PGdP Tutorium: Fünfte Stunde

Benedikt Werner

München, 21. November 2017

Hausaufgaben

- Die Hausaufgaben diese Woche sind etwas aufwändiger
  - Rechtzeitig anfangen
  - Aufgabenstellung genau lesen
  - Variablenamen sinnvoll benennen und evtl. Kommentare schreiben
  - Halbe Lösungen bringen auch Punkte

Aufgabe 5.1

Folgende Programmstücke sind semantisch äquivalent:

```
int x = read(); int y = x/x;
```

und

```
int x = read(); int y = 1;
```

Falsch

Lösung: Die Methode `read()` könnte die Zahl 0 zurückliefern, womit im ersten Programmstück eine `ArithmeticException` (division by zero) geworfen würde, was im zweiten Fall nicht passieren kann. Somit sind die Programme nicht semantisch äquivalent.

Aufgabe 5.1

Der Ausdruck `0.3 == 0.1 + 0.1 + 0.1` evaluiert zu **`true`**

Falsch

Lösung: Mathematisch betrachtet sollte der Ausdruck wahr sein. Die Gleitkommazahl 0.1 kann in Java aber nicht exakt dargestellt werden. D.h. es wird nur mit einer Approximation/Annäherung der Zahl 0.1 gerechnet. Somit kann das Ergebnis aber auch nicht exakt sein. Als Lehre nehmen wir mit, dass Gleitkommazahlen im Generellen nicht auf Gleichheit überprüft werden sollten, sondern ob diese in einem *Wertebereich* liegen.

Aufgabe 5.1

Mit dem 32-bit-Datentyp `float` lassen sich größere Zahlen darstellen als mit dem 64-bit-Datentyp `long`

Wahr

Lösung: Obwohl für den Datentyp `float` weniger Bits zur Verfügung stehen als in einem `long`, können damit größere Zahlen dargestellt werden. Das liegt an der Darstellung von Gleitkommazahlen. Diese werden mittels Vorzeichen, Mantisse und Exponent dargestellt. Durch die Interpretation einiger Bits als Exponenten lassen sich somit große Zahlen darstellen.

Aufgabe 5.1

Jeder 32-bit-`Integer` kann in einem 32-bit-`Float` dargestellt werden.

Falsch

Lösung: Try it out with the `Integer` 16777217

Aufgabe 5.1

Der Ausdruck `(x + y) - y == x` evaluiert immer zu `true` unter der Annahme, dass `x` und `y` vom selben Zahlentyp sind.

Falsch

Lösung: Für Gleitkommazahlen gilt dies nicht. Wenn `y` den Wert `NaN` oder `POSITIVE_INFINITY` bzw. `NEGATIVE_INFINITY` hat, muss die Gleichheit nicht gelten. Ein weiteres Argument ist ähnlich zu der Aufgabe weiter oben. Versuchen Sie es mit den Werten `0.3` für `x` und `0.1` für `y`.

Aufgabe 5.1

Angenommen, `x` und `y` sind vom Typen `int`, dann sind folgende Ausdrücke semantisch äquivalent:

```
x + 10 > y + 10
```

und

```
x > y
```

Falsch

Lösung: Angenommen, dass `x + 10` zu einem *Integer-Overflow* führt und `y + 10` nicht, dann sind die Ausdrücke nicht äquivalent. Beispiel: `x = Integer.MAX_VALUE - 1; y = 0;` **Achtung:** Treten zwei Integer-Overflows auf, dann sind die Ausdrücke wieder äquivalent!

Aufgabe 5.1

Jede Menge von Wörtern, die sich mit einer kontextfreien Grammatik beschreiben lässt, die *nicht* rekursiv ist, lässt sich auch mit einem regulären Ausdruck beschreiben und umgekehrt.

Wahr

Aufgabe 5.1

Die Grammatik mit den Regeln

S ::= aAb

A ::= ab | S

ist rekursiv.

Wahr

Aufgabe 5.1

Der Ausdruck `2 + 5 + ">=" + 1 + 1` evaluiert zu `"7>=11"`

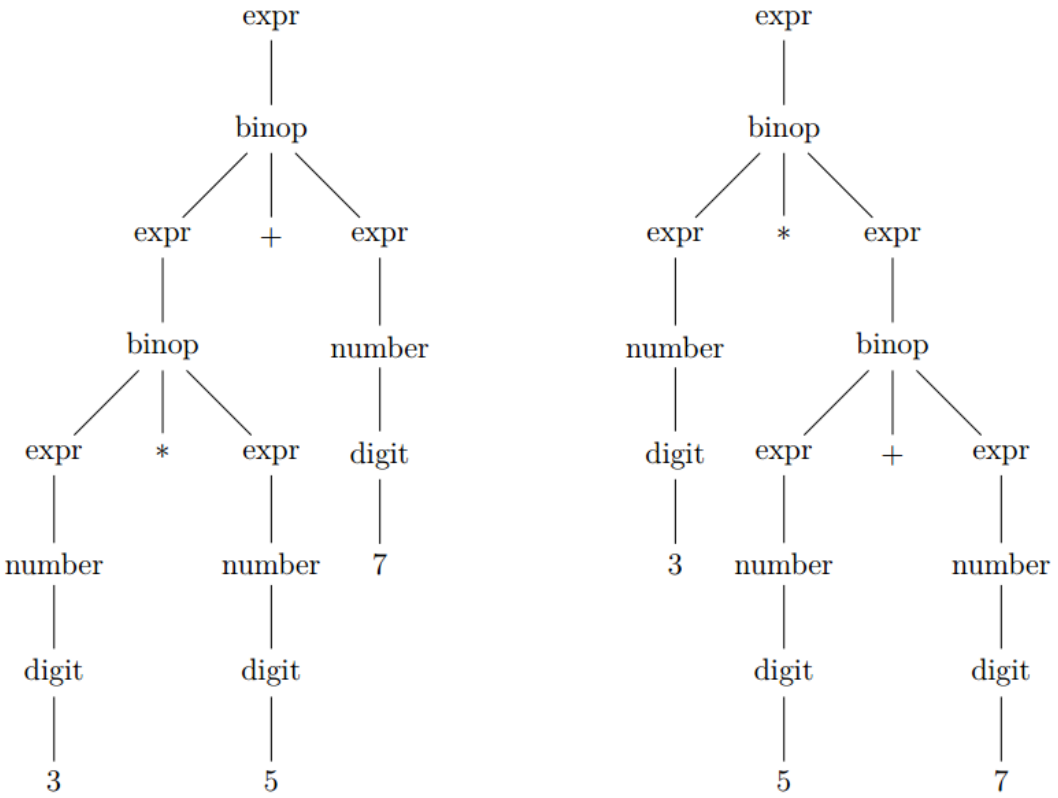
Wahr

Lösung: *Operator Overloading* is evil! Machen Sie sich mit der Auswertungsreihenfolge von Ausdrücken bekannt: `((((2 + 5) + ">=") + 1) + 1)`. Vereinfacht gesagt wird bei Java wie folgt ein Additionsausdruck ausgewertet: Zahl + Zahl ergibt eine Zahl, Zahl + String oder String + Zahl ergibt einen String.

Aufgabe 5.1

Gegeben ist folgender Ausdruck `3*5+7`. Ist der Syntaxbaum/Ableitungsbaum, der mithilfe der MiniJava-Grammatik erzeugt werden kann, eindeutig?

Nein

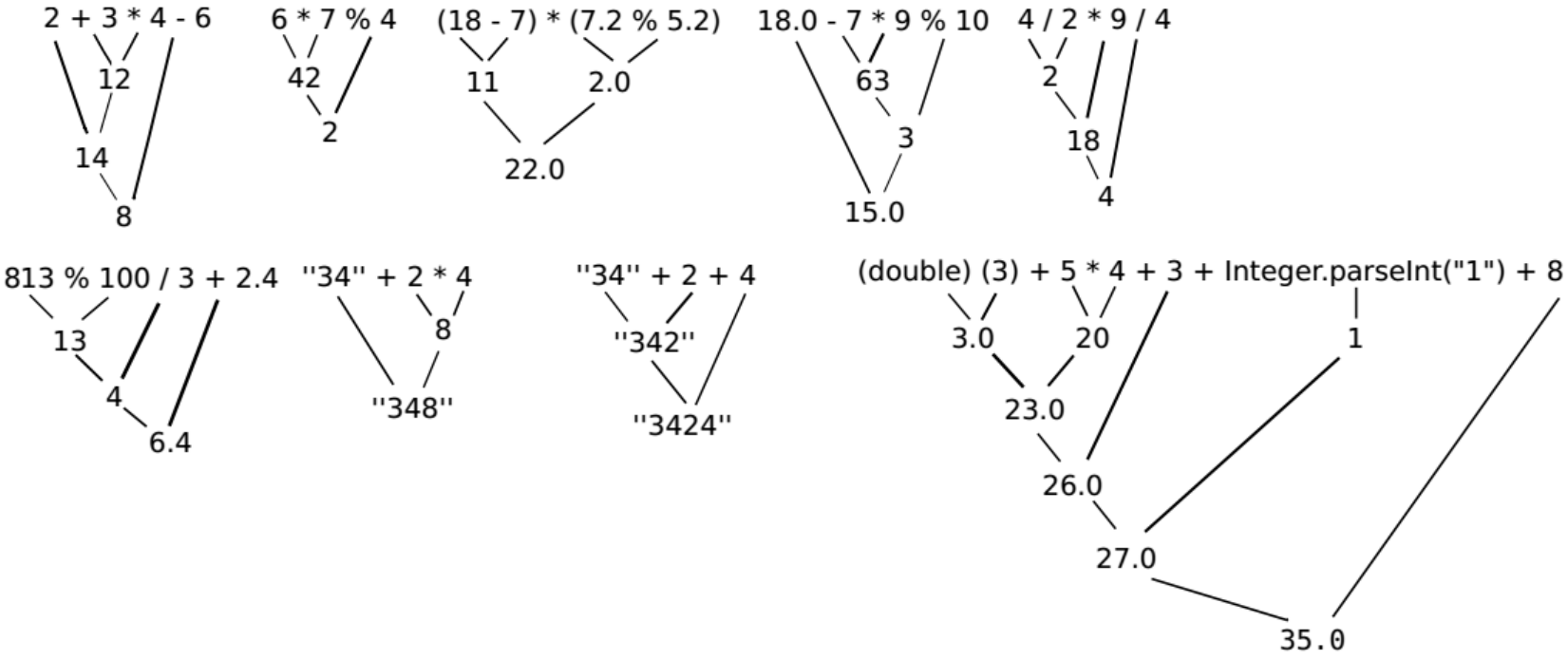


Aufgabe 5.2

- Geben Sie zu jedem Ausdruck den Auswertungsbaum an. Achten Sie darauf, dass anhand des Wertes auf den Typ geschlossen werden kann (String `"hallo"`, Integer `5`, Double `2.0`).
- Ändern Sie jeden Ausdruck (nur falls nötig) so ab, dass der Ausdruck den Typen
  - Integer,
  - Double,
  - String,

  hat. Verwenden Sie dabei nur die Casts `(int) (...)`, `(double) (...)` und die beiden Methoden `Integer.parseInt(String s)`, `Integer.toString(int i)`. Welche Werte haben nun die Ausdrücke? Beachten Sie, dass es verschiedene Möglichkeiten gibt, die Ausdrücke anzupassen, und sich daraus verschiedene Werte ergeben können.

Aufgabe 5.2


2 + 3 * 4 - 6
12
14
8
6 * 7 % 4
42
2
(18 - 7) * (7.2 % 5.2)
11
2.0
22.0
18.0 - 7 * 9 % 10
63
3
15.0
4 / 2 * 9 / 4
2
18
4
813 % 100 / 3 + 2.4
13
4
6.4
''34'' + 2 * 4
8
''348''
''34'' + 2 + 4
''342''
''3424''
(double) (3) + 5 * 4 + 3 + Integer.parseInt("1") + 8
3.0
20
1
23.0
26.0
27.0
35.0