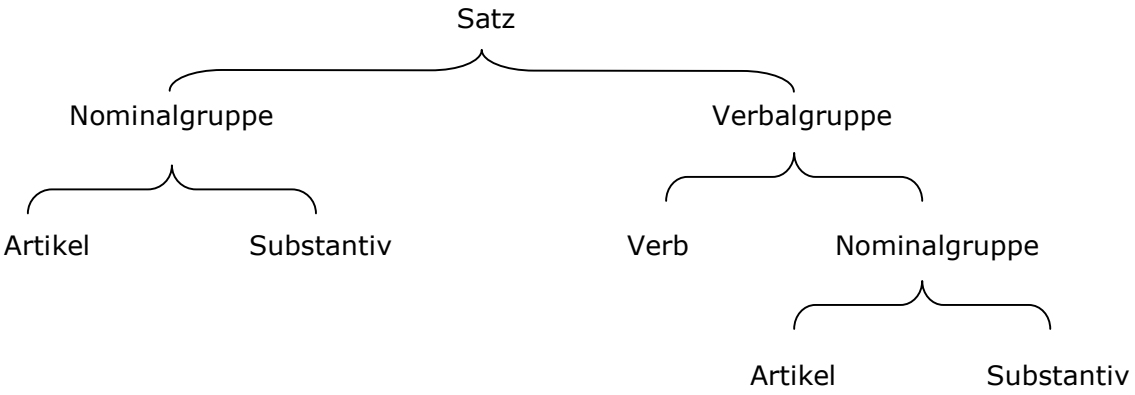
FORMALE SPRACHEN

Analyse folgenden Satzes:

„Die Katze jagt die Maus"

Syntaxbaum:



Eine Grammatik wird beschrieben durch ein System von Regeln, z.B. in der ***Backus-Naur-Form*** (BNF-Notation):

<SATZ> ::= <NOMINALGRUPPE> <VERBALGRUPPE>

<NOMINALGRUPPE> ::= <ARTIKEL> <SUBSTANTIV>

<VERBALGRUPPE> ::= <VERB> <NOMINALGRUPPE>

<VERB> ::= jagt | sieht | beißt | frißt

<ARTIKEL> ::= der | die | das

<SUBSTANTIV> ::= Katze | Maus | Merlin | Tablet

Bemerkung: In spitzen Klammern eingeschlossene Symbole sind nicht endgültige Zeichen, sogenannte **Non-Terminalzeichen (Nonterminals)**; die anderen Zeichen, wie z. B. "der" oder "Katze" im obigen Beispiel, werden nicht mehr durch andere Zeichen ersetzt und heißen endgültige Zeichen, **Terminalzeichen (Terminals)**.
Die Ersetzungsregel „<ARTIKEL> ::= der | die | das" bedeutet, daß das Non-Terminal „<ARTIKEL>" durch die Terminals „der" oder „die" oder „das" ersetzt werden kann und auch zu ersetzen ist, denn der endgültige Satz besteht aus lauter Terminals.

Beachte: der Leser hat gewiß schon bemerkt, daß der Begriff „Zeichen" nicht einen Buchstaben oder eine Ziffer im Sinne von „character" (char) meint, sondern die Terminalzeichen sind bei einer natürlichen Sprache die Wörter, bei einer Programmiersprache die Schlüsselwörter (z. B. ***input***, ***print***, ***if***, ***else*** *etc. in Python). Folglich sind die Sätze, die gemäß den Syntaxregeln einer die Programmiersprache definierenden Grammatik gebildet werden können, nichts anderes als die in dieser Sprache formulierten Programmtexte.*

Sätze, die gemäß obenstehenden Regeln aufgebaut sind:

"*der Merlin beißt das Tablet*"

"die Maus sieht die Katze"

"das Katze frißt die Maus"

Die Syntaxregeln müssen zur Bildung korrekter Sätze eingehalten werden; andererseits impliziert deren Einhaltung nicht zwingend, daß ein korrekter Satz gebildet wird. Ein nach den Regeln der Grammatik syntaktisch korrekt gebildeter Satz garantiert keineswegs, daß der Satz auch semantisch korrekt ist.

DEFINITION:

Wenn A eine endliche Menge von Zeichen ist, erhält man durch deren Hintereinanderschreiben Zeichenketten. Die Menge A heißt auch Alphabet, die Zeichenketten heißen Wörter über dem Alphabet A; das leere Wort, das keine Zeichen enthält, heißt ε.
Jede Menge von Wörtern über A heißt eine formale Sprache; ein System von Regeln, welches entscheidet, ob ein Wort über A zur Sprache gehört, heißt Grammatik (oder Syntax) einer formalen Sprache.

In *Python* sind

- die Zeichen oder Symbole der *formalen Sprache*: Schlüsselwörter (**`print`**, **`if`**, **`input`**, **`else`**, **`elif`**, **`return`**)
- die Wörter der *formalen Sprache*: Python-Programme

Eine *Grammatik* besteht aus *Regeln*, mit Hilfe derer entschieden wird, ob ein *Wort* (also ein *Programm-Text*) ein gültiges *Python*-Programm ist. Dieser Vorgang heißt *Syntax-Analyse*. Ein syntaktisch korrekter *Programm-Text* ist nicht hinreichend, daß das Programm auch etwas "Vernünftiges" leistet; die Bedeutung eines *Programm-Textes* (oder eines Textes einer *natürlichen Sprache*) wird durch den Begriff *"Semantik"* beschrieben.

Wir unterscheiden bei einer *formalen Sprache terminale* ("endgültige") und *nicht-terminale* ("nicht endgültige") Zeichen oder Symbole.

REGULÄRE SPRACHEN (TYP 3)

Eine einfache *formale Sprache*

Sei **S** ein *nicht-terminales* Symbol, **a**, **b** seien *terminale* Symbole.
Eine *Grammatik* ist gegeben durch folgende *Ersetzungsregeln*:

(1) S ::= a
(2) S ::= a S a
(3) S ::= S b

Bemerkung:
Die in der BNF-Notation für Nonterminals vorgesehenen spitzen Klammern wurden hier weggelassen.

Aufgabe 1

a) Bilde einige *Wörter* (*Programm-Texte*), die zu der oben beschriebenen *Grammatik* gehören.
b) Wie lassen sich *Wörter* charakterisieren, die durch diese *Grammatik* beschrieben werden?

Lösung:

a) Beachte: Solange noch ein **S** vorkommt, muß **S** ersetzt werden, bis das entstandene *Wort* aus lauter *Terminal-Zeichen* besteht.
Linksableitung (ausgehend vom Startsymbol **S**, „top-down"):

$S \xrightarrow{2} a\,S\,a \xrightarrow{1} a\,a\,a$

$S \xrightarrow{2} a\,S\,a \xrightarrow{3} a\,S\,b\,a \xrightarrow{1} a\,a\,b\,a$

$S \xrightarrow{3} S\,b \xrightarrow{3} S\,b\,b \xrightarrow{2} a\,S\,a\,b\,b \xrightarrow{3} a\,S\,b\,a\,b\,b \xrightarrow{1} a\,a\,b\,a\,b\,b$

b) Zu dieser *Sprache* gehören offenbar *Wörter*,
- die genau aus ungeradzahlig vielen **a**'s bestehen
- die als erstes Zeichen ein **a**, gefolgt von beliebig vielen **b**'s, haben (wende wiederholt (3) und zuletzt (1) an)

Beispiel: $S \xrightarrow{3} S\,b \xrightarrow{3} S\,b\,b \xrightarrow{3} S\,b\,b\,b \xrightarrow{1} a\,b\,b\,b$

DEFINITION:

Eine *Satzgliederungsgrammatik* **G** ist durch folgende Bestandteile gegeben:

(1) eine *endliche Menge* **T**; ihre Elemente heißen *Terminalzeichen*.
(2) eine *endliche Menge* **N**; ihre Elemente heißen *nicht-terminale Zeichen*; in dieser Menge **N** ist ein *Startzeichen* **S** ausgezeichnet.
(3) endlich viele *Ersetzungsregeln*, genannt *Produktionen* **P**.

Die von der *Grammatik* **G** bestimmte *formale Sprache* **L(G)** besteht aus allen *Wörtern* (bzw. *Zeichenketten, Sätzen, Programmtexten*) über **T**, die – ausgehend vom Startzeichen **S** – durch endlich viele Anwendungen der Produktionen erzeugt werden können.

Aufgabe 2

T := { x , + , (,) }
N := { A , B , C , **S** }

Produktionen **P**:
(1) S ::= x | (B)
(2) B ::= S C
(3) C ::= + S | ε

Erzeuge das Wort:
((x + (x + x)))

Aufgabe 3

Lexikalische Analyse von Namen (Bezeichner, identifier):

T := {a, b, . . . , z, A, B, . . . , Z, _, 0, 1, ... , 9}
N := { <name>, <buchstabe>, <ziffer> }
Startzeichen: **S** = <name> (beachte: **S** ∈ **N**)

Produktionen **P**:

(1) <name> ::= <buchstabe> | <name> <buchstabe> | <name> <ziffer>
(2) <buchstabe> ::= a | b | c | . . . . . | z | _ | A | B | C | . . . . . | Z
(3) <ziffer> ::= 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9

Zeige: Diese Grammatik **G** ist linkslinear vom Typ 3.
Zeichne den Graph eines endlichen Automaten, der Identifier erkennt (beachte die Seiten 4 und 5; Lösung Seite 7).

DEFINITION:

Eine Grammatik **G** heißt *regulär*, wenn alle Produktionen von der Form

(R) A ::= aB , A::= a (Rechtslinearität)

sind, oder wenn alle Produktionen die Form

(L) A ::= Ba , A ::= a (Linkslinearität)

haben. Die zugehörige formale Sprache **L(G)** heißt regulär.

Wir beschränken uns im folgenden auf linkslineare Grammatiken, was keine Einschränkung darstellt; eine linksreguläre Grammatik nennen wir auch Grammatik vom Typ 3, die zugehörige reguläre Sprache heißt vom Typ 3.

DEFINITION:

Ein **endlicher Automat (Akzeptor)** ist bestimmt durch

- eine nichtleere, endliche Menge **Z** von Zuständen,
- eine nichtleere, endliche Menge **E** von Eingabesymbolen (Eingabealphabet),
- eine Überführungsfunktion f : Z x E $\rightarrow$ Z, die jedem Paar aus aktuellem Zustand und Eingabe einen Folgezustand zuordnet,
- einen Anfangszustand $\mathbf{z_0}$ aus der Menge **Z**,
- mindestens einen Endzustand $\mathbf{z_E}$ aus der Menge **Z**.

Wir verdeutlichen einen endlichen Automaten durch einen Graphen: Für jedem Zustand aus **Z** zeichnen wir einen Knoten. Von den Knoten gehen gerichtete Kanten aus, wobei eine Kante mit dem jeweiligen Eingabesymbol aus der Menge **E** beschriftet wird. Eine Kante endet bei demjenigen Knoten (Zustand), in den der Automat nach Lesen des Eingabesymbols übergeht.

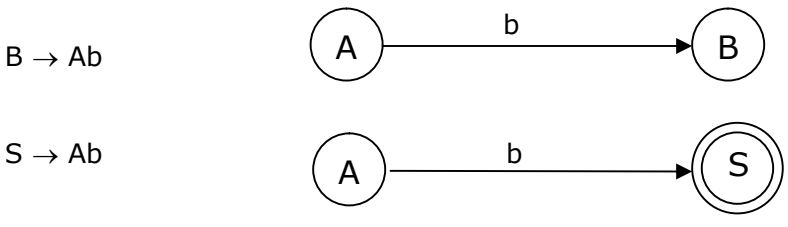
Es gilt folgender

SATZ: Ein endlicher Automat erkennt eine Sprache genau dann, wenn sie regulär ist.

Der strenge Beweis dieses Satzes ist schwierig; für linkslineare Grammatiken führen wir konstruktiv eine Plausibilitätsbetrachtung durch:

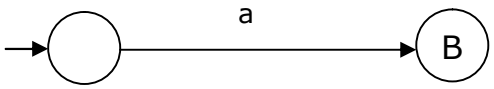
(1) Jedem Element der Menge **N** der Nonterminals einer Sprache ist ein Zustand (Knoten) des endlichen Automaten zugeordnet; Ausnahme: dem Anfangszustand entspricht kein Nonterminalzeichen. Das Nonterminal **S** (Startzeichen) entspricht dem Endzustand.

(2) Jeder Produktion B $\rightarrow$ Ab entspricht eine gerichtete Kante mit der Bewertung b (b $\in$ **T**) vom Knoten A (Zustand A) zum Knoten B (Zustand B).



(3) Jeder Produktion B $\rightarrow$ a entspricht eine gerichtete Kante vom Anfangszustand zum Knoten B (Zustand B) mit der Bewertung a, a $\in$ **T**. Auf den Anfangszustand dürfen keine Kanten hinführen.

B $\rightarrow$ a



Um den Zusammenhang zwischen einer regulären Sprache und einem endlichen Automat, der diese Sprache erkennt, klarzumachen, betrachten wir folgende durch das Quadrupel **(N, T, P, S)** gegebene Grammatik **G** vom Typ 3:

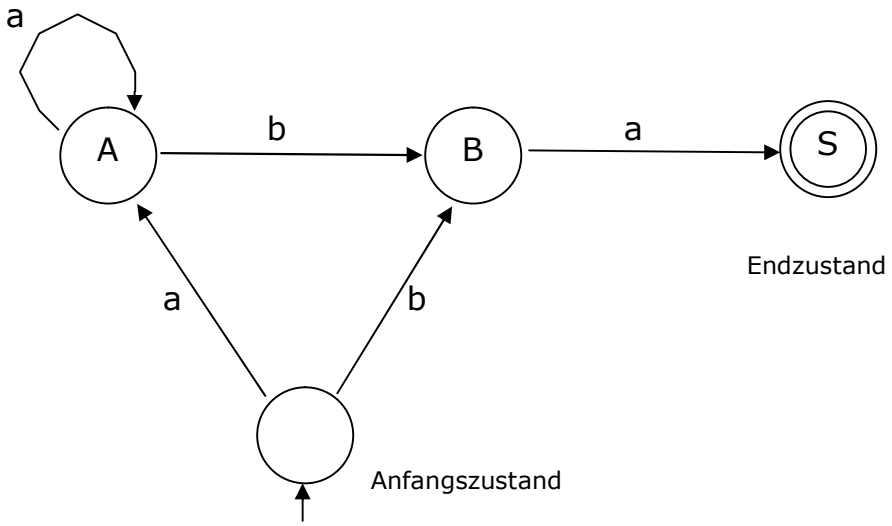
G = (N, T, P, S)

T := {a, b} (Eingabealphabet des endlichen Automaten)
N := {A, B, **S**} (Menge der Zustände mit **S** = Startzeichen = Endzustand)

Produktionen **P**:

(1) A $\rightarrow$ a | Aa
(2) B $\rightarrow$ b | Ab
(3) S $\rightarrow$ Ba

Der endliche Automat **DFA**, der zu dieser Grammatik gehört:



DFA = „deterministic finite acceptor"

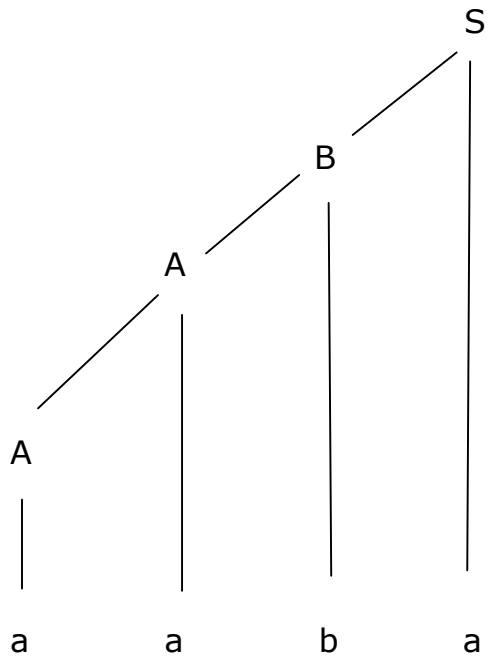
Die zu dieser Grammatik **G** gehörende Sprache **L(G):**

L(G) = {ba, aba, aaba, aaaba, aaaaba, aaaaaba, ... }
= {w | w = a^nba mit n $\in \mathbb{N}_0$}

Linksableitung des Wortes **aaaaba** („top-down"; Linksableitung bedeutet, daß das jeweils am weitesten links stehende Nonterminal-Zeichen ersetzt wird):

S $\rightarrow$ Ba $\rightarrow$ Aba $\rightarrow$ Aaba $\rightarrow$ Aaaba $\rightarrow$ Aaaaba $\rightarrow$ aaaaaba

Syntaxbaum für das Wort **aaba** („bottom-up"):



Ein Beispiel für eine Sprache **L**, die nicht regulär ist und folglich von einem endlichen Automaten nicht erkannt wird:

Eingabe-Alphabet = **T** := {a, b}

L = {w | w = $a^n b^n$ mit n $\in$ **N** } = {ab, aabb, aaabbb, aaaabbbb, ... }

Daß wir gerade diese Sprache betrachten, hat folgenden Grund:

Interpretiert man a als öffnende, b als schließende Klammer, so stellt **L** die Menge der Klammerstrukturen beliebiger Tiefe dar. Solche Klammern treten nicht nur bei arithmetischen Ausdrücken auf, sondern auch bei allen blockorientierten Sprachen wie C++, Pascal, Java oder Python; in Pascal erfolgt die Klammerung eines Blocks mit ***begin*** und ***end***, in Java mit geschweiften Klammern ***{*** und ***}***, in Python wird ein Anweisungsblock durch Einrücken gekennzeichnet.

Versuch, ein Regelsystem für eine die Sprache **L** beschreibende reguläre Grammatik zu finden:

N := {A, **S**}

Produktionen **P**:

(1) S $\rightarrow$ Sb | Ab
(2) A $\rightarrow$ Aa | a

a) Konstruiere den endlichen Automat Au, der zu dieser Grammatik gehört.
b) Zeige, daß auch die "falschen" Wörter $a^n b^m$ mit $n \neq m$ erkannt werden.
c) Gib ein Regelsystem (Produktionen) an, so daß diese Sprache erkannt wird. (Diese Sprache ist nicht regulär, sondern heißt ***contextfrei*** oder ***vom Typ 2***.)

Lösung von Aufgabe 3 (Seite 3):

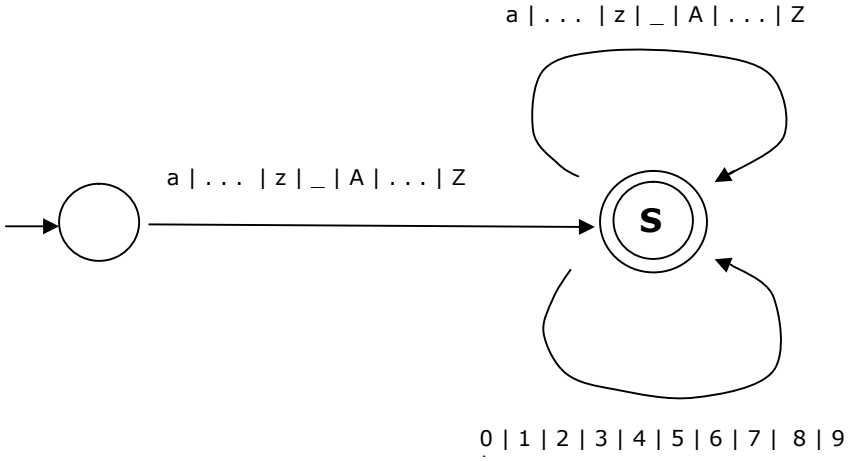
Für das Nonterminal <name>, welches auch Startzeichen ist, schreiben wir **S**; dann läßt sich die Grammatik **G = (N, T, P, S)**, welche Bezeichner in der Programmiersprache Python erzeugt, als linkslineare Grammatik formulieren:

T := {a, b, c, . . . , z, _, A, B, C, . . . , Z, 0, 1, 2, 3, ... , 9}
N := { **S** } mit **S** = Startsymbol

Produktionen **P:**

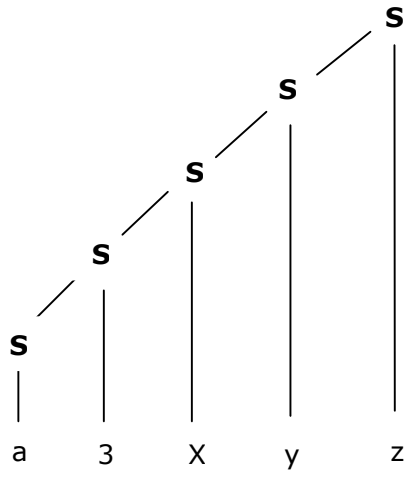
(1) **S** → a | b | . . . | z | _ | A | B | . . . | Z
(2) **S** → **S**a | **S**b | . . . | **S**z | **S**_ | **S**A | **S**B | . . . | **S**Z
(3) **S** → **S**0 | **S**1 | **S**2 | . . . . | **S**9

Der zu dieser Grammatik **G** gehörende **DFA**:



Beispiel:
Wir zeigen, daß die Zeichenkette a3Xyz ein gültiger Bezeichner, also ein Wort der von der Grammatik **G** erzeugten Sprache **L(G)** ist, indem diese Zeichenkette (Programmtext oder hier: Teil eines Programmtextes) solange reduziert wird, bis die gesamte Zeichenkette auf das Startsymbol **S** zurückgeführt wurde:

a) Syntaxbaum für das Wort a3Xyz („bottom-up"):



b) Linksableitung für das Wort a3Xyz („top-down"):

S $\overset{(2)}{\rightarrow}$ **S**z $\overset{(2)}{\rightarrow}$ **S**yz $\overset{(2)}{\rightarrow}$ **S**Xyz $\overset{(3)}{\rightarrow}$ **S**3Xyz $\overset{(1)}{\rightarrow}$ a3Xyz

Lokale Teilbereiche (hier: Syntax von Bezeichnern) einer Programmiersprache können durch eine reguläre Grammatik (Grammatik vom Typ 3) beschrieben werden.

Aufgaben:

4. Gegeben ist die (linkslineare) Grammatik **G** mit

   T = {a; b} **N** = {A; B; **S**} **S** = Startzeichen

   Produktionen **P**:
   (1) A → a
   (2) A → Ab
   (3) B → Sb
   (4) S → Aa
   (5) S → Sa
   (6) S → Ba

   a) Gib den zu dieser Grammatik gehörenden DFA („deterministic finite acceptor") an.
   b) Charakterisiere die zur Grammatik **G** gehörende Sprache **L(G)**.
   c) Konstruiere die Syntaxbäume zu den Wörtern aaba, ababa, abab („bottom-up")

5. Gegeben sei zum Eingabealphabet **T** = {0;1} die Menge T^* aller Wörter (Zeichenketten), die aus den Elementen von **T** gebildet werden können; betrachte die Sprache

   L(G) := {w ∈ T^* | die Anzahl der Einsen in w ist gerade} .

   a) Konstruiere einen DFA, der **L(G)** erkennt.
   b) Gib die Syntaxregeln (Produktionen **P**) an.
   c) Zeige, daß es für das Wort 01101100 einen korrekten Syntaxbaum gibt, nicht hingegen für das Wort 01011.

6. Ein Grammatik **G** sei definiert durch

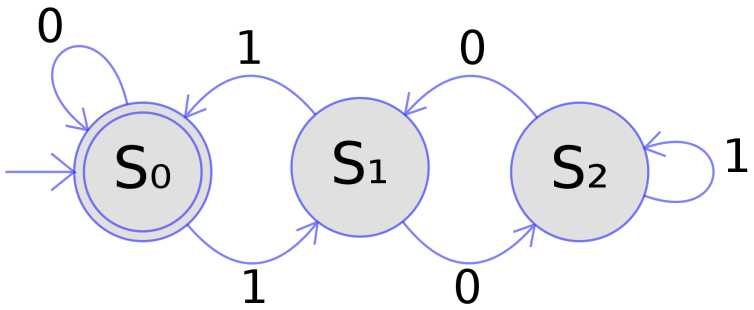
   T = {a, b, c} **N** = {R, **S**}

   Produktionen **P**:

   (1) S ::= c
   (2) S ::= aR
   (3) R ::= Sb

   Zeige: a) **L(G)** = { w | w = $a^n cb^n$, n ∈ $\mathbf{N_0}$} = { c, acb, aacbb, aaacbbb, ... }
   b) Es gibt keine reguläre Grammatik für diese Sprache (siehe S. 9 oben).

7. Gegeben ist der folgende **DFA** (https://en.wikipedia.org/wiki/Deterministic_finite_automaton):



   a) Gib die Mengen **T** und **N** sowie das Startsymbol **S** an.
   b) Wie lauten die Produktionen **P**?
   c) Zeige: 11, 1001, 10101, 1011101 ∈ **L(G)**
   d) Gib für die Wörter aus c) jeweils die Linksableitung und den Syntaxbaum an; Hinweis: Ergänze obenstehenden Graphen geeignet und formuliere eine weitere Produktionsregel, beachte hierzu den folgenden Originaltext zu obenstehendem DFA:

   An example of a deterministic finite automaton that accepts only binary numbers that are multiples of 3. The state S_0 is both the start state and an accept state. For example, the string "1001" leads to the state sequence S_0, S_1, S_2, S_1, S_0, and is hence accepted.

KONTEXTFREIE GRAMMATIKEN UND KONTEXTFREIE SPRACHEN (TYP 2)

Context free grammars (CFG) and context free languages (CFL)

Zur Grammatik **G** = (**T**, **N**, **P**, **S**) mit Eingabealphabet **T** = {a; b; c} und **N** = {A; **S**} (**S**=Startsymbol) sei die Sprache

L(G) := {w | w = $a^n cb^n$, n = 0, 1, 2, . . . } = {c, acb, aacbb, aaacbbb, ... }

gegeben. Falls man versucht, zu dieser Sprache **L(G)** ein linksreguläre Grammatik mit den Produktionen **P**

(1) S → Sb | Ac | c
(2) A → Aa | a

zu formulieren (Aufgabe 6.b) auf der vorigen Seite), sieht man, daß der zugehörige DFA zwar die „richtigen" Wörter c, acb, aacbb, aaacbbb, ... erkennt und daß es korrekte Syntaxbäume für diese Wörter gibt, daß allerdings ebenso die „falschen" Wörter ac, acbb, aaacb, . . . (also $a^n cb^m$ mit $n \neq m$) erkannt werden; denn zum Abarbeiten der a bedarf es der rekursiven Regel A → Aa, zum Abarbeiten der b der rekursiven Regel S → Sb. Und der DFA ermöglicht nicht zu zählen und festzuhalten, wie oft diese rekursiven Regeln jeweils angewandt wurden!
Bemerkung: Eine Produktionsregel heißt rekursiv, wenn ein Nonterminal auf der linken Seite der Regel auch auf deren rechter Seite vorkommt.

Mehrfach geschachtelte Klammerungen, wie durch **L(G)** = {w | w = $a^n cb^n$ } beschrieben, treten nicht nur arithmetischen Termen, sondern auch als Programmstruktur in allen blockorientierten Sprachen wie Python, Pascal, Java usw. auf. Um die oben formulierte Sprache **L(G)** zu erkennen, muß man das Regelsystem erweitern.

Vereinbarung:
Im Folgenden verstehen wir unter dem Symbol ω eine beliebige Aneinanderreihung von Terminals oder Nonterminals, z. B. ω = AbaSa.

DEFINITION:
Eine zur Grammatik G gehörende Sprache L(G) heißt kontextfrei (oder kontextunabhängig, engl.: contextfree) genau dann, wenn alle Produktionen die Form

A → ω (andere Schreibweise: **A ::= ω**)

mit A ∈ N haben.

Bemerkungen:
Eine kontextfreie Grammatik nennen wir auch Grammatik vom Typ 2, die zugehörige kontextfreie Sprache heißt vom Typ 2.
Eine Produktion aAb ::= aBaS ist dagegen nicht kontextfrei in dem Sinne, daß man das Nonterminal A nicht einfach durch aBaS ersetzen darf, sondern nur dann, wenn es im Zusammenhang (im Kontext) mit einem voranstehenden a und einem folgenden b vorkommt; hier ist die Zeichenkette aAb durch aBaS zu ersetzen. Bei kontextfreien Sprachen steht das auf der linken Seite einer Produktion stehende Nonterminal in keinem Kontext anderer Zeichen.
Teilbereiche von Python (z. B. Identifier) lassen sich durch eine reguläre Grammatik (Aufgabe 3, S. 3 und S. 7) beschreiben, insgesamt ist Python mindestens eine kontextfreie Programmiersprache, also vom Typ 2.
Natürliche Sprachen sind nicht kontextunabhängig (daß man den Satz "die Maus jagt die Katze" bilden konnte, liegt daran, daß die Produktionen kontextfrei definiert waren; solche semantisch unsinnigen Sätze kann man dadurch ausschließen, indem die Produktionen kontextabhängig formuliert werden.).

Aufgabe 8

Zur Grammatik **G** = (**T**, **N**, **P**, **S**) mit Eingabealphabet **T** = {a; b; c}, **N** = {A; **S**} (**S**=Startsymbol) und den Produktionen **P**

(1) S ::= c
(2) S ::= aR
(3) R ::= Sb

gehört die Sprache

L(G) := {w | w = $a^n c b^n$, n = 0, 1, 2, . . . } (siehe Aufg. 6).

Definiere die Grammatik **G'** = (**T**, **N**, **P**, **S**) mit **T** = {a; b; c}, **N** = {**S**} (**S**=Startsymbol) und den Produktionen **P**

(1) S ::= c
(2) S ::= aSb (zentralrekursive Regel)

Zeige: **L(G)** = **L(G')**

Definition:
Zwei Grammatiken **G** und **G'** heißen äquivalent genau dann, wenn gilt: **L(G)** = **L(G')**

Aufgabe 9

a) Formuliere eine Grammatik **G**, damit die Sprache
L(G) = { w | w = $a^n b^n$ } = { ab, aabb, aaabbb, aaaabbbb, ... } ; n = 1, 2, . . .
erkannt wird.

b) Zeige, daß es für das Wort aaabbb einen korrekten Syntaxbaum und eine korrekte Linksableitung gibt, für aabbb dagegen nicht.

DEFINITION:
Eine Grammatik G heißt strukturell mehrdeutig (ambigous) genau dann, wenn die zugehörige Sprache L(G) Wörter (bei Programmiersprachen: Quelltexte) enthält, für die es unterschiedliche Syntaxbäume gibt.

Bemerkung: Von lexikalischer Mehrdeutigkeit spricht man, wenn ein Terminal (in einer natürlichen Sprache: ein Wort) mehrere Bedeutungen besitzt; Beispiel: In dem Satz „Das Schloß wurde im 16. Jahrhundert gebaut" kann mit dem Wort „Schloß" ein Gebäude oder eine Schließvorrichtung gemeint sein.

Weitere Beispiele für lexikalische Mehrdeutigkeit in natürlichen Sprachen:
"Der Gefangene floh" ↔ *"Der gefangene Floh"*
"Time flies like an arrow" ↔ *"Fruit flies like a banana".*

Aufgabe 10

Gegeben ist die Sprache **L(G)** zur Grammatik **G** = (**T**, **N**, **S**, **P**) mit
T = { +, *, (,), a, b, c, . . . , z }
N = { **S**, V }, **S** = Startzeichen
Produktionen **P**:
(1) S → V | (S) | S + S | S * S
(2) V → a | b | c | . . . | z

Zeige:
a) **(a + b) * c** ∈ **L(G)** (Linksableitung, Syntaxbaum)
b) Für das Wort **a + b * c** lassen sich Syntaxbäume auf zwei strukturell verschiedene Arten angeben! Erläutere die Konsequenzen für die Abfolge der Rechenschritte.
Lösung: Seite 17

Aufgabe 11

Gegeben ist die Sprache **L(G)** zur Grammatik **G** = (**N**, **T**, **A**, **P**) mit

- Menge der Terminals: **T** := {+ , – , * , / , (,) , a , b , c , d , e }
- Menge der Nonterminals: **N** := { **A** , S , V } mit **A** = Startsymbol
- Produktionsregeln **P**:

  (1) A → V
  (2) S → A + A
  (3) S → A – A
  (4) A → A * A
  (5) A → A / A
  (6) A → S * S
  (7) A → S / S
  (8) A → (S) | S
  (9) V → a | b | c | d | e

a) Zeige, daß das Wort (a + b) / (c – d) zur Sprache **L(G)** gehört! (Syntaxbaum und Linksableitung)

b) Beweise, daß die Grammatik **G** strukturell mehrdeutig ist, indem man zu dem Wort a * b + c zwei strukturell verschiedene Syntaxbäume entwickelt.

Bezeichnungen für die Nonterminals: A(usdruck), S(umme), V(ariable)

Aufgabe 12

„Dangling-else" ambiguity

Gegeben: Grammatik **G** = (**N**, **T**, **S**, **P**) mit

- **T** := { if, else, s1, s2, c1, c2 }
- **N** := { E, **S** } mit **S** = Startsymbol
- Produktionsregeln **P**:

  (1) S → if E S
  (2) S → if E S else S
  (3) S → s1 | s2
  (4) E → c1 | c2

Bedeutung der Terminals:
s1, s2 (statement1, statement2) stehen jeweils für Anweisungen oder Anweisungsblöcke
c1, c2 (condition1, condition2) stehen jeweils für Boolesche Terme

Zeige:
Für das Wort **`if c1 if c2 s1 else s2`** gibt es verschiedene Syntaxbäume.

Möglichkeiten, um der Mehrdeutigkeit zu begegnen:

- *Der else-Zweig bezieht sich immer auf das nächststehende if.*
- *Kennzeichnung von Anweisungsblöcken durch entsprechende Strukturierung des Quelltextes (in Python: Strukturierung durch Einrücken; in Pascal: Strukturierung mit den Schlüsselwörtern* **`begin`** *und* **`end`***, durch die ein Anweisungsblock jeweils „geklammert" wird)*

```
if c1:                  if c1:
    if c2:                  if c2:
        s1                      s1
    else:               else:
        s2                  s2
```

Bemerkungen:

- Es gibt kontextfreie Sprachen (CFLs; Sprachen vom Typ 2), die inhärent mehrdeutig (inherently ambiguous) sind, d. h. jede Grammatik für diese Sprache ist mehrdeutig. Eine kontextfreie Sprache heißt eindeutig, sobald sich eine eindeutige Grammatik angeben läßt, die diese Sprache erzeugt.
- Eine reguläre Sprache (Sprache vom Typ 3) kann nicht inhärent mehrdeutig sein, da sich stets eine eindeutige Grammatik angeben läßt, die diese Sprache erzeugt.
- Die Frage, ob zwei Grammatiken dieselbe Sprache erzeugen und damit äquivalent sind, ist allgemein nicht entscheidbar.
- Es ist grundsätzlich nicht möglich, für eine gegebene kontextfreie Grammatik mit einem allgemeinen Algorithmus zu entscheiden, ob sie eindeutig oder mehrdeutig ist.
- Gleichwohl gelingt es in der Praxis in aller Regel, eine eindeutige kontextfreie Grammatik zu formulieren (indem man z. B. die möglichen Fälle durchspielt).
- Syntaxbäume, bei denen das Startzeichen als Wurzel, die Nonterminals als innere Knoten und die Terminals als Endknoten (Blätter) auftreten, lassen sich nur bei Typ-3 oder Typ-2-Sprachen sinnvoll erstellen, also bei Sprachen, bei denen die „linke" Seite jeder Produktionsregel aus genau einem Nonterminal-Zeichen besteht.

Aufgabe 13

Gegeben ist die Grammatik **G**, bestehend aus der Menge **T** der Terminalzeichen, der Menge **N** der Nonterminalzeichen, der Menge **P** der Produktionen und dem Element **S**$\in$**N** als Startzeichen:

T := { a, b, p, q, if, then, else}

N := {**S**, S_1, S_2, B, T} mit **S**=Startzeichen

Produktionen **P**:

(1) S $\rightarrow$ S_1 | S_2

(2) S_1 $\rightarrow$ T | if B then S_1 else S_2

(3) S_2 $\rightarrow$ T | if B then S | if B then S_1 else S_2

(4) B $\rightarrow$ p | q

(5) T $\rightarrow$ a | b

Zeige: Das Wort

`if p then if q then a else b`

besitzt in dieser Grammatik nur einen einzigen Syntaxbaum!

Beispiel einer **mehrdeutigen Grammatik** bei einer natürlichen Sprache
Quelle:
https://www.uni-ulm.de/fileadmin/website_uni_ulm/iui.inst.040/Formale_Methoden_der_Informatik/Vorlesungsskripte/FMdI-06--2010-01-10--FormaleSprachen_Vorlesung.pdf

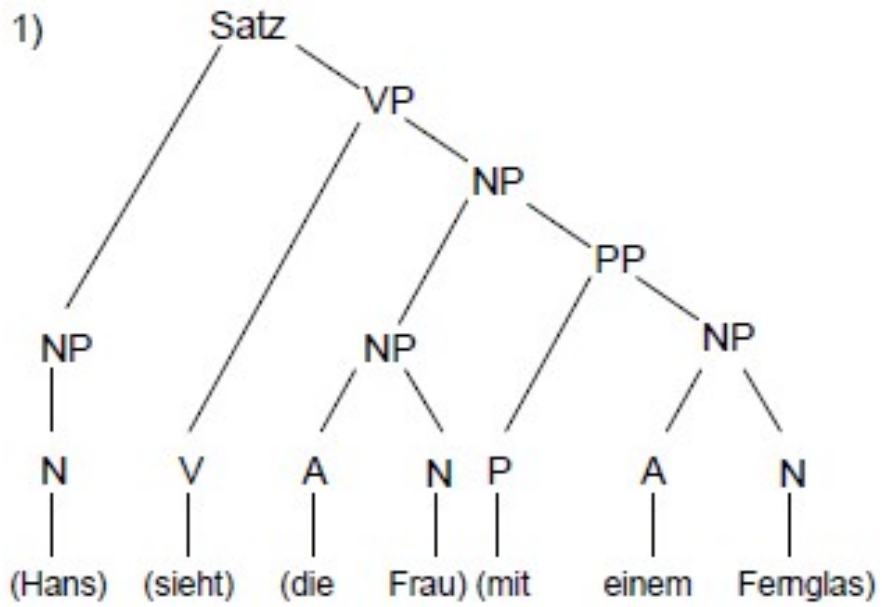
T = {mit, in, auf, Hans, Frau, Fernglas, Park, sieht, geht, der, die, das, einem}
N = {**Satz**, NP, VP, PP, N, A, V, P} mit **Satz**=Startsymbol

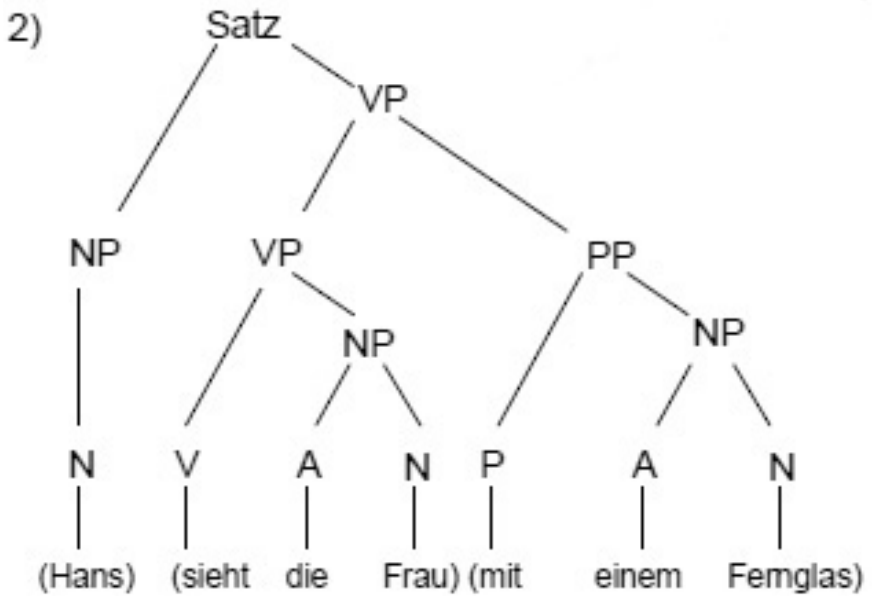
Produktionen **P**:
(1) Satz $\rightarrow$ NP VP
(2) NP $\rightarrow$ NP PP | A N | N
(3) VP $\rightarrow$ VP PP | V NP | V
(4) PP $\rightarrow$ P NP
(5) P $\rightarrow$ mit | in | auf
(6) N $\rightarrow$ Hans | Frau | Fernglas | Park
(7) V $\rightarrow$ sieht | geht
(8) A $\rightarrow$ der | die | das | einem

Zu dem Satz

„**Hans sieht die Frau mit einem Fernglas**"

lassen sich zwei strukturell verschiedene Syntaxbäume angeben:





Beachte: Die Klammern sind nicht Bestandteil des zu analysierenden Satzes, sondern dienen dazu, die unterschiedliche Semantik zu verdeutlichen.

Aufgabe 14

Gegeben ist die Menge der Terminalzeichen **T** = {a, b, c, (,), +, *}.
Wir definieren die folgenden Grammatiken $\mathbf{G_1}$ und $\mathbf{G_2}$:

$\mathbf{G_1}$ = (T, N, P, S) mit **N** = {I, R, Q, S}, **S**=Startsymbol
Produktionen **P**:

(1) I → a | b | c
(2) Q → R | Q * R
(3) R → I | (S)
(4) S → Q | S + Q

G_2 = (T, N, P, S) mit **N** = {I, S} , **S**=Startsymbol
Produktionen **P**:

(1) I → a | b | c
(2) S → I | I * S | I + S | (S)

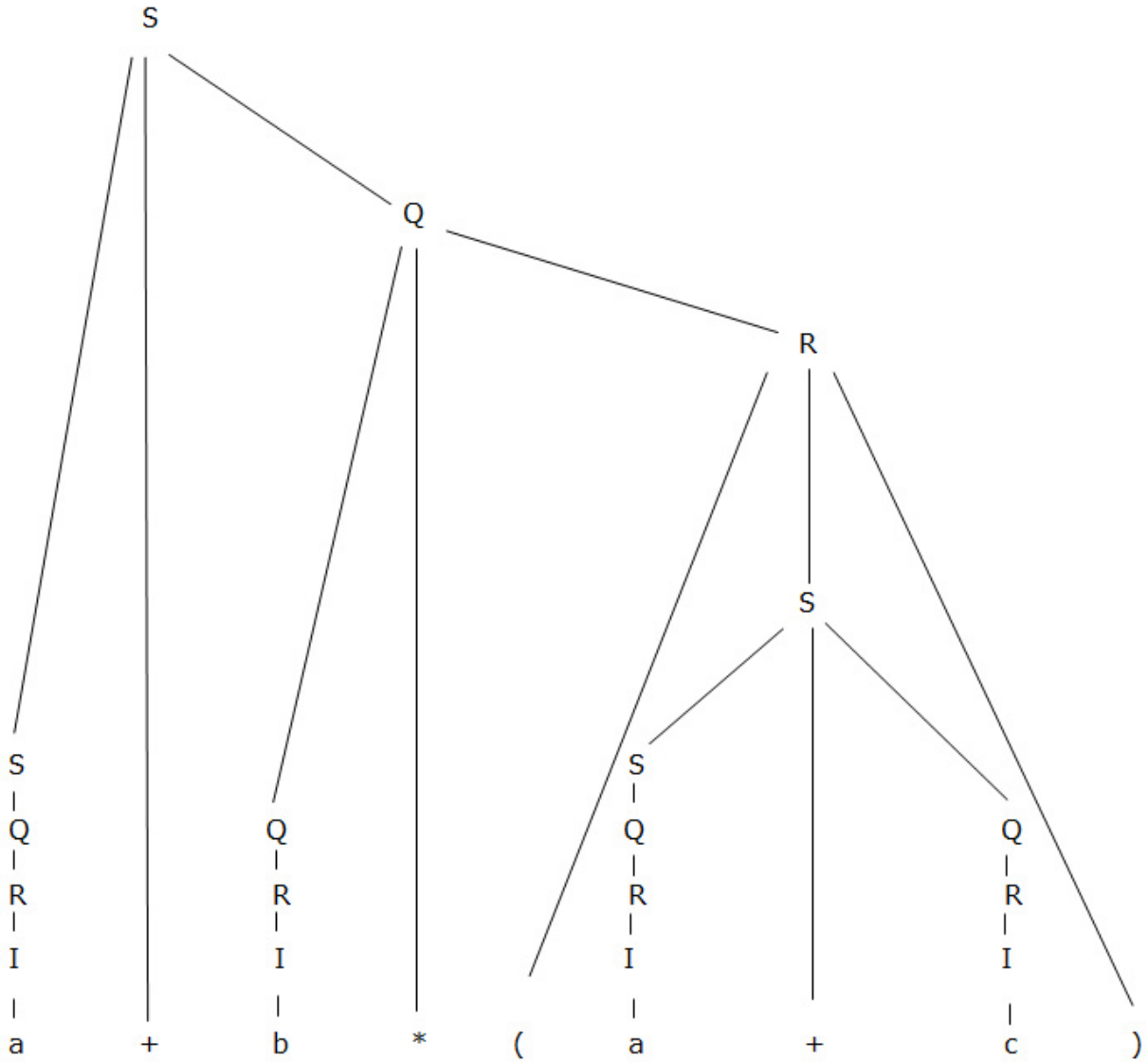
a) Zeige: Das Wort a + b * (a + c) gehört sowohl zur Sprache **$L(G_1)$** als auch zur Sprache **$L(G_2)$**, indem man bei **G_1** und **G_2** jeweils einen Syntaxbaum und eine Linksableitung angibt. (Bemerkung: G_1 und G_2 sind äquivalent.)

b) Analysiere das Wort a * b + a * c sowohl nach **G_1** als auch nach **G_2**.

c) Analysiere das Wort a * (b + c) sowohl nach **G_1** als auch nach **G_2**.

Lösungen zu Aufgabe 14 a), b):

a) α) **a + b * (a + c) ∈ $L(G_1)$**



Linksableitung:
Beachte: das am „weitesten links" stehende Nonterminal wird jeweils ersetzt.

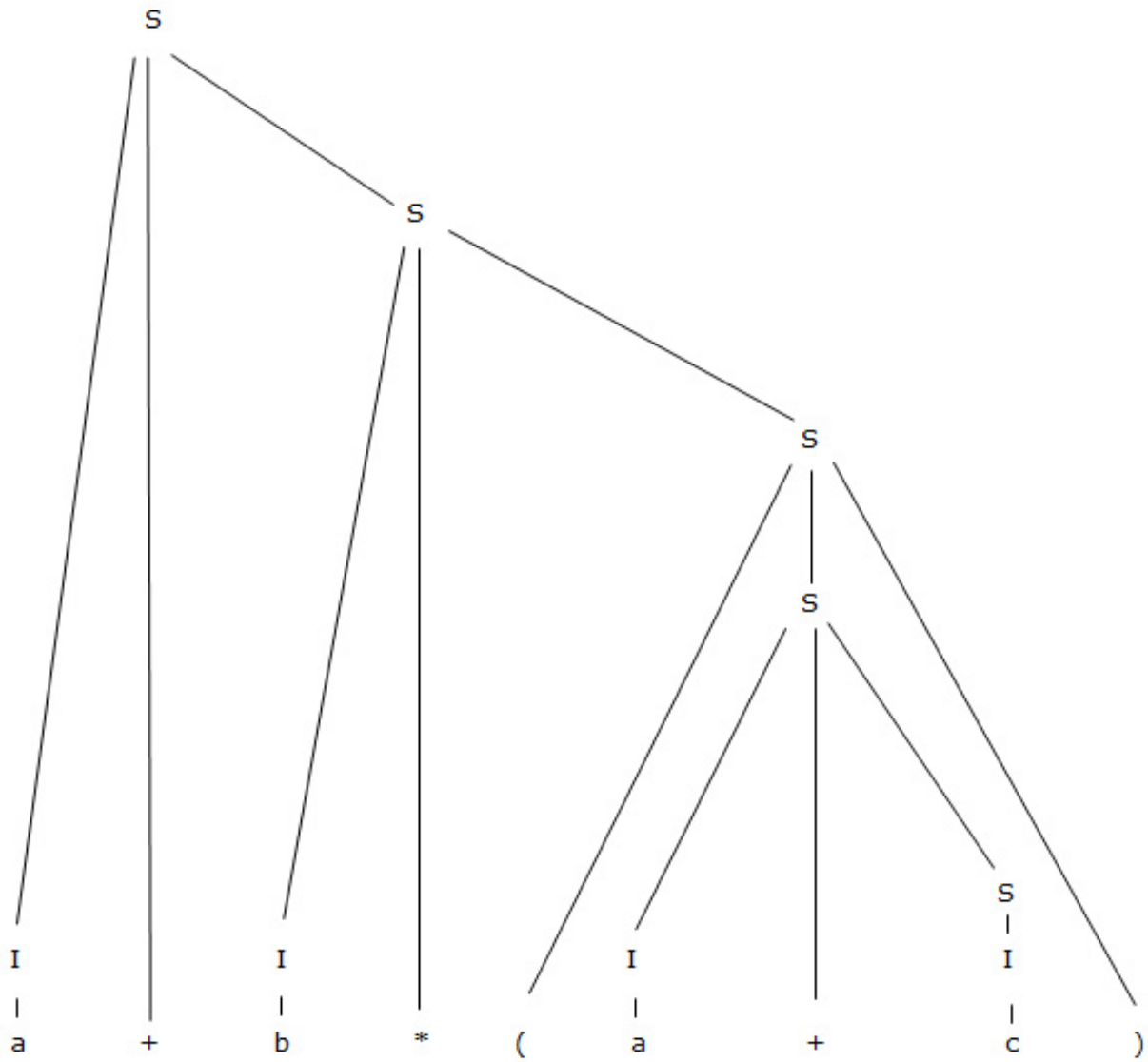
$S \rightarrow S + Q \rightarrow Q + Q \rightarrow R + Q \rightarrow I + Q \rightarrow a + Q \rightarrow a + Q * R$

$\rightarrow a + R * R \rightarrow a + I * R \rightarrow a + b * R \rightarrow a + b * (S) \rightarrow a + b * (S + Q)$

$\rightarrow a + b * (Q + Q) \rightarrow a + b * (R + Q) \rightarrow a + b * (I + Q) \rightarrow a + b * (a + Q)$

$\rightarrow a + b * (a + R) \rightarrow a + b * (a + I) \rightarrow a + b * (a + c)$

a) β) **a + b * (a + c) ∈ $L(G_2)$**

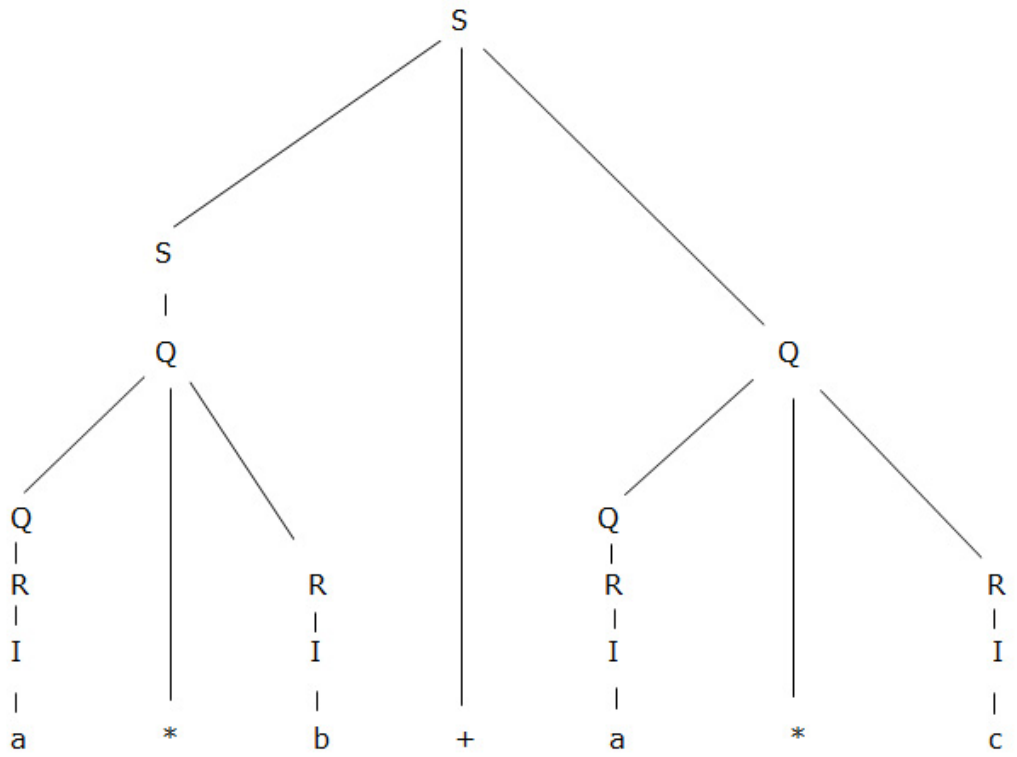


Linksableitung:

$S \rightarrow I + S \rightarrow a + S \rightarrow a + I * S \rightarrow a + b * S \rightarrow a + b * (S) \rightarrow a + b * (I + S)$

$\rightarrow a + b * (a + S) \rightarrow a + b * (a + I) \rightarrow a + b * (a + c)$

b) α) **a * b + a * c ∈ L(G_1)**



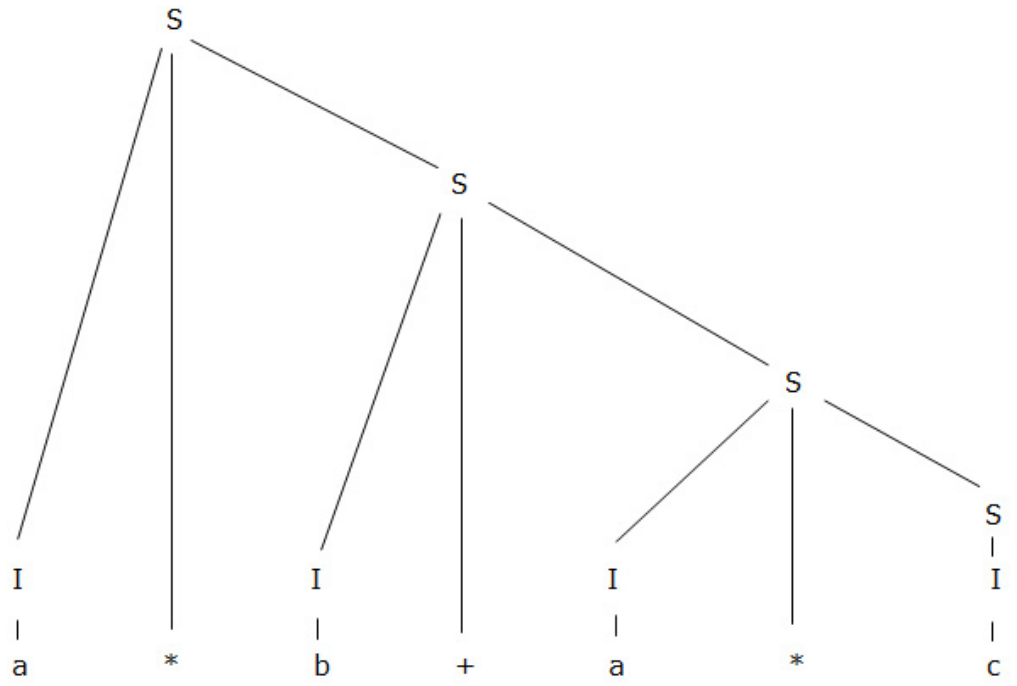
Linksableitung:

S → S + Q → Q + Q → Q * R + Q → R * R + Q → I * R + Q → a * R + Q

→ a * I + Q → a * b + Q → a * b + Q * R → a * b + R * R → a * b + I * R

→ a * b + a * R → a * b + a * I → a * b + a * c

b) β) **a * b + a * c ∈ L(G_2)**



Linksableitung:

$S \rightarrow I * S \rightarrow a * S \rightarrow a * I + S \rightarrow a * b + S \rightarrow a * b + I * S$

$\rightarrow a * b + a * S \rightarrow a * b + a * I \rightarrow a * b + a * c$

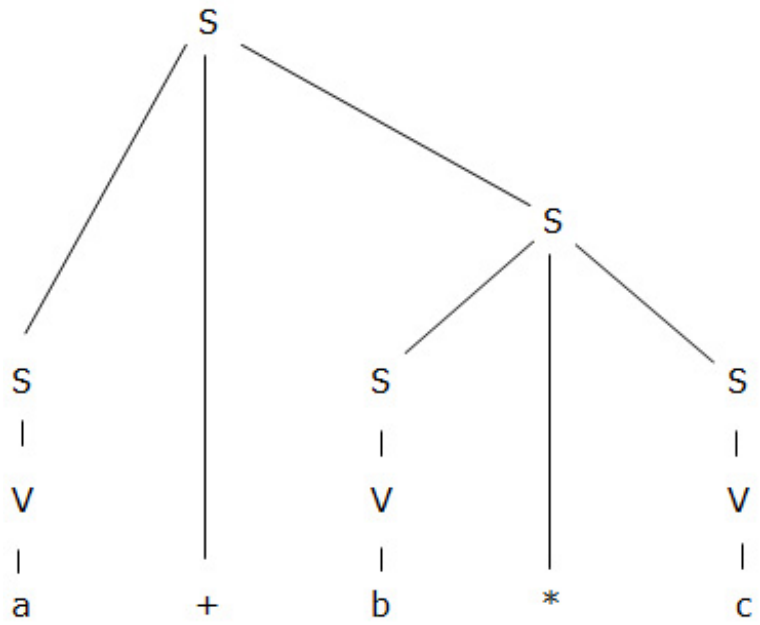
Der Term **a * b + a * c** wird in der Grammatik $\mathbf{G_1}$ als Summe, deren Summanden jeweils die Produkte **a * b** und **a * c** sind, verstanden; dagegen faßt die Grammatik $\mathbf{G_2}$ den Term **a * b + a * c** als Produkt mit den Faktoren **a** und (**b + a * c**) auf und beachtet nicht die allgemeingültige Vereinbarung „Punkt vor Strich". Daher ist die „kompliziertere" Grammatik $\mathbf{G_1}$ der „einfacheren" Grammatik $\mathbf{G_2}$ vorzuziehen, obwohl beide Grammatiken $\mathbf{G_1}$ und $\mathbf{G_2}$ äquivalent sind, denn $\mathbf{L(G_1) = L(G_2)}$.

Lösung zu Aufgabe 10 b), Seite 10:

a + b * c $\in$ **L(G)**

1. Lösung

Syntaxbaum 1:



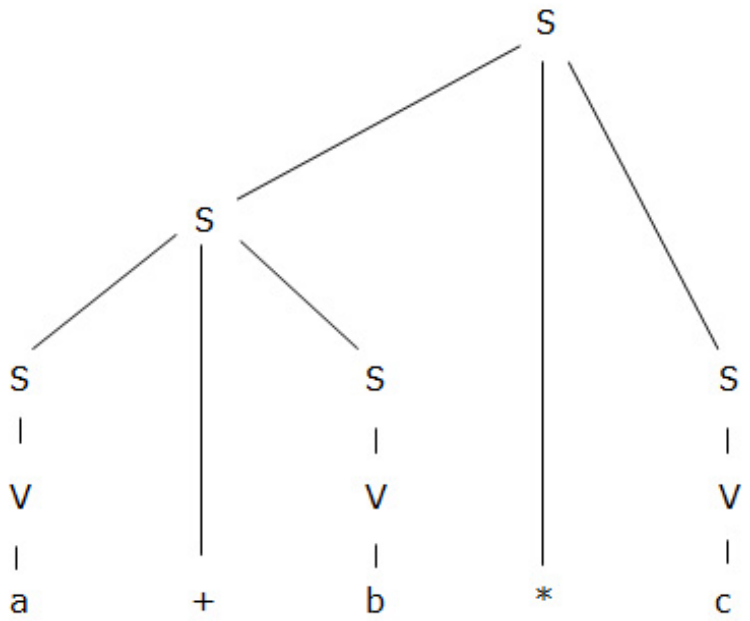
Linksableitung:

$S \rightarrow S + S \rightarrow V + S \rightarrow a + S \rightarrow a + S * S \rightarrow a + V * S \rightarrow a + b * S$

$\rightarrow a + b * V \rightarrow a + b * c$

2. Lösung

Syntaxbaum 2:



Linksableitung:

$S \rightarrow S * S \rightarrow S + S * S \rightarrow V + S * S \rightarrow a + S * S \rightarrow a + V * S \rightarrow a + b * S$

$\rightarrow a + b * V \rightarrow a + b * c$

Der Term **a + b * c** wird gemäß dem ersten Syntaxbaum als Summe mit den Summanden **a** und **b * c**, gemäß dem zweiten Syntaxbaum als Produkt mit den Faktoren (**a + b**) und **c** aufgefaßt.

Da sich zu dem Wort **a + b * c** zwei strukturell verschiedene Syntaxbäume in der Grammatik **G** angeben lassen, ist die Grammatik **G** strukturell mehrdeutig.

KONTEXTSENSITIVE GRAMMATIKEN UND KONTEXTSENSITIVE SPRACHEN (TYP 1)

Folgende Grammatik **G** sei geben durch das Quadrupel (T; N; S; P):

T := {a, b, c}

N:= {B, C, **S**} mit **S** = Startsymbol

Produktionen **P**:

(1) S → aSBC | aBC

(2) CB → BC

(3) aB → ab

(4) bB → bb

(5) bC → bc

(6) cC → cc

Zeige anhand von Beispielen (Linksableitungen für die Wörter abc, aabbcc, aaabbbccc): Die zur Grammatik **G** gehörende Sprache ist

L(G) = { w | w = $a^n b^n c^n$, n ∈ **N** } = { abc, aabbcc, aaabbbccc, aaaabbbbcccc, ... } .

Die Grammatik **G** ist nicht kontextfrei im Sinne der Definition auf Seite 9; vielmehr verlangen die Regeln (2) bis (6), daß die Nonterminals auf der linken Seite nur dann ersetzt werden können, wenn sie in einem bestimmten Kontext mit anderen Zeichen (Terminals oder Nonterminals) stehen. Die Ersetzungsregeln (2) bis (6) sind folglich kontextsensitiv.

Hinweis:
*Zu dieser Sprache **L(G)** läßt sich keine kontextfreie Grammatik (Grammatik vom Typ 2) angeben. Die oben definierte Grammatik **G** ist kontextsensitiv (Grammatik vom Typ 1).*

Auf die exakte Definition einer Typ-1- und einer Typ-0-Grammatik verzichten wir an dieser Stelle.

Es erhebt sich die Frage, von welchem Typ natürliche Sprachen sind. Folgende Beispiele erhellen, daß neben höheren Programmiersprachen (Pascal, C++, Python, Java) auch natürliche Sprachen mindestens kontextfrei, also mindestens vom Typ 2 sind:

Beispiel 1:

Ein Schüler, der die Qualifikation Block I, für die 35 Kurse, von denen höchstens sieben mit weniger als 5 Punkten bewertet wurden, gemäß §10 (1)-(8) einzubringen sind, erreicht hat, wird zur mündlichen Prüfung zugelassen.

Die Struktur dieses Satzes wird durch eine geeignete Formatierung des Textes deutlich:

```
Ein Schüler,                                                    wird zur mdl. Prüfung zugelassen.
   der die Qualifikation Block I,                                                  erreicht hat,
       für die 35 Kurse,                                gemäß §10(1)-(8) einzubringen sind,
          von denen höchstens sieben mit weniger als 5 Punkten    bewertet wurden,
```

Damit hat dieser Satz eine Syntax, die dem Regelsystem von Aufgabe 6 (Seite 8) entspricht (hier: 4 mal „Klammer auf", gefolgt von genau 4 mal „Klammer zu") und der folglich eine kontextfreie Grammatik (Typ 2) zugrunde liegt.

Beispiel 2:

Das Mädchen, das den Hund, der die Katze, die schnurrte, biß, sah, weinte.

Die Sätze aus diesen Beispielen sind syntaktisch korrekt gebildet; dennoch werden in der Praxis solche vierfachen Verschachtelungen gemieden, dreifache kommen kaum vor, zweifache dagegen sind durchaus üblich:

Dreifach:

Der Schüler, der die Qualifikation Block I, für die er mindestens 200 Punkte benötigt, erreicht hat, wird zur mündlichen Prüfung zugelassen.

Zweifach:

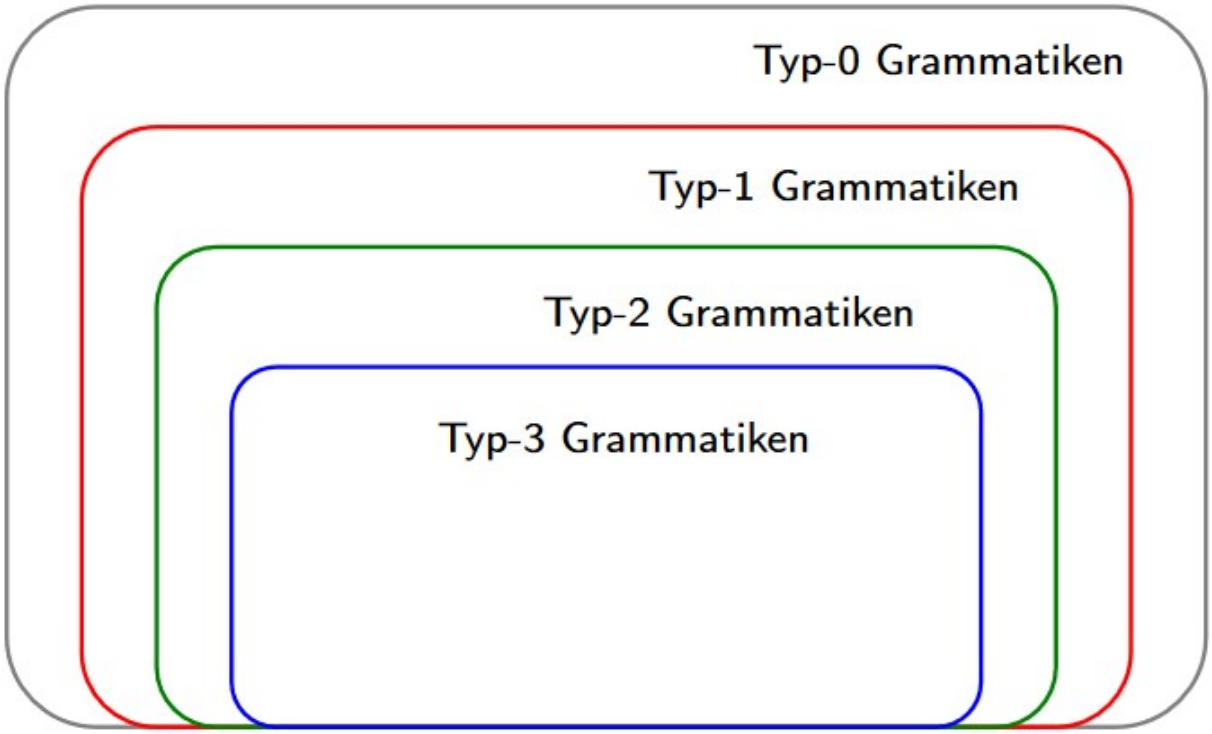
Der Schüler, der die Qualifikation Block I erreicht hat, wird zur mündlichen Prüfung zugelassen.

Wenn man solcher grammatikalischer Strukturen vom Typ 2 nicht mächtig ist, wird man den Inhalt des Beispiels 2 auch so formulieren können:

Das Mädchen weinte, das den Hund sah, der die schnurrende Katze biß.

Seit **NOAM CHOMSKY** grundlegende Arbeiten zur Klassifizierung formaler Sprachen (Typ 3 ↔ regulär, Typ 2 ↔ kontextfrei, Typ 1 ↔ kontextsensitiv, Typ 0 ↔ rekursiv-aufzählbar) verfaßt hat, ist man der Auffassung, daß natürliche Sprachen mindestens die Komplexität einer kontextsensitiven Sprache aufweisen. Allerdings ist zu vermuten, daß kontextsensitive grammatikalische Konstruktionen in der Praxis eher gemieden werden, was sogar für kontextfreie Konstruktionen gilt (siehe obige Beispiele).

Hierarchie der Grammatiken nach Noam Chomsky:



Typ-3 Grammatiken bilden eine echte Teilmenge der Typ-2 Grammatiken usw.