. von  $d$  und  $e$ . Also  $(d_0, e_0) \sqsubseteq (d, e)$ .
- $(d, e)$  ist kleinste obere Schranke: Sei  $(d_0, e_0)$  obere Schranke von  $A$ . Dann ist  $d_0$  obere Schranke von  $A_D$  und  $e_0$  obere Schranke von  $A_E$  (denn ist zB.  $d_1 \in A_D$ , so existiert  $e_1 \in E$  mit  $(d_1, e_1) \in A$ , somit  $(d_1, e_1) \sqsubseteq (d_0, e_0)$ , folglich  $d_1 \sqsubseteq d_0$ ; analog für  $e_0$ ). Da  $d$  und  $e$  kleinste obere Schranken von  $A_D$  bzw.  $A_E$ , gilt  $d \sqsubseteq d_0$  und  $e \sqsubseteq e_0$ , also  $(d, e) \sqsubseteq (d_0, e_0)$ .

b) Wir zeigen nur die Stetigkeit von  $\pi_1$ , für  $\pi_2$  analog. Sei  $(d_n, e_n)_{n \in \omega}$  Kette in  $D \times E$ . Dann sind  $(d_n)$  und  $(e_n)$  Ketten in  $D$  bzw.  $E$ , also insbesondere gerichtet. Mit  $d = \bigsqcup_{n \in \omega} d_n$  und  $e = \bigsqcup_{n \in \omega} e_n$  gilt also nach Teilaufgabe (a):  $\bigsqcup_{n \in \omega} (d_n, e_n) = (d, e)$ . Also  $\bigsqcup_{n \in \omega} \pi_1(d_n, e_n) = \bigsqcup_{n \in \omega} d_n = d = \pi_1(d, e) = \pi_1(\bigsqcup_{n \in \omega} (d_n, e_n))$ .

c) Sei  $(k_n)_{n \in \omega}$  Kette in  $F$ . Setze  $k = \bigsqcup_{n \in \omega} k_n$ ,  $d = \bigsqcup_{n \in \omega} f(k_n)$  und  $e = \bigsqcup_{n \in \omega} g(k_n)$ . Wegen der Stetigkeit von  $f$  und  $g$  also  $f(k) = d$  und  $g(k) = e$ . Mit Teilaufgabe (a) erhalten wir:  $\bigsqcup_{n \in \omega} \langle f, g \rangle(f_n) = \bigsqcup_{n \in \omega} (f(k_n), g(k_n)) = (\bigsqcup_{n \in \omega} f(k_n), \bigsqcup_{n \in \omega} g(k_n)) = (d, e) = \langle f, g \rangle(k) = \langle f, g \rangle(\bigsqcup_{n \in \omega} k_n)$ .

d) Betrachte zwei Ketten  $(a^n)_{n \in \omega}$  und  $(b^n)_{n \in \omega}$ . Schreibe  $a^n = (a_k^n)_{k < \lg(a^n)}$  und  $b^n = (b_k^n)_{k < \lg(b^n)}$  und setze  $a = (a_k)_{k < \lg(a)} = \bigsqcup_{n \in \omega} a^n$ ,  $b = (b_k)_{k < \lg(b)} = \bigsqcup_{n \in \omega} b^n$ .

Dann  $\bigsqcup_{n \in \omega} a^n \oplus b^n = \bigsqcup_{n \in \omega} (a_k^n + b_k^n)_{k < \min(\lg(a^n), \lg(b^n))} = \bigsqcup_{n \in \omega} (a_k + b_k)_{k < \min(\lg(a^n), \lg(b^n))} = (a_k + b_k)_{k < s}$ , wobei  $s = \sup_{n \in \omega} \min(\lg(a^n), \lg(b^n))$ .

Ausserdem  $\bigsqcup_{n \in \omega} a^n \oplus \bigsqcup_{n \in \omega} b^n = (a_k + b_k)_{k < \min(\lg(a), \lg(b))}$ .

Es verbleibt also zu zeigen:  $m = \min(\lg(a), \lg(b)) = \sup_{n \in \omega} \min(\lg(a^n), \lg(b^n))$ .

- $m$  ist obere Schranke von  $\{\min(\lg(a^n), \lg(b^n)) \mid n \in \omega\}$ : Sei  $n \in \omega$ . Dann  $\lg(a^n) \leq \lg(a)$ ,  $\lg(b^n) \leq \lg(b)$ , also  $\min(\lg(a^n), \lg(b^n)) \leq \min(\lg(a), \lg(b))$ .
- $m$  ist kleinste obere Schranke von  $\{\min(\lg(a^n), \lg(b^n)) \mid n \in \omega\}$ : Sei  $s$  eine weitere obere Schranke. Dann  $s \geq \min(\lg(a^n), \lg(b^n))$  für alle  $s$ , also  $s \geq \sup_{n \in \omega} \min(\lg(a^n), \lg(b^n)) = m$ .

e) Folgt sofort aus (b), (c) und (d) und der Tatsache, dass die Komposition von stetigen Funktionen stetig ist.

f) Sei  $x = (x_n)_{n \in \omega}$  die Folge der Fibonacci-Zahlen. Setze  $y = (y_n)_{n \in \omega}$  mit  $y_n = x_{n+1}$  und  $z = (z_n)_{n \in \omega}$  mit  $z_n = y_{n+1}$ .

Wir betrachten zunächst  $g = f \circ f$  und zeigen folgende Hilfsaussage durch Induktion nach  $n$ : Für alle  $n \geq 1$  gilt

$$g^n \begin{pmatrix} () \\ () \\ () \end{pmatrix} = \begin{pmatrix} (x_0, \dots, x_{n+1}) \\ (y_0, \dots, y_n) \\ (z_0, \dots, z_n) \end{pmatrix}.$$

Dies ist für  $n = 1$  offensichtlich; für  $n > 1$  klar aus der Definition von  $f$ .

Wenn  $x^n, y^n, z^n = f^n((), (), ())$  erhalten wir also

$$x^n = (x_0, \dots, x_k) \quad \text{mit } k = (n + 2) \text{div } 2$$

Nach dem Satz von Kleene gilt  $\bigsqcup_{n \in \omega} (x^n, y^n, z^n)$  is kleinster Fixpunkt von  $f$ , nach Teilaufgabe (a) dürfen wir das Supremum komponentenweise bilden. Aus  $\bigsqcup_{n \in \omega} x^n = x$ , folgt die Behauptung.