Der zweite Algorithmus

Der jetzt folgende Algorithmus hat zum Ziel, dass alle rechten Seiten der Regeln mit einem Terminalzeichen beginnen.

Wir stellen fest, dass das bei A_m-Regeln schon der Fall ist.

Bei A_{m-1}-Regeln wissen wir bereits, dass die rechten Seiten nur mit A_m oder mit Terminalzeichen beginnen können.

Durch Einsetzen der rechten Seiten von A_m-Regeln erhalten wir nun auch in allen A_{m-1}-Regeln Terminalzeichen zu Beginn der rechten Seiten.

So fahren wir fort mit A_{m-2}, dann A_{m-3} und so weiter. Wenn wir A_2 und zuletzt A_1 bearbeitet haben, wissen wir, dass nun alle A_i-Regeln auf der rechten Seite mit Terminalen beginnen.

B-Regeln

Sind wir schon fertig?

NEIN! Wir müssen noch einmal auf die im ersten Algorithmus eingeführten B-Regeln schauen, ob dort die rechten Seiten die gewünschte Form haben.

Aber: Die rechten Seiten der B-Regeln beginnen entweder mit A_i (für ein $i \in \{1, \ldots, m\}$) oder mit einem Terminal!

Da alle A_i-Regeln mit Terminalen auf der rechten Seite beginnen, ist nun klar, dass wir durch Einsetzen das selbe auch für alle B-Regeln erreichen können.

Letzter Schritt: Weiter hinten erscheinende Terminale durch Pseudoterminale ersetzen...

Beispiel

Wir bearbeiten die folgende Grammatik, bei der wir die Variablen der Einfachheit halber schon von A_1 bis A_3 durchnummeriert haben:

$$A_1 \to A_1A_2 \mid A_2A_3 \mid a$$
$$A_2 \to A_3A_1$$
$$A_3 \to b$$

Bitte versuchen Sie es selbst – führen Sie die beiden Algorithmen Schritt für Schritt durch, wie in den vorigen Folien angegeben...

Binärbäume und Pfade

Bevor wir uns dem Pumping Lemma für Typ-2 Sprachen widmen, wollen wir einen einfachen Satz über Binärbäume beweisen.

Als *Binärbaum* bezeichnen wir hier einen Baum, bei dem jeder Knoten, der kein Blatt ist, genau zwei Nachfolgeknoten besitzt.

Die Syntaxbäume für CNF-Grammatiken sind Binärbäume, wenn wir die jeweils letzten Schritte (der Form $A \to a$) ignorieren!

Satz: Hat ein Binärbaum 2^k oder mehr Blätter, so hat er auch einen Pfad der Länge mindestens k.

Beweis per Induktion über k, Induktionsanfang für $k = 1$ ist klar.

Für den Induktionsschritt betrachten wir die beiden Teilbäume der Wurzel, von denen mindestens einer 2^k oder mehr Blätter hat. Der hat einen Pfad der Länge $\geq k$, mit der Wurzel also $\geq k + 1$.

Pumping Lemma für Typ-2

Das Pumping Lemma für Typ-2 Sprachen wird oft auch als *uvwxy*-Theorem bezeichnet. Es lautet wie folgt:

Satz: Sei $L \subseteq \Sigma^*$ eine Typ-2 Sprache. Dann gibt es eine Zahl n so, dass für alle $z \in L$ mit $|z| \geq n$ eine Zerlegung $z = uvwxy$ in $u, v, w, x, y \in \Sigma^*$ existiert, für die die folgenden drei Bedingungen erfüllt sind:

1) $|vx| \geq 1$
2) $|vwx| \leq n$
3) $\forall i \geq 0 : \quad uv^i wx^i y \in L$

Wie beim Pumping Lemma für Typ-3 ist es auch hier besonders wichtig, die logische Struktur der Aussage zu beachten!

Beweis des Pumping Lemmas

Die Sprache L sei eine Typ-2 Sprache, d.h. es gibt eine Typ-2 Grammatik $G = (V, \Sigma, P, S)$ in CNF, so dass $L = L(G)$ gilt.
Wir fixieren eine solche Grammatik G und wählen $n = 2^{|V|}$.

Nun müssen wir zeigen, dass für diese Wahl von n die Behauptung aus dem Pumping Lemma tatsächlich gilt.

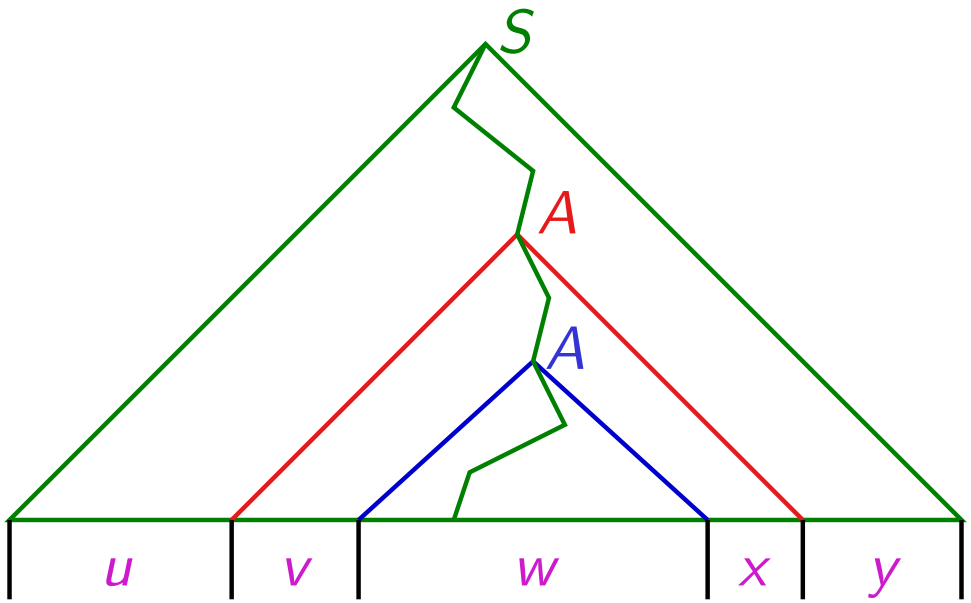
Es sei also $z \in L$ mit $|z| \geq n$ gegeben. Wir müssen zeigen, dass z sich zerlegen lässt in $z = uvwxy$, und zwar so, dass die Bedingungen 1), 2) und 3) erfüllt sind.

Weil $z \in L$ ist, gibt es einen Syntaxbaum, dessen Blätter genau das Wort z ergeben. Auch ohne die Terminalschritte hat dieser Baum mindestens $2^{|V|}$ viele Blätter, also gibt es in ihm einen Pfad (von der Wurzel zu einem Blatt), dessen Länge mindestens $|V|$ beträgt. Auf diesem Pfad muss sich eine Variable wiederholen.

Beweis (2)

Wir fixieren im Syntaxbaum einen *längsten* Pfad.

Von unten nach oben prüfen wir diesen Pfad und suchen den Knoten, in dem erstmals eine Variable steht, die weiter unten schon vorgekommen war. Daher erhalten wir folgendes Bild:



Beachte: Der (grüne) Teilpfad unterhalb des roten A's hat höchstens die Länge $|V|$.