

Programmanalysen und Compilerbau

Wintersemester 2007/8
Prof. Plödereder, Stefan Staiger

1. Übungsblatt

Organisatorische Hinweise

Die Übungen zur Vorlesung Programmanalysen und Compilerbau finden dienstags im Wechsel mit der Vorlesung statt. Die Aufgabenblätter werden jeweils in der Vorlesung verteilt. Zu den Übungsaufgaben werden keine schriftlichen Lösungen ausgegeben. Dieses Übungsblatt wird am 23. Oktober besprochen.

Aktuelle Informationen zur Vorlesung finden Sie im Web:

<http://www.iste.uni-stuttgart.de/ps/Lehre/>

Aufgabe 1.1

Gegeben seien die folgenden Deklarationen:

```
type Rec is record A: Integer; B: Real; end;
type Matrix is array (1..30, 2..21, 3..12) of Rec;
M: Matrix;
```

Objekte vom Typ `Integer` und `Real` benötigen jeweils 4 Bytes. Die Adressierung sei byte-orientiert. Mehrdimensionale Arrays seien *spaltenweise* gespeichert. Das physikalische Layout von Record-Komponenten erfolgt in der Ordnung, in der die Komponenten in der Record-Definition deklariert sind.

- Geben Sie die Byte-Adresse von `M(I, J, K) .B` an. Dabei sei `base` die Anfangsadresse (relativ oder absolut) von `M`.
- Bestimmen Sie den zur Übersetzungszeit berechenbaren Anteil der Adresse von `M(I, J, K) .B`. Welche Berechnungen können erst zur Laufzeit stattfinden? Die Werte von `I`, `J` und `K` seien zur Übersetzungszeit nicht bekannt.

Aufgabe 1.2

Gegeben sei das folgende Programmfragment aus einer Modula-ähnlichen Sprache:

```
PROCEDURE Test (A: Integer; B: Integer);
  VAR I: Integer;
  TYPE Rec = RECORD
    Flags: ARRAY [1..3] OF Boolean;
    Arr:   ARRAY [1..A] OF Integer;
    Arr2:  ARRAY [1..B] OF Integer;
    Status: Boolean;
    Count: Integer;
  END RECORD;
  VAR RObj: Rec;
  K:   Integer

BEGIN ... END Test;
```

Entwickeln Sie eine möglichst effiziente Speicherstruktur für die lokalen Datenstrukturen der Prozedur Test. Die Halde soll nicht verwendet werden. Geben Sie die Relativadressen aller Recordkomponenten an. `Integer` benötigen 4 Byte, `Boolean` 1 Byte. Die Programmiersprache enthält keine Vorschriften über die physikalische Struktur. Zugriffe auf Komponenten, die nicht an einer 8-Bit Wortgrenzen beginnen, sind auf der Zielmaschine teuer. Sie sollen primär die Zugriffszeiten und sekundär den Speicherbedarf optimieren. Erläutern Sie kurz, warum die von Ihnen gewählte Struktur effizient ist. (Hinweis: mehrere Lösungen sind möglich; gefragt ist nach *einer* Lösung.)

Aufgabe 1.3

Die folgende semantische Spezifikation soll bestimmen, ob alle Elemente einer Liste konstante Werte besitzen. In diesem Fall soll für die Liste ein Gesamtwert errechnet werden, der sich aus den Werten der Listenelemente, multipliziert mit der Zehnerpotenz ihrer Position (beginnend bei $10^0 = 1$) berechnet. „literal.Wert“ sei der numerische Wert eines vom Scanner gelieferten Literals.

Z. B.:

- Liste \rightarrow (3, 4, 5) Liste.Konst = true
Liste.Wert = 543
- Liste \rightarrow (5, 6, var) Liste.Konst = false
Liste.Wert = ? (undefiniert)

Syntax	Semantik
Liste \rightarrow (W_Liste)	Liste.Konst := W_Liste.Konst if W_Liste.Konst then Liste.Wert := W_Liste.Acc fi
W_Liste \rightarrow Wert , W_Liste ₁	W_Liste.Konst := W_Liste ₁ .Konst and Wert.Konst if W_Liste.Konst then W_Liste.Acc := W_Liste ₁ .Acc + Wert.Val * W_Liste.Gew fi W_Liste ₁ .Gew := W_Liste.Gew * 10
W_Liste \rightarrow Wert	W_Liste.Konst := Wert.Konst if Wert.Konst then W_Liste.Acc := Wert.Val * W_Liste.Gew fi
Wert \rightarrow Ausdruck	Wert.Val := Ausdruck.Wert
Ausdruck \rightarrow <i>id</i>	Ausdruck.Konst := false
Ausdruck \rightarrow <i>literal</i>	Ausdruck.Wert := literal.Wert

- Welche Attribute sind ererbt (bitte genaue Angaben)?
- Welche Attribute sind zusammengesetzt (synthetisiert)?
- Die semantische Spezifikation ist unvollständig. Fügen Sie Ihre Ergänzungen in die obige Spezifikation ein. Die vorgegebenen Berechnungen sollen dabei nicht geändert werden.
- Stellt die korrigierte Spezifikation eine LAG(1)-Attributierung dar? Begründen Sie Ihre Antwort.

- e. Die Grammatik ist nicht LL(1). Geben Sie den Grund an und formen Sie die Grammatik in eine LL(1)-Grammatik um. Ergänzen Sie auch die semantische Attributierung, um zu gleichen Ergebnissen zu kommen.

Aufgabe 1.4

- a. Welche Terminale und/oder Nichtterminale der Grammatik in Aufgabe 1.3 (ohne die Änderungen in Teilaufgabe 1.3 e.) werden für einen abstrakten Syntaxbaum benötigt, welche können weggelassen werden?
- b. Geben Sie für die Knoten des abstrakten Syntaxbaums Ada-Deklarationen an, die die Attribute und die zum Traversieren benötigten Kanten beschreiben.
- c. Schreiben Sie unter Verwendung dieser Deklarationen Ada-Routinen, die einen abstrakten Syntaxbaum traversieren und alle Attribute berechnen. Verwenden Sie dafür die in Teilaufgabe 1.3 c. vervollständigten Attributregeln.