

## Einführung in den Compilerbau

# Lösungsblatt Nr. 1

von Patrick Elsen, Viola und Michael Matthé

Andreas Koch

Wintersemester 2018-2019

Technische Universität Darmstadt

### Einleitung

Auf diesem Aufgabenblatt sollen Sie sich mit der Matrix and Vector Language, kurz MAVL, vertraut machen. Studieren Sie bitte zunächst die MAVL-Sprachspezifikation, die Sie im Moodle-Kurs der Veranstaltung finden.

### Aufgabe 1.1: MAVL-Syntax

Die MAVL-Sprachspezifikation enthält nur eine informelle Beschreibung der Syntaxelemente der Sprache. In den folgenden Teilaufgaben sollen Sie einige der Syntaxelemente in Produktionen einer kontextfreien Grammatik überführen.

#### Aufgabe 1.1a: Produktionen

Geben Sie Produktionen für die Nichtterminale `mulExpr` (Multiplikations-Operator), `subvectorExpr` (Subvektor-Operator), sowie `recordElementSelectExpr` (Selektion von Record-Elementen) an.

Ein Multiplikationsausdruck ist in der Sprachspezifikation unter § 7.5 *Ternärer Operator* definiert. Ein solcher Ausdruck nimmt `int` oder `float`-Werte als Parameter und ist Linksassoziativ. Also kann man diesen grammatikalisch folgendermaßen definieren.

```
mulExpr ::= (INT | FLOAT) '*' expr
```

Die `subvectorExpr` ist in dem Sprachstandard unter § 7.7.4 *Submatrix und Subvektor* definiert. Hier wird definiert, dass eine solche Beispielsweise als `v{-1:i:1}` geschrieben werden kann, wobei `v` ein Vektor und `i` eine Zahl sein muss. Dieser Ausdruck extrahiert einen Subvektor mit den Elementen  $[i-1, i+1]$ .

```
subvectorExpr ::= '{' expr ':' expr ':' expr '}'
```

Unter § 7.8 *Selektion von Record-Elementen* ist definiert, wie der Syntax funktioniert.

```
recordElementSelectExpr ::= ID '@' ID
```

## Aufgabe 1.1b

Geben Sie Produktionen für die Nichtterminale `primitiveType` (primitive Typen) und `vectorType` (Vektortypen) an.

Die primitiven Typen sind unter § 4.2 *Primitive Datentypen* definiert. Hier sind nur `int`, `float` und `bool` als eingebaute, primitive Typen angegeben. Also könnte eine Grammatik folgendermaßen aussehen:

```
primitiveType ::= 'int' | 'float' | 'bool'
```

Der `vectorType` ist bei § 4.5 *Vektoren* definiert. Ein Vektor muss, mit einem Elementtyp (entweder `int` oder `float`) und einer Länge (positive, ganze Zahl) definiert werden.

```
vectorType ::= 'vector' '<' ('int' | 'float') '>' '[' constExpr ']'
```

## Aufgabe 1.1c

Geben Sie Produktionen für die Nichtterminale `returnStmt` (Rückgabebefehl), `varDecl` Variablen-deklaration), `callStmt` (Aufruf-Befehl, ohne Rückgabewert) sowie `forStmt` (For-Schleife) an.

## Aufgabe 1.2: AST zu MAVL

Abstrakte Syntaxbäume (engl. *Abstract Syntax Trees, AST*) sind eine weitverbreitete Zwischendarstellung, die nur essentielle Informationen enthält und Details der konkreten Syntax einer Programmiersprache abstrahiert.

In dieser Aufgabe zeigen wir Ihnen eine mögliche Repräsentation von MAVL-Code als AST. Die darin verwendeten AST-Knoten korrespondieren auf natürliche Weise mit den in der Spezifikation beschriebenen Syntaxelementen.

## Artikel 1.2a

Geben Sie den zum folgenden AST zugehörigen MAVL-Code an.

Den Syntaxbaum kann man, von oben nach unten und links nach rechts, einfach wie Code lesen.

```
if(id && r > q) {  
    r = -1;  
    q(q, r);  
} else {  
    r = a - (q * d);  
}
```

## Artikel 1.2b

Geben Sie den zum folgenden AST zugehörigen MAVL-Code an.

Antwort.

### **Artikel 1.3: Ausdrücke**

Ausdrücke in typischen Programmiersprachen lassen sich einfach durch mehrdeutige Grammatiken beschreiben, die aber als Grundlage für die syntaktische Analyse ungeeignet sind.

#### **Artikel 1.3a**

Zeichnen Sie den AST für den MAVL-Ausdruck  $q \ \& \ a == b \ \# \ c$ .

Antwort.

#### **Artikel 1.3b**

Zeichnen Sie den AST für den MAVL-Ausdruck  $- \ v2 \ . * \ v1 \ + \ (v2 \ \# \ m) [0]$ .

Antwort.

#### **Aufgabe 1.3c**

Gegeben seien folgende Wertedefinitionen.

Welchen Wert liefert der Ausdruck aus Teilaufgabe 1.3b?

Antwort.