Übungen zu CFGs (**Daniel Siebert 2011, cc-by-nc-sa**)

Anmerkungen:

1. Wenn nicht explizit angegeben gilt für alle CFGs das **Startsymbol S**. Die Terminal- und Nichtterminalsymbole ergeben sich aus den Produktionsregeln.
2. Aufgabentypen zur Einschätzung des Schwierigkeitsgrades:
   (a) Typ1: Nicht verschachtelte Abhängigkeiten.
   (b) Typ2: Einfach verschachtelte Abhängigkeiten.
   (c) Typ3: Mehrfach verschachtelte Abhängigkeiten.
   (d) Typ4: Komplexe Abhängigkeiten.
3. Zeitangaben geben nur eine sehr grob geschätzte Einschätzung des Bearbeitungsaufwandes.

1. Aufwärmübung: $L = \{a^n b^k | k = \frac{n}{2}, k \geq 1\}$, geben Sie G an. `(Typ1, 5min)`
2. G hat die Produktionsregeln: $S \rightarrow BA$, $B \rightarrow Bb|b$, $A \rightarrow aAc|ac$, geben Sie L an. Tipp: Vergessen Sie nicht die Wertebereiche für die Indizes! `(Typ1, 10min)`
3. $L = \{(da)^n b^l c^{n+1} | n, l \geq 0\}$, geben Sie G an. `(Typ2, 15min)`
4. G hat die Produktionsregeln: $S \rightarrow NaD$, $N \rightarrow cbNa|\epsilon$, $D \rightarrow Dd|\epsilon$, geben Sie L an. `(Typ2, 10min)`
5. $L = \{a^k (b^{n+2} c^m d^n)^{k+1} \mid k \bmod 2 = 0; k, n, m \geq 0\}$, geben Sie G an. `(Typ3, 20min)`
6. G hat die Produktionsregeln $S \rightarrow aNb$, $N \rightarrow cSd|\epsilon$, geben Sie L an. `(Typ2, 15min)`

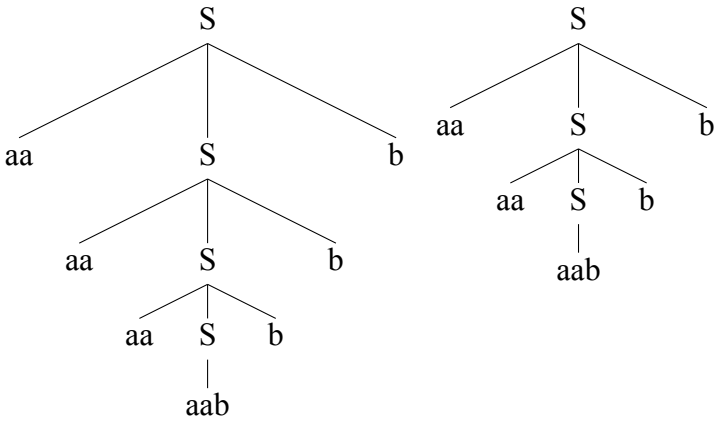
   Für Schlaue: Überlegen Sie was sich ändert, wenn außerdem $S \rightarrow \epsilon$ (also $S \rightarrow aNb|\epsilon$) gilt. `(Typ2, zusätzlich 5min)`
7. $L = \{a^n b^m a^n d^l | l, m, n \geq 0\} \setminus \{w^R w | w \in \{a, b, c, d\}^+\}$, geben Sie G an. Tipp: Überlegen Sie welche Wörter mit $|w| \leq 4$ es gibt. `(Typ4, 20min)`
8. G hat folgende Produktionsregelns: $S \rightarrow aSb|cSa|D$, $D \rightarrow dD|d$, geben Sie L an? `(Typ4, 20min)`

Übungen zu CFGs: Lösungen

1. Lösung: $S \to aaSb|aab$

   Da sowohl a^n als auch b^k durch die Bedingung $k = \frac{n}{2}$ von n abhängen müssen alle Terminal a's und b's vom selben Nichtterminal erzeugt werden. Um immer halb so viele b's wie a's zu erhalten erzeugen wir einfach immer zwei a's und ein b gleichzeitig.

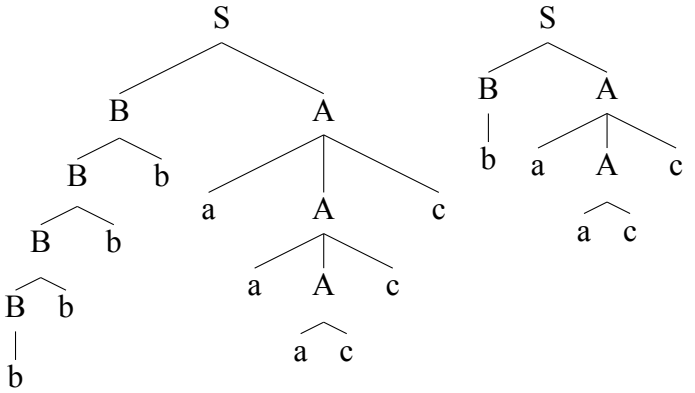
   Mögliche Parsbäume sind:



2. Lösung: $L = \{b^n a^m c^m | m, n \geq 1\}$

   Durch das Startsymbol wird direkt ein B und ein A erzeugt und es führt keine Regel zurück zu S, somit wissen wir, dass B und A vollständig unabhängig von einander sind. Es ist einfach zu erkennen, dass B beliebig lange Ketten von b's, jedoch mindestens eins, erzeugt. A erzeugt wiederum beliebig viele a's und c's jedoch immer gleich viele und in fester Reihenfolge.

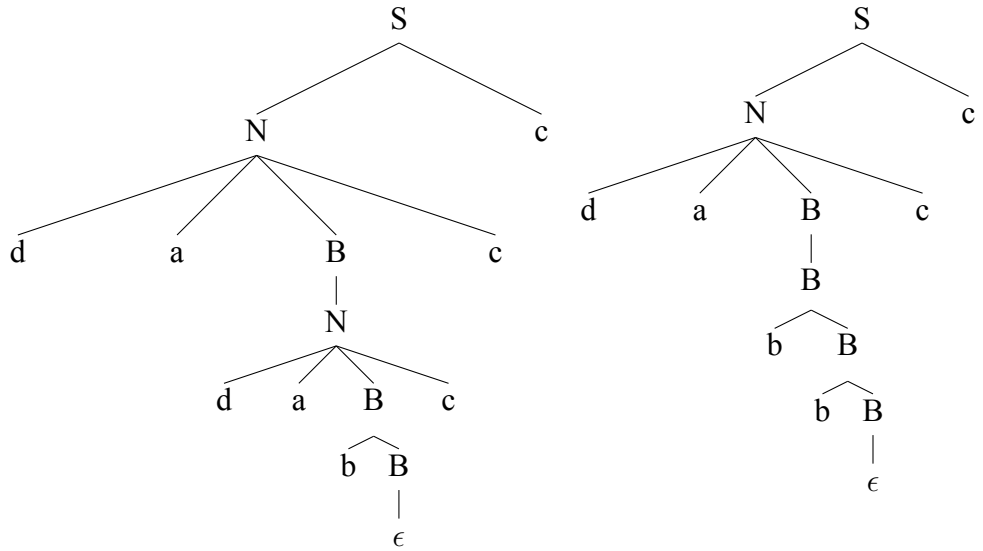
   Mögliche Parsbäume:



3. Lösung: $S \to Nc$, $N \to daBc|\epsilon$, $B \to N|bB|\epsilon$

Da $(da)^n$ und c^{n+1} vom gleichen Index n abhängen müssen sie vom selben Nichtterminal erzeugt werden. Um genau ein c mehr zu erhalten als da's hängt unsere erste Regel einfach ein c ganz ans Ende aller erzeugten Wörter. Die zweite Regel erzeugt nun jeweils da's und c's in gleicher Menge. Da es jedoch auch b's zwischen den da's und c's geben kann expandieren wir zunächst nach B. Die dritte Regel ermöglicht es und nun entweder zurück zu N zu expandieren und somit in einer $N \rightarrow B \rightarrow N$-Schleife beliebig viele da's und c's zu erzeugen, oder beliebig viele b's zu erzeugen, oder direkt aufzuhören.

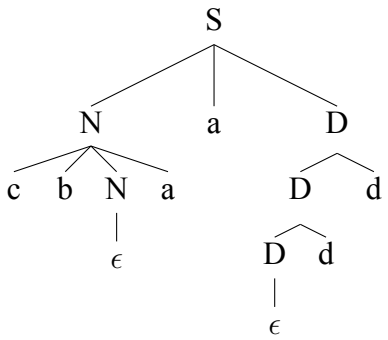
Mögliche Parsbäume:



4. Lösung: $L = \{(cb)^n a^{n+1} d^m | n, m \geq 0\}$

   S erzeugt zwei von einander unabhängige Nichttermina N und D wie daran zu erkennen ist, dass S in keiner weiteren Regel vorkommt und N und D jeweils nur sich selbst rekursiv enthalten. Außerdem wird ein zusätzliches a in der Mitte erzeugt, dies führt zum +1 des a^{n+1} in der Kettensprache. N erzeugt rekursiv beliebig viele Ketten der Form $(cb)^n a^n$, also gleich viele cb's und a's, oder terminiert mit ϵ. Zu beachten ist, dass alle von N erzeugten a's direkt vor dem durch S erzeugten a stehen, somit ergibt sich $(cb)^n a^{n+1}$. Zuletzt erzeugt die Expansion von D rekursiv eine beliebige Anzahl d's oder gar keine, wodurch wir ein unabhängiges d^m erhalten.

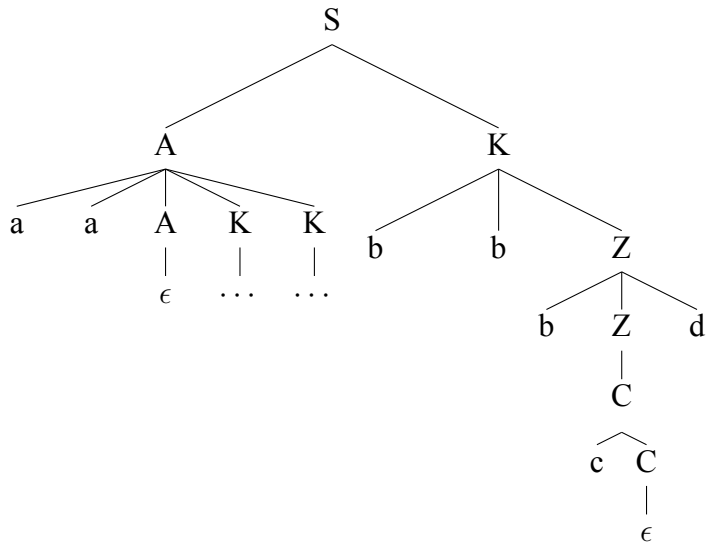
   Ein möglicher Parsbaum:



5. Lösung: $S \to AK$, $A \to aaAKK|\epsilon$, $K \to bbZ$, $Z \to bZd|C$, $C \to cC|\epsilon$

   Zuallererst überlegt man sich die möglichen Wertebereiche der Indizes. $k = \{0, 2, 4, 6, ...\}, n{+}2 = \{2, 3, 4, 5, ...\}, n, m = \{0, 1, 2, 3, ...\}, k{+}1 = \{1, 3, 5, 7, ...\}$. Hieraus ergibt sich das kleinste mögliche Wort "bb". Um den nächsten Schritt zu vereinfachen blenden wir jetzt den gesamten Klammerinhalt aus und ersetzen ihn durch K, also $a^k(\underbrace{b^{n+2}c^m d^n}_{K})^{k+1} = a^k K^{k+1}$ Wir benötigen jetzt also eine Regel die das +1 ausgleicht damit wir nur noch $a^k K^k$ abbilden müssen was mit dem üblichen Verfahren zur Produktion gleich langer Ketten funktioniert. Unsere ersten Regel lautet also $S \to AK$, sie produziert ein einzelnes extra K (+1) und übergibt die weitere Produktion an A. Als nächstes benötigen wir eine Regel zur Produktion gleich langer Ketten von a's und K's. Zu beachten ist hier, dass diese Ketten eine Länge von 0 haben können oder eine gerade Anzahl an Zeichen haben müssen. $A \to aaAKK|\epsilon$ erfüllt diese Anforderungen. Jetzt können wir uns der Produktion des Inhaltes von K zuwenden. Auch hier haben wir wieder ein Ungleichheit zwischen zwei Indizes mit gleicher Basis auszugleichen, nämlich das +2 in $(b^{n+2} \cdots d^n)$. Dies erreichen wir wie oben durch die Einführung einer Regel die zwei b's vorne anhängt und die Produktion des verbleibenden $(b^n \cdots d^n)$ an die nächte Regel weitergibt. Wir nennen diesen verbleibenden Teil willkürlich Z und erhalten so unsere Regel $K \to bbZ$. Z muss nun nach dem üblichen Verfahren gleich viele b's und d's produzieren, also $Z \to bZd$. Da wir am Ende jedoch vielleicht noch c's produzieren müssen erweitern wir Z um eine mögliche C Expansion und erhalten $Z \to bZd|C$. C expandiert nun ganz einfach zu einer beliebigen, auch leeren, Kette von c's $C \to cC|\epsilon$.

   Möglicher Parsbaum (übersichtshalber mit nur einem K-Element-Teilbaum):



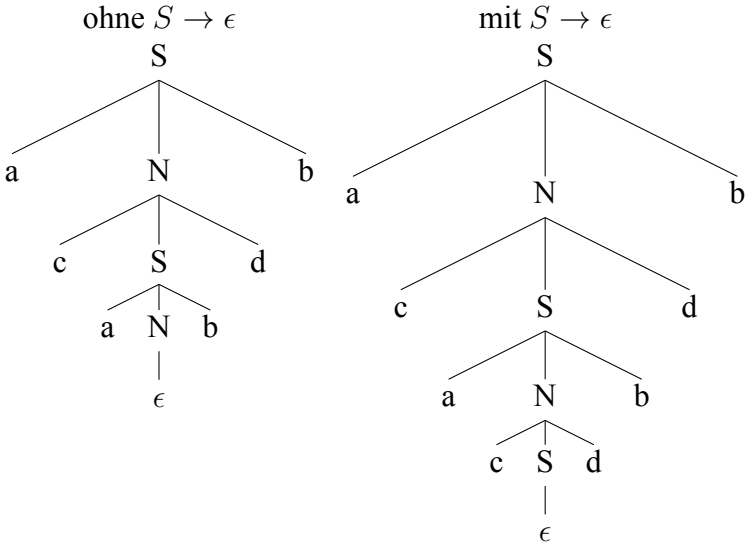
6. Lösung: $L = \{a(ca)^k(bd)^k b | k \geq 0\}$

   Regel Eins produziert zunächst ein a und ein b, dies stellt das minimale Wort dar (da N nach ϵ expandieren kann), wir schreiben $a \cdots b$. Nun haben wir zwei rekursive Schleifen übrig die zusammen eine gleiche Anzahl and a's, b's, c's und d's produzieren. Wir müssen diese jetzt nur noch in die richtige Reihenfolge bringen. Da wir mit der Expansion von N beginnen müssen erhalten wir zunächst $c \cdots d$. Zwischen dieses c und d muss jedoch jeweils ein $a \cdots b$ das bei der Expansion von S entsteht, also $ca \cdots bd$. Berücksichtigen wir nun noch die mögliche Rekursion und fügen unser anfängliches $a \cdots b$ an so erhalten wir die Lösung: $a(ca)^k(bd)^k b$ zu der wir lediglich noch den Wertebreich von k $k \geq 0$ als definieren müssen um auch das Minimalwort 'ab' zuzulassen.

   Zusatzlösung: $L = \{a(ca)^k(cd)^m(bd)^k b | k \geq 0$ und $1 \geq m \geq 0\}$

   Wie oben jedoch müssen wir jetzt beachten, dass das Wort auch durch eine $S \rightarrow \epsilon$ Expansion enden kann. Wir fügen also einfach in der Mitte ein $(cd)^m$, welches die letzten entstandenen Terminalsymole vor der S-Expansion erstellt. Wir begrenzen dies durch Definition des entsprechenden Wertebereichs von m auf ein (Wort endet durch S-Expansion) oder null (Wort endet auf N-Expansion) Vorkommen.

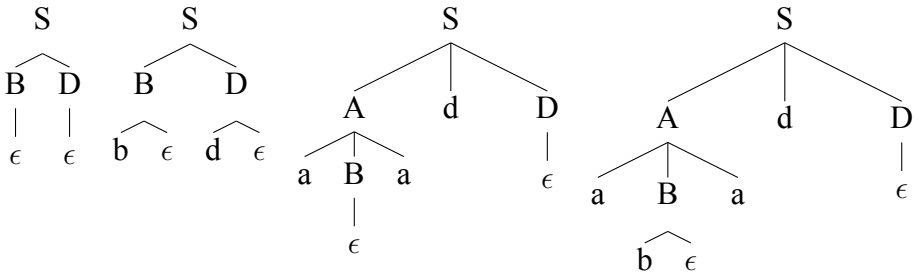
   Mögliche Parsbäume:



7. Lösung:
   $S \to AdD|BD$, $A \to aAa|B$,
   $B \to bB|\epsilon$, $D \to dD|\epsilon$

   Zuerst einmal müssen wir alle Wörter ausschließen die kein d enthalten, da diese automatisch Palindrome $(w^R w)$ sind. Einzige Ausnahme sind Ketten der Form $b^m d^l$ und ϵ, somit erhalten wir $S \to AdD|BD$, das entweder mindestens ein d erzwingt oder a's ausschließt. Regel zwei muss nun Ketten der Form $a^n \cdots a^n$ generieren, was nach dem üblichen Schema zu $A \to aAa$ führt. Da wir noch b's zwischen den a's benötigen erweitern wir A, so dass b's nach den a's produziert werden können und erhalten $A \to aAa|B$. Zuletzt fügen wir noch zwei Regeln hinzu die beliebig lange Ketten von b's und d's produzieren. Zu beachten ist hier, dass diese Regeln auch ϵ produzieren können um Ketten aus nur b's bzw. nur d's zu ermöglichen.

   Mögliche Parsbäume:

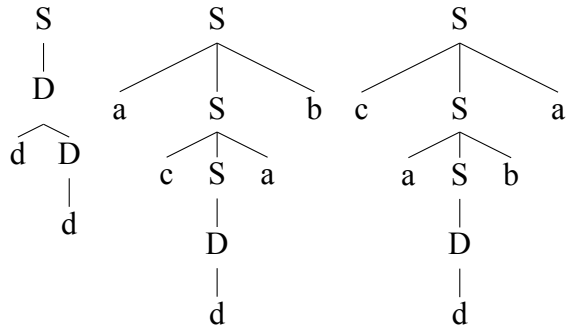


8. Lösung: $L = \{w_1^n d^m w_2^n | n \geq 0; m \geq 1; w_1 \in \{a, c\}; w_2 \in \{a, b\}\}$

   Wir sehen auf den ersten Blick, dass D eine beliebig lange Kette von d's jedoch mindestens eines produziert und das diese Kette in der Mitte jedes Wortes der

Zielsprache stehen muss, also notieren wir zunächst $L = \{\cdots d^m \cdots | m \geq 1\}$. Wir sehen weiterhin, dass die Teilketten vor und hinter den d's gleich lang sein müssen, da $S \rightarrow aSb$ und $S \rightarrow cSa$ jeweils genau ein Terminalsymbol auf beiden Seiten produzieren, also $L = \{w_1 \cdots w_2 || w_1| = |w_2|\}$. Da S beliebig nach aSb oder cSa expandieren kann muss die linke Teilkette eine Mischung aus a's und c's und die rechts eine aus b's und a's sein. Also $w_1 \in \{a, c\}^*$ und $w_2 \in \{a, b\}^*$. Da wir jedoch die Länge von w_1 und w_2 vergleichen wollen schreiben wir stattdessen $\{w_1^n \cdots w_2^n | w_1 \in \{a, c\}; w_2 \in \{a, b\}\}$. Zuletzt fehlt noch die Bedingung $n \geq 0$ da bei direkter Anwendung von $S \rightarrow D$ auf das Startsymbol keine $w_1^n \cdots w_2^n$ Ketten entstehen.

Mögliche Parsbäume:



Parsing-Zwischenklausuraufgaben

Hier folgen Aufgaben aus der Zwischenklausur des Parsing-Kurses von Laura Kallmeyer aus dem SS 2011.

1. (Zwischenklausur: Aufgabe 6)
   Welche Sprache wird von der CFG mit $N = \{A, B, S\}$, $T = \{a, b\}$, Startsymbol S und folgenden Produktionen erzeugt?
   $S \rightarrow aBB|bAB|bBA$,
   $A \rightarrow a|aS|bAAB|bABA|bBAA$,
   $B \rightarrow b|bS|aBBB$.

Parsing-Zwischenklausuraufgaben: Lösungen

1. Lösung: $L = \{w|w \in \{a, b\}^+, |w|_b = 2|w|_a\}$

   Beim ersten Blick auf die Produktionsregeln stellt man fest, dass alle Terminalsymbole aus Expansionen beliebiger Nichtterminalsymbole erzeugt werden können, jedoch kann nur A eine Teilkette mit a und nur B eine Teilkette mit b final abschließen. Außerdem scheint es als ob die Produktionsregeln beliebige "gemischte"Rekursion zwischen den verschiedenen Regeln ermöglichen. Daraus können wir schließen, dass es sich bei der erzeugten Sprache nicht um eine Sprache mit einfachen Abhängigkeiten wie etwa $(ab)^n(ba)^n$ handelt.

   Gehen wir also Schritt für Schritt vor: Zuerst ermitteln wir welche minimalen Wörter die gesuchte Sprache hat, dazu schauen wir uns $S \rightarrow aBB|bAB|bBA$ an und stellen fest, das nur 'abb', 'bab' und 'bba' in Frage kommen. Bei sehr scharfem hinsehen könnten wir hier bereits feststellen, dass diese 3 Wörter die Gemeinsamkeit haben jeweils ein a und zwei b's zu haben und es kann keine weiteren Wörter mit $|w| = 3$ geben die diese Eigenschaft haben.

   Analysieren wir nun zunächst $A \rightarrow a|aS$ und $B \rightarrow b|bS$, wir sehen, dass A und B jeweils ein a und b generieren oder eine neue Teilkette ans Ende anhängen, wir können also davon ausgehen, dass unsere Zielsprache von der Form S^n mit $n \geq 1$ ist.

   Es gilt nun also alle Eigenschaften der Teilketten S zu finden. Schauen wir uns also nun $A \rightarrow bAAB|bABA|bBAA$ an. Wir stellen fest, dass hier das Nichtterminal A unüblicherweise zu einem Terminal b und weiteren Nichttermina expandiert. Betrachten wir nun den Fall in dem alle diese Nichttermina durch $A \rightarrow a$ und $B \rightarrow B$ expandieren so erhalten wir $A \rightarrow baab|baba|bbaa$. Diese 3 Teilketten haben jeweils 2 a's und 2 b's, bedenken wir jedoch, dass wir ursprünglich ein A expandiert haben und zählen entsprechend ein a weniger,

so haben wir wieder unser Verhältnis von einem a zu zwei b's für die Restkette, dass wir bereits bei der Ermittlung der Minimalwörter festgestellt hatten. Schauen wir uns $B \to aBBB$ unter den selben Gesichtspunkten an, so stellen wir fest, das auch hier ein a auf zwei b's kommt ($B \to abbb$ abzüglich einem b wegen B-Expansion).

Wir wissen nun also, dass alle Teilketten die Bedingung $|w|_b = 2|w|_a$ einhalten, also doppelt so viele a's wie b's enthalten. Es ist jedoch nicht zu erkennen, dass diese a's und b's in einer bestimmten Reihenfolge stehen, es ist also von $\{a, b\}^*$ auszugehen. Allerdings haben wir bei der Suche nach Minimalwörtern eine Mindestlänge von 3 festgestellt, also $\{a, b\}^+$. Fügen wir nun alle unsere Informationen zusammen so erhalten wir die korrekte Lösung $L = \{w | w \in \{a, b\}^+, |w|_b = 2|w|_a\}$, wobei Wörter der Länge 1 und 2 durch das Verhältnis von a's zu b's automagisch ausgeschlossen werden.

Mögliche Parsbäume: