PHILIPPS-UNIVERSITÄT MARBURG D-35032 Marburg
Fachbereich Mathematik und Informatik Hans Meerwein Straße
Prof. Dr. R. Loogen
J.Berthold

10.November 2005

3. Übung zu „Grundlagen des Compilerbaus“, WS 2005/06

Abgabe schriftlicher Aufgaben: Do, 17.November 2005 (vor der Vorlesung)
Besprechung mündlicher Aufg.: ab 14.November 2005 in der Übung

---

Mündliche Aufgaben

3.1 Zerlegung und Symbolzuordnung

(a) Das Prinzip der “Longest-Match-Zerlegung” garantiert, dass eine Zerlegung der Eingabe eindeutig wird, falls sie existiert. Könnte man dies auch durch eine “Shortest-Match-Zerlegung” erreichen? Erläutern Sie ggf. an einem einfachen Beispiel Vor- und Nachteile.

(b) Beweisen oder widerlegen Sie:

> Seien $\alpha_1, \ldots, \alpha_n \in \mathrm{RA}(\Sigma)$ reguläre Ausdrücke und $w \in \Sigma^*$ ein Wort. Wenn es eine Zerlegung von w bzgl. $(\alpha_1, \ldots, \alpha_n)$ gibt, so gibt es auch eine lm-Zerlegung von w bzgl. $(\alpha_1, \ldots, \alpha_n)$.

(c) In Fortran können Typnamen als Variablenbezeichner verwendet werden. Dies ist also korrekter Fortran-Code:

```
integer real;        ! zwei Deklarationen
real integer;
real = integer + 10; ! eine Zuweisung
```

In der Sprache PL/I sind ebenfalls Variablenbezeichner erlaubt, die normalerweise als Schlüsselwörter reserviert sind, etwa so wie hier:

```
if then = else then if = then else then = if
```

Welches massive Problem ergibt sich daraus für die Scanner-Konstruktion, wie sie in der Vorlesung vorgestellt wurde?

3.2 Alex

(a) Erstellen Sie mit Alex einen Scanner zur Erkennung der in Aufgabe 2.2. gegebenen regulären Ausdrücke und geben Sie den von Alex erzeugten DFA an.

(b) Testen Sie Ihren Scanner mit gültigen und ungültigen Eingaben. Erweitern Sie den Scanner, so dass er bei ungültigen Eingaben die Position des Fehlers angibt.

Bitte wenden!

Schriftliche Aufgaben

3.3 Linearer Scan-Aufwand

5 Punkte

In der Vorlesung wurde die *Maximal-Munch*-Methode (Longest-Match) mit quadratischem Aufwand eingeführt. Mit einer von Thomas Reps vorgeschlagenen Optimierung [1] kann ein Scanner dagegen sogar in linearer Zeit arbeiten.

(a) Lesen Sie die genannte Veröffentlichung[1] und erläutern Sie die darin beschriebenen Verfahren an einem eigenen Beispiel.

(b) Stellen Sie fest, ob ein von Alex generierter Scanner mit linearem oder quadratischem Aufwand arbeitet.

[1] T. Reps:*Maximal-Munch Tokenization in Linear Time* In: *ACM TOPLAS* 20(1998), 2, S. 259ff.

3.4 Scanner für umgangssprachliche Uhrzeitangaben

7 Punkte

Für eine Sprache zur umgangssprachlichen Angabe von Uhrzeiten seien die nebenstehenden Symbolklassen definiert.

Haskell Code

```
data Token =
    Nach | Vor
    | Halb | Viertel
    | Dreiviertel
    | Zahl Int
```

Stunden und Minuten werden als Zahlen angegeben, die im gültigen Bereich (0-59) bleiben sollen. Zeitangaben wie z.B. "drei viertel 12" oder "5 nach 1" lassen sich in diese Token zerlegen.

(a) Spezifizieren Sie unter Verwendung der angegebenen Token, was gültige Zeitangaben sind. Gehen Sie davon aus, dass Leerzeichen überlesen werden.

(b) Erstellen Sie mit dem Scanner-Generator Alex einen Scanner für die in a) spezifizierten Zeitangaben.

(c) Schreiben Sie ein Haskell-Programm, das den erzeugten Scanner importiert und aus der gelesenen umgangssprachlichen Angabe die entsprechende Zeit in Minuten seit 0:00 Uhr berechnet.

Beispiele:
`Viertel nach 7` $= 7 \cdot 60 + 15 = 435$, `drei viertel 12` $= 12 \cdot 60 - 15 = 705$.

[1] Von Fachbereichsrechnern aus haben Sie Zugang zur ACM Digital Library. Sie können den Artikel unter *http://portal.acm.org/dl.cfm* herunterladen.