

# Grundlagen Programmiersprachen und Compilerbau

Wintersemester 2005/2006

## 4. Übungsblatt

7.12.2005

Dieses Übungsblatt wird am 14,15,16.12.2005 besprochen.

### Aufgabe 4.1

*Diese Aufgabe steht bereits auf dem vorigen Übungsblatt als Aufgabe 3.5*

Mit der Methode des rekursiven Abstiegs kann man Parser für die Klasse der LL(1)-Grammatiken implementieren.

- a) Geben Sie ein allgemeines Schema an, mit dessen Hilfe man aus einer in EBNF-Notation vorliegenden LL(1)-Grammatik einen in Ada95 formulierten Parser nach der Methode des rekursiven Abstiegs konstruieren kann. Dazu muss jeder Struktur der EBNF eine passende Anweisung oder Anweisungsfolge zugeordnet werden. Dabei seien folgende Vereinbarungen vorgegeben:

```
package Rekursiver_Abstieg_Parser is
  type Token_Type is (was auch immer);
  Token : Token_Type;
  -- enthaelt das aktuelle Token
  procedure Get_Next-Token;
  -- Get_Next-Token enthaelt den Scanner und liefert jeweils das naechste Token
  -- des Eingabetextes in der Variablen Token zurueck.
  procedure Error;
  -- diese Prozedur wird zur Fehlerbehandlung aufgerufen
end Rekursiver_Abstieg_Parser;
```

Zu Beginn der Syntaxanalyse enthalte `Token` bereits das erste Symbol. Zur Abkürzung kann folgende Notation verwendet werden:  $E_1 \dots E_n$  sind EBNF-Ausdrücke,  $M(E_i)$  ist der einem EBNF-Ausdruck zugeordnete Programmtext,  $A$  bezeichnet ein Nichtterminalsymbol und  $F(E)$  ist die FIRST-Menge des EBNF-Ausdrucks  $E$ . `t` steht für ein Terminalsymbol.

- b) Wenden Sie das Schema auf die folgende Deklaration an:

```
CaseStatement = "CASE" Expression "OF" CaseEntry {"|" CaseEntry}
               ["ELSE" StatementSequence] "END".
```

Für das Parsen der Nichtterminale `Expression`, `StatementSequence` und `CaseEntry` existieren bereits gleichnamige Unterprogramme.

## Aufgabe 4.2

Gegeben sei folgende attributierte Grammatik:

$$\begin{aligned} \text{Binaerzahl} &\rightarrow \text{Ziffernfolge} \\ \{ &\text{Binaerzahl.Wert} = \text{Ziffernfolge.Wert} \\ &\text{Ziffernfolge.Position} = 0 \} \end{aligned}$$

$$\begin{aligned} \text{Ziffernfolge} &\rightarrow \text{Ziffernfolge}_1 \text{ Ziffer} \\ \{ &\text{Ziffernfolge.Wert} = \text{Ziffernfolge}_1.\text{Wert} + \text{Ziffer.Wert} \\ &\text{Ziffer.Position} = \text{Ziffernfolge.Position} \\ &\text{Ziffernfolge}_1.\text{Position} = \text{Ziffernfolge.Position} + 1 \} \end{aligned}$$

$$\begin{aligned} \text{Ziffernfolge} &\rightarrow \text{Ziffer} \\ \{ &\text{Ziffernfolge.Wert} = \text{Ziffer.Wert} \\ &\text{Ziffer.Position} = \text{Ziffernfolge.Position} \} \end{aligned}$$

$$\begin{aligned} \text{Ziffer} &\rightarrow 0 \\ \{ &\text{Ziffer.Wert} = 0 \} \end{aligned}$$

$$\begin{aligned} \text{Ziffer} &\rightarrow 1 \\ \{ &\text{Ziffer.Wert} = 2^{\text{Ziffer.Position}} \} \end{aligned}$$

Verwenden Sie in den nachfolgenden Aufgaben für die Nichtterminalsymbole zur Abkürzung  $B$ ,  $F$  (für Ziffernfolge) und  $Z$ .

- Konstruieren Sie den Ableitungsbaum des Wortes 1010. Versehen Sie dabei die mit *Ziffernfolge* und *Ziffer* beschrifteten Knoten zusätzlich mit einem Index, um in der nächsten Teilaufgabe eindeutige Bezeichnungen zu erhalten.
- Tragen Sie in jeden Knoten des Ableitungsbaumes die Attribute ein.
- Tragen Sie in jeden Knoten Attributregeln ein. Verwenden Sie dabei die weiter oben definierten Indizes.
- Berechnen Sie den dezimalen Wert der Binärzahl 1010 und tragen Sie den dazu zwischen den Knoten erforderlichen Informationsfluss in den Ableitungsbaum ein.
- Welche Attribute sind zusammengesetzt, welche sind ererbt ?
- Verändern Sie die Attributregeln der Grammatik so, dass keine ererbten Attribute mehr auftreten.

Berechnen Sie den dezimalen Wert des Wortes 1010 mit Ihrer Grammatik.

### Aufgabe 4.3

Die folgende LL(1)-Grammatik ist aus der Grammatik in der vorigen Aufgabe durch Umformen entstanden:

$$\begin{aligned} \text{Binaerzahl} &\rightarrow \text{Ziffernfolge } \$ \\ \{ \text{Binaerzahl.Wert} &= \text{Ziffernfolge.Wert} \} \end{aligned}$$
$$\begin{aligned} \text{Ziffernfolge} &\rightarrow \text{Ziffer Rest} \\ \{ \text{Rest.EWert} &= \text{Ziffer.Wert}; \text{Ziffernfolge.Wert} = \text{Rest.Wert} \} \end{aligned}$$
$$\begin{aligned} \text{Rest} &\rightarrow \text{Ziffer Rest}_1 \\ \{ \text{Rest}_1.\text{EWert} &= 2 \cdot \text{Rest.EWert} + \text{Ziffer.Wert}; \text{Rest.Wert} = \text{Rest}_1.\text{Wert} \} \end{aligned}$$
$$\begin{aligned} \text{Rest} &\rightarrow \epsilon \\ \{ \text{Rest.Wert} &= \text{Rest.EWert} \} \end{aligned}$$
$$\begin{aligned} \text{Ziffer} &\rightarrow 0 \\ \{ \text{Ziffer.Wert} &= 0 \} \end{aligned}$$
$$\begin{aligned} \text{Ziffer} &\rightarrow 1 \\ \{ \text{Ziffer.Wert} &= 1 \} \end{aligned}$$

Verwenden Sie in den nachfolgenden Aufgaben für die Nichtterminalsymbole zur Abkürzung  $B$ ,  $F$  (für *Ziffernfolge*),  $R$  und  $Z$ .

- Welche Attribute sind ererbt, welche zusammengesetzt?
- Handelt es sich um eine S-Attributierung, eine L-Attributierung oder keines von beiden? Begründen Sie Ihre Antwort.
- Schreiben Sie für das Nichtterminalsymbol *Rest* ein Unterprogramm zur Syntaxanalyse und Auswertung der semantischen Attributgleichungen nach der Methode des rekursiven Abstiegs. Verwenden Sie Pascal-, Modula- oder Ada-ähnliche Notation. Folgende Deklarationen seien vorgegeben:

```
type Token is (ziffer0, ziffer1, ende);

function Get-Token return Token; -- gibt das jeweils naechste
                                -- lexikalische Token zurueck
```

(Klausuraufgabe)

#### Aufgabe 4.4

Sind die folgenden Aussagen wahr oder falsch?

- a. Bei der L-Attributierung darf die Berechnung eines Attributes eines gegebenen Knotens von Attributen des Vaterknotens und der Geschwisterknoten abhängen, die „links“ vom gegebenen Knoten im Ableitungsbaum liegen.
- b. Eine LL(1)-Grammatik kann nicht mehrdeutig sein.
- c. Alle LL(1)-Grammatiken eignen sich zur Implementierung von Parsern auf der Basis des rekursiven Abstiegs und der Kellerautomaten.
- d. Die L-Attributierung einer LL(1)-Grammatik kann durch einen Parser nach dem Prinzip des rekursiven Abstiegs problemlos ausgewertet werden.
- e. Die Berechnung zusammengesetzter Attribute entspricht einer Postfix-Durchquerung des Strukturbaumes.
- f. Gemeinsame Präfixe in alternativen Produktionen eines Nichtterminals verhindern die LL(1)-Eigenschaft einer Grammatik.
- g. Eine LL(1)-Grammatik ist immer auch eine LR(1)-Grammatik.

(aus alten Klausuraufgaben zusammengestellt)