

Syntax und Semantik – ein Tutorial

Jan Sürmeli
suermeli@googlemail.com

3. März 2015

1 Syntax vs. Semantik: Ideen haben und hinschreiben

In diesem Abschnitt untersuchen wir den Unterschied zwischen Syntax und Semantik. Kurz gesagt ist Syntax die *Art*, Dinge hinzuschreiben, und Semantik die *Bedeutung* des Geschriebenen. Den Unterschied untersuchen wir zunächst anhand einiger konkreter Beispiele. Anschließend formalisieren wir die Begriffe der Syntax und der Semantik auf eine recht abstrakte Weise.

Erstes Beispiel: Programmiersprachen. Bevor wir ein Programm schreiben, müssen wir uns für eine *Programmiersprache* entscheiden. Damit wir ein Programm in einer Programmiersprache L hinschreiben können, müssen wir die *Syntax* von L kennen: Ein Compiler (oder Interpreter) für die Sprache L akzeptiert nur solche Quelltexte, die der Syntax von L folgen. Die Syntax von Programmiersprachen umfasst unter anderem die Schlüsselwörter und die Namen eingebauter Datentypen. Ein Programm in L ist jedoch (meist) mehr als eine bloße Aneinanderreihung von Schlüsselwörtern. Tatsächlich ist festgelegt, welche *Zeichenketten* in welcher Reihenfolge und Kombination stehen dürfen. Die Syntax einer Programmiersprache L beantwortet also die Frage, *ob ein gegebener Quelltext Q ein Programm in L ist*. Haben wir ein Programm Q in L geschrieben, möchten wir dieses meist ausführen. Dazu geben wir Q in einen Compiler oder Interpreter, um Q *auszuführen*. Die Semantik von L legt fest, was passiert, wenn wir Q ausführen. Dazu legt die Semantik unter anderem fest, welche Schlüsselwörter welche Bedeutung haben. Genauer legt sie genau fest, *welche Bedeutung Q bezüglich der Programmiersprache L hat*. Ein Quelltext Q kann dabei ein Programm gleich zweier Programmiersprachen L_1 und L_2 sein, und in L_1 etwas völlig anderes bedeuten als in L_2 . Ein gutes Beispiel dafür ist das Gleichheitszeichen, das in manchen Programmiersprachen *Gleichheit* und in anderen *Zuweisung* bedeutet. Dieser Punkt ist äußerst wichtig: Selbst wenn wir die Syntax einer Programmiersprache kennen, kennen wir nicht automatisch auch ihre Semantik. Auch wenn wir Syntax und Semantik einer Programmiersprache häufig gleichzeitig lernen, sind es tatsächlich zwei unterschiedliche Gedanken.

Zweites Beispiel: Mathematische Funktionen In der Schule haben wir gelernt, dass wir eine Funktion f hinschreiben können, in dem wir erst

$$f : \mathbb{R} \rightarrow \mathbb{R} \quad (1)$$

und dann

$$f(x) = x^2 + 6x + 2 \quad (2)$$

schreiben. Wir haben auch gelernt, dass wir damit eine Funktion f hinschreiben, die eine Parabel im zweidimensionalen Koordinatensystem beschreibt, oder etwas abstrakter: Eine Menge P_f von Punkten im zweidimensionalen Koordinatensystem, wobei für jede reelle Zahl x genau ein Punkt in P_f existiert, dessen X -Koordinate x ist. Außerdem ist die Y -Koordinate dieses einen Punktes genau $f(x)$, also das Ergebnis davon, x in $x^2 + 6x + 2$ einzusetzen. Angenommen, wir schreiben noch folgendes dazu:

$$g : \mathbb{R} \rightarrow \mathbb{R} \quad (3)$$

$$g(x) = (x + 3)^2 - 7 \quad (4)$$

Damit haben wir eine Funktion g beschrieben – eine Menge P_g von Punkten im zweidimensionalen Koordinatensystem. Gucken wir ein bisschen länger auf die P_f und P_g stellen wir fest, dass sie *gleich* sind: Für jede reelle Zahl x hat der Punkt in P_f mit X -Koordinate x die gleiche Y -Koordinate wie der Punkt in P_g mit X -Koordinate x . Was ist hier passiert? Wir haben die Punktmenge P_f einmal durch die Zeichenketten (1) und (2) beschrieben. Anschließend haben wir dieselbe Menge von Punkten durch die Zeichenketten (3) und (4) beschrieben. Ähnlich zum vorherigen Beispiel haben wir eine Syntax gelernt, um Funktionen zu beschreiben, und gelernt, was eigentlich die Bedeutung solch einer Beschreibung ist. Diesmal haben wir den Fall kennen gelernt, dass zwei unterschiedliche Beschreibungen dieselbe Bedeutung haben können.

Eine abstrakte Definition für Syntax und Semantik Eine *Syntax* legt stets ein Universum erlaubter *Zeichenketten* fest – also nichts anderes als eine formale Sprache über einem Alphabet. Eine *Semantik* weist jeder syntaktisch erlaubten Zeichenkette eine *Bedeutung* zu. Eine Zeichenkette kann zu mehr als nur einer Syntax passen. Zwei Zeichenketten können bezüglich derselben Semantik dieselbe Bedeutung haben.

2 Signaturen und Strukturen

In diesem Abschnitt definieren wir konkret die Begriffe der Syntax und Semantik in der Welt der Algebra: Wir legen Syntax durch Signaturen, Semantik durch Strukturen fest. Dazu wenden wir uns zunächst der Semantik – also den Strukturen – zu. Anschließend betrachten wir die Syntax – also die Signaturen.

2.1 Strukturen

Wenn wir an die Schulmathematik denken, dreht sie sich doch größtenteils um Funktionen auf reellen Zahlen. Manchmal betrachten wir bestimmte Teilmengen der reellen Zahlen, wie die positiven reellen Zahlen oder die rationalen Zahlen. Manche dieser Funktion berechnen zu einer reellen Zahl eine andere reelle Zahl, wie zum Beispiel das Quadrieren. Andere Funktionen berechnen zu zwei reellen Zahlen eine reelle Zahl wie zum Beispiel die Addition.

Wir haben jedoch auch andere Funktionen kennen gelernt, zum Beispiel die Ableitung einer Funktion: Die Ableitung einer Funktion f über den reellen Zahlen ist die Funktion f' , die jedem $x \in \mathbb{R}$ die Steigung der Funktion f im Punkt x zuordnet.

Was macht eine Funktion aus? Diese Frage wird in unterschiedlichen Werken unterschiedlich beantwortet. In diesem Tutorial wollen wir uns auf folgendes einigen:

1. Eine Funktion hat eine Stelligkeit n , wobei $n \in \{0, 1, 2, \dots\}$.
2. Eine n -stellige Funktion hat einen Definitionsbereich $A_1 \times \dots \times A_n$, wobei $A_1 \dots A_n$ Mengen sind.
3. Eine Funktion hat einen Wertebereich A .
4. Eine 0-stellige Funktion f heißt *Konstante* und beschreibt einen Wert aus A .
5. Eine n -stellige Funktion f mit $n \geq 1$ weist jedem Element aus dem Definitionsbereich $A_1 \times \dots \times A_n$ ein Element aus dem Wertebereich A zu.

Jetzt sind wir bereit den Begriff der *Struktur* einzuführen. Eine Struktur S besteht aus endlich vielen Mengen $B_1 \dots B_\ell$ sowie endlich vielen Funktionen $f_1 \dots f_k$, so dass für jedes i mit $1 \leq i \leq k$ gilt: f_i ist eine n -stellige Funktion mit Definitionsbereich $A_1 \times \dots \times A_n$ und Wertebereich A , so dass

1. $\{A_1, \dots, A_n\} \subseteq \{B_1 \dots B_\ell\}$ und
2. $A \in \{B_1, \dots, B_n\}$.

Wir benutzen für Strukturen gerne eine Tupel-Schreibweise: Sind B_1, \dots, B_n und f_1, \dots, f_k bekannt, schreiben wir S auch als $(B_1, \dots, B_n; f_1, \dots, f_k)$.

2.2 Signaturen

Als ein einführendes Beispiel betrachten wir drei Strukturen:

1. $S_1 = (\mathbb{N}; \text{succ}, 0)$, wobei \mathbb{N} die Menge der natürlichen Zahlen und succ die einstellige Funktion ist, die jeder natürlichen Zahl ihren Nachfolger zuordnet.
2. $S_2 = (\mathbb{R}; \text{abs}_{\mathbb{R}}, 17)$, wobei \mathbb{R} die Menge der reellen Zahlen und $\text{abs}_{\mathbb{R}}$ die einstellige Funktion ist, die jeder reellen Zahl ihren Betrag zuordnet.

3. $S_3 = (\mathbb{F}; \text{diff}, \text{quadriere})$, wobei \mathbb{F} die Menge der 1-stelligen Polynom-Funktionen¹ über \mathbb{R} , \mathbb{R} die Menge der reellen Zahlen, diff die Funktion ist, die jeder Funktion aus \mathbb{F} seine Ableitung zuordnet, und quadriere jeder Zahl ihr Quadrat zuordnet.

Wir beobachten, dass die drei Strukturen S_1 , S_2 und S_3 folgende Gemeinsamkeit aufweisen: Jede der drei Strukturen besteht aus einer Menge zusammen mit einer einstelligen Funktion und einer Konstante. Formal sagen wir, dass S_1 , S_2 und S_3 dieselbe *Signatur* haben. Bevor wir das konkretisieren, betrachten wir ein etwas komplizierteres Beispiel:

1. $S_4 = (\mathbb{Q}, \mathbb{Z}; \text{rundeAuf})$, wobei \mathbb{Q} die Menge der rationalen Zahlen, \mathbb{Z} die Menge der ganzen Zahlen und rundeAuf die Funktion ist, die jede rationale Zahl x auf die kleinste ganze Zahl y aufrundet, die nicht echt kleiner als x ist.
2. $S_5 = (\mathbb{Z}, \mathbb{N}; \text{abs}_{\mathbb{Z}})$, wobei \mathbb{Z} die Menge der ganzen Zahlen, \mathbb{N} die Menge der natürlichen Zahlen $0, 1, 2, \dots$ und $\text{abs}_{\mathbb{Z}}$ die Funktion ist, die jeder ganzen Zahl ihren Betrag zuordnet.

Hier beobachten wir, dass S_4 und S_5 jeweils aus zwei Mengen bestehen und einer einstelligen Funktion bestehen, wobei der Definitionsbereich die erste und der Wertebereich die zweite Menge ist. Auch hier sagen wir, dass S_4 und S_5 dieselbe *Signatur* haben.

Was ist eine Signatur? Wie bei den Funktionen finden wir hier unterschiedliche Definitionen in unterschiedlichen Büchern. Wir einigen uns auf die folgenden Axiome:

1. Eine Signatur Σ besteht aus endlich vielen *Symbolen*.
2. Jedes Symbol aus Σ ist entweder ein *Sortensymbol* oder ein *Funktionssymbol*.
3. Jedes Funktionssymbol f aus Σ hat eine Stelligkeit $\text{ar}_{\Sigma}(f) \in \{0, 1, 2, \dots\}$.
4. Jedes Funktionssymbol aus Σ hat einen *Typ* $\text{typ}_{\Sigma}(f) = s_1 \dots s_n s$, wobei $n = \text{ar}_{\Sigma}(f)$ und s_1, \dots, s_n, s jeweils Sortensymbole sind.

Wie Strukturen schreiben wir Signaturen gerne als Tupel, wobei wir zunächst alle Sortensymbole auflisten und nach einem Semikolon alle Funktionssymbole mit ihrem Typ auflisten. Am Typ erkennen wir auch leicht die Stelligkeit. Wir stellen im Folgenden den Zusammenhang zwischen Signaturen und Strukturen dar.

Wie schon angedeutet, hat eine Struktur eine Signatur. Tatsächlich hat eine Struktur sogar unendlich viele Signaturen: Sei $\Sigma = (s_1, \dots, s_k; f_1 : \text{typ}_{\Sigma}(f_1), \dots, f_{\ell} : \text{typ}_{\Sigma}(f_{\ell}))$ eine Signatur und $S = (B_1, \dots, B_m; g_1, \dots, g_n)$ eine Struktur. Wenn $k = m$ und $\ell = n$, dann bezeichne $S(s_i)$ die Menge A_i für alle $1 \leq i \leq m$. Dann ist S eine Σ -Struktur, falls:

¹Eine 1-stellige Polynom-Funktion ist eine Funktion, die durch ein Polynom beschrieben wird.

1. $k = m$,
2. $\ell = n$,
3. Für alle $1 \leq i \leq n$ gilt: Sei $r = \text{ar}_\Sigma(f_i)$ und $\text{typ}_\Sigma(f_i) = t_1 \dots t_r t$. Dann ist g r -stellig, $S(t_1) \times \dots \times S(t_r)$ der Definitionsbereich von g_i und $S(t)$ der Wertebereich von g_i .

Die Strukturen S_1 , S_2 und S_3 sind jeweils Σ_A -Strukturen für $\Sigma_A = (A; f : AA, c : A)$. Die Strukturen S_4 und S_5 sind jeweils Σ_B -Strukturen für $\Sigma_B = (A, B; f : AB)$.