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

**WS 2001**
**Lösungsvorschläge zu Blatt 4**
**08. November 2001**

# Übungen zu Einführung in die Informatik I

## Aufgabe 13 Zeichensequenzen (Lösungsvorschlag)

a) Die Aufgabe demonstriert u.a., das eine vollständige und eindeutige Beschreibung eines Algorithmus in natürlicher Sprache sehr schwierig ist. Eine mögliche (unvollständige) Beschreibung eines Algorithmus könnte wie folgt lauten:

Eine Lösungsidee ist, am Beginn des Worts, den man an dem Symbol „<“ erkennt, ein zusätzliches Zeichen einzuführen. Initial steht dieses Zeichen für „gerade“. Mit diesem Zeichen wandern wir von links nach rechts. Bei Überschreiten eines Striches von links nach rechts, ersetzen wir unser zusätzliches Zeichen durch ein Zeichen für „ungerade“ und umgekehrt. Das zusätzliche Zeichen gibt somit stets an, ob die Anzahl der bereits verarbeiteten Striche gerade oder ungerade war. Bei Erreichen des Wortendes ist abschließend das Zeichen für „gerade“ durch *gerade* bzw. „ungerade“ durch *ungerade* zu ersetzen.

b) Ein Textersetzungssystem, das die umgangssprachlich formulierte Lösungsidee aus der Tailaufgabe a) realisiert, wäre beispielsweise:

- Variante 1

$$
\begin{aligned}
\text{Zeichenvorrat } V = \; & \{\; |, <, >, G, U, gerade, ungerade\} \\
\text{Regelmenge } T = \; & \{\; < \to G, \\
& \quad G\,| \to U, \\
& \quad U\,| \to G, \\
& \quad G> \to gerade, \\
& \quad U> \to ungerade\}
\end{aligned}
$$

Dieses Textersetzungssystem ist deterministisch, determiniert und terminierend.

- Variante 2
Würde man noch folgende Regeln hinzunehmen, so wäre das Textersetzungssystem nicht mehr deterministisch, aber nach wie vor determiniert und terminierend.

$$
\begin{aligned}
G\,|> &\to ungerade \\
U\,|> &\to gerade
\end{aligned}
$$

- Variante 3 (elegante Lösung ohne zusätzliche Zeichen)

$$
\begin{aligned}
\text{Zeichenvorrat } V = \; & \{\; |, <, >, gerade, ungerade\} \\
\text{Regelmenge } T = \; & \{<||\to <, \\
& \quad <|> \to ungerade, \\
& \quad <> \to gerade\}
\end{aligned}
$$

c) Bei Variante 2 ergäbe sich folgende Ableitung (Ersetzungsstelle ist jeweils unterstrichen):

$$\underline{\leq}|||||||> \longrightarrow \underline{G|}||||||> \longrightarrow \underline{U|}|||||> \longrightarrow \underline{G|}||||> \longrightarrow \underline{U|}|||> \longrightarrow \underline{G|}||> \longrightarrow$$
$$\underline{U|}|> \longrightarrow \underline{G|}> \longrightarrow \underline{U>} \longrightarrow \mathit{ungerade}$$
$$\searrow \mathit{ungerade}$$

## Aufgabe 14 Markov (Lösungsvorschlag)

a) $ba \to ab \to ab \to \ldots$
$abaaba \to aababa \to aaabba \to aaabab \to aaaabb \to aabb \to bb \to \varepsilon$
$baaab \to abaab \to aabab \to aaabb \to abb \to a \to a \to \ldots$

b) Für alle Wörter aus $S = \{v \in V^* \,:\,$ in $v$ ist sowohl die Anzahl der $a$ als auch die Anzahl der $b$ gerade $\}$ terminiert der Markov-Algorithmus $M$ mit dem Ergebnis $\varepsilon$. Für alle Wörter aus $V^* \setminus S$ terminiert $M$ nicht.

c) Wenn das Eingabewort ein $a$ enthält, terminiert der Algorithmus wegen der Regel $a \to a$ nicht mehr. Die Regel $bb \to \varepsilon$ ist die einzige, die zur Terminierung führen kann. Also terminiert der Algorithmus für die Wortmenge $S' = \{b^{2n} \,:\, n \in \mathbb{N}\}$.

## Aufgabe 15 Substitution mittels Markov-Algorithmen (Lösungsvorschlag)

a) Falls $x_1$ in $t_1$ nicht vorkommt, besteht die einfachste Lösung aus nur einer Regel

$$x_1 \to t_1 \tag{1}$$

(Dies ist streng genommen ein Regelschema für beliebige vorgegebene $x_1$ und $t_1$.) Unter Umständen muss in dieser Regel der Term $t_1$ noch geklammert werden, damit nach Substitution wieder ein korrekter Term entsteht. Das hängt von $t_1$ ab und sei vorher erledigt:

$$x_1 \to (t_1) \tag{2}$$

Falls $x_1$ in $t_1$ vorkommt, ist dieses Textersetzungssystem ungeeignet. Sobald nämlich $x_1$ in $t$ vorkommt, würde eine Berechnung das mit dem Term $t_1$ neu eingesetzte $x_1$ auch wieder substituieren, d.h. die Berechnung terminiert nicht.

Wir formulieren einen Markov-Algorithmus, der den Term $t$ von links nach rechts durchläuft und dabei die Substitution aller $x_1$ durch $t_1$ vornimmt. Die aktuelle Arbeitsposition wird durch ein "Schiffchen" # angezeigt.

| | | | |
|---|---|---|---|
| #( | $\to$ | (# | Schiffchen weiter nach rechts schieben |
| #) | $\to$ | )# | - " - |
| #$\neg$ | $\to$ | $\neg$ # | - " - |
| #$\wedge$ | $\to$ | $\wedge$# | - " - |
| #$\vee$ | $\to$ | $\vee$# | - " - |
| #$\Rightarrow$ | $\to$ | $\Rightarrow$# | - " - |
| #$\Leftrightarrow$ | $\to$ | $\Leftrightarrow$# | - " - |
| #true | $\to$ | true# | - " - |
| #false | $\to$ | false# | - " - |
| #$x_1$ | $\to$ | $t_1$# | Substitution ausführen, ggf. Regel #$x_1 \to (t_1)$# |
| #$x_2$ | $\to$ | $x_2$# | Schiffchen weiter nach rechts schieben |
| $\vdots$ | | $\vdots$ | |
| #$x_n$ | $\to$ | $x_n$# | Schiffchen weiter nach rechts schieben |
| # | $\to .$ | $\varepsilon$ | Schiffchen löschen und terminieren |
| $\varepsilon$ | $\to$ | # | Schiffchen ganz links einführen |

Die letzten beiden Regeln müssen unbedingt am Schluss stehen, und zwar in der angegebenen Reihenfolge. Davor ist die Reihenfolge unerheblich.

b) Das in (a) angegebene Textersetzungssystem würde nicht terminieren ohne die vorletzte Regel, da immer wieder ein neues Schiffchen erzeugt würde. Es kann kein Textersetzungssystem ohne Punktregel geben, das die Substitution korrekt durchführt und terminiert. Nehmen wir an, $R$ wäre so eines. Dann würde beim Beispiel

$$\begin{aligned} t &=_{def} x_1 \\ t_1 &=_{def} (\neg x_1) \end{aligned}$$

für die durch $R$ induzierte Abbildung $f_R$ gelten:

$$\begin{aligned} f_R(x_1) &= (\neg x_1) \\ f_R((\neg x_1)) &= (\neg(\neg x_1)) \end{aligned}$$

das heißt, $R$ liefert in der Berechnung zu $x_1$ das Ergebnis $(\neg x_1)$. Darauf ist aber $R$ wieder anwendbar und soll das Ergebnis $(\neg(\neg x_1))$ liefern. Ohne eine terminierende Punktregel ist also $R$ immer wieder anwendbar und die Berechnung zu $t$ terminiert nicht. Dies ist ein Widerspruch zur Annahme.

c) Damit die vorgegebene simultane Substitution definiert ist, müssen $x_1, \ldots, x_n$ paarweise verschieden sein. Das Textersetzungssystem ist analog zu dem in (a)

| | | | |
|---|---|---|---|
| #( | $\to$ | (# | Schiffchen weiter nach rechts schieben |
| #) | $\to$ | )# | - “ - |
| #$\neg$ | $\to$ | $\neg$ # | - “ - |
| #$\wedge$ | $\to$ | $\wedge$# | - “ - |
| #$\vee$ | $\to$ | $\vee$# | - “ - |
| #$\Rightarrow$ | $\to$ | $\Rightarrow$# | - “ - |
| #$\Leftrightarrow$ | $\to$ | $\Leftrightarrow$# | - “ - |
| #true | $\to$ | true# | - “ - |
| #false | $\to$ | false# | - “ - |
| #$x_1$ | $\to$ | $t_1$# | (*) Subst. ausführen, ggf. Regel #$x_1 \to (t_1)$# |
| #$x_2$ | $\to$ | $t_2$# | (*) Subst. ausführen, ggf. Regel #$x_2 \to (t_2)$# |
| $\vdots$ | | $\vdots$ | |
| #$x_n$ | $\to$ | $t_n$# | (*) Subst. ausführen, ggf. Regel #$x_n \to (t_n)$# |
| # | $\to .$ | $\varepsilon$ | Schiffchen löschen und terminieren |
| $\varepsilon$ | $\to$ | # | Schiffchen ganz links einführen |

d) Für die mit (*) markierten Substitutionsregeln in (c) haben wir in diesem speziellen Beispiel:

| | | | |
|---|---|---|---|
| #$x$ | $\to$ | $y$# | (*) Substitution ausführen |
| #$y$ | $\to$ | $x$# | (*) Substitution ausführen |

Die Berechnung für den Term $(x \wedge y)$ liefert:
$(x \wedge y) \to \#(x \wedge y) \to (\#x \wedge y) \to (y\# \wedge y) \to (y \wedge \#y) \to (y \wedge x\#) \to (y \wedge x)\# \to (y \wedge x)$.

## Aufgabe 16 Markov–Algorithmus (Lösungsvorschlag)

a) **Variante 1:** Zeichenvorrat `V = { a, b, c }`, Schiffchen `S = ∅`

Regelmenge `T = {`

| | | |
|---|---|---|
| ab | $\rightarrow$ | ε |
| ba | $\rightarrow$ | ab |
| ac | $\rightarrow$ | ε |
| ca | $\rightarrow$ | ac |
| a | $\rightarrow.$ | a |
| b | $\rightarrow.$ | b |
| c | $\rightarrow.$ | c |
| ε | $\rightarrow.$ | ε |

}

**Variante 2:** Zeichenvorrat `V = { a, b, c }`, Schiffchen `S = ∅`

Regelmenge `T = {`

| | | |
|---|---|---|
| c | $\rightarrow$ | b |
| ab | $\rightarrow$ | ε |
| ba | $\rightarrow$ | ε |
| a | $\rightarrow.$ | a |
| b | $\rightarrow.$ | b |
| ε | $\rightarrow.$ | ε |

}

b) Folgendes Skript `regeln.gs` definiert die in Variante 1 angegebenen Regeln des Markov-Algorithmus. (Beim Test ist darauf zu achten, dass zuerst das Skript `markov.gs` geladen wird und anschließend mit `:also regeln.gs` die Definition der Regeln.)

```
p1 :: (String, String, Bool)
p2 :: (String, String, Bool)
p3 :: (String, String, Bool)
p4 :: (String, String, Bool)
p5 :: (String, String, Bool)
p6 :: (String, String, Bool)
p8 :: (String, String, Bool)

p1 = ("ab", "", False)
p2 = ("ba", "ab", False)
p3 = ("ac", "", False)
p4 = ("ca", "ac", False)
p5 = ("a", "a", True)
p6 = ("b", "b", True)
p7 = ("c", "c", True)
p8 = ("", "", True)

regeln :: [(String, String, Bool)]

regeln = [p1, p2, p3, p4, p5, p6, p7, p8]

annahme :: (String, String)
annahme = markov "ababccaa" regeln
ablehnung :: (String, String)
ablehnung  = markov "aabcc" regeln
```