Prof. Dr. R. Loogen, M. Dieterle
Fachbereich Mathematik und Informatik
Hans-Meerwein-Straße
D-35032 Marburg

11. November 2009

Philipps Universität Marburg

Übungen zu „Grundlagen des Compilerbau“, Winter 2009/10

Nr. 4, Abgabe der Aufgaben: 18. November 2009 vor der Vorlesung

---

Aufgaben

4.1 Deterministischer LL(1)-TDA in Haskell

6 Punkte

Üblicherweise benutzen Grammatiken für Programmiersprachen die Ausgabe des Scanners, also `[Token]`, als Terminalsymbole. Zur Vereinfachung sollen für diese Aufgabe die Grammatiken in Haskell statt dessen Buchstaben als Symbole benutzen:

```
type Nonterminal = Char     -- Konvention: Großbuchstaben
type Terminal    = Char     -- Konvention: Kleinbuchstaben und Sonderz.
type Ruleindex   = Int      -- Index einer Regel in einer Grammatik

data CFG = CFG {nterm :: [Nonterminal],
                term  :: [Terminal],
                start :: Nonterminal,
                rules :: [Rule]}

data Rule = Rule {leftp  :: Nonterminal,
                  rightp :: [Char]} deriving Show

type LASet = [(Ruleindex,[Maybe Terminal])] --Nothing: Epsilon
data LAnalysis = LAnalysis [Ruleindex] --Indexe der abgeleiteten Regeln
               | SynErr [Terminal] [Ruleindex] --Resteingabe Indexe
```

Dabei seien (als Konvention, *nicht* typsicher) Nonterminale stets Großbuchstaben, Terminale hingegen stets Kleinbuchstaben oder Sonderzeichen.

(a) Drücken Sie die Grammatik für arithmetische Ausdrücke G'_{AE} (Skript S. 37) und die dazugehörenden *la*-Mengen als `gAE :: CFG` und `la_gAE :: LASet` in Haskell aus. Ersetzen Sie die gestrichenen Buchstaben dabei mit den nächst kleineren Buchstaben im Alphabet. / 1

(b) Schreiben Sie die Funktion: / 2

```
mkAnaTable :: CFG → LASet → Maybe Terminal → Nonterminal
                  → Maybe ([Char],Int),
```

welche zu einer Grammatik und den *la*-Mengen eine Funktion definiert, mit deren Hilfe der TDA gesteuert wird. Letztere soll (wie die linke Hälfte der Analysetabelle) für ein Eingabezeichen und ein Nonterminal eine abgeleitete rechte Regelseite und die Regelnummer liefern. Dabei steht ein **`Nothing`** auf der Eingabe (3. Parameter) für das Eingabeende und ein **`Nothing`** im Ergebnis für einen undefinierten Tabelleneintrag (Fehler).

(c) Implementieren Sie einen deterministischen LL(1)-TDA: / 3

```
ll1TDA :: CFG → LASet → [Terminal] → LAnalysis.
```

Bitte wenden!

4.2 **LL(2)-Grammatik** 6 Punkte

(a) Linksfaktorisieren Sie die nebenstehende Grammatik und beseitigen Sie die Linksrekursion. / 1

(b) Konstruieren Sie die LL(1)-Analysetabelle zu der resultierenden Grammatik. Benennen Sie den vorhandenen Konflikt. / 3

(c) Zeigen Sie, dass der auftretende Konflikt durch einen Lookahead von zwei aufgelöst werden kann. / 2

$$
\begin{array}{lcl}
S & \rightarrow & H \ \texttt{\$} \\
H & \rightarrow & P \\
 & | & P \ H \\
P & \rightarrow & \texttt{id} : \\
 & | & P \ \texttt{id}
\end{array}
$$