Technische Universität München
Fakultät für Informatik
Prof. Dr. A. Knoll

WS 2002/2003
3. April 2003

Lösungsvorschläge der Zwischenklausur zu Einführung in die Informatik I

Aufgabe 1 Markow-Algorithmen (Lösungsvorschlag) ((1+2+2+2) + 3 = 10 Punkte)

a) Gesucht ist ein Textersetzungssystem, das natürliche Zahlen $n \geq 0$ in Binärdarstellung. d.h. Wörter der Form $b_k \dots b_1$ mit $b_i \in \{0, L\}$, $1 \leq i \leq k$ invertiert. Beispiel:

das Wort LLL00L wird zu 000LL0 .

(i) $V = \{O, L, \#\}$.

(ii) $T = \{\#L \to 0\#, \#0 \to L\#, L\# \to .L, 0\# \to .0, \varepsilon \to \#\}$.

(iii)
```
L00LLL00 ==> #L00LLL00   (Regel: "" -> "#")
#L00LLL00 ==> 0#00LLL00  (Regel: "#L" -> "0#")
0#00LLL00 ==> 0L#0LLL00  (Regel: "#0" -> "L#")
0L#0LLL00 ==> 0LL#LLL00  (Regel: "#0" -> "L#")
0LL#LLL00 ==> 0LL0#LL00  (Regel: "#L" -> "0#")
0LL0#LL00 ==> 0LL00#L00  (Regel: "#L" -> "0#")
0LL00#L00 ==> 0LL000#00  (Regel: "#L" -> "0#")
0LL000#00 ==> 0LL000L#0  (Regel: "#0" -> "L#")
0LL000L#0 ==> 0LL000LL#  (Regel: "#0" -> "L#")
0LL000LL# ==> 0LL000LL   (Regel: "L#" ->. "L")
```

(iv) terminierend, deterministisch, determiniert, sequentiell.

b) Geben Sie einen Markow-Algorithmus an, der für Wörter über dem Zeichenvorrat $\{a, b, c\}$ und der Beziehung $B := \{a \leq b \leq c\}$ eine sortierte Ausgabe gemäß der Beziehung B liefert. Beispiel:

Eingabewort = bcaaa, Ausgabewort = aaabc

Lösungsvorschlag:
Zeichenvorrat: $V = \{a, b, c\}$.
Regeln: $R = \{ba \to ab, ca \to ac, cb \to bc\}$.

Aufgabe 2 Ringpuffer in OCaml (1 + 2 + 2 + 2 + 3 = 10 Punkte)

a)
```
type 'a ringbuffer = Ring of (int * int * int * 'a list);;
```

b)
```
let emptyRing n = Ring(0,0,n,[]);;
```

c)
```
let memory ring = match ring with
        | Ring(schreib,lese,n,_) -> n - (schreib + n - lese) mod n
;;
```

d)
```
let rec push_emb pos elem q = match q with
      | []  -> [elem] (* Liste ist noch nicht voll *)
      | hd::tail when pos == 0 -> elem::tail
      | hd::tail -> hd::(push_emb (pos-1) elem tail)
;;

let push elem ring = match ring with
      | Ring(s,l,n,q) when ((s+1)mod n) == l  ->
                        Ring((s+1) mod n, (l+1) mod n, n, push_emb s elem q)
      | Ring(s,l,n,q) -> Ring((s+1) mod n, l, n, push_emb s elem q)
;;
```

e)
```
let rec pop_emb pos q ring = match q with
      | [] -> failwith("Empty Ring")
      | elem::tail when pos == 0 -> (elem, ring)
      | hd::tail -> pop_emb (pos-1) tail ring
;;

let pop ring = match ring with
      | Ring(s,l,n,q) when l == s   -> failwith("Buffer underrun")
      | Ring(s,l,n,q) -> pop_emb l q (Ring(s,(l+1) mod n, n, q))
;;
```

Aufgabe 3 Listen in Prolog (2 + 4 + 4 = 10 Punkte)

a) Eine mögliche Implementierung lautet:

```
sortiert([]).
sortiert([E]).
sortiert([E|[F|Rest]]) :- E=<F, sortiert([F|Rest]).
```

b) Ein Programm, das aus einer Liste `A` ein Element `E` streicht, lautet:

```
streiche(E, [E|Rest], Rest).
streiche(E, [F|Rest], [F|Z]) :- streiche(E, Rest, Z).
```

c) Die Menge aller Permutationen einer Liste `A` läßt sich durch folgendes Programm bestimmen:

```
permutation([], []).
permutation(A, [E|C]) :- streiche(E, A, A2), permutation(A2, C).
```

d) Das gesuchte Prädikat lautet:

```
sortiere(A, B) :- permutation(A, B), sortiert(B).
```

Aufgabe 4 Termersetzung (Lösungsvorschlag) (2+5+3=10 Punkte)

Gegeben sei ein Termersetzungssystem mit folgender rekursiver Definition: ε ist ein Term, und für alle Terme A gilt: $zero(A)$ und $one(A)$ sind auch Terme.

$$\begin{aligned}
xor(zero(A), zero(B)) &\mapsto zero(xor(A,B)) && (1)\\
xor(one(A), zero(B)) &\mapsto one(xor(A,B)) && (2)\\
xor(zero(A), one(B)) &\mapsto one(xor(A,B)) && (3)\\
xor(one(A), one(B)) &\mapsto zero(xor(A,B)) && (4)\\
xor(\varepsilon, A) &\mapsto A && (5)\\
xor(A, \varepsilon) &\mapsto A && (6)\\
zero(\varepsilon) &\mapsto \varepsilon && (7)
\end{aligned}$$

a) Vereinfachen Sie folgenden Term mit den angegebenen Regeln bis zu seiner terminalen Grundform:

$$\begin{array}{lll}
 & & xor(one(one(one(\varepsilon))), xor(one(one(\varepsilon)), one(zero(one(\varepsilon)))))\\
(4) & \mapsto & xor(one(one(one(\varepsilon))), zero(xor(one(\varepsilon), zero(one(\varepsilon)))))\\
(2) & \mapsto & xor(one(one(one(\varepsilon))), zero(one(xor(\varepsilon, one(\varepsilon)))))\\
(5) & \mapsto & xor(one(one(one(\varepsilon))), zero(one(one(\varepsilon))))\\
(2) & \mapsto & one(xor(one(one(\varepsilon)), one(one(\varepsilon))))\\
(4) & \mapsto & one(zero(xor(one(\varepsilon), one(\varepsilon))))\\
(4) & \mapsto & one(zero(zero(xor(\varepsilon, \varepsilon))))\\
(5) & \mapsto & one(zero(zero(\varepsilon)))\\
(7) & \mapsto & one(zero(\varepsilon))\\
(7) & \mapsto & one(\varepsilon)
\end{array}$$

b) Zeigen Sie (induktiv) daß in dem gegebenen Termersetzungssystem für einen beliebigen Term A immer gilt: Die terminale Grundform von $xor(A,A)$ ist ε.

Wir beweisen die Aussage mit Hilfe von Induktion über den Termaufbau:

Induktionsannahme: $xor(X,X) \cong \varepsilon$ für Terme die in n Schritten aufgebaut wurden.

Induktionsanfang: für den Term ε gilt nach Regel (5): $xor(\varepsilon, \varepsilon) = \varepsilon$

Induktionsschritt: Sei A ein Term der in einem Schritt aus X aufgebaut werden kann. Dann ist entweder $A = zero(X)$ oder $A = one(X)$. In beiden Fällen führt die Termersetzung nach Regel (1) oder (4) zu dem selben Ergebnis:

$$\begin{aligned}
xor(A,A) = xor(zero(X), zero(X)) &\mapsto zero(xor(X,X))\\
xor(A,A) = xor(one(X), one(X)) &\mapsto zero(xor(X,X))
\end{aligned}$$

Nach Induktionsannahme ist aber $xor(X,X) \cong \varepsilon$ und damit mit Regel (7) auch

$$zero(xor(X,X)) \mapsto zero(\varepsilon) \cong \varepsilon$$

c) Sei weiter folgende Interpretation des Termersezungssystems gegeben:

$$\begin{aligned}
zero(A) &\mathrel{\hat{=}} (2 \cdot a)\\
one(A) &\mathrel{\hat{=}} (2 \cdot a) + 1\\
\varepsilon &\mathrel{\hat{=}} 0\\
xor(A,B) &\mathrel{\hat{=}} a \oplus b
\end{aligned}$$

Zeigen Sie hiermit daß für ein geradzahliges $g \in \mathbb{N}$ und ein ungeradzahliges $u \in \mathbb{N}$ gilt, daß $g \oplus u$ wieder ungeradzahlig ist.

Mit dieser Interpretation entsprechen die Terme genau der Binärdarstellung einer natürlichen Zahl. Diese Zahl ist genau dann geradzahlig, wenn an der kleinstwertigen Bitstelle *least significant bit*, in unserem Fall also der äußsersten Funktion, das Bit gesetzt ist, also $one(A)$; geradzahlig genau dann wenn hier $zero(A)$ steht.

$$g \oplus u \hat{=} xor(G,U) = xor(zero(G'), one(U')) \mapsto one(xor(G',U'))$$

Dies ist nach selbiger Argumentation wieder ungeradzahlig. qed.