Übungsblatt 2: Algorithmen und Grammatiken

Abgabe bis 16.11.2015, 09:00 über **Goya**

Die Lösung dieses Übungsblatts soll in Gruppen von je 2 Personen erfolgen. Schreiben Sie die Lösungen in die vorgefertigten Abgabedateien (abgabeX.txt, X ∈ {1,2,3,4}) und geben Sie diese über Goya ab. Für die Besprechung der Aufgabe in den Übungen machen Sie sich bitte Notizen zu Ihren Lösungen.

Aufgabe 1 (2 Punkte)
Betrachten Sie die folgende Grammatik mit Startsymbol S:
(1) `S` ::= `A`
(2) `A` ::= `AA`
(3) `A` ::= `B`
(4) `A` ::= “a”
(5) `B` ::= “b”

Geben Sie eine Ableitung für das Wort abba an. Schreiben Sie dazu in jede Zeile die angewendete Regel, gefolgt von der bisher abgeleiteten Zeichenkette:

Start: S
1: A

Aufgabe 2 (10 Punkte)
Betrachten Sie den folgenden Algorithmus:

```
if a < 0
then
  return 0
else
  return a
end if
```

Der Algorithmus ist ein Wort der Sprache, die durch die folgende Grammatik mit Startsymbol Program beschrieben wird (Nicht-Terminale stehen in serifenloser Schriftart):

(01) Program ::= {Instruction}
(02) Instruction :: = Input | Assignment | Branch | Loop | Return
(03) Input ::= "read(" Variable ")"
(04) Assignment ::= Variable "=" Expression
(05) Expression ::= "("Expression")" | Variable | Digit {Digit} | Expression ArithOperator Expression | "Random()"
(06) ArithOperator ::= "+" | "-" | "*" | "/" | "DIV" | "MOD"
(07) Branch ::= "if " BooleanExpression "then" Program "end if" | "if " BooleanExpression "then" Program "else" Program "end if"
(08) BooleanExpression ::= "not" BooleanExpression | Expression CompOperator Expression
(09) CompOperator ::= "<" | "<=" | "==" | ">=" | ">" | "!="
(10) Loop ::= WhileLoop | ForLoop
(11) WhileLoop ::= "while " BooleanExpression " do " Program " end while"
(12) ForLoop ::= "for " Assignment " to " Expression " do " Program " end for "
(13) Return ::== "return " { Letter | Digit }
(14) Variable ::= Letter { Letter | Digit }
(15) Letter ::= "a" | "b" | "c" | "d" | "e" | "f" | "g" | "h" | "i" | "j" | "k" | "l" | "m" | "n" | "o" | "p" | "q" | "r" | "s" | "t" | "u" | "v" | "w" | "x" | "y" | "z"
(16) Digit ::= "0" | "1" | "2" | "3" | "4" | "5" | "6" | "7" | "8" | "9"

Finden Sie eine Ableitung für das oben angegebene Wort der Sprache. Nennen Sie alle Regeln, die Sie verwendet haben. Schreiben Sie dazu die Nummern der Regeln, getrennt durch Kommata, in die Abgabevorlage (die Ableitung selbst müssen Sie nicht angeben). Tipp: Ersetzen Sie beim Ableiten immer das am weitesten links stehende Nicht-Terminal.
Beispiel: 1, 7, 2

Aufgabe 3 (4 Punkte)
Betrachten Sie den folgenden Algorithmus, gegeben in der Syntax von Aufgabe 2:

```
read(a)
read(b)
if a == 0
then
  return b
else
  while b > 0 do
    if a > b
    then
      a = a - b
    else
      b = b - a
    end if
  end while
end if
return a
```

Dabei sollen die syntaktischen Konstrukte folgende Bedeutung (Semantik) haben:

- read(x): liest eine Eingabe in die Variable x
- if B then P1 else P2 end if: Wertet den boolschen Ausdruck B aus; bei Ergebnis "wahr" wird P1 ausgeführt, sonst P2.
- while B do P end while: Führe P solange aus, wie der boolesche Ausdruck B wahr ist. Prüfe vor jedem Eintritt in P.
- x = y: Weise der Variable x den Wert von y zu, wobei Variablen durch ihre Werte ersetzt werden.
- return x: Gibt x zurück

Geben Sie für die folgenden Eingaben eine *Variablenbelegungstabelle* an. Die Spalten der Tabelle seien die Variablen; jede Zeile stehe für eine Zuweisung und die nach der Zuweisung vorliegenden Werte der Variablen. Trennen Sie die Werte in einer Zeile mit einem Komma, schließen Sie jede Zeile mit einem Semikolon ab.
Eingaben:

(a) a = 0, b = 11
(b) a = 49, b = 14

Aufgabe 4 (4 Punkte)
Geben Sie einen Algorithmus in Pseudocode (Syntax s. Aufgabe 2) an, der als Eingabe eine natürliche Zahl *z* mit genau *n* Stellen und eine natürliche Zahl 1 <= *k* <= *n* erhält, und als Ausgabe die *k*-te Ziffer von *z* (von rechts) zurückgibt.

Beispiel: Zu der Eingabe *z* = 8924 und *k* = 3 soll der Algorithmus die Ziffer 9 zurückgeben.

Vervollständigen Sie das vorgegebene Fragment:

```
read(z)
read(k)
…
return ziffer
```