PHILIPPS-UNIVERSITÄT MARBURG
Fachbereich Mathematik und Informatik
Prof. Dr. R. Loogen J.Berthold

D-35032 Marburg
Hans Meerwein Straße
Lahnberge

15. Dezember 2003

Übungen zu „Grundlagen des Compilerbaus“, WS 2003/04

Nr. 8 , Abgabe und Besprechung: 12. Januar 2004 in der Übung

Hinweise: In dieser Übung wird die zurückgestellte Aufgabe 7.5 bearbeitet; daher sind statt 12 Punkten auf diesem Blatt 15 Punkte erreichbar.

Mündliche Aufgaben

8 .1 **Vergleich von Grammatiken**
Welche der folgenden Grammatiken sind in LL(k) / in LR(k)? Begründen Sie Ihre Antworten und geben Sie ggf. k explizit an.

$$\begin{array}{lcl} S & \to & B \mid C \\ B & \to & bB \mid a \\ C & \to & bC \mid c \end{array} \qquad \begin{array}{lcl} S & \to & Ba \mid Cc \\ B & \to & bB \mid \varepsilon \\ C & \to & bC \mid \varepsilon \end{array} \qquad \begin{array}{lcl} S & \to & aBc \\ B & \to & Bbb \mid b \end{array} \qquad \begin{array}{lcl} S & \to & aBc \\ B & \to & bBb \mid b \end{array}$$

8 .2 **Mehrdeutige Grammatik und LR(k)**
Beweisen Sie, daß eine mehrdeutige Grammatik für kein $k \in I\!N$ LR(k)-Grammatik ist.

Schriftliche Aufgaben

8 .3 *Bearbeiten Sie Aufgabe 7.5 von Blatt 7:*
LR(0)-Mengen 3 Punkte
Bestimmen Sie zur folgenden Grammatik G die Menge LR(0)(G):

$$\begin{array}{lcl} \mathrm{S} & \to & \mathrm{CC} \\ \mathrm{C} & \to & \mathrm{cC} \mid \mathrm{d} \end{array}$$

Ist G LR(0)?

8 .4 **Grammatik G_{bool} für Bool'sche Ausdrücke** 6 Punkte
Die Grammatik G_{bool} sei gegeben durch:

$$\begin{array}{lcl} \mathrm{S} & \to & \mathrm{B} \\ \mathrm{B} & \to & (\mathrm{B} \wedge \mathrm{B}) \mid \neg \mathrm{B} \mid \mathrm{t} \mid \mathrm{f} \end{array}$$

(a) Berechnen Sie die LR(0)-Informationen von G_{bool} mit Hilfe der Potenzmengenkonstruktion.

(b) Geben Sie die LR(0)-Analysetabelle von G_{bool} an.

(c) Bestimmen Sie die Konfigurationsfolge, die der LR(0)-Analyseautomat bei Eingabe des Wortes $((\neg t \wedge f) \wedge (f \wedge t))$ durchläuft.

8 .5 **Bottom-Up-Analyseautomat in Haskell** 6 Punkte

In der Vorlesung wurde der (unter Umständen nichtdeterministische) Bottom-Up-Analyseautomat definiert. In dieser Aufgabe wird der Automat in Haskell implementiert.

(a) Implementieren Sie die Transitionsfunktion `transition` eines nichtdeterministischen Bottom-Up-Analyseautomaten gemäß den Definitionen der Vorlesung. Verwenden Sie dabei die folgenden Typen und Funktionen:

Haskell Code

```
-- CFG = (Nonterminals,Terminals,StartSymbol,Produktionen)
type CFG = ([Nonterminal], [Terminal], Nonterminal, Rules)
type Rules = [(Nonterminal,Satzform)]
type Nonterminal = Char
type Terminal    = Char
type Satzform    = String

-- Automatenkonfigurationen:
data Config = C Satzform [Terminal] [Int]    deriving (Eq,Show)
            --  stack     eingabe  ausgabe

transition ::  CFG -> Config -> [Config]    -- Transitionsfunktion
```

Die nichtdeterministische Transitionsfunktion erzeugt also stets eine (evtl. leere) Menge von Nachfolgekonfigurationen zu einer Konfiguration. Man kann außerdem bereits hier die Ausgabe spiegeln, weil hinterher ohnehin eine Rechtsanalyse ausgegeben wird.

(b) Schreiben Sie eine Funktion `analyse`, welche mit Hilfe von `transition` zu einer startseparierten Grammatik und einem eingegebenen Wort eine Rechtsanalyse erzeugt bzw. einen Misserfolg meldet.

Haskell Code

```
-- analyse liefert die Rechtsanalyse zu einem eingegebenen Wort.
-- Die Analyse [-1] wird als Fehlermeldung interpretiert
analyse :: CFG -> [Terminal] -> [Int]
```

(c) i. Testen Sie den Automaten mit (jeweils startseparierten) Grammatiken aus Aufgabe 8.1 sowie mit gültigen und ungültigen Eingaben.
Welche Grammatiken führen (aus welchem Grund) zu Problemen?

ii. Unter welchen Bedingungen können mehrere verschiedene Endkonfigurationen erreicht werden?

Frohe Weihnachten
und alles Gute in 2004