Blatt 03: ANTLR

Carsten Gips, BC George (HSBI)

Zusammenfassung

Ziel dieses Aufgabenblattes ist die Erstellung eines einfachen *Pretty Printers* für eine einfache fiktive Sprache mit Expressions und Kontrollstrukturen.

Dazu werden Sie eine passende kontextfreie Grammatik definieren mit Lexer- und Parser-Regeln und dabei auch übliche Vorrangregeln beachten.

Für diese Grammatik erstellen Sie mit Hilfe von ANTLR einen Lexer und einen Parser, die zu einem Eingabeprogramm einen Parse-Tree erzeugen.

Den im Parse-Tree repräsentierten Code des Eingabeprogramms können Sie mit Hilfe einer Traversierung konsistent eingerückt wieder auf der Standardausgabe ausgeben - das ist der *Pretty Printer*.

Sie werden merken, dass viele Strukturen im Parse-Tree für diese Aufgabe nicht relevant sind und den Baum mit einer weiteren Traversierung in einen vereinfachten Baum, den sogenannten Abstract-Syntex-Tree (*AST*), transformieren und diesen erneut als formatierten Code auf der Konsole ausgeben.

Methodik

Nutzen Sie das Starter-Projekt in der Vorgabe.

Laden Sie sich das Projekt herunter, binden Sie es in Ihre IDE ein und vergewissern Sie sich, dass alles funktioniert: Führen Sie das enthaltene Programm aus, ändern Sie die mitgelieferten Beispielgrammatik.

Bauen Sie dann Ihre Grammatik für die Aufgabe schrittweise auf. Testen Sie diese mit Hilfe der Beispielprogramme der Zielsprache (s.u.) und überlegen Sie sich selbst weitere Code-Schnipsel, die Sie mit Ihrem Parser einlesen bzw. die Ihr Parser zurückweisen sollte.[1] Es empfiehlt sich, in dieser Phase mit dem ANTLR-Plugin für IntelliJ zu arbeiten.

Erkunden Sie dann die Strukturen Ihres Parse-Trees. Diese sind an die Regeln Ihrer Grammatik gekoppelt und sind deshalb so individuell wie Ihre Grammatik. Mit einer Traversierung des Baumes können Sie die gewünschte Ausgabe programmieren und auch die Erstellung des vereinfachten Baumes (AST).

[1] Um den Text lesbar zu halten, wird hier oft nur von "Parser" gesprochen - gemeint ist aber die gesamte auf diesem Blatt zu erstellende Toolchain: Lexer - Parser - AST - Ausgabe.

Sprachdefinition

Ein Programm besteht aus einer oder mehreren Anweisungen (*Statements*).

Anweisungen (*Statements*)

Eine Anweisung ist eine einzeilige Befehlsfolge, beispielsweise eine Zuweisung oder eine Operation. Sie muss immer mit einem Newline abgeschlossen werden. Eine Anweisung hat keinen Wert.

```
a := 10 - 5  # Zuweisung des Ausdruckes 10-5 (Integer-Wert 5) an die Variable a
b := "foo"   # Zuweisung des Ausdrucks "foo" (String) an die Variable b
```

Kontrollstrukturen (s.u.) zählen ebenfalls als Anweisungen.

Ausdrücke (*Expressions*)

Die einfachsten Ausdrücke sind Integer- oder String-Literale. Variablen sind ebenfalls Ausdrücke. Komplexere Ausdrücke werden mit Hilfe von Operationen gebildet, dabei sind die Operanden jeweils auch wieder Ausdrücke. Ein Ausdruck hat/ergibt immer einen Wert.

Die Operatoren besitzen eine Rangfolge, um verschachtelte Operationen aufzulösen. Sie dürfen daher nicht einfach von links nach rechts aufgelöst werden. Die Rangfolge der Operatoren entspricht der üblichen Semantik (vgl. Java, C, Python).

Aufgaben

Grammatik (4P)

Definieren Sie für die obige Sprache eine geeignete ANTLR-Grammatik.

Sie werden sowohl Lexer- als auch (rekursive) Parser-Regeln benötigen. Beachten Sie die üblichen Vorrangregeln für die Operatoren, orientieren Sie sich hier an Sprachen wie Java oder Python oder C.

Pretty Printer (3P)

Erzeugen Sie mithilfe der Grammatik und ANTLR einen Lexer und Parser. Damit können Sie syntaktisch korrekte Eingabe-Programme in einen Parse-Tree überführen.

Programmieren Sie eine Traversierung Ihres Parse-Trees, in der Sie syntaktisch korrekte Programme konsistent eingerückt ausgeben können.

Jede Anweisung soll auf einer eigenen Zeile stehen. Die Einrückung soll mit Leerzeichen erfolgen und konsistent sein. Sie brauchen keine Begrenzung der Zeilenlänge implementieren.

Demonstrieren Sie die Fähigkeiten an mehreren Beispielen mit unterschiedlicher Komplexität.

Beispiel:

Aus

Last modified: *f7ac9d2 (reformat using shorter lines, 2025-08-09)*