Auswertungsregeln

Einführung in die Programmierung

Johannes Brauer

21. 11. 2020

Syntax und Semantik einer mathematischen Formelsprache

Die Auswertung von Ausdrücken in *Racket* basiert auf der Anwendung einfacher äquivalenter Umformungen, wie sie aus der Schulmathematik bekannt sein sollten.

Betrachten wir folgende mathematische Funktion:

$$bf(a, b) = a \cdot a + 2ab + b \cdot b \tag{1}$$

Halten wir zunächst fest, dass diese Funktionsdefinition in einer definierten mathematischen Formelsprache aufgeschrieben ist. Die Sprache definiert einige grammatikalische Regeln, die beim Aufschreiben von Formeln einzuhalten sind. Diese Regeln bezeichnet man auch als die *Syntax* der Sprache.

Syntax der mathematischen Formelsprache

Sie legt z. B. fest, dass

- Konstanten (Zahlen bzw. Ziffernfolgen),
- Variablen (a, b) und
- bestimmte Operationszeichen (z. B. $+$, $-$ oder der Bruchstrich) verwendet werden dürfen.

Zu den Syntaxregeln gehört aber z. B auch, dass

- links und rechts von Operationszeichen $+$ jeweils ein Operand stehen muss und dass
- die Zeichenfolge $2ab$ als Abkürzung für $2 \cdot a \cdot b$ geschrieben werden darf, das Operationszeichen $\cdot$ also in bestimmten Situationen weggelassen werden darf.

Die Syntaxregeln der Sprache sagen nun aber zunächst nichts darüber aus, was eine syntaktisch korrekte Zeichenfolge bedeutet. Eine syntaktisch nicht korrekte Zeichenfolge – wie z. B. 3+ – ist von vornherein bedeutungslos. Die Definition der Bedeutung syntaktisch korrekter Zeichenfolgen bezeichnet man als die *Semantik* der Sprache.

Semantik der mathematischen Formelsprache

Sie bestimmt z. B., dass

- das Operationszeichen $+$ für die Addition steht und
- der Ausdruck $3 + 5$ äquivalent zu 8 ist.

Man beachte, dass dabei vorausgesetzt wird, dass die Semantik der Addition bereits anderswo definiert ist. Die Addition natürlicher Zahlen ist in der Mathematik z. B. durch die bekannten Peano-Axiome definiert. Darauf soll aber hier nicht näher eingegangen werden. Wir setzen die Wohldefiniertheit der Semantik der mathematischen Formelsprache als gegeben voraus.

Ein wichtiger Bestandteil der Semantik der mathematischen Formelsprache sind die Regeln, die erlauben, Teile einer Formel durch äquivalente, einfachere Teile zu ersetzen. Beinhaltet dieser Prozess auch das Ersetzung von Variablen durch Konstanten, bezeichnet man ihn auch als *Auswertung* einer Formel oder eines

Ausdrucks. Dabei werden die Ersetzungsregeln solange angewendet, bis keine weitere Regel mehr anwendbar ist.

Aus der Mathematik wissen wir, dass die Gleichung

$$a \cdot a + 2ab + b \cdot b = (a + b)(a + b) \tag{2}$$

gilt, das Ersetzen der linken durch die rechte Seite also eine äquivalente Umformung darstellt. Hierbei handelt es sich aber nicht um eine Auswertung, denn erstens werden hier die Variablen nicht durch Konstanten ersetzt und zweitens ist auch nicht klar, welcher der beiden Ausdrücke der einfachere ist.

Eine Auswertung könnte nun stattfinden, wenn wir danach fragen, welchen Wert die Funktion $bf(a, b)$ hat, wenn z. B. a durch 3 und b durch 4 ersetzt werden. In der folgenden Gleichung ist diese Ersetzung vorgenommen worden:

$$bf(3, 4) = 3 \cdot 3 + 2 \cdot 3 \cdot 4 + 4 \cdot 4 \tag{3}$$

Um den Ausdruck auf der rechten Seite korrekt auswerten zu können, müssen weitere semantische Regeln benutzt werden; hier insbesondere die Regel: Punktrechnung geht vor Strichrechnung.

Damit könnte nun der Ausdruck wie folgt äquivalent umgeformt werden:

$$\begin{aligned} bf(3, 4) &= 3 \cdot 3 + 2 \cdot 3 \cdot 4 + 4 \cdot 4 && (4)\\ &= 9 + 2 \cdot 3 \cdot 4 + 4 \cdot 4 && (5)\\ &= 9 + 6 \cdot 4 + 4 \cdot 4 && (6)\\ &= 9 + 24 + 4 \cdot 4 && (7)\\ &= 9 + 24 + 16 && (8)\\ &= 33 + 16 && (9)\\ &= 49 && (10) \end{aligned}$$

Syntax und Semantik von *Racket* (vorläufig)

Funktionale Programmiersprachen zeichnen sich u. a. dadurch aus, dass sie sich hinsichtlich ihrer Semantik sehr strikt an der Mathematik orientieren. Das bedeutet z. B., dass die Auswertung von Ausdrücken durch eine Folge von äquivalenten Umformungen definiert ist.

Betrachten wir dies exemplarisch anhand der Funktion f aus dem vorhergehenden Abschnitt. Die Syntax von *Racket* verlangt zunächst das wir die mathematische Funktionsdefintion

$$bf(a, b) = a \cdot a + 2ab + b \cdot b \tag{11}$$

umschreiben. Wollen wir in *Racket* eine Funktion mit Namen `bf` und den Parametern `a` und `b` definieren, sieht das so aus:

```
(define bf (lambda [a b] ... ))
```

Wollen wir nun die rechte Seite der Funktionsdefinition nach *Racket* übertragen, müssen wir die Syntaxregel von *Racket* beachten, wonach Grundform eines (arithmetischen) Ausdrucks (in *Racket*) immer so aussieht:

```
(operator operand-1 operand-2 ... operand-n)
```

Damit erhalten wir die vollständige Funktionsdefinition:

```
(define bf
  (lambda [a b]
    (+ (* a a)
       (* 2 a b)
       (* b b))))
```

Die Semantik von *Racket* definiert nun insbesondere, wie Ausdrücke, d. h. Funktionsanwendungen, ausgewertet werden. Wie in der mathematischen Formelsprache können durch die Anwendung einfacher Regeln Ausdrücke in *Racket* in einfachere Ausdrücke äquivalent umgeformt werden. Die Frage lautet also z. B., wie aus der Funktionsanwendung

```
(bf 3 4)
```

das Resultat `49` entsteht.

Betrachten wir dazu eine verallgemeinerte Definition einer Funktion `f` mit den formalen Parametern `x-1 ... x-n` und der Berechnungsvorschrift `exp`:

```
(define f (lambda [x-1 ... x-n] exp))
```

Für die Anwendung der Funktion gilt dann die folgende Ersetzungsregel.

Ersetzungsmodell für Funktionsanwendungen:

- Formal: Mit `(define f (lambda [x-1 ... x-n] exp))` gilt:

$$(f\ v_1\ ...\ v_n) = exp^{x_i}_{v_i} \tag{12}$$

  Dabei bedeutet $exp^{x_i}_{v_i}$, dass in `exp` jedes Auftreten eines x_i durch v_i ersetzt wird. (x_i entspricht `x-i`.)

- Werte den Rumpf der Funktion aus, wobei jeder formale Parameter durch das korrespondierende ausgewertete Argument ersetzt wird.

- Wenden wir diese Regel auf die Anwendung der Funktion `bf` an:

```
(bf 3 4) = (+ (* 3 3)
              (* 2 3 4)
              (* 4 4))
```

- Wie in der Mathematik gehen wir davon aus, dass die Addition und Multiplikation auch in *Racket* bereits definiert sind.

- D. h., weitere Auswertungsschritte erfolgen analog zur mathematischen Formelsprache. Man beachte dabei, dass aufgrund der Klammerstruktur in *Racket* keine Vorrangregel für die Multiplikation und Addition erforderlich ist:

```
(bf 3 4) = (+ (* 3 3)
              (* 2 3 4)
              (* 4 4))
         = (+ 9
              (* 2 3 4)
              (* 4 4))
         = (+ 9
              24
              (* 4 4))
         = (+ 9 24 16)
         = 49
```

(s. Aufgabe 9)

Zusammenfassung

- Die **Syntax** einer Programmiersprache legt zulässige Elemente und Strukturen fest.
- Die **Semantik** einer Programmiersprache legt die Bedeutung syntaktisch korrekter Ausdrücke fest.
- Die Semantik funktionaler Programmiersprachen orientiert sich an der Mathematik.
- Ausdrücke werden schrittweise über **äquivalente Umformungen** durch das **Ersetzungsmodell für Funktionsanwendungen** vereinfacht.