

Boost::Spirit Eine Einführung

Tobias Gerg

4. April 2006

1 Theoretische Informatik: Eine kurze Einführung

Dieses Kapitel dient vor allem zum Verständnis für die spätere Benutzung von Spirit. Sie werden hier keine ausführlichen Erläuterungen bezüglich Sprachen, Notationen und sonstigen Sachen finden, die die absolute mathematische Korrektheit bezüglich einer Definition von Grammatiken darstellen oder beweisen.

1.1 Grammatiken und Regeln

Um Ausdrücke hinsichtlich ihrer syntaktischen Korrektheit zu überprüfen, werden neben Automaten sehr gerne Grammatiken definiert.

Definition:¹

Eine *Grammatik* ist ein 4-Tupel

$$\mathcal{G} = (\mathcal{V}, \Sigma, \mathcal{P}, \mathcal{S}) \quad (1)$$

...

Erläuterungen:

\mathcal{V} = Menge der Variablen (können abgeleitet werden)

Σ = Menge der Terminalsymbole (können nicht mehr abgeleitet werden)

\mathcal{P} = Die Menge der Regeln

\mathcal{S} = Die Startvariable (wo geht's los)

Nun ist es vor allem wichtig zu wissen, was Regeln sind und wie sie angewandt werden.

Definition:

Eine Regel sei wie folgt definiert:

$$\text{LinkeSeite} \rightarrow \text{RechteSeite} \quad (2)$$

wobei die linke Seite aus mindestens einer Variable und beliebigen Terminalsymbolen besteht und die rechte Seite aus Variablen und Terminalsymbolen.

Genauere Definitionen finden Sie im schon erwähnten Buch:

Theoretische Informatik - kurzgefasst

Auf Grund dieser Definition können wir jetzt ein kleines Beispiel betrachten. Wir definieren eine Grammatik, die korrekt geklammerte arithmetische Ausdrücke darstellen kann.

¹Auszug aus dem Titel:Theoretische Informatik - kurzgefasst von Uwe Schöning

Beispiel:

Sei

$$\mathcal{G} = (\{E, T, F\}, \{(\cdot), a, +, *\}, \mathcal{P}, E)$$

wobei

$$\mathcal{P} = \{ E \rightarrow T, E \rightarrow E + T, T \rightarrow F, T \rightarrow T * F, F \rightarrow a, F \rightarrow (E) \}$$

Daraus lassen sich jetzt beliebige korrekt geklammerte arithmetische Ausdrücke darstellen, z.B.

$$a * a * (a + a) + a \in \mathcal{L}(\mathcal{G})$$

Die Ableitung:

$$\begin{aligned} E &\Rightarrow E + T \Rightarrow T + T \Rightarrow T * F + T \Rightarrow T * F * F + T \Rightarrow \\ &F * F * F + T \Rightarrow a * F * F + T \Rightarrow a * a * F + T \Rightarrow a * A * (E) + T \Rightarrow \\ &a * a * (E + T) + T \Rightarrow a * a * (T + T) + T \Rightarrow a * a * (F + T) + T \Rightarrow a * a * (a + T) + T \Rightarrow \\ &a * a * (a + F) + T \Rightarrow a * a * (a + a) + T \Rightarrow a * a * (a + a) + F \Rightarrow \\ &a * a * (a + a) + a \end{aligned}$$

So viel zum Thema Grammatiken und Regeln.

*Genauere Definitionen finden Sie im schon erwähnten Buch:
Theoretische Informatik - kurzgefasst*

1.2 Backus Naur Form

Im Folgenden noch ein kleiner Abriss über die Backus Naur Form.

Wie im obigen Beispiel gesehen, gibt es für das Symbol (die Variable) E zwei Regeln. Anstatt die so zu beschreiben wie oben, erlaubt die Backus Naur Form

$$E \rightarrow T \mid E + T$$

Das heißt, E geht entweder in T über oder in $E + T$.

Des Weiteren gibt es in der Erweiterten Backus Naur Form noch

$$E \rightarrow \alpha[\beta]\gamma \tag{3}$$

und

$$E \rightarrow \alpha\{\beta\}\gamma \tag{4}$$

Was soviel bedeutet wie β kann einmal kommen, muss aber nicht und β kann beliebig oft kommen, muss aber nicht.

Das war's aber jetzt mit Theorie ;-). Nun geht's an die Praxis.