Fakultät für Mathematik und Informatik
Dr. Christoph Herrmann

SS 2006

Übungen zur Vorlesung: Struktur und Implementierung von Programmiersprachen I

Blatt 2 (Syntaxanalyse (1))

Aufgabe 3 (Recursive-Descend-Parsing)

Gegeben ist eine Grammatik G für geklammerte arithmetische Ausdrücke.

- Terminalzeichen (Token): $(id \cup \{\texttt{+}, \texttt{-}, \texttt{*}, \texttt{(}, \texttt{)}, \texttt{EOF}\})$, wobei id für die Menge der Identifikatoren steht und `EOF` das *end-of-file*-Token bezeichnet. Die Menge id wird durch den regulären Ausdruck `[a-zA-Z]+` beschrieben.
- Nichtterminale: $\{E', E, T, F\}$
- Startsymbol: E'
- Produktionen:

$$
\begin{array}{lcl}
E' & \rightarrow & E\ \texttt{EOF} \\
E & \rightarrow & E\ \texttt{+}\ T \mid E\ \texttt{-}\ T \mid T \\
T & \rightarrow & T\ \texttt{*}\ F \mid F \\
F & \rightarrow & id \mid \texttt{(}\ E\ \texttt{)}
\end{array}
$$

Teilaufgaben:

(a) Berechnen Sie manuell die FIRST- und FOLLOW-Mengen für die Nichtterminale von G. Beschreiben Sie unter Bezug auf diese Mengen und die Grammatik, welches Problem beim Versuch der Konstruktion eines Abstiegsparsers für G entstehen würde.

(b) Transformieren Sie manuell nach dem in der Vorlesung behandelten Schema diese Grammatik in eine äquivalente Grammatik G' $(L(G) = L(G'))$, die linksfaktorisiert und nicht linksrekursiv ist. Damit bei der Besprechung keine zusätzliche Erschwernis durch die Namensgebung entsteht, vergeben Sie bitte die Namen für zusätzliche Nichtterminale so, dass die Produktionen $(E \rightarrow T\ R)$ und $(S \rightarrow *\ F\ S)$ vorkommen.

(c) Berechnen Sie manuell die FIRST- und FOLLOW-Mengen für die Nichtterminale von G'. Zur Erinnerung: es handelt sich dabei um Hüllenalgorithmen, die erst dann terminieren, wenn an keiner Stelle mehr eine Regel anwendbar ist.

(d) Schreiben Sie unter Verwendung des auf der Webseite gegebenen Scanners einen Abstiegsparser für G', der für ein gegebenes Wort aus $L(G')$ einen Parsebaum als Element eines algebraischen Datentyps in OCaml (`type`) erzeugt. In diesem Datentyp sollen die Klammern, Operatoren und auch das Meta-Symbol ε eine explizite Repräsentation besitzen.

(e) Schreiben Sie eine Funktion, die aus dem Parsebaum einen abstrakten Syntaxbaum (AST) des folgenden Typs erzeugt:

```
type syntaxTree =
    SId    of string                    (* Identifikator *)
  | SPlus  of syntaxTree * syntaxTree   (* Knoten für +  *)
  | STimes of syntaxTree * syntaxTree   (* Knoten für *  *)
```

Die Prioritäten und Assoziativitäten der Operatoren sollen dabei dieselben sein, die durch die ursprüngliche Grammatik G vorgegeben wurden. Dabei müssen die Umstrukturierungen, die durch die Elimination der Linksrekursion entstanden sind, ausgeglichen werden. Verwenden Sie in bestimmten Fällen den Ergebnis-AST eines linken Parsebaumteils als Zusatz-Argument bei der Transformation des rechten Parsebaumteils in einen gemeinsamen Ergebnis-AST.

Besprechung: in der Übung am Mittwoch, 17.05.2006.