

Grundlagen der Programmiersprachen und Compilerbau

Wintersemester 2007/2008

4. Übungsblatt

28. November 2007

Aufgabe 4.1

Die Programmiersprache MANGER[©] unterstützt zusammengesetzte Datentypen [MRA05]. Die Syntax wird durch folgende attributierte Grammatik beschrieben¹:

```
Verbund → verbund ID ist Liste ende ID;
          { Verbund.gepackt = Liste.gepackt;
            Verbund.ungepackt = Liste.ungepackt }
```

```
Liste → Liste1 Element
        { Element.start = Liste1.ungepackt
          Liste.gepackt = Liste1.gepackt + Element.groesse
          Liste.ungepackt = ((Liste1.ungepackt + Element.groesse + 3)/4) * 4 }
```

```
Liste → Element
        { Liste.gepackt = Element.groesse
          Liste.ungepackt = ((Element.groesse + 3)/4) * 4 }
```

```
Element → ID : Typ ;
          { Element.groesse = Typ.groesse
            ID.start = Element.start }
```

```
Typ → zahl
        { Typ.groesse = 4 }
```

```
Typ → bool
        { Typ.groesse = 1 }
```

Verwenden Sie in den nachfolgenden Teilaufgaben für die Nichtterminalsymbole zur Abkürzung *V*, *L*, *E* und *T*.

1. Konstruieren Sie den Ableitungsbaum des Wortes

```
verbund R ist
  eins : zahl;
  zwei : bool;
ende R;
```

¹Der / Operator steht für die ganzzahlige Division

Versehen Sie dabei die mit *Liste*, *Element* und *Typ* beschrifteten Knoten zusätzlich mit einem Index, um in der nächsten Teilaufgabe eindeutige Bezeichnungen zu erhalten.

2. Tragen Sie in jeden Knoten des Ableitungsbaumes die Attribute ein.
3. Tragen Sie in jeden Knoten Attributregeln ein. Verwenden Sie dabei die weiter oben definierten Indizes.
4. Berechnen Sie die gepackte und die ungepackte Größe des Verbundes und tragen Sie hierzu zwischen den Knoten erforderlichen Informationsfluss in den Ableitungsbaum ein.
5. Welche Attribute sind zusammengesetzt, welche sind ererbt?
6. Kann die Grammatik so verändert werden, dass keine ererbten Attribute mehr benötigt werden?

Aufgabe 4.2

Ist die Grammatik aus der letzten Aufgabe eine LL(1)-Grammatik? Wenn nein formen Sie die Grammatik in eine LL(1)-Grammatik um; passen Sie notwendigerweise die Attributregeln an und/oder führen Sie zusätzliche Attribute ein.

1. Klassifizieren Sie die Attribute der entstanden Grammatik.
2. Welche Art der Attributierung liegt vor? Begründen Sie Ihre Antwort.
3. Schreiben Sie für das Nichtterminalsymbol *Liste* ein Unterprogramm zur Syntaxanalyse und Auswertung der semantischen Attributgleichungen nach der Methode des rekursiven Abstiegs. Verwenden Sie Ada-ähnliche Notation. Folgende Deklarationen seien vorgegeben:

```
type Token is (ID, ziffer, bool);

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

Aufgabe 4.3

Sind die folgenden Aussagen wahr oder falsch?

1. 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.
2. Eine LL(1)-Grammatik kann nicht mehrdeutig sein.

3. Alle LL(1)-Grammatiken eignen sich zur Implementierung von Parsern auf der Basis des rekursiven Abstiegs und der Kellerautomaten.
4. Die L-Attributierung einer LL(1)-Grammatik kann durch einen Parser nach dem Prinzip des rekursiven Abstiegs problemlos ausgewertet werden.
5. Die Berechnung zusammengesetzter Attribute entspricht einer Postfix-Durchquerung des Strukturbaumes.
6. Gemeinsame Präfixe in alternativen Produktionen eines Nichtterminals verhindern die LL(1)-Eigenschaft einer Grammatik.
7. Eine LL(1)-Grammatik ist immer auch eine LR(1)-Grammatik.

Literatur

[MRA05] MANGER: Referenz und Anmerkungen. Second Edition, 2005.