

Übung zur Vorlesung Compiler 1: Grundlage

Prof. Dr. Andreas Koch
Jens Huthmann, Florian Stock



TECHNISCHE
UNIVERSITÄT
DARMSTADT

Wintersemester 11/12
Aufgabenblatt 1

Abgabemodalitäten

Die Aufgabe sind in 2er Gruppen zu bearbeiten. Ihre Lösungen reichen Sie als PDF per E-Mail bis zum 24.11.2011 um 23:59 MET an der Adresse oc@esa.informatik.tu-darmstadt.de ein. Die E-Mail muss als Betreff "Compiler 1 Aufgabe 1" haben. Geben Sie in Ihrem Lösungsdokument den Namen und die Matrikelnummer aller Gruppenmitglieder an.

Aufgabe 1.1 Triangle zu AST

10 Punkte

Zeichnen Sie für das folgende Triangle Programm den abstrakten Syntaxbaum. Halten Sie sich dabei an den Regeln auf Seite 109 und 110 im Buch. Zum Zeichnen können Sie zum Beispiel Visio verwenden, welches Sie im MSDNAA bekommen können.

```
while ( a < 25) do
    a := a - 1
```

Aufgabe 1.2 AST zu Triangle

10 Punkte

Geben Sie für den abstrakten Syntaxbaum auf Seite 3, 4 und 5 das zugehörige Triangle Programm an. Die AST Regeln auf Seite 109 und 110 im Buch wurden um eine Regel für Array erweitert. Diese Regel gehört in die Gruppe der V-name ASTs. Zusätzlich gibt es noch einen SkipCommand welcher einem leeren Befehl entspricht. Ein Beispiel hierzu gibt es auf Seite 21 des Buches (Example 1.9).

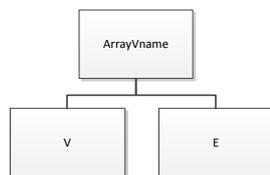


Abbildung 1: Regel für Arrays

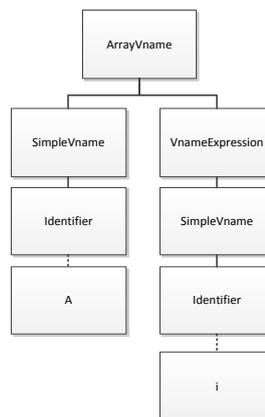


Abbildung 2: Beispiel



Abbildung 3: Leerer Befehl

Aufgabe 1.3 LL(1) Grammatiken

15 Punkte

Welche der gegebenen Grammatiken sind durch einen LL(1) Parser zu verarbeiten? Falls eine Grammatik dies nicht ist, so formen Sie diese entweder um, so dass sie verarbeitet werden kann, oder begründen Sie, warum eine Umformung gegebenenfalls nicht möglich ist.

- a) $G = (\{S, T, B, ID\}, \{if(,), else, fi, true, false, a, b, c\}, P, S), P :$
 $S ::= if (B) S T \mid ID$

$T ::= \text{else } S \text{ fi} \mid \text{fi}$
 $B ::= \text{true} \mid \text{false}$
 $ID ::= a \mid b \mid c$

b) $G = (\{S, A, B\}, \{x, y, z, 0, 1\}, P, S), P :$
 $S ::= A x \mid B y \mid z \mid S S$
 $A ::= B$
 $B ::= 0 \mid 1$

c) $G = (\{S, X, Y\}, \{a, b, c, g, h\}, P, S), P :$
 $S ::= X \mid Y$
 $X ::= aXb \mid g$
 $Y ::= aYc \mid h$

Aufgabe 1.4 LL(k) Grammatiken

10 Punkte

Bestimmen Sie die minimale Größe von k für die gegebene LL(k) Grammatik. Die Grammatik darf nicht umgeformt werden.

$G = (\{S, X\}, \{x, y\}, P, S), P :$
 $S ::= xXxx \mid yXyx$
 $X ::= y|\epsilon$

Aufgabe 1.5 Parser

15 Punkte

Geben Sie die Namen der Java Klassen des AST für die gegebene Grammatik an. Unterstreichen sie hierbei die Klassen welche die Schreibweise des Tokens benötigen. Schreiben Sie dann die benötigten `parseN`-Methoden. Halten Sie sich bei Ihrer Umsetzung an das Beispiel in den Folien bzw. des Buches (Seite 95, Schritt 2). Jedes Terminalsymbol außer denen aus der Produktion `ID` sind als Schlüsselwort zu betrachten.

$G = (\{S, T, B, ID\}, \{\text{while}(_), \text{end}, \text{true}, \text{false}, a, b, c\}, P, S), P :$
 $S ::= \text{while} (B) S E \mid ID$
 $E ::= \text{end}$
 $B ::= \text{true} \mid \text{false}$
 $ID ::= a \mid b \mid c$

Aufgabe 1.6 Scanner

15 Punkte

Nehmen Sie an, dass die lexikalische Grammatik von Mini-Triangle um hexadezimale Literale erweitert wird. Geben Sie hierzu analog zu den Verfahren im Kapitel 4.5 ab Seite 118 im Buch einen passenden Scanner an. Hierbei reicht es aus die `scanToken` Methode zu schreiben. Die Fälle für einzelnen Buchstaben oder Ziffern die keine Sonderfälle betreffen können sie mit `case 'a' ... case 'z'` zusammenfassen. Beachten Sie hierbei dass es passieren kann das Sie von dem Verfahren für einige Regeln leicht abweichen müssen.

`Token ::= Identifier | Operator | IntegerLiteral | HexLiteral | ...`
`Identifier ::= Letter (Letter | Digit)*`
`IntegerLiteral ::= (Digit)+`
`HexLiteral ::= 0x(Digit)+`

Nennen Sie anschliesend das wesentliche Problem dieser Grammatik welches Sie zur Abweichung von dem Verfahren gezwungen hat.

Plagiarismus

Der Fachbereich Informatik misst der Einhaltung der Grundregeln der wissenschaftlichen Ethik großen Wert bei. Zu diesen gehört auch die strikte Verfolgung von Plagiarismus. Weitere Infos unter www.informatik.tu-darmstadt.de/plagiarism

