

Sprachverarbeitung: Musterlösung zur Übung 14

Formale Sprachen und Grammatiken

Aufgabe 1: Chomsky-Sprachtypen

Welche Regeln die verschiedenen Grammatiktypen enthalten dürfen, ist aus der Chomsky-Sprachhierarchie (siehe Buch Seite 142) ersichtlich.

- a) Eine mögliche Typ-3-Grammatik G_a zur Sprache

$$L_a = \{w \in V_T^* \mid w \text{ enthält mindestens fünf } a\text{'s}\}$$

mit $L(G_a) = L_a$ ist $G_a = (V_N, V_T, P, S)$, wobei $V_N = \{A_0, A_1, A_2, A_3, A_4, S\}$ und

$$P = \{ \begin{array}{l} S \rightarrow aA_4 \mid bS \mid cS \\ A_4 \rightarrow aA_3 \mid bA_4 \mid cA_4 \\ A_3 \rightarrow aA_2 \mid bA_3 \mid cA_3 \\ A_2 \rightarrow aA_1 \mid bA_2 \mid cA_2 \\ A_1 \rightarrow aA_0 \mid bA_1 \mid cA_1 \\ A_0 \rightarrow aA_0 \mid bA_0 \mid cA_0 \mid \varepsilon \end{array} \}.$$

- b) Wenn es in der Sprache L_b kein einziges Wort gibt, welches das Terminalsymbol c enthält, dann ändert sich an der Sprache L_b nichts, wenn die Menge der Terminalsymbole auf $V_{T_b} = \{a, b\}$ reduziert wird.

Eine Typ-2-Grammatik G_b zur Sprache

$$L_b = \{w \in V_T^* \mid w \text{ enthält mehr } a\text{'s als } b\text{'s, aber keine } c\text{'s}\}$$

mit $L(G_b) = L_b$ ist also beispielsweise $G_b = (V_N, V_{T_b}, P, S)$, wobei $V_N = \{X, S\}$ und

$$P = \{ \begin{array}{l} S \rightarrow XaS \mid XaX \\ X \rightarrow aXbX \mid bXaX \mid \varepsilon \end{array} \}.$$

Die Idee dabei ist, dass durch die X-Regel jeweils Paare der Terminalsymbole a und b generiert werden, wobei die Paare in jeder beliebigen Reihenfolge auftreten können und auch von anderen Symbolen unterbrochen werden. Die S-Regel produziert diejenigen a die überzählig sind.

Selbstverständlich könnte man für die Sprache L_b auch eine Typ-1-Grammatik schreiben, nämlich $G'_b = (V_N, V_{T_b}, P, S)$, wobei $V_N = \{A, S\}$ und

$$P = \{ \begin{array}{l} S \rightarrow aA \\ A \rightarrow AA \mid ab \mid a \mid \varepsilon \\ ab \rightarrow ba \\ ba \rightarrow ab \end{array} \}.$$

Obwohl es Regeln mit mehreren Symbolen im Regelkopf gibt, und G'_b somit eine Typ-1-Grammatik (kontextsensitiv) ist, ist L_b keine Typ-1-Sprache, weil es eine Typ-2-Grammatik G_b mit $L(G_b) = L_b$ gibt und damit gilt: $L_b \in \mathcal{L}_2$.

c) Eine mögliche Typ-1-Grammatik G_c zur Sprache

$L_c = \{w \in V_T^* \mid w \text{ enthält doppelt so viele } a\text{'s wie } c\text{'s, mehr } a\text{'s als } b\text{'s und mehr } b\text{'s als } c\text{'s}\}$
mit $L(G_c) = L_c$ ist $G_c = (V_N, V_T, P, S)$, wobei $V_N = \{A, S\}$ und

$$P = \{ \begin{array}{l} S \rightarrow aaaabbbcc \mid aaaabbbccA \\ A \rightarrow abc \mid abbc \mid AA \\ ab \rightarrow ba \\ ac \rightarrow ca \\ ba \rightarrow ab \\ bc \rightarrow cb \\ ca \rightarrow ac \\ cb \rightarrow bc \end{array} \}.$$

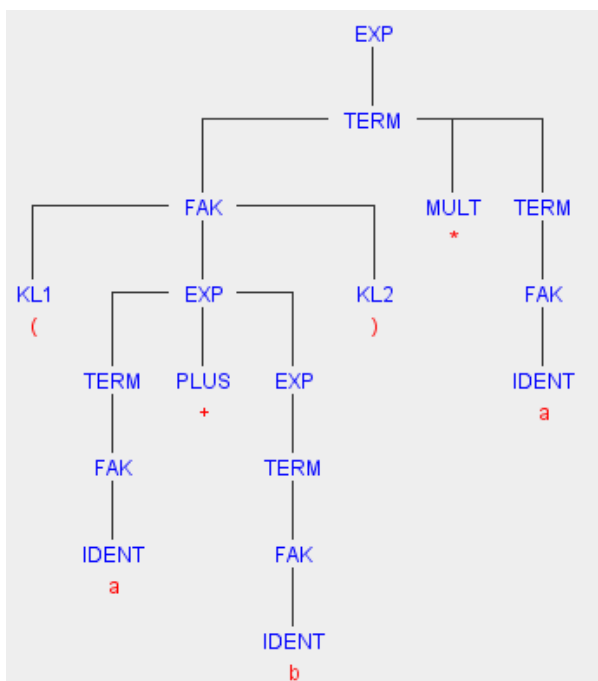
Hier erzeugen die S- und A-Regel zuerst einmal Wörter mit der richtigen Anzahl von Terminalsymbolen. Die übrigen Regeln erlauben dann eine beliebige Permutation der Terminalsymbole und erzeugen dadurch jedes erlaubte Wort.

Die Regeln der Form $ab \rightarrow ba$ sind zwar nicht kontextsensitiv, aber G_c ist monoton, eine für Typ-1-Grammatiken notwendige und hinreichende Bedingung. Somit ist G_c eine Typ-1-Grammatik (kontextsensitiv) $\longrightarrow L_c = L(G_c) \in \mathcal{L}_1$.

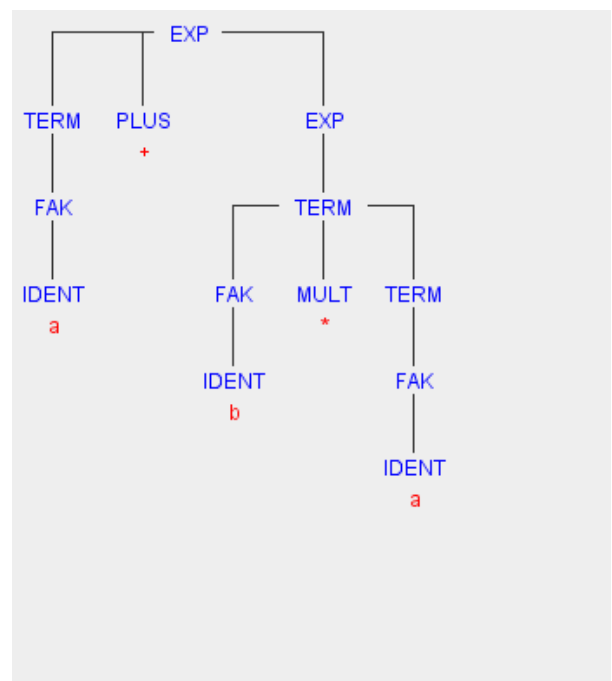
Aufgabe 2: Ableitungsbaum mathematischer Ausdrücke

Die Ableitungsbäume der gegebenen mathematischen Ausdrücke können für die gegebene Grammatik (ist mit der Grammatik von Aufgabe 1 der Übung 15 identisch) auch mit dem Parser von Übung 15 erzeugt werden. Die Resultate sind dann:

a) $(a + b) * a$



b) $a + b * a$



Der Ableitungsbaum b) zeigt, dass zuerst der *Term* $b * a$ ermittelt werden muss, weil dieser benötigt wird, um die Summe aus a und diesem Term zu bestimmen. Die Baumstruktur entspricht also genau der Präzedenz, also dass die Operation $*$ gegenüber der Operation $+$ den Vorrang hat.

Im mathematischen Ausdruck a) bewirken die Klammern, dass die Präzedenz lokal ausser Kraft gesetzt wird, was sich selbstverständlich im Ableitungsbaum äussern muss.

Aufgabe 3: Grammatik für mathematische Ausdrücke

In mathematischen Ausdrücken mit den Operationszeichen $+$ und $*$ müssen nur dort Klammern gesetzt werden, wo eine Summe als Faktor in einem Produkt gebraucht wird. An der gegebenen Grammatik G_M sind demnach die folgenden Änderungen nötig:

- eine neue Regel für die Summe *Sum* einfügen,
- eine neue Regel für das Produkt *Prod* einfügen,
- die Term-Regel ändern (ein Term ist entweder eine einfache Grösse oder ein Produkt),
- die Faktor-Regel ändern (ein Faktor ist entweder eine einfache Grösse oder eine Summe in Klammern) und
- die Regel für den Ausdruck *Exp* anpassen.

Die resultierende Grammatik ist dann $G'_M = (V'_N, V_T, P', S)$ mit:

$$V'_N = \{Exp, Sum, Term, Fak, Ident\}$$

$$\begin{aligned}
 P' = \{ & Exp \rightarrow Term \mid Sum, \\
 & Sum \rightarrow Term + Term \mid Term + Sum \\
 & Term \rightarrow Ident \mid Prod \\
 & Prod \rightarrow Fak * Fak \mid Fak * Prod \\
 & Fak \rightarrow Ident \mid (Sum) \\
 & Ident \rightarrow a \mid b \mid c \}.
 \end{aligned}$$

Anhang: Beweisskizze für Aufgabe 1b

Es ist oft nicht einfach zu beweisen, dass eine Grammatik eine bestimmte Sprache beschreibt. Das Führen solcher Beweise ist auch nicht Teil der Vorlesung. Für diejenigen, die sich trotzdem dafür interessieren, wie ein solcher Beweis aussehen könnte, wird hier eine halbformale Beweisskizze für Aufgabe 1b aufgeführt. *Es handelt sich dabei nicht um Prüfungsstoff.*

Es ist zu zeigen, dass $L(G_b) = L_b$. Dies ist genau dann der Fall, wenn sowohl $L(G_b) \subseteq L_b$ als auch $L(G_b) \supseteq L_b$.

Nachweis von $L(G_b) \subseteq L_b$

Zu zeigen ist, dass die Grammatik G_b nur solche Worte produzieren kann, die mehr a 's als b 's enthalten. Dies ist auch ohne formalen Beweis ersichtlich: Das Nichtterminalsymbol X produziert nur Worte, die gleich viele a 's wie b 's enthalten. Das Startsymbol S produziert mindestens ein a , ansonsten aber nur Nichtterminalsymbole X .

Nachweis von $L(G_b) \supseteq L_b$

Zu zeigen ist, dass jedes Wort w , das mehr a 's als b 's enthält, auch von der Grammatik G_b produziert werden kann.

Es ist möglich, jedes Wort $w = c_1 c_2 \dots c_n$, das mehr a 's als b 's enthält, in eine Folge $X_1 a X_2 \dots X_{l-1} a X_l$ zu zerlegen, wobei die Symbolfolgen X_i jeweils gleich viele a 's enthalten wie b 's. Um dies zu zeigen, definieren wir eine Funktion f , die für ein Wort $x_1 x_2 \dots x_k$ die Anzahl der a 's minus die Anzahl der b 's berechnet:

$$f(x_1 x_2 \dots x_k) = \sum_{i=1}^k \delta(x_i), \quad \text{wobei } \delta(x) = \begin{cases} +1 & \text{falls } x = a \\ -1 & \text{falls } x = b \end{cases}$$

Wir betrachten nun den Fall $c_1 = b$. Es gilt nun einerseits, dass $f(c_1 c_2 \dots c_n) \geq 1$, andererseits ist $f(c_1) = -1$. Da f bei Hinzunahme eines weiteren c_i entweder um eins grösser oder um eins kleiner wird, muss es ein m geben, so dass $c_m = a$ und $f(c_1 c_2 \dots c_{m-1}) = 0$. Dies ist in der folgenden Figur illustriert:

c_i	$c_1 = b$	c_2	...	c_{m-1}	$c_m = a$...	c_n
$f(c_1 c_2 \dots c_i)$	-1			0	1	...	≥ 1

Die Symbolfolge $c_1 c_2 \dots c_n$ kann also zerlegt werden in einen Präfix $c_1 c_2 \dots c_{m-1}$, der gleich viele a 's wie b 's enthält, ein Symbol a , und einen Rest $c_{m+1} \dots c_n$, der mindestens so viele a 's wie b 's enthält. Im Fall $c_1 = a$ ist eine entsprechende Aufteilung möglich, wobei der Präfix leer ist (d.h. $m = 1$).

Falls der Rest mehr a 's als b 's enthält, kann er auf analoge Weise zerlegt werden, so dass wir schliesslich die gesuchte Zerlegung $w = X_1 a X_2 \dots X_{l-1} a X_l$ erhalten.

Es ist nun leicht einzusehen, dass die Grammatik G_b korrekt ist: Das Startsymbol produziert alle möglichen Sequenzen der Form $XaX \dots XaX$, und das Nichtterminalsymbol X produziert alle möglichen Buchstabenfolgen, die gleichviele a 's wie b 's enthalten. Letzteres müsste natürlich erst noch bewiesen werden, beispielsweise wiederum mit Hilfe der Funktion f und durch Betrachten von Präfixen.