1.3 Kontextfreie Sprachen

Wir haben uns im letzten Abschnitt sehr ausführlich mit den Typ-3 Sprachen beschäftigt, und das hatte einen guten Grund: Die regulären Sprachen sind sehr einfach und übersichtlich, haben aber dennoch eine große Bedeutung. In der Informatik begegnet man oft Strukturen, die durch endliche Automaten oder reguläre Ausdrücke adäquat beschrieben werden können.

Aber die Typ-3 Welt hat auch enge Grenzen!

Schon die eigentlich sehr einfache Sprache

$$L = \{a^n b^n \mid n \geq 1\}$$

ist keine Typ-3 Sprache, wie wir durch Anwendung des Pumping Lemmas sogar nachweisen konnten.

Aber diese Sprache ist bekanntlich eine Typ-2 Sprache!

Klammerungen

Intuitiv ist klar, wie ein korrekt angeordneter Klammerausdruck auszusehen hat. Wir können daher eine formale Definition der korrekt angeordneten Klammerausdrücke leicht angeben:

Eine Folge von a's (entsprechen öffnenden Klammern) und b's (entsprechen schließenden Klammern) ist korrekt angeordnet, wenn der gesamte Ausdruck gleich viele a's und b's enthält, und zudem in jedem Präfix mindestens so viele a's wie b's enthalten sind.

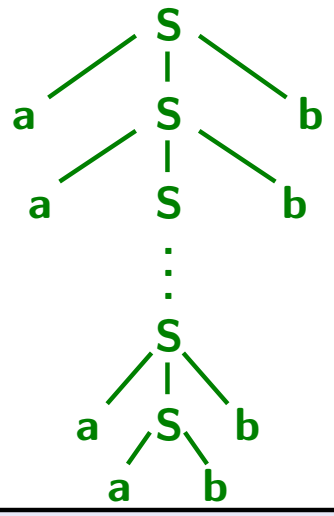
Korrekte Klammerausdrücke nennt man auch *Dyck-Wörter*.

Für die Menge aller Dyck-Wörter der Länge mindestens 2 kann man eine Typ-2 Grammatik $G = (\{S\}, \{a, b\}, P, S)$ angeben:

$$P: \quad S \to aSb, \quad S \to ab, \quad S \to SS$$

Ableitungsbäume

Wir haben bereits früher gesehen, dass eine Ableitung $S \Rightarrow_G^* w$ bzgl. einer Typ-2 Grammatik G immer durch einen Syntaxbaum visualisiert werden kann. Dabei wird von der speziellen Reihenfolge der Weiterverarbeitung vorhandener Variablen abstrahiert; d.h. mehrere Ableitungen können den selben Syntaxbaum haben. Legt man sich allerdings auf *Linksableitungen* fest, ist die Zuordnung Syntaxbaum $\leftrightarrow$ Linksableitung eine 1:1 Entsprechung. Beispiele:



Zeichnen Sie sich selbst den Syntaxbaum für die folgende Ableitung:

$$S \Rightarrow_G SS \Rightarrow_G aSbS \Rightarrow_G aabbS \Rightarrow_G aabbSS \Rightarrow_G aabbabS \Rightarrow_G aabbabaSb \Rightarrow_G aabbabaabb$$

Die Form der Syntaxbäume

Die Syntaxbäume können bei entsprechenden Typ-2 Grammatiken praktisch beliebige Formen annehmen. Einzige Einschränkung: Es gibt offensichtlich einen maximalen Verzweigungsgrad, weil in jedem inneren Knoten die Verzweigung durch eine der Regeln aus P gegeben ist – und das sind nur endlich viele.

Zudem muss man bei beliebigen Typ-2 Grammatiken damit rechnen, dass zu P Regeln (u, v) gehören, bei denen v viele Terminale und viele Variablen – in beliebigem Wechsel – enthält.

Da das zu sehr unübersichtlichen Ableitungsbäumen führen kann, entstehen folgende Wünsche für eine *Normalform* der Typ-2 Grammatiken:

1) Innere Knoten sollen normalerweise zwei Nachfolger haben.
2) Entweder zwei Variablen oder ein Terminal als Nachfolger.

Definition der Chomsky-Normalform

Was wir soeben formuliert haben, ist genau die nach Noam Chomsky benannte *Chomsky-Normalform* (CNF), die wir nun auch formal definieren wollen:

Definition: Eine Typ-2 Grammatik G mit Regelmenge P ist in Chomsky-Normalform, wenn für alle $(u, v) \in P$ gilt:

$$v \in V^2 \cup \Sigma$$

In der nächsten Einheit werden wir zeigen, dass es zu jeder Typ-2 Grammatik G eine Grammatik G' in CNF gibt, für die gilt: $L(G) = L(G')$. Das rechtfertigt den Ausdruck *Normalform.*

Definition der Greibach-Normalform

Ähnlich wie die CNF wollen wir noch eine zweite Normalform einführen, nämlich die *Greibach-Normalform* (GNF), benannt nach Sheila Greibach:

Definition: Eine Typ-2 Grammatik G mit Regelmenge P ist in Greibach-Normalform, wenn für alle $(u, v) \in P$ gilt:

$$v \in \Sigma V^*$$

Hier bestehen die rechten Seiten der Regeln also immer aus einem Terminalsymbol, gefolgt von einer (möglicherweise leeren) Folge von Variablen. Auch hier werden wir zeigen, dass es zu jeder Typ-2 Grammatik eine äquivalente Grammatik in GNF gibt.

Erste Schritte zur CNF

Wir gehen grundsätzlich von einer Typ-2 Grammatik G mit $\varepsilon \notin L(G)$ aus.

Als ersten generellen Schritt wollen wir eine zu G äquivalente Typ-2 Grammatik G_1 erzeugen, in der folgendes gilt:

$$(u, v) \in P \implies [\,|v| > 1 \text{ oder } v \in \Sigma\,]$$

Also müssen Regeln der Form $A \to B$ aus P eliminiert werden.

Wir werden das in zwei Schritten bewerkstelligen:

- Zunächst werden sogenannte *Ringableitungen* entfernt.
- Dann werden die Variablen so angeordnet, dass $A \to B$ nur vorkommen kann, wenn B in der Anordnung hinter A kommt.