Automatentheorie

kontextfreie Grammatiken

Prof. Dr. Franz-Karl Schmatzer
schmatzf@dhbw-loerrach.de

Literatur

- C.Wagenknecht, M.Hielscher; Formale Sprachen, abstrakte Automaten und Compiler; 3.Aufl. Springer Vieweg 2022;
- Sipser M.; Introduction to the Theory of Computation; 2.Aufl.; Thomson Course Technology 2006
- Hopecroft, T. et al; Introduction to Automata Theory, Language, and Computation; 3. Aufl. Pearson Verlag 2006
- Vossen,G. Witt K.; Grundkurs Theoretische Informatik; 4.Aufl.; Vieweg Verlag 2006
- Cohen, D; Introduction to Computer Theory; John Wiley 1990

Agenda

- kontextfreie Grammatiken
  - Ableitungsbäume
  - Eindeutig und mehrdeutige Grammatiken
- Normalformen kontextfreier Grammatiken
- Erweitere Backus-Naur-Form
- Eigenschaften kontextfreier Grammatiken
- CYK-Algorithmus

Kontextfreie Sprache

Einführung

- Die Sprache L(P) mit

  $$L(P) = \{a^n b^n \mid n \geq 0\}$$

  gehört nicht zu einer Typ-3-Grammatik

- Bei einer Typ-3-Sprache kann höchstens 1 Terminalsymbol pro Produktionsregel erzeugt werden.
- Aufheben dieser Beschränkung:
  - $S \rightarrow aSb$
  - (Anwenden der Regel: $S \Rightarrow aSb \Rightarrow aaSbb \Rightarrow aaaSbbb \Rightarrow \ldots$)
  - d.h. $S \Rightarrow^* a^n S b^n$

  Wir brauchen noch eine Terminierung $S \rightarrow \varepsilon$ und haben L(P)
  - $G = (\{S\}, \{a,b\}, \{S \rightarrow aSb, S \rightarrow \varepsilon\}, S)$ erzeugt die Sprache L(P)

Kontextfreie Sprache

Definition

- Eine Grammatik $G = (N,\Sigma,P,S)$ heißt Typ-2-Grammatik, wenn
  - N eine Menge von Nicht-Terminalsymbolen, die zu Σ disjunkt ist.
  - Σ eine Menge von Terminalsymbolen
  - P eine Relation $P \subseteq N \times (\Sigma \cup N)^*$ und $|P| < \infty$
  - $S \in N$, dem Startsymbol
  - Eine Element $p \in P$ mit $p = (l, r)$ heißt Produktion oder Regel mit

    $l \in N$ und $r \in (\Sigma \cup N)^*$ und der

    Notation: $l \rightarrow r$ (l geht über in r oder l wird durch r ersetzt)
- Eine Sprache L heißt kontextfrei über Σ, falls es eine kontextfreie Grammatik G über Σ gibt mit $L = L(G)$

Kontextfreie Sprache

Beispiel 2 Palindrome

- $L_P = \{ww^r \mid w \in \Sigma^*\}$ (Palindrome)
  - L gehört nicht zu den regulären Sprachen, sondern zu den kontextfreien Sprachen.
  - Die Grammatik dazu:
    - $G_P = (\{S\},\{0,1\},\{S \rightarrow 0S0 \mid 1S0 \mid \varepsilon\},S)$
  - Erstellen Sie die Grammatik mit Flaci und produzieren sie alle Worte der Länge $l < 5$

Aufgaben
kontextfreie Grammatiken

- Konstruieren Sie eine kontextfreie Grammatik für Sprachen L mit
  - $L = \{ 0^n 1^m 2^n \mid n, m \geq 0 \}$
  - $L = \{ 0^n 1^m \mid n, m \geq 0 , n < m \}$
  - $L = \{ w \in \Sigma^* \mid$ Anzahl der 0-Ziffern = Anzahl der 1-Ziffern in w $\}$
  - Die Sprache L der ausgewogenen Klammerausdrücke. D.h. jede öffnende Klammer muss auch eine schließende Klammer haben.
- Nutzen Sie FLACI und prüfen Sie ihre Implementierung auf Korrektheit

Kontextfreie Sprache

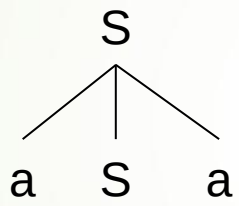
Ableitungsbaum

- $L_P = \{ww^r \mid w \in \Sigma^*\}$ (Palindrome)
  - L gehört nicht zu den regulären Sprachen, sondern zu den kontextfreien Sprachen.
  - Die Grammatik dazu: $G_P = (\{S\},\{a,b\},\{S \rightarrow aSa \mid bSb \mid \varepsilon\},S)$
  - Ableitung abbbba
  - $S \Rightarrow aSa \Rightarrow abSba \Rightarrow abbSbba \Rightarrow abbbba$
  - Dazu kann man auch einen Ableitungsbaum erstellen

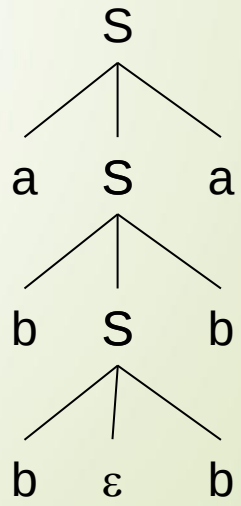
Kontextfreie Sprache

Beispiel Ableitungsbaum

- Dazu kann man einen Ableitungsbaum erstellen
  - w = abbbba
  - Regel: S → aSa ergibt zum Beispiel:



  - und damit folgenden Ableitungsbaum
  - Erstellen Sie Ableitungsbäume mit FLACI



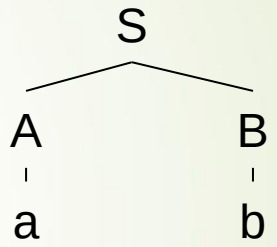
Kontextfreie Sprache

Grammatik: eindeutig

- Die Worte einer Grammatik G sind nicht immer in eindeutiger Weise anhand der Regeln ableitbar.

  Beispiel: $S \rightarrow AB$, $A \rightarrow a$, $B \rightarrow b$

  - Das Wort w=ab lässt sich ableiten als
  - $S \Rightarrow AB \Rightarrow aB \Rightarrow ab$ (1) oder
  - $S \Rightarrow AB \Rightarrow Ab \Rightarrow ab$ (2)



- Der Ableitungsbaum für beide Ableitungen (1) oder (2) ist jedoch identisch.
- Die Grammatik ist daher eindeutig.

Kontextfreie Sprache

Grammatik: mehrdeutig

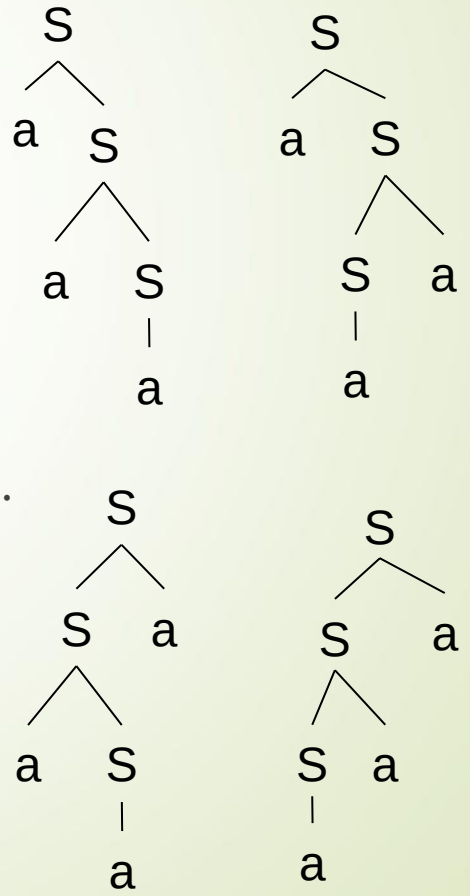
- Die Worte einer Grammatik G sind nicht immer in eindeutiger Weise anhand der Regeln ableitbar und können auch verschiedene Ableitungsbäume ergeben.

Beispiel: $S \rightarrow aS \mid Sa \mid a$

- Das Wort w=aaa lässt sich ableiten als

1. $S \Rightarrow aS \Rightarrow aaS \Rightarrow aaa$ (1) oder
2. $S \Rightarrow aS \Rightarrow aSa \Rightarrow aaa$ (2) oder
3. $S \Rightarrow Sa \Rightarrow Saa \Rightarrow aaa$ (3) oder
4. $S \Rightarrow Sa \Rightarrow aSa \Rightarrow aaa$ (4)

- Der Ableitungsbäume sind alle verschieden.
- Die Grammatik ist somit mehrdeutig.



Aufgabe
Ableitungsbäume

Erstellen Sie einen Ableitungsbaum für Worte der Länge 4 zu folgenden Grammatiken G = ({S, A, B}, {0,1}, P, S). Welche Grammatiken sind mehrdeutig?

1. $P=\{S \rightarrow 0S \mid 1S \mid 0\}$
2. $P=\{S \rightarrow 0S0S \mid 1\}$
3. $P=\{S \rightarrow 0S1 \mid 1A,\ A \rightarrow 1A \mid 1\}$
4. $P=\{S \rightarrow 1A \mid 0B,\ A \rightarrow 1AA \mid 0S \mid 0,\ B \rightarrow 0BB \mid 1S \mid 1\}$

Kontextfreie Sprache

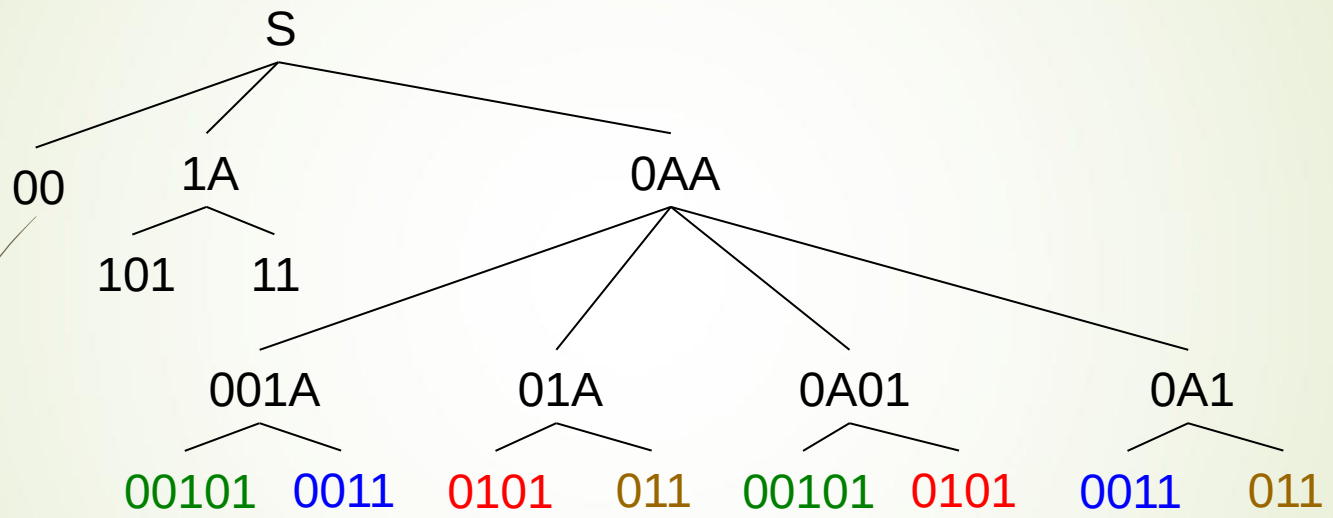
Grammatik: mehrdeutig

- Die Grammatik $G_m = (\{S\}, \{0\}, \{ S \rightarrow aS \mid Sa \mid a\}, S)$ ist mehrdeutig, aber sie lässt sich leicht durch eine eindeutige Grammatik ersetzen.
- $G = (\{S\}, \{0\}, \{ S \rightarrow aS \mid a\}, S)$
- Eine mehrdeutigen Grammatik impliziert noch lange nicht, dass die Sprache mehrdeutig ist.
- Ziel
  - für eine eindeutige Sprache L auch eine eindeutige Grammatik G mit L(G) = L zu erhalten.

Kontextfreie Sprache

Grammatik: eindeutig/mehrdeutig

- Beispiel: G=({S,A},{0,1},{S → 00 | 1A | 0AA, A → 01 | 1},S)



- Die Sprache L(G) hat nur 7 Worte: L(G) = {00, 101, 11, 00101, 0011, 0101,011)
- Die Worte: 00101, 0011, 0101 und 011 können auf zwei verschiedene Arten abgeleitet werden, aber es gibt nur einen Ableitungsbaum.

⇒ die Grammatik G ist daher eindeutig.

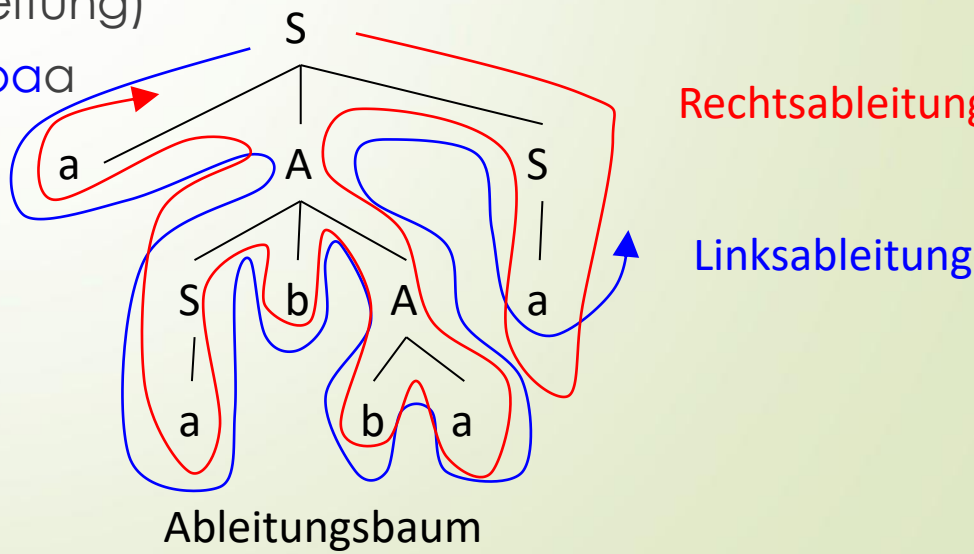
Kontextfreie Sprache

Grammatik: eindeutig, aber Ersetzungen mehrdeutig

- Mehrdeutigkeit in der Ersetzung ist möglich
- daher erzielt man Eindeutigkeit dadurch, dass:
  - Nichtterminale, die ganz links stehen, werden zuerst ersetzt oder
  - Nichtterminale, die ganz rechts stehen, werden zuerst ersetzt.
- Beispiel: $G_2 = (\{S,A\}, \{a,b\}, \{S \rightarrow aAS \mid a, A \rightarrow SbA \mid SS \mid ba\}, S)$
- Ableiten des Wortes: aabbaa

$S \Rightarrow aAS \Rightarrow aSbAS \Rightarrow aabAS$
$\Rightarrow aabbaS \Rightarrow aabbaa$ (Linksableitung)

$S \Rightarrow aAS \Rightarrow aAa \Rightarrow aSbAa \Rightarrow aSbbaa$
$\Rightarrow aabbaa$ (Rechtsableitung)



Ableitungsbaum

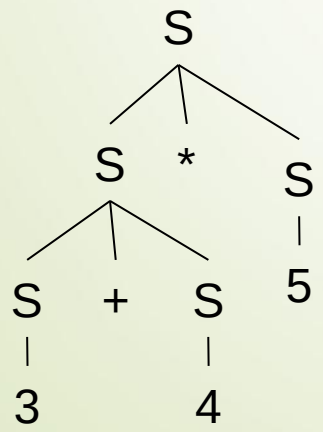
Aufgabe Mehrdeutigkeit

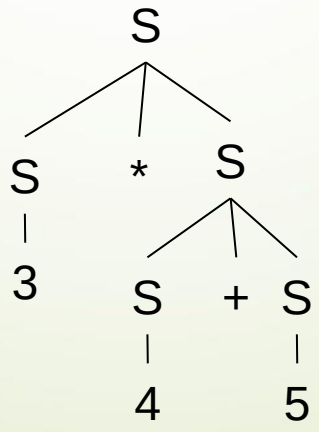
- Zeigen Sie, dass die Grammatik G mehrdeutig ist und die dazugehörige äquivalente Grammatik G_e eindeutig.
  - $G = (\{E\},\{a,+,*,(,)\},\{E \rightarrow E+E \mid E*E \mid (E) \mid a\}, E)$
  - $G_e = (\{E,T,F\},\{a,+,*,(,)\},\{E \rightarrow E+T \mid T, T \rightarrow T*F \mid F, F \rightarrow (E) \mid a\}, E)$
- Beispiel einer inhärent mehrdeutige Sprache
  - $L = \{a^i b^j c^k \mid i=j \text{ oder } j=k \text{ mit } i,j,k > 0\}$

Kontextfreie Sprache

Ableitungsbäume

- Neben der Generierung von Worte einer Sprache, können auch Ableitungsbäume zur Berechnung von Ausdrücken verwandt werden
- Beispiel:
  - G = ({S}, {a∈N}, {S → S+S | S*S | a },S)
  - Mögliche Ausdrücke 3+4*5, 1+7+8
  - Was heißt 3+4*5 ?
    - 3+(4* 5) oder (3+4)*5
    - Wie bekomme ich die Mehrdeutigkeit weg?

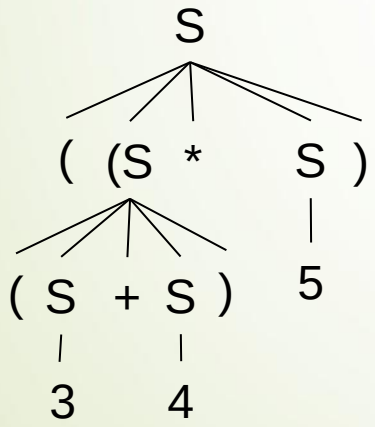


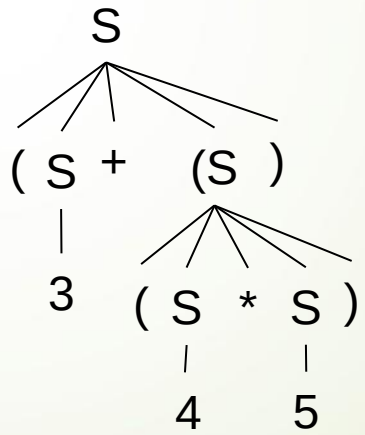


Kontextfreie Sprache

Ableitungsbäume

- Durch Klammerung
- G = ({S}, {(,),a∈N}, {S → (S+S) | (S*S) | a },S)
- Was heißt ((3+4)*5) oder (3+(4*5)) ?
- Es gibt jedoch noch einen anderen Weg



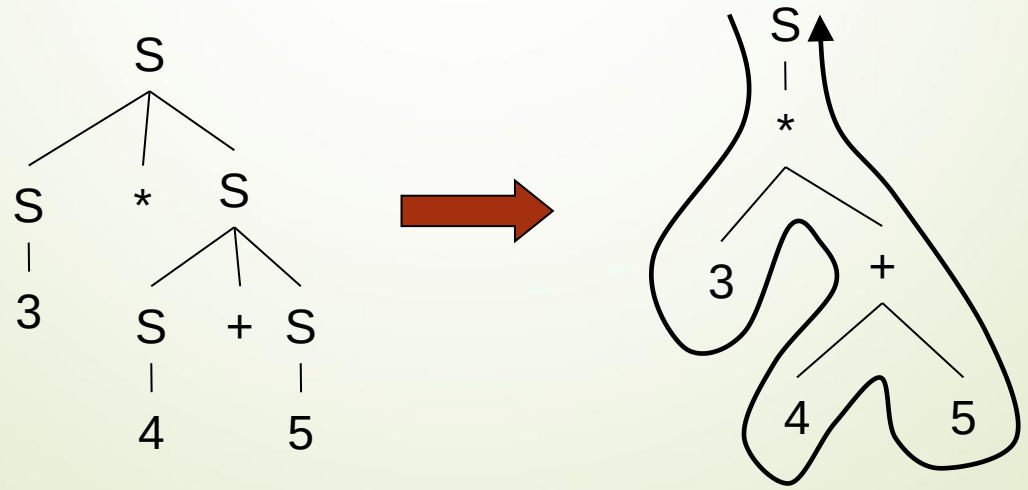


Kontextfreie Sprache

Berechnungsbäume

- Die Ableitungsbäume erlauben arithmetische Ausdrücke zu berechnen
- Wenn man den Baum durchwandert erhält man *3+4 5
  - Dies nennt man Prefix-Notation (oder Polnische Notation)

    (Operation steht vor den beiden Operanden)
  - erlaubt die Klammerfreie Berechnung von Ausdrücken



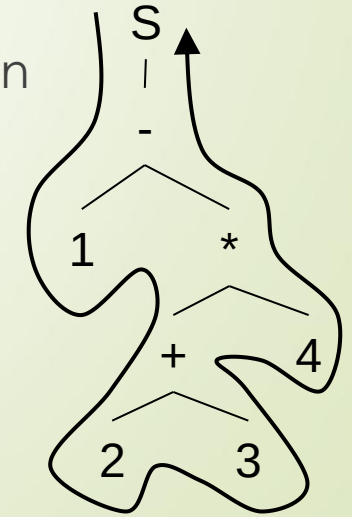
Kontextfreie Sprache

Berechnungsbäume

- Durchwandern von Bäumen
  - **Prefix Notation** (Operation wird zuerst ausgegeben, dann der linke Ast rekursiv durchwandert und dann der rechte Ast rekursiv durchwandert)

    Nennt man auch Polnische Notation
  - **Infix Notation** (zuerst der linke Ast durchwandert dann die Operation ausgegeben und dann der rechte Ast durchwandert)
  - **Postfix Notation** (Der linke Ast wird rekursiv durchwandert und dann der rechte Ast rekursiv durchwandert dann wird die Operation ausgegeben)

    Nennt man auch Umgekehrte Polnische Notation



Beispiel: (1-(2+3)*4)

- Prefix: - 1 * + 2 3 4
- Infix: 1 - 2+3*4
- Postfix: 1 2 3 + 4 * -

Kontextfreie Sprache

Beispiel: Berechnungsbäume

- Beispiel: ((1+2)*(3+4)+5)*6)
- Polnische Notation: * + * +1 2 +3 4 5 6
- Berechnung:

  + * +1 2 +3 4 5 6 ⇒ * + * 3 +3 4 5 6

  + * 3 +3 4 5 6 ⇒ * + * 3 7 5 6

  * + * 3 7 5 6 ⇒ * + 21 5 6

  + 21 5 6 ⇒ * 26 6

  * 26 6 ⇒ 156

Aufgabe Polnische Notation

- Wandeln Sie folgende Infix-Notation in Polnische Notation um.

1. 1*2*3
2. 1*2+3
3. 1*(2+3)
4. ((1+2)*3)+4)
5. 1+2*3+4

Kontextfreie Sprache

Beispiel arithmetische Ausdrücke

- Erzeugen von arithmetischen Ausdrücken

  Als Stellvertreter für Variablen und Ausdrücken wählen wir a

  Ausdrücke wäre: a, a+a, a*a, a+(a+a),a*((a-a)/a)-((a+a)*a)).

- Was ist das Terminalalphabet?
  - a als Bezeichner für Konstanten und Variablen
  - () Klammersymbolen
  - +,-,*,/ als Operatoren
  - Hilfssymbole E für Ausdrücke und O für Operationen
- D.h wir haben
  - $E \rightarrow a$ und $O \rightarrow + \mid - \mid * \mid /$
  - $E \rightarrow E\ O\ E$ zum Verknüpfen von Ausdrücken
  - $E \rightarrow (E)$ zum Einklammern
- $G_A = (\{E,O\}, \{a,(,),+,-,*,/\}, P,E)$

  $P = \{E \rightarrow a \mid E\ O\ E \mid (E) \mid , O \rightarrow + \mid - \mid * \mid / \}$

Kontextfreie Sprache

Beispiel 2 Ableitung von Wörter

$G_A = (\{E,O\}, \{a,(,),+,-,*,/\}, \{E \rightarrow a \mid E\ O\ E \mid (E) \mid , O \rightarrow + \mid - \mid * \mid / \},E)$

- Ableiten des Ausdrucks: a*(((a-a)/a)-((a+a)*a))

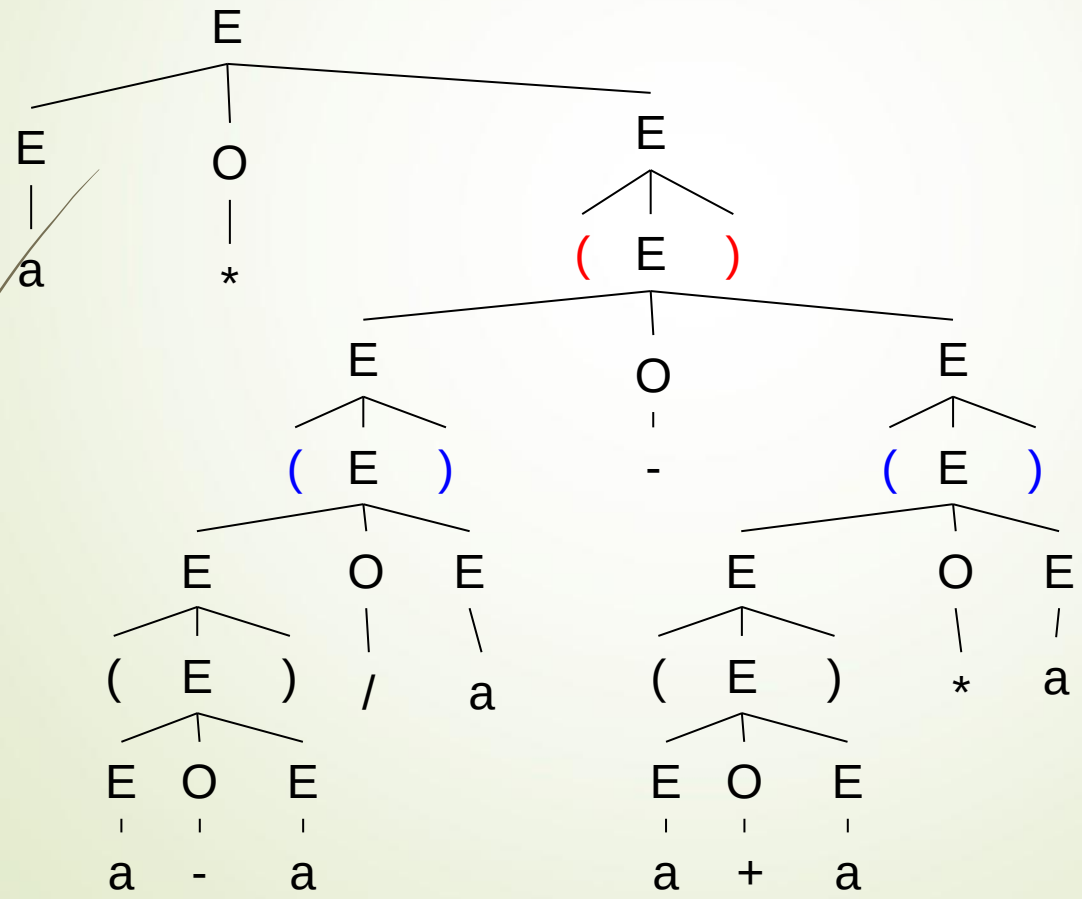
$E \Rightarrow EOE$
$\Rightarrow E*E$
$\Rightarrow E*(E)$
$\Rightarrow E*(EOE)$
$\Rightarrow E*(E-E)$
$\Rightarrow E*(E-(E))$
$\Rightarrow E*(E-(EOE))$
$\Rightarrow E*(E-(E*E))$
$\Rightarrow E*(E-((E)*E))$
$\Rightarrow E*(E-((EOE)*E))$
$\Rightarrow E*(E-((E+E)*E))$

$E \Rightarrow E*((E)-((E+E)*E))$
$\Rightarrow E*((EOE)-((E+E)*E))$
$\Rightarrow E*((E/E)-((E+E)*E))$
$\Rightarrow E*(((E)/E)-((E+E)*E))$
$\Rightarrow E*(((EOE)/E)-((E+E)*E))$
$\Rightarrow E*(((E-E)/E)-((E+E)*E))$
$\Rightarrow^* a*(((a-a)/a)-((a+a)*a))$

Kontextfreie Sprache

Beispiel2 Ableitungsbaum

- Ableitungsbaum für: a*(((a-a)/a)-((a+a)*a))



Kontextfreie Grammatiken

Normalformen

- Jede kontextfreie Grammatik lässt sich vereinfachen und in eine Normalform überführen.
  - Elimination der ε-Regeln
  - Elimination von Kettenregeln
  - Elimination von nutzlosen Variablen
- Gängige Normalformen sind:
  - Chomsky-Normalform

    (Produktionen P mit $A, B, C \in N$ und $a \in \Sigma$ nur in der Form:

    $A \rightarrow BC$ oder $A \rightarrow a$
  - Greibach-Normalform

    (Produktionen P mit $a \in \Sigma$ und $X \in N^*$ nur in der Form: $A \rightarrow aX$)

Kontextfreie Grammatiken

Elimination der ε-Regeln

- In jeder kontextfreie Grammatik G lassen sich alle ε-Regeln mit Ausnahme der Regel $S \rightarrow \varepsilon$ eliminieren.
  - Die Grammatiken sind dabei äquivalent.
- Verfahren:
  1. Sei $X \rightarrow vAw$ mit $A \rightarrow \varepsilon$ (A eine ε-Variable), dann füge $X \rightarrow vw$ als Regel hinzu.
  2. Mache das iterativ bis keine neue ε-Variable mehr hinzu kommt.
  3. Lösche alle ε-Regeln bis auf $S \rightarrow \varepsilon$

Kontextfreie Grammatiken

Elimination der ε-Regeln

- Beispiel: G = ({S,A,B,C}, {0,1,2}, {S → 0A | 1B, A → BC, B →B2 | ε, C → ε },S}
- 1.Schritt (neue Regeln für die ε-Variablen B und C)
  - S → 0A | 1B | 1
  - A → BC | B | C | ε
  - B → B2 | 2 | ε
  - C → ε
- Nun ist A eine ε-Variablen (Iteration über A)
  - S → 0A | 0 | 1B | 1
  - A → BC | B | C | ε
  - B → B2 | 2 | ε
  - C → ε
- Löschen aller ε-Regeln
  - S → 0A | 0 | 1B | 1
  - A → BC | B | C
  - B → B2 | 2

Kontextfreie Grammatiken

Elimination von Kettenregeln

- In jeder kontextfreie Grammatik G lassen sich alle Kettenregeln eliminieren.
- Eine Kettenregel ist eine Regeln von der Form $A \rightarrow B$ mit $B \in N$
- Def: $[A]^* = \{B \in N \mid \text{mit } A \rightarrow^* B\}$
- Verfahren:
  1. Zunächst Zyklen eliminieren. Ein Zyklus ist eine Menge $A_1, \ldots, A_k$ von Variablen mit: $A_1 \rightarrow A_2$, $A_2 \rightarrow A_3$, … $A_{k-1} \rightarrow A_k$, $A_k \rightarrow A_1$. Der Zyklus wird entfernt, indem die zyklische Regel gelöscht wird und alle Variablen $A_1, \ldots, A_k$ durch eine Variable B ersetzt werden.
  2. Für alle $A \in N$
     1. bestimme $[A]^*$. Lösche anschließend alle Kettenregeln
     2. Für jedes $B \in [A]^*$ und jede Regel $B \rightarrow w$: Füge die Regel $A \rightarrow w$ hinzu.

Kontextfreie Grammatiken

Elimination von Kettenregeln

- Beispiel: G = ({S,A,B,C}, {a,b,c,d}, P, S} mit P:
  - $S \rightarrow A \mid bB$
  - $A \rightarrow B \mid C$
  - $B \rightarrow A \mid Cc$
  - $C \rightarrow c \mid d$
- 1.Schritt Zyklus $A \rightarrow B$, $B \rightarrow A$ entfernen. A,B werden durch X ersetzt.
  - $S \rightarrow X \mid bX$
  - $X \rightarrow C \mid Cc$
  - $C \rightarrow c \mid d$
- 2. Schritt [S]* , [X]* und [C]* bestimmen
  - $[S]^* = \{X,C\}$, $[X]^* = \{C\}$, $[C]^* = \varnothing$
  - $X \rightarrow C$ und $S \rightarrow X$ löschen.
- 3. Schritt Regeln hinzufügen ($S \rightarrow c \mid d \mid Cc$ und $X \rightarrow c \mid d$)
  - $S \rightarrow c \mid d \mid Cc \mid bX$
  - $X \rightarrow c \mid d \mid Cc$
  - $C \rightarrow c \mid d$

Kontextfreie Grammatiken

Elimination nutzloser Variablen

- Eine Variable heißt **nützlich**, wenn sie in der Ableitung eines Terminalwortes vorkommt, d.h. $S \Rightarrow^* uAv \Rightarrow^* w$ mit $w \in \Sigma^*$ und $u,v \in (N \cup \Sigma)^*$
- A ist nutzlos,
  - wenn sich daraus kein Terminalwort ableiten lässt oder
  - wenn A von Startsymbol aus nicht erreichbar ist.
- Beispiel: Grammatik G = ({S,A,B,C}, {0,1,2}, P,S} mit P:

  $S \rightarrow 0A \mid 0 \mid 1B \mid 1$

  $A \rightarrow BC \mid B \mid C$

  $B \rightarrow B2 \mid 2$

  - die Variable C ist nutzlos. D.h. eliminieren alle Produktionen, wo C vorkommt.

    $S \rightarrow 0A \mid 0 \mid 1B \mid 1$

    $A \rightarrow B$

    $B \rightarrow B2 \mid 2$

Aufgabe Vereinfachung

- Falls möglich, vereinfachen Sie folgende Grammatiken G
- G = ({S,A,B,C,D},{0,1},

  {S→00B | 1A, A→B | C, B→1B | 0 | AD, C→BD | AD, D→ε}, S)

Chomsky Normalform

- Definition: Chomsky Normalform

  Eine kontextfreie Grammatik mit $\varepsilon \notin L(G)$ heißt in Chomsky-Normalform, wenn alle ihre Regeln von einer der beiden Formen sind:

  $A \rightarrow BC$ oder

  $A \rightarrow a$

  mit $A, B, C \in N$ und $a \in \Sigma$

- **Jede kontextfreie Grammatik G mit $\varepsilon \notin L(G)$ lässt sich in Chomsky-Normalform transformieren.**

Chomsky Normalform

Transformation 1: Beispiel

Beispiel: Grammatik G = ({S,A,B,C}, {0,1,2,3}, P,S} mit P

$S \rightarrow 0A1 \mid 0C$

$A \rightarrow B0B \mid 0 \mid C$

$B \rightarrow BA \mid 1$

$C \rightarrow C2 \mid 3C$

- **Elimination von C (nutzlos)**

  $S \rightarrow 0A1$

  $A \rightarrow B0B \mid 0$

  $B \rightarrow BA \mid 1$

- **Für jedes Terminalsymbol {0,1} eine Regel $X_0 \rightarrow 0$, $X_1 \rightarrow 1$ einführen und entsprechend in der Regelmenge ersetzten.**

  $S \rightarrow X_0AX_1$

  $A \rightarrow BX_0B \mid 0$

  $B \rightarrow BA \mid 1$

  $X_0 \rightarrow 0$

  $X_1 \rightarrow 1$

Chomsky Normalform

Transformation 2: Beispiel

$S \rightarrow X_0AX_1$
$A \rightarrow BX_0B \mid 0$
$B \rightarrow BA \mid 1$
$X_0 \rightarrow 0$
$X_1 \rightarrow 1$

- Schließlich mehrfach Produktionen: $A \rightarrow B_1B_2 \ldots B_k$ ersetzen durch $A \rightarrow B_1W_1$, $W_1 \rightarrow B_2\ W_2 \ldots\ W_{k-2} \rightarrow B_{k-1}\ B_k$

  $S \rightarrow X_0W_1$

  $W_1 \rightarrow AX_1$

  $A \rightarrow B\ W_2 \mid 0$

  $W_2 \rightarrow X_0B$

  $B \rightarrow BA \mid 1$

  $X_0 \rightarrow 0$

  $X_1 \rightarrow 1$

Aufgabe Chomsky-Normalform

- Überführen Sie folgende Grammatiken G = ({S, A, B}, {0,1}, P, S) in Chomsky-Normalform.

1. $P = \{S \rightarrow 0S0 \mid SS0 \mid 0\}$
2. $P = \{S \rightarrow S0S \mid S0S1S \mid S1S0S \mid \varepsilon\}$
3. $P = \{S \rightarrow ABABAB, A \rightarrow 0 \mid \varepsilon, B \rightarrow 1\}$
4. $P = \{S \rightarrow 0A \mid B1, A \rightarrow S, B \rightarrow 1B \mid 1\}$

Greibach-Normalform

- Neben der Chomsky-Normlform gibt es auch Greibach-Normalform. Die Regeln sind alle vom Typ:
  - L -> aN* (N: Nichterminale, a die Terminale der Grammatik)
  - In jeden Schritt lässt sich ein Zeichen ableiten. Um ein Wort der Länge n zu erkennen braucht man höchstens n-Schritte.
- Die Umwandlung einer kontextfreie Grammatik in die Greibach-Normalform ist recht aufwändig. Daher belassen wie es mit zwei Beispielen für Grammatiken in der Greibach-Normalform
- Beispiele
  - $G_1 = (\{S,B\},\{a,b\}, \{S \rightarrow a\ B \mid a\ S\ B, B \rightarrow b\}, S)$
  - $G_2 = (\{S,A,B\},\{a,b\},\{S \rightarrow aA \mid bB \mid aSA \mid bSB, A \rightarrow a, B \rightarrow b\}, S)$

  Welche Sprachen erzeugen diese Grammatiken?

Darstellung kontextfreier Grammatiken

- Die erweiterterte Backus-Naur-Form (EBNF) ist eine effiziente Darstellungsform für kontextfreie Grammatiken.
- Verwendet für die Darstellung von Programmier- und Dialogsprachen.
- Entwickelt durch Backus und Naur.
  - Diese waren in den 50 und 60 Jahren wesentlich an der Entwicklung von Fortan und Algol beteiligt.

Erweiterterte Backus-Naur-Form (EBNF)
Definition

- Seine Σ und N zwei Alphabete mit Σ terminales und N nicht terminales Alphabet. Dann gilt:
  - Jedes $a \in \Sigma$ und jedes $A \in N$ sowie ε sind Elemente von EBNF
  - Sind $a_1, a_2, \ldots a_k, k \geq 1 \in$ EBNF dann auch $(a_1 a_2 \ldots a_k)$ sowie $(a_1 \mid a_2 \mid .. \mid a_k)$
  - Sind a, b und c $\in$ EBNF, dann auch a{b}*c oder alternativ a{b}c $\in$ EBNF
  - Sind a, b und c $\in$ EBNF, dann auch $a\{b\}^1_0 c$ oder alternativ a[b]c $\in$ EBNF
- Sei $A \in N$ und x eine EBNF, dann heißt A::= x eine erweiterte Backus-Naur-Regel
- Eine Grammatik G ={ N, Σ, P, S} , wobei $S \in N$ das Startsymbol und P eine endliche Menge von erweiterten Backus-Naur-Regeln ist, heißt erweiterte Backus-Naur Grammatik.

EBNF

Definition der Sprache

- Festlegen der Ableitungsregeln ($A \in N$, $a,b,c \in \Sigma$)
  - Bisher wurde ein nicht-terminales Symbol A durch ein Wort w über $N \cup \Sigma$ ersetzt. (Expansionsschritt)

    $aAc \Rightarrow abc$, falls eine Regel $A \rightarrow b$ existierte mit $A \in N$ und $a, b, c \in \Sigma$
- Außer diesen Regeln gibt es noch (Reduktionsschritt):
  - Ist ein Wort $w = a(b_1 \mid b_2 \mid .. \mid b_k)c$ abgeleitet, dann gilt $a(b_1 \mid b_2 \mid .. \mid b_k) \Rightarrow ab_ic$ für ein $1 \leq i \leq k$
  - Ist ein Wort $w = a\{b\}c$ abgeleitet, dann gilt $a\{b\}c \Rightarrow ab^ic$ für ein $i \geq 0$
  - Ist ein Wort $w = a[b]c$ abgeleitet, dann gilt $a[b]c \Rightarrow ac$ oder $a[b]c \Rightarrow abc$
- Die Symbole | und * werden wie bei den regulären Ausdrücken interpretiert und [a] entspricht dem regulären Ausdruck ($\varepsilon \mid a$)
- Damit lässt sich die Sprache der erweiterten Backus-Naur-Grammatik definieren: $L(G) = \{ w \in \Sigma^* \mid S \Rightarrow^* w\}$

EBNF

Beispiel: Ableitung

- Gegeben $G_I = (\{int, v_z, z_f, z_1, z_0\}, \{+,-,0,1,2,3,4,5,6,7,8,9\}, P, int)$

$P = \{$ $int ::= v_z\ z_f,$

$v_z ::= [+ \mid -],$

$z_f ::= (0 \mid z_1\{z_0\}),$

$z_1 ::= (1 \mid 2 \mid 3 \mid 4 \mid 5 \mid 6 \mid 7 \mid 8 \mid 9),$

$z_0 ::= (0 \mid z_1)\}$

Ableiten der Zahl: 7305:

$int \Rightarrow v_z\ z_f \Rightarrow [+ \mid -]\ z_f \Rightarrow z_f$

$\Rightarrow (0 \mid z_1\{z_0\} \Rightarrow z_1\{z_0\} \Rightarrow z_1 z_0 z_0 z_0 z_0$

$\Rightarrow (1 \mid 2 \mid 3 \mid 4 \mid 5 \mid 6 \mid 7 \mid 8 \mid 9)\ z_0 z_0 z_0 z_0 \Rightarrow 7 z_0 z_0 z_0 z_0$

$\Rightarrow 7(0 \mid z_1)\ z_0 z_0 \Rightarrow 7 z_1 z_0 z_0 \Rightarrow 7(1 \mid 2 \mid 3 \mid 4 \mid 5 \mid 6 \mid 7 \mid 8 \mid 9)\ z_0 z_0 \Rightarrow 73 z_0 z_0$

$\Rightarrow 73(0 \mid z_1) z_0 \Rightarrow 730 z_0$

$\Rightarrow 730(0 \mid z_1) \Rightarrow 730 z_1 \Rightarrow 730(1 \mid 2 \mid 3 \mid 4 \mid 5 \mid 6 \mid 7 \mid 8 \mid 9) \Rightarrow 7305$

EBNF

EBNG ist äquivalent zu den kontextfreien Grammatiken

- Die Sprache einer erweiterten Backus-Naur-Grammatik L_{BN} ist äquivalent der Sprache einer kontextfreien Grammatik L_{kf}

  Beweis durch Konstruktion:

  1) Jedes $A \rightarrow a \Rightarrow A ::= a$

  d.h jede kontextfreie Sprache lässt sich durch eine EBNG beschreiben

  2) Jede Backus-Naur-Regel lässt sich in eine kontextfreie Regel transformieren.

$A ::= (a_1 a_2 \ldots a_k)$	$\Rightarrow A \rightarrow a_1 a_2 \ldots a_k$
$A ::= (a_1 \mid a_2 \mid .. \mid a_k)$	$\Rightarrow A \rightarrow a_1, A \rightarrow a_2, .., A \rightarrow a_k$
$A ::= a\{b\}c$	$\Rightarrow A \rightarrow aBc, B \rightarrow bB, B \rightarrow \varepsilon$
	mit B ein neues nicht-terminales Symbol
$A ::= a[b]c$	$\Rightarrow A \rightarrow ac, A \rightarrow abc$

  Mit diesem Verfahren alle Regeln transformieren.

EBNF

Beispiel: EBNG ist äquivalent zu den kontextfreien Grammatiken

- Gegeben $G_I = (\{int, v_z, z_f, z_1, z_0\}, \{+,-,0,1,2,3,4,5,6,7,8,9\}, P, int)$

$P = \{$ $int ::= v_z\ z_f,$

$v_z ::= [+ \mid -],$

$z_f ::= (0 \mid z_1\{z_0\}),$

$z_1 ::= (1 \mid 2 \mid 3 \mid 4 \mid 5 \mid 6 \mid 7 \mid 8 \mid 9),$

$z_0 ::= (0 \mid z_1)\}$

$int ::= v_z\ z_f \quad \Rightarrow int \rightarrow v_z\ z_f$

$v_z ::= [+ \mid -] \quad \Rightarrow v_z \rightarrow + \mid - \mid \varepsilon$

$z_f ::= (0 \mid z_1\{z_0\}) \Rightarrow z_f \rightarrow 0$ und $z_f \rightarrow z_1\{z_0\}$

$z_f \rightarrow z_1\{z_0\}$ wird transformiert in $z_f \rightarrow z_1 B,\ B \rightarrow z_0\ ,B \rightarrow \varepsilon$

$z_1 ::= (1 \mid 2 \mid 3 \mid 4 \mid 5 \mid 6 \mid 7 \mid 8 \mid 9) \Rightarrow z_1 \rightarrow 1 \mid 2 \mid 3 \mid 4 \mid 5 \mid 6 \mid 7 \mid 8 \mid 9$

$z_0 ::= (0 \mid z_1) \quad \Rightarrow z_0 \rightarrow 0 \mid z_1$

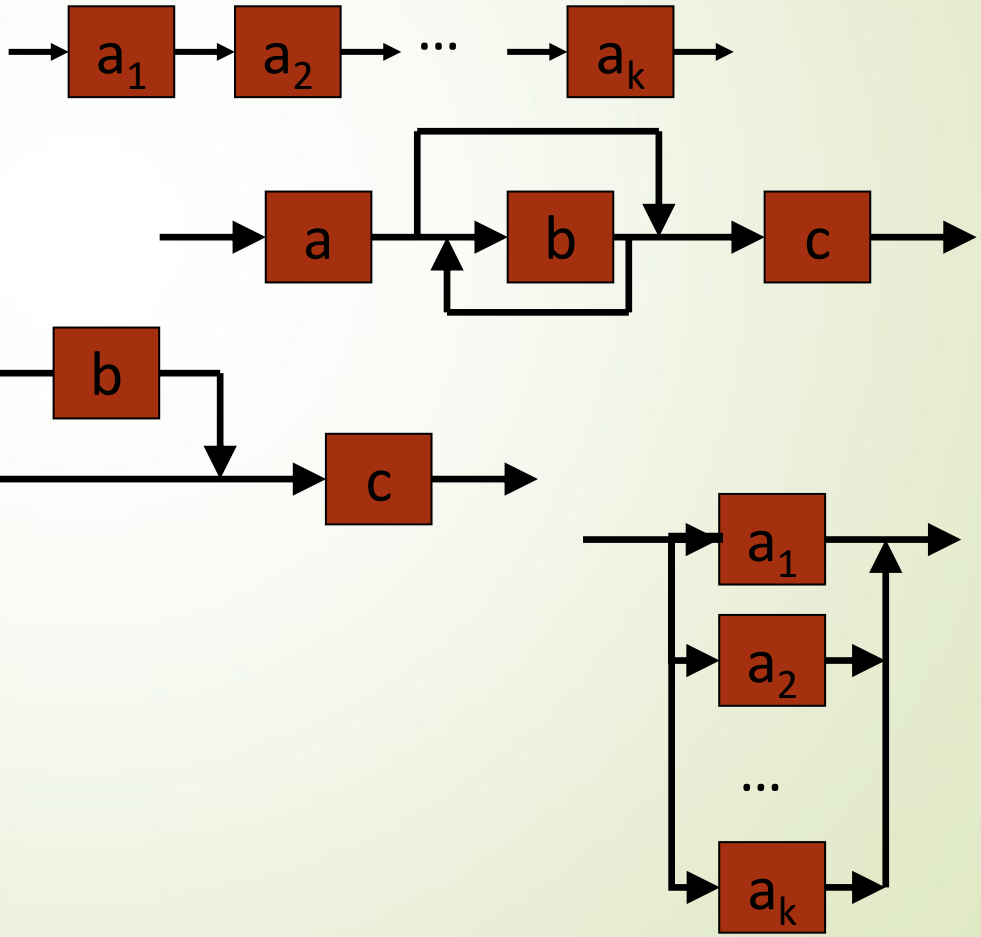
EBNF
vereinfachte Regel Darstellung

- Wenn man sich EBNF genauer ansieht, lassen sich mit der dortigen Notation die Regeln regulärer Grammatiken kompakter schreiben
  - $A \rightarrow a_1, A \rightarrow a_2, ..., A \rightarrow a_k$ kompakter $A \rightarrow a_1 \mid a_2 \mid .. \mid a_k$
  - $A \rightarrow aBc, B \rightarrow bB, B \rightarrow \varepsilon$ kompakter $A \rightarrow a\{b\}c$
  - $A \rightarrow ac, A \rightarrow abc$ kompakter $A \rightarrow a[b]c$
- und graphisch darstellen (Syntaxdiagramme).

EBNF

Syntaxdiagramme

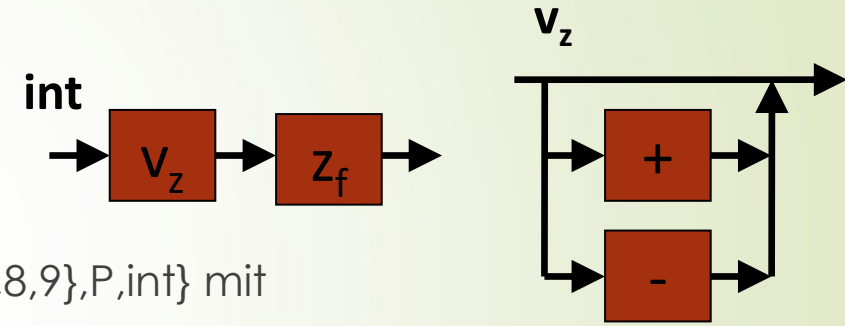
- Grafische Darstellung der Regeln
  - $A \rightarrow a_1 a_2 .. a_k$
  - $A \rightarrow a\{b\}c$
  - $A \rightarrow a[b]c$
  - $A \rightarrow a_1 \mid a_2 \mid .. \mid a_k$

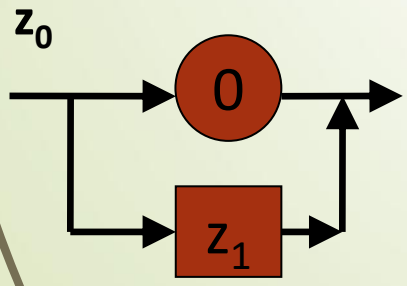


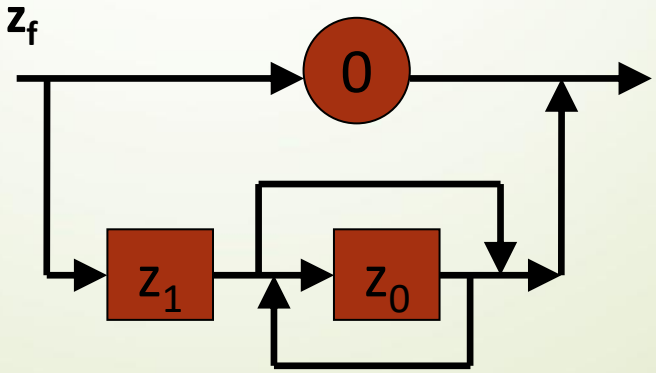
EBNF
Syntaxdiagramme

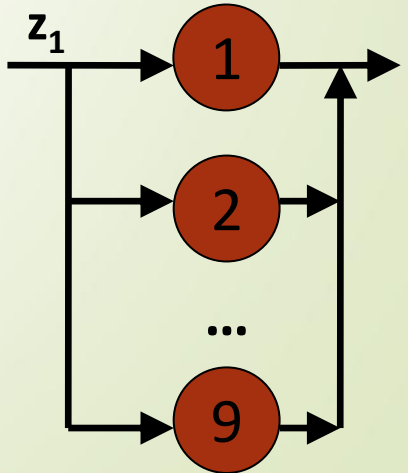
- Darstellung kontextfreien Grammatiken durch Syntaxdiagramme
  - Ein Pfeil markiert das Startsymbol
  - Terminale sind Kreise
  - Nicht-Terminale sind Rechtecke
  - Beispiel $G_I = (\{int, v_z, z_f, z_1, z_0\}, \{+,-,0,1,2,3,4,5,6,7,8,9\}, P, int\}$ mit

$P = \{ int ::= v_z\ z_f,\ v_z ::= [+ | -],\ z_f ::= (0 | z_1\{z_0\}),\ z_1 ::= (1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9),\ z_0 ::= (0 | z_1)\}$









Eigenschaften kontextfreier Sprachen

- Es gibt ein Pumping-Lemma für diese Sprachen
  - Sei L eine kontextfreie Sprache. Dann existiert eine Zahl $p \geq 0$ mit $p \in N$, so dass sich jedes Wort $w \in L$ mit $|w| \geq p$ in der folgenden Form schreiben lässt:

    $$w = uvwxy \text{ mit } |vx| \geq 1 \text{ und } |vwx| \leq p$$

    und $\forall\, i \geq 0$ gilt uv^iwx^iy gehört auch zu L.
- Die kontextfreien Sprachen sind abgeschlossen unter:
  - Vereinigung
  - Konkatenation
  - Kleene-Stern
- Die kontextfreien Sprachen sind nicht abgeschlossen unter:
  - Durchschnittsbildung
  - Komplement

Kontextfreier Sprachen

Abgeschlossenheit

- Seien $L_1 = L(G_1)$ und $L_2 = L(G_2)$ zwei kontextfreie Sprachen, die durch die Grammatiken
  - $G_1 = (N_1, \Sigma, P_1, S_1\}$ und
  - $G_2 = (N_2, \Sigma, P_2, S_2\}$ definiert sind

Kontextfreie Sprachen sind abgeschlossen unter

- **Vereinigung** $L(G) = L_1 \cup L_2$
  - $G = (N_1 \cup N_2 \cup \{S\}, \Sigma, P_1 \cup P_2 \cup \{S \rightarrow S_1 \mid S_2\}, S\}$
  - ist eine kontextfreie Sprache.
- **Konkatenation** $L(G) = L_1 L_2$
  - $G = (N_1 \cup N_2 \cup \{S\}, \Sigma, P_1 \cup P_2 \cup \{S \rightarrow S_1 S_2\}, S\}$
  - ist eine kontextfreie Sprache.
- **Kleene-Stern** $L(G) = L_1^*$
  - $G = (N_1 \cup \{S\}, \Sigma, P_1 \cup \{S \rightarrow SS_1 \mid \varepsilon\}, S\}$
  - ist eine kontextfreie Sprache.

Kontextfreier Sprachen

Nichtabgeschlossenheit

Kontextfreie Sprachen sind nicht abgeschlossen unter

- Durchschnittsbildung

  Gegenbeispiel:

  - $L_1 = \{a^i b^i c^k \mid i,k \geq 1\}$ und $L_2 = \{a^i b^k c^k \mid i,k \geq 1\}$
  - Sowohl L_1 als auch L_2 gehören zu den kontextfreien Sprachen
    - mit $G_1 = (\{S,A,B\},\{a,b,c\},\{S \rightarrow AB,\ A \rightarrow aAb \mid ab,\ B \rightarrow cB \mid c\},S)$
    - und $G_2 = (\{S,C,D\},\{a,b,c\},\{S\text{->}CD,\ C \rightarrow aC \mid a,\ D \rightarrow bDc \mid bc\},S)$
  - Die Schnittmenge $L = L_1 \cap L_2 = \{a^i b^i c^i \mid i \geq 1\}$ ist nicht kontextfrei.
- und Komplement.

  Da $L_1 \cap L_2 = (L_1^c \cup L_2^c)^c \Rightarrow L^c$ kann auch nicht kontextfrei frei.

Entscheidungsprobleme

- Der Zweck eines endlichen Automaten ist die Prüfung, ob ein gegebenes Wort zu seiner Sprache gehört. (Wortproblem)
- Auch andere Entscheidungsprobleme sind hier relevant.
- Für kontextfreie Sprachen können diese fast alle mit ja beantwortet werden.

Problem	**Gegeben**	**Gefragt**	**Entscheidbar**
Wortproblem	L und $w \in \Sigma^*$	Gilt $w \in L$?	Ja
Leerheitsproblem	L	Gilt $L = \emptyset$?	Ja
Endlichkeitsproblem	L	Gilt $\|L\| < \infty$?	Ja
Äquivalenzproblem	L_1 und L_2	$L_1 = L_2$?	nein

Entscheidungsprobleme

- Das Wortproblem ist entscheidbar
  - Z.B. lösen mit Hilfe des CYK-Algorithmus
- Das Leerheitsproblem ist entscheidbar
  - Die Sprache L(G) ist genau dann leer, wenn S eine nutzlose Variable ist. (Siehe dazu Kapitel Vereinfachung der Grammatik von kontextfreien Sprachen)
- Das Endlichkeitsproblem ist entscheidbar
  - Transformation in Chomsky-Normalform
  - Aufstellen eines gerichteten Graphen.
    - Die Knoten sind die Nichtterminalen.
    - Die gerichtete Kantenmenge V wird gebildet anhand der Regeln. Jede Regel $S \rightarrow AB$ ergibt zwei Kanten $S \rightarrow A$ und $S \rightarrow B$.
  - Enthält der gerichtete Graph keine Zyklen, dann ist die Sprache endlich.
- Das Äquivalenzproblem ist nicht entscheidbar

Das Endlichkeitsproblem

Beispiel

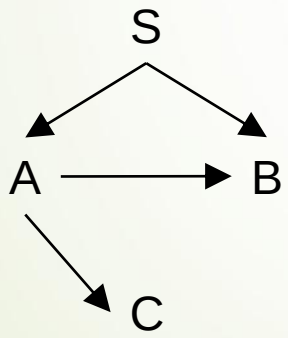
- Gegeben die Grammatiken:

  $G_1 = (\{S,A,B,C\},\{a,b,c\},\{S \to AB,\ A \to BC \mid a,\ B \to b,\ C \to c\},S)$

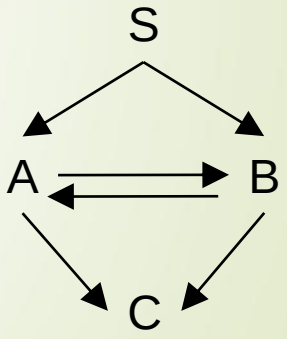
  $G_2 = (\{S,A,B,C\},\{a,b,c\},\{S \to AB,\ A \to BC,\ B \to CA \mid b,\ A \to a,\ C \to c\},S)$

- Die zugehörigen Graphen sind:

Graph zu G_1



Graph zu G_2



- G_1 nicht zyklisch $\Rightarrow$ $L(G_1)$ endlich
- G_2 zyklisch $\Rightarrow$ $L(G_2)$ unendlich

Das Wortproblem

CYK Algorithmus

$w \in L(G)$ und G eine kontextfreie Grammatik ?

- Diese Frage lässt sich mit ja beantworten.
- Ein einfacher Algorithmus mit Komplexität $O(n^3)$ stammt von Cocke, Younger und Kasami (CYK Algorithmus)
- Methode dynamische Programmierung
  - In der Chomsky-Normlform sind alle Regeln ($A,B,C \in N$ und $a \in \Sigma$) entweder
    - $A \rightarrow a$ oder
    - $A \rightarrow BC$
  - Sei $w \in \Sigma^*$ und es soll geprüft werden ob $w \in L(G)$ ist.
    - Ist $|w| = 1$ so muss es eine Regel $S \rightarrow w$ geben, andernfalls $w \notin L(G)$
    - Ist $|w| > 1$ so ist die erste Regel $S \Rightarrow XY \Rightarrow^* w$.

      d.h w lässt sich zerlegen in $w = uv$ mit $X \Rightarrow^* u$ und $Y \Rightarrow^* v$

      Dies wird nun rekursive auf die Teilworte u und v von w angewandt bis $|u| = 1$ und $|v| = 1$
  - Dieser rekursiver Algorithmus lässt sich auch in einen iterativen Algorithmus überführen

Das Wortproblem

CYK Algorithmus: iterative Variante

- Methode der Tabellierung
  - Für jedes Teilwort des Eingabeworts w wird notiert aus welchem Nichtterminal es sich ableiten lässt. Dies wird sukzessiv angewandt bis man beim Startsymbol endet.
- Sei $w = a_1 a_2 \ldots a_n$ das zu erkennende Wort.
- Man erstellt eine Tabelle $T \in (n+1) \times n$
  - in T[i,j] wird notiert aus welchem Nichtterminal das Teilwort $u = a_j a_{j+1} \ldots a_{j+1-i}$ das an Position j beginnt und Länge i hat ableiten lässt.
  - Das Ergebnis ist dann in T[n,1] abzulesen.
  - Enthält T[n,1] das Startsymbol S ist w aus dem S ableitbar und damit $w \in L(G)$

Das Wortproblem

CYK Algorithmus: Beispiel

- Beispiel w_1=0011 und w_2=1001 Frage $w_1 \in L(G)$, $w_2 \in L(G)$?

  mit G=({S,A,B,C},{0,1},S → BC | BA, A → SC | BS, B → 0, C → 1},S}

- Aufstellen der Tabelle

w_1	j=1	j=2	j=3	j=4
i=0	0	0	1	1
i=1	B	B	C	C
i=2		S		
i=3	A	A		
i=4	S			

⇒ w_1=0011 ∈ L(G) und

⇒ w_2=1001 ∉ L(G)

w_2	j=1	j=2	j=3	j=4
i=0	1	0	0	1
i=1	C	B	B	C
i=2			S	
i=3		A		
i=4				

Aufgabe CYK Algorithmus

- Überprüfen Sie mit Hilfe des CYK-Algorithmus, ob folgende Worte w zur Sprache L(G) mit der Grammatik
- G = ({S, A, B, C, D, {a,b,c,S},P,S} mit

  P={S $\rightarrow$ AB | AC | ε, C $\rightarrow$ DB, D $\rightarrow$ AB | AC, A $\rightarrow$ a, B $\rightarrow$ b | c} gehören. (Überprüfen Sie ihr Ergebnis mit FLACI).

  - w= aacbc
  - w= aaabcb