Letzter Abgabetermin: Mittwoch, 27.01.2016, 18^{00} Uhr

Coco/R (24 Punkte)

2. Komplexitätsanalyse von MicroJava-Code (24 Punkte)

Implementieren Sie mit Coco/R ein Werkzeug zur Analyse der Komplexität von MicroJava-Code. Verwenden Sie dafür von unserer Website die EBNF-Form der MicroJava-Grammatik, die sie einfach in den Produktionen-Teil der Coco/R-Eingabedatei übernehmen können. Definieren Sie Zeichen, Terminalsymbole und Kommentarregeln nach den Informationen aus den Unterlagen (Abschnitt 4.2 Syntax). Attributieren Sie die schließlich die Grammatik, um nach folgenden Regeln für jede Methode ein Komplexitätsmaß zu berechnen:

- Jede Anweisung zählt einen Punkt
  - *while*-Anweisungen zählen zwei Punkte zusätzlich (also insgesamt 3 Punkte)
  - ein *else*-Zweig bei einem *if*-Konstrukt zählt einen zusätzlichen Punkt
- Die Schachtelung von Anweisungen wird durch Multiplikation mit einem Faktor $f = 1.5$ berücksichtigt, was zu einer exponentiell steigenden Gewichtung führt (1, 1.5, 2.25, 3.38, 5.06, 7.09, …). Als Schachtelung gilt:
  - Zweige einer *if*-Anweisung (mit oder ohne *{ }*)
  - Schleifenkörper
  - Blöcke, die keinen *if*-Zweig oder Schleifenkörper darstellen

Abgabe und Hinweise

- Die Abgabe der Übungen muss elektronisch erfolgen. Geben Sie folgende Dateien ab:
- Elektronisch in das Repository: ATG-Datei und Java-Dateien mit der Lösung.
- svn://ssw.jku.at/2015W/UB/k<MatrNr>/branches/UE8
- Coco/R kann steht auf http://ssw.jku.at/Coco/#Java zur Verfügung. Sie benötigen die Dateien Scanner.frame, Parser.frame und Coco.jar. Um den Scanner und Parser zu erzeugen, führen Sie in der Konsole aus: `java -jar Coco.jar mygrammer.atg`

Rechenbeispiel:

```
void primes (int n)
  int[] numbers;
  int i, j;
{
  numbers = new int[n];                 1
  i = 0;                                1
  while(i < n) {                        3
    if(1 < i && numbers[i] == 0) {      1
      print(i);                         1
      j = i;                            1
      while(j < n) {                    3
        numbers[j]++;                   1  ] (1 + 1) * 1.5
        j += i;                         1  ] = 3
      }
    } else {                            1
      print('.');                       1  ] 1 * 1.5 = 1.5
    }
    i++;                                1
  }
}
```

*16.5 * 1.5 = 24.75*

29.75

Geben Sie in Ihrem Werkzeug die Komplexität jeder Methode und des Programms (d.h. die Summe der Komplexitäten) aus. Beispielausgabe:

```
Method "fib" has complexity 5.00
Method "primes" has complexity 29.75
Method "main" has complexity 2.00
Program "Numbers" has complexity 36.75
```

Geben Sie ein Testprogramm ab, das eine MicroJava-Datei lesen kann und die Komplexität ausgibt.