Formale Sprachen

Sei Σ eine endliche, nicht-leere Menge. Wir bezeichnen Σ als ein *Alphabet* und die Elemente als *Symbole*. Σ^n bezeichnet alle endlichen Folgen der Länge n von Symbolen aus Σ und $\Sigma^* := \bigcup_{i \in \mathbb{N}_0} \Sigma^i$ die Menge von *Worten* von Symbolen aus Σ, d.h. endliche Folgen beliebiger Länge. Eine Teilmenge $A \subseteq \Sigma^*$ bezeichnen wir als *Sprache*.

Wir definieren mit A, B Sprachen über Σ:

- ε das eindeutige Wort der Länge 0, das *leere Wort*
- $\{\varepsilon\} := \Sigma^0$
- $\circ \colon \Sigma^* \times \Sigma^* \to \Sigma^*$ die gewöhnliche Konkatenation von Worten[1]
- $w^n := w \circ w \circ \ldots \circ w$ (n-mal) die n-fache Konkatenation für $w \in A$ ($w^0 = \varepsilon$)
- $\bar{A} := \Sigma^* - A$
- $wA := \{w \circ w' \mid w' \in A\}$, $Aw := \{w' \circ w \mid w' \in A\}$
- $A \circ B := \bigcup_{w \in A} wB$
- $A^n := A \circ A \circ \ldots \circ A$ (n-mal), $A^0 = \{\varepsilon\}$, $A^1 = A$
- $A^? := A^0 \cup A$
- $A^* := \bigcup_{n \in \mathbb{N}_0} A^n$
- $A^+ := \bigcup_{n \in \mathbb{N}, n \geq 1} A^n$

Extended Backus-Naur Form (EBNF)

Sprachen stets nur mit Mengenschreibweise und den bisher eingeführten Operationen zu beschreiben wäre sehr umständlich. Wir beschreiben daher die *Grammatik* G einer Sprache als *Extended Backus-Naur Form*[2]. Eine Grammatik G besteht hierbei aus dem Alphabet Σ, einer davon disjunkten Menge von *Non-Terminalen* (auch *Variablen*) V, einer Menge von *Produktionen* P der Form „linke Seite → rechte Seite“ und einem speziell ausgezeichneten Startsymbol $S \in V$.

Für $v \in V$ bezeichnen wir mit $L(v)$ die von v induzierte Sprache nach folgenden Regeln:

$\langle v \rangle \to$ “w”	$L(v) = \{w\}$
$\langle v \rangle \to$ /[c_1-c_n]/	$L(v) = \{c_1, \ldots, c_n\}$
$\langle v \rangle \to [r_1]$	$L(v) = L(r_1)^?$
$\langle v \rangle \to \{r_1\}$	$L(v) = L(r_1)^*$
$\langle v \rangle \to \{r_1\}^+$	$L(v) = L(r_1)^+$
$\langle v \rangle \to r_1\ r_2$	$L(v) = L(r_1) \circ L(r_2)$
$\langle v \rangle \to r_1 \mid r_2$	$L(v) = L(r_1) \cup L(r_2)$

Die durch die Grammatik G beschriebene Sprache ist dann $L(G) := L(S)$.

Beispiel (Arithmetische Ausdrücke mit EBNF). Wir geben eine EBNF an um korrekt geklammerte Arithmetische Ausdrücke (+, ·) mit natürlichen Zahlen zu beschreiben, Startsymbol ist ⟨Ausdruck⟩:

[1] $(\Sigma^*, \circ)$ ist ein Monoid, der *freie Monoid* über Σ

[2] Im Allgemeinen lässt sich nicht jede Sprache durch eine Grammatik beschreiben. EBNFs beschreiben wiederrum eine Teilmenge aller Grammatiken, die *kontextfreien* Grammatiken.

$$
\begin{aligned}
\langle \text{Faktor} \rangle &\longrightarrow \{\texttt{/[0-9]/}\}^{+} \\
&\quad\ \mid \text{“}\texttt{(}\text{”}\ \langle \text{Ausdruck} \rangle\ \text{“}\texttt{)}\text{”} \\
\langle \text{Produkt} \rangle &\longrightarrow \langle \text{Faktor} \rangle\ \{\text{“}\texttt{*}\text{”}\ \langle \text{Faktor} \rangle\} \\
\langle \text{Ausdruck} \rangle &\longrightarrow \langle \text{Produkt} \rangle\ \{\text{“}\texttt{+}\text{”}\ \langle \text{Produkt} \rangle\}
\end{aligned}
$$

EBNFs für C++

Wir geben in dieser Vorlesung oft EBNFs für C++-Syntax an. In diesen Fällen versteht sich die Syntax angewandt nicht auf einzelne Buchstaben als Alphabet sondern sog. *Tokens*. Bevor der C++-Compiler den Code syntaktisch analysiert trennt er ihn zunächst auf in eine Folge von Tokens. Als Trenner zwischen Tokens fungiert i.W. ein Übergang zwischen alphanumerischen (also Buchstaben und Ziffern) und nicht-alphanumerischen Symbolen. Nicht-druckbare Zeichen (Leerzeichen, Zeilenumbrüche, etc.) sind i.W. ohne Bedeutung außer dass sie geeignet sind zwei Tokens voneinander zu trennen.