**Technische Universität München**
**Fakultät für Informatik**
**Dr. H. Ehler, Dr. M. Pizka,**
**W. Prenninger, Dr. M. Schneider**

**WS 2001**
**Lösungsvorschläge zu Blatt 3**
**1. November 2001**

# Übungen zu Einführung in die Informatik I

## Aufgabe 8 Subsitution (Lösungsvorschlag)

a) (i) Durch Einsetzen von $t = x \vee y$ ergibt sich folgender Term:

$$(x \vee y) \wedge (y \vee z) \wedge \neg(x \vee z) \tag{1}$$

Wir wählen folgende Belegung $\beta$: $\beta(x) = \mathtt{0}$, $\beta(y) = \mathtt{L}$, $\beta(z) = \mathtt{0}$. Dann gilt:

$$\begin{aligned} & I_\beta[(x \vee y) \wedge (y \vee z) \wedge \neg(x \vee z)] = \\ & = and(and(or(I_\beta[x], I_\beta[y]), or(I_\beta[y], I_\beta[z])), not(or(I_\beta[x], I_\beta[z]))) \\ & = and(and(or(\mathtt{0}, \mathtt{L}), or(\mathtt{L}, \mathtt{0})), not(or(\mathtt{0}, \mathtt{0}))) \\ & = \mathtt{L} \end{aligned}$$

Term (1) ist also erfüllbar.

(ii) Durch Einsetzen von $t = x \Rightarrow y$ ergibt sich folgender Term:

$$s \overset{\text{def}}{=} (x \Rightarrow y) \wedge (y \Rightarrow z) \wedge \neg(x \Rightarrow z) \tag{2}$$

Da die Implikation transitiv ist, vermuten wir sofort, dass der Term $s$ nicht erfüllbar ist. Dies zeigen wir mit der Widerspruchsannahme: *Term $s$ ist erfüllbar, d.h. es gibt eine Belegung $\beta$ mit $I_\beta[s] = \mathtt{L}$.*
Term $s$ ist eine Konjunktion aus den drei Teiltermen $(x \Rightarrow y)$, $(y \Rightarrow z)$ und $\neg(x \Rightarrow z)$.
Also muss für die drei Teilterme gelten: $I_\beta[x \Rightarrow y] = I_\beta[y \Rightarrow z] = I_\beta[\neg(x \Rightarrow z)] = \mathtt{L}$.
Betrachten wir $I_\beta[\neg(x \Rightarrow z)] = \mathtt{L}$, so schließen wir $\beta(x) = \mathtt{L}$ und $\beta(z) = \mathtt{0}$.
Betrachten wir $I_\beta[x \Rightarrow y] = \mathtt{L}$ und $\beta(x) = \mathtt{L}$, so schließen wir $\beta(y) = \mathtt{L}$.
Betrachten wir schließlich $I_\beta[y \Rightarrow z] = \mathtt{L}$, dann ergibt sich ein Widerspruch zu $\beta(y) = \mathtt{L}$ und $\beta(z) = \mathtt{0}$. Term $s$ ist also nicht erfüllbar.
Alternativ können wir mit Hilfe einer Wertetafel prüfen, ob es eine Belegung $\beta$ mit $I_\beta[s] = \mathtt{L}$ gibt.

| $\beta(x)$ | $\beta(y)$ | $I_\beta[\beta(z)]$ | $I_\beta[x \Rightarrow y]$ | $I_\beta[y \Rightarrow z]$ | $I_\beta[\neg(x \Rightarrow z)]$ | $I_\beta[s]$ |
|---|---|---|---|---|---|---|
| L | L | L | L | L | 0 | 0 |
| L | L | 0 | L | 0 | L | 0 |
| L | 0 | L | 0 | L | 0 | 0 |
| L | 0 | 0 | 0 | L | L | 0 |
| 0 | L | L | L | L | 0 | 0 |
| 0 | L | 0 | L | 0 | 0 | 0 |
| 0 | 0 | L | L | L | 0 | 0 |
| 0 | 0 | 0 | L | L | 0 | 0 |

b) Die Aufgabenstellung enthält bereits den Hinweis, dass ein weiterer Identifikator (nämlich $z$) eine Rolle spielt. Die folgenden 3 Substitutionen

$$[z/x] \;\; [x/y] \;\; [y/z]$$

haben, in dieser Reihenfolge angewandt, den gleichen Effekt wie die gegebene Subsitution:

$$((t[z/x])[x/y])[y/z] = t[y/x, x/y].$$

(Vergleiche Übungsbuch Broy/Rumpe, Aufgabe 1.6 S. 57)

## Aufgabe 9 Zeichensequenzen (Lösungsvorschlag)

a) Sei $u$ ein Wort der Länge $n$, $v$ ein Wort der Länge $m$ und $w$ ein Wort der Länge $o$; d.h.

$$\begin{array}{ll} u = <u_1 \dots u_n> & u_i \in C,\ 0 \le i \le n \\ v = <v_1 \dots v_m> & v_i \in C,\ 0 \le i \le m \\ w = <w_1 \dots w_o> & w_i \in C,\ 0 \le i \le o \end{array}$$

Beweisidee (der vollständige Beweis erfordert Induktion über die Länge der Zeichensequenzen!):

$$\begin{array}{lll} (u \circ v) \circ w = & & \\ & = (<u_1 \dots u_n> \circ <v_1 \dots v_m>) \circ w & \text{Def. } u, v \\ & = (<u_1 \dots u_n v_1 \dots v_m>) \circ w & \text{Def. Konkatenation} \\ & = <u_1 \dots u_n v_1 \dots v_m> \circ w & \text{Klammer auswerten} \\ & = <u_1 \dots u_n v_1 \dots v_m> \circ <w_1 \dots w_o> & \text{Def. } w \\ & = <u_1 \dots u_n v_1 \dots v_m w_1 \dots w_o> & \text{Def. Konkatenation} \\ & = <u_1 \dots u_n> \circ <v_1 \dots v_m w_1 \dots w_o> & \text{Def. Konkatenation} \\ & = u \circ <v_1 \dots v_m w_1 \dots w_o> & \text{Def. } u \\ & = u \circ (<v_1 \dots v_m w_1 \dots w_o>) & \\ & = u \circ (<v_1 \dots v_m> \circ <w_1 \dots w_o>) & \\ & = u \circ (v \circ w) & q.e.d \end{array}$$

b) (i) Die gesuchten Elemente aus $H^*$ sind: $\varepsilon$, $<a>$, $<aa>$, $<ab>$, $<ac>$, $<b>$, $<ba>$, $<bb>$, $<bc>$, $<c>$, $<ca>$, $<cb>$ und $<cc>$.

(ii) $H^2$ ist die Menge der geordneten Paare aus $H$. Elemente von $H^2$ in lexikographischer Ordnung: $<aa>, <ab>, <ac>, <ba>, <bb>, <bc>, <ca>, <cb>, <cc>$.

(iii)
- $(H^2)^0 = \{\varepsilon\}$
- Beispiele aus $(H^2)^1$: $<<aa>>$, $<<ab>>$
- Beispiele aus $(H^2)^2$: $<<aa><aa>>$, $<<ab><cc>>$

(iv) Wir widerlegen die Behauptung durch das Gegenbeispiel $v = <<a><ab><b>>$. Offensichtlich gilt $v \in (H^*)^3$. Allerdings ist $v \notin (H^3)^*$.

c) Eine *formale Sprache* ist eine Teilmenge $S$ von $C^*$, wobei $C$ eine Menge von Zeichen ist. D.h. eine formale Sprache ist eine Menge von Wörter.
Palindrome sind Zeichensequenzen mit der besonderen Eigenschaft, dass man sie sowohl von links nach rechts als auch „rückwärts", von rechts nach links lesen kann, und dabei identische Zeichen des Zeichenvorrats in genau der gleichen Reihenfolge auftreten; z.B. „otto". Präzise lässt sich dieser Sachverhalt wie folgt formulieren:

$$P = \{w \in H^* : w = <w_1 \dots w_n>, w_i \in H\ \wedge w_i = w_{(n-i+1)}\}$$

## Aufgabe 10 Ordnungen auf Zeichensequenzen (Lösungsvorschlag)

a) Es wird gezeigt, dass die Präfixrelation reflexiv, antisymmetrisch und transitiv ist:

**Reflexivität:** Sei $x \in C^*$. Da $x = x \circ \varepsilon$ gilt für alle $x \in C^*$ folgt $x \sqsubseteq x$ und die Präfixrelation ist reflexiv.

**Antisymmetrie:** Sei $x, y \in C^*$; außerdem gelte $x \sqsubseteq y$ und $y \sqsubseteq x$. Dann gibt es ein $u, u' \in C^*$ derart, dass $y = x \circ u$ und $x = y \circ u'$. Daraus folgt $y = y \circ u \circ u'$ und $u = u' = \varepsilon$. Damit gilt $y = x \circ \varepsilon$ und schließlich $y = x$. Die Präfixrelation ist somit antisymmetrisch.

**Transitivität:** Sei $x, y, z \in C^*$; außerdem gelte $x \sqsubseteq y$ und $y \sqsubseteq z$. Dann gibt es ein $v, v' \in C^*$ derart, dass $y = x \circ v$ und $z = y \circ v'$. Wir erhalten also $z = x \circ v \circ v'$. Mit $v'' = v \circ v'$ folgt $z = x \circ v''$ oder $x \sqsubseteq z$. Die Präfixrelation ist somit transitiv.

Insgesamt ist also gezeigt, dass die Präfixrelation eine Halbordnung ist.

b) Wir zeigen durch Angabe eines Gegenbeispiels: Die Präfixrelation ist keine totale Ordnung. Sei $C = \{a, b\}$ und $x, y \in C^*$ mit $x = a$ und $y = b$. Dann gilt weder $x \sqsubseteq y$ noch $y \sqsubseteq x$ und die Präfixrelation ist nicht total.

## Aufgabe 11 Informelle Algorithmenbeschreibung: Türme von Hanoi (Lösungsvorschlag)

a) Für $n = 0$ ist die Lösung klar: Es ist nichts zu tun.
Hat man bereits eine Zugfolge für $n - 1$ Scheiben, dann ergibt sich daraus unmittelbar eine Zugfolge für $n$ Scheiben:

```
(A)     Verlege n Scheiben von a nach b mittels Platz c
        wie folgt:
                Verlege n-1 Scheiben von a nach c mittels Platz b
                Setze eine Scheibe vom Platz a auf Platz b
                Verlege n-1 Scheiben von c nach b mittels Platz a
```

Man beachte, dass beim Umlegen der $n - 1$ oberen Scheiben die unterste nicht stört, weil sie größer als alle anderen ist.

b) Es wird nur ein elementarer Schritt verwendet: `Setze eine Scheibe vom Platz a auf Platz b`.

c) Die Anzahl $Anz(n)$ der benötigten elementaren Schritte, um einen Turm aus $n$ Scheiben zu versetzen, ist:

| | | | | |
|---|---|---|---|---|
| Anz(1) | = | 1 | | |
| Anz(2) | = | 1+2*Anz(1) | = | 1+2 |
| Anz(3) | = | 1+2*Anz(2) | = | 1+2*(1+2) |
| Anz(4) | = | 1+2*Anz(3) | = | 1+2*(1+2*(1+2)) |
| | | | = | 1+ 2+ 4+8 |
| Anz(n) | = | $2^n - 1$ | | |

d) Das Verfahren ist deterministisch, weil die Schrittfolge eindeutig festgelegt ist. Es ist determiniert, weil das Ergebnis eindeutig ist, es ist terminierend, weil es nach endliche vielen Schritten endet.

Eine andere Lösung: Wir denken uns die drei Plätze im Kreis angeordnet, und zwar, wenn $n$ ungerade ist, so:

auf $a$ folgt $b$, darauf $c$, dann wieder $a$

und wenn $n$ gerade ist, so

auf $a$ folgt $c$, darauf $b$, dann wieder $a$.

Das folgende Verfahren gibt für $0 < n$ die Schrittfolge als Wiederholung von zwei elementaren Schritten an.

```
(B)     Verlege n Scheiben von a nach b mittels Platz c
        wie folgt:
                Lege die kleinste Scheibe einen Platz weiter
                Wiederhole (2^(n-1) - 1)-mal:
                    Bewege eine Scheibe, aber nicht die kleinste
                    Lege die kleinste Scheibe einen Platz weiter
```

Im folgenden Beispiel sind Scheiben durch Zahlen dargestellt, die ihre Durchmesser wiedergeben. Fett gedruckt ist jeweils die Scheibe, die zuletzt bewegt wurde. Man beachte, wie die Turmspitze wandert, und zwar zunächst die 1 nach $b$, dann 12 nach $c$, dann 123 nach $b$ - unabhängig jeweils vom darunterliegenden Rest des Turms:

| a | b | c |
|---|---|---|
| 12345 | ..... | ..... |
| .2345 | ....**1** | ..... |
| ..345 | ....1 | ....**2** |
| ..345 | ..... | ...**1**2 |
| ...45 | ....**3** | ...12 |
| ..**1**45 | ....3 | ....2 |
| ..145 | ...**2**3 | ..... |
| ...45 | ..**1**23 | ..... |
| | ..... | |

Dieses Verfahren führt zur selben Schrittfolge und zum selben Ergebnis wie Verfahren (A). Es ist somit auch deterministisch, determiniert und terminierend.

Anmerkung: Der Turm der buddistischen Mönche, der auf diese Weise täglich um einen Stein versetzt wird und die Tage bis zum Weltuntergang zählt, hat eine Höhe von 64 Scheiben. Das Umsetzen wird also mehr als $10^{15}$ Jahre dauern. Übrigens ist $(2^{64} - 1)$ auch die Zahl der Reiskörner, die jenem König dadurch als Belohnung abverlangt wurde, dass er beginnend bei 1 auf jedes folgende Feld des Schachbretts die doppelte Anzahl Körner legen sollte.

## Aufgabe 12 Operationen auf Zeichensequenzen (Lösungsvorschlag)

a) Eine Implementierung des gesuchten Goferprogramms lautet:

```
vorname = "Markus"
nachname = "Schneider"

name = last (init (init (init (init (init vorname))))) :
       last (init (init (init (init vorname)))) :
       last (init (init (init vorname))) :
       last (init (init vorname)) :
       last (init vorname) :
       last vorname : nachname
```

b) Das Revertieren der gegebenen Zeichensequenz kann mit folgendem Goferprogramm durchgeführt werden:

```
wort = "abcdef"

wortneu = head (tail (tail (tail (tail (tail wort))))) :
          head (tail (tail (tail (tail wort)))) :
          head (tail (tail (tail wort))) :
          head (tail (tail wort)) :
          head (tail wort) :
          [head wort]
```

c) Wir definieren zunächst die Sequenz von Zeichensequenzen mit

```
satz = ["sator", "arepo", "tenet", "opera", "rotas"].
```

Anschließend definieren eine Hilfsfunktion `f`, die das Zeichen in der i-ten Spalte und der j-ten Zeile des Palindroms auswählt:

```
f :: Int -> Int -> Char
f i j = (satz !! i) !! j .
```

Nun läßt sich die gewünschte Zeichensequenz durch mehrfaches Anwenden der Hilfsfunktion erstellen:

```
zeichenkette = (f 0 0 : (f 1 0 : (f 2 0 : (f 3 0 : [f 4 0]))))
```