

Grundlagen der Informatik

Prof. Dr. Stefan Enderle

NTA Isny

4. Kontextfreie Sprachen

Einführung / Wiederholung

- Bisher: Reguläre Sprachen
- Eine reguläre Sprache konnte
 - durch einen regulären Ausdruck oder,
 - durch ein Syntaxdiagramm beschrieben und
 - durch einen endlichen Automaten akzeptiert werden.

Symbole / Alphabete / Wörter

- *Symbole* oder Zeichen sind die atomaren Einheiten.
Allgemein: a_1, \dots, a_n
- Das *Alphabet* ist die Menge der Symbole.
Allgemein: $\Sigma = \{a_1, a_2, \dots, a_n\}$
- Ein *Wort über* Σ ist eine endlich lange Zeichenfolge, die über dem Alphabet Σ gebildet werden kann.
Allgemein: $w = a_i a_j \dots a_k$
- Σ^* ist die Menge aller Wörter über Σ .

Beispiel

- Symbole: $a_1=0$ $a_2=1$
- Alphabet: $\Sigma = \{ 0, 1 \}$
- Wörter: 011000, 11111, 1, 000000001
- Menge aller Wörter:
 $\Sigma^* = \{\varepsilon, 0, 1, 00, 01, 10, 11, 000, 001, 010, \dots\}$

Sprache

- Definition:
Eine Sprache L über einem Alphabet Σ ist eine Menge von Wörtern über Σ .
- D.h. L ist eine Teilmenge von Σ^* .

Beispiel

- Alphabet $\Sigma = \{0, 1\}$
- Menge aller Wörter:
 $\Sigma^* = \{\varepsilon, 0, 1, 00, 01, 10, 11, 000, 001, 010, \dots\}$
- Sprache $L = \{0, 1, 10, 11, 100, 101, 110, \dots\}$
= Menge der natürlichen Zahlen in
Binärdarstellung

Beispiel

- Alphabet $\Sigma = \{a, +, -, *, /, (,)\}$
- Menge aller Wörter:
 $\Sigma^* = \{\varepsilon, a, a+, -+-aa,)^*(a-, ******, \dots)\}$
- Sprache $L = \{a, a+a, (a+a), (a+(a^*a)), \dots\}$
= Menge der korrekt geklammerten
Ausdrücke

Reguläre Ausdrücke

- Durch einen regulären Ausdruck kann eine bestimmte Menge von Sprachen beschrieben werden → Reguläre Sprachen
- Reguläre Ausdrücke sind definiert durch:
 1. \emptyset ist ein regulärer Ausdruck
 2. Für jedes $a \in \Sigma$ ist a ein regulärer Ausdruck
 3. Sind r und s reguläre Ausdrücke, so auch
 - $(r|s)$ (Vereinigung)
 - rs (Konkatenation)
 - r^* (Kleene Stern)

Beispiele

- Geg: $\Sigma = \{ a, +, - \}$
 - $L(a+a) := \{ a+a \}$
 - $L(a(+a)^*) := \{ a, a+a, a+a+a, a+a+a+a, \dots \}$
 - $L(a(+|-)a) := \{ a+a, a-a \}$
 - $L(a((+|-)a)^*) := \{ a+a, a-a, a+a+a, a+a-a, a-a+a, a-a-a, a+a+a+a, a+a+a-a, \dots \}$
 - $L((\epsilon|-)a) := \{ a, -a \}$

Syntaxdiagramm

- Ein *Syntaxdiagramm* ist die graphische Darstellung eines regulären Ausdrucks:

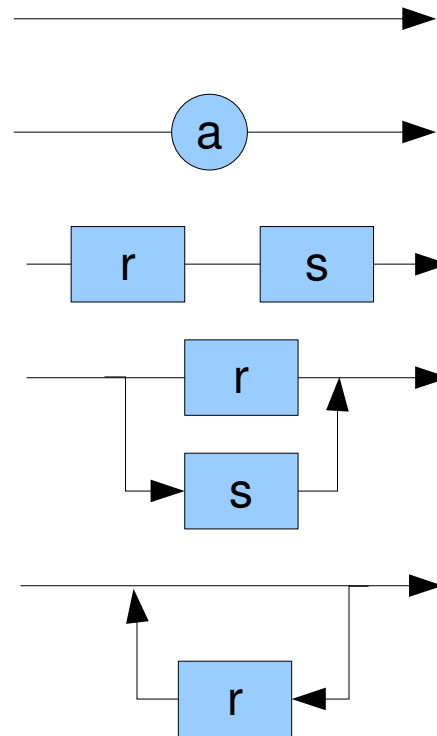
- \emptyset

- $a \in \Sigma$

- rs

- $r|s$

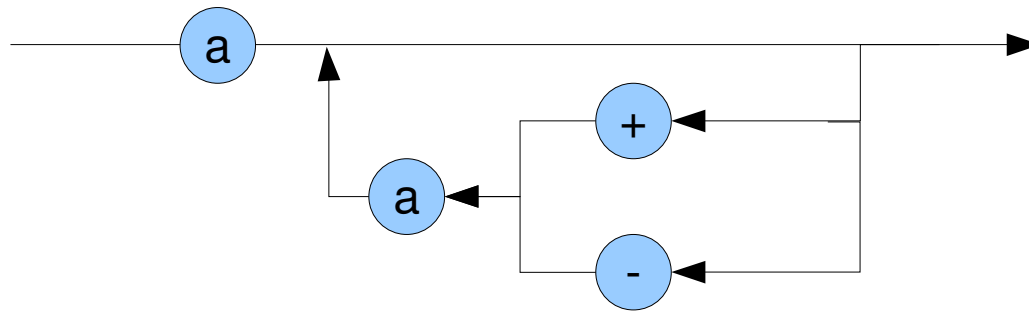
- r^*



Beispiel

- Geg: $\Sigma = \{ a, +, - \}$

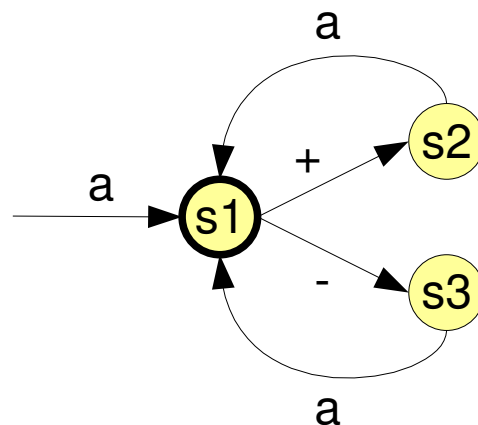
$L(a((+|-)a)^*) := \{ a+a, a-a, a+a+a, a+a-a, a-a+a, a-a-a, a+a+a+a, a+a+a-a, \dots \}$



Endliche Automaten

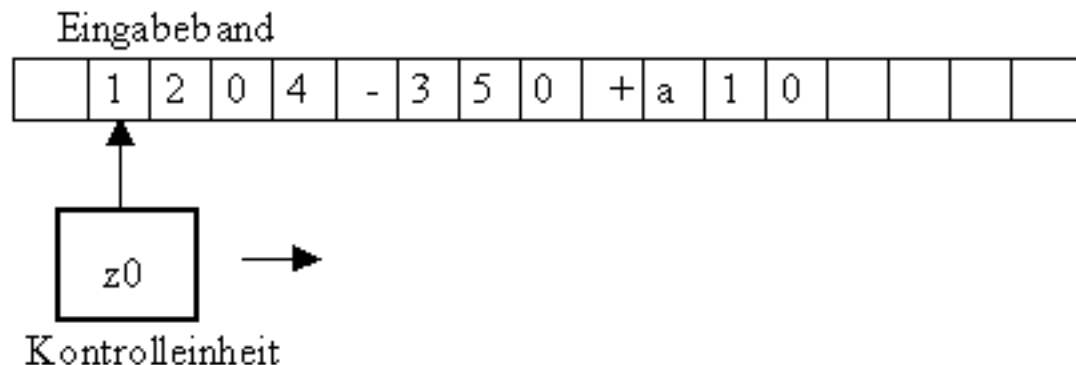
- Ein *endlicher Automat* für eine Sprache L akzeptiert genau die Wörter in L.

$L(a((+|-)a)^*) := \{ a+a, a-a, a+a+a, a+a-a, a-a+a, a-a-a, a+a+a+a, a+a+a-a, \dots \}$



Endlicher Automat - Implementierung

- Aufbau:
 - **Eingabeband** mit Eingabesymbolen
(wird von links nach rechts gelesen)
→ Datei mit Text / Programmcode
 - **Zustandsspeicher** mit aktuellem Zustand
→ Zustand = ganze Zahl (Integer)
 - **Zustandstabelle** mit Zustandsübergängen
 - **Kontrolleinheit** mit Zustandsübergängen



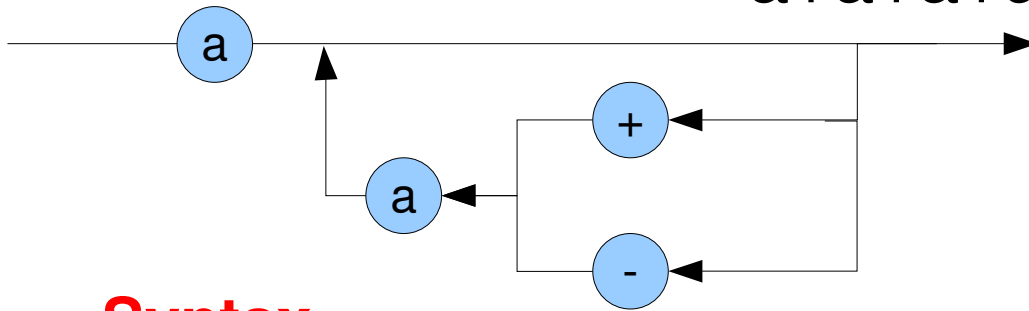
Beispiel

- Geg: $\Sigma = \{ a, +, - \}$

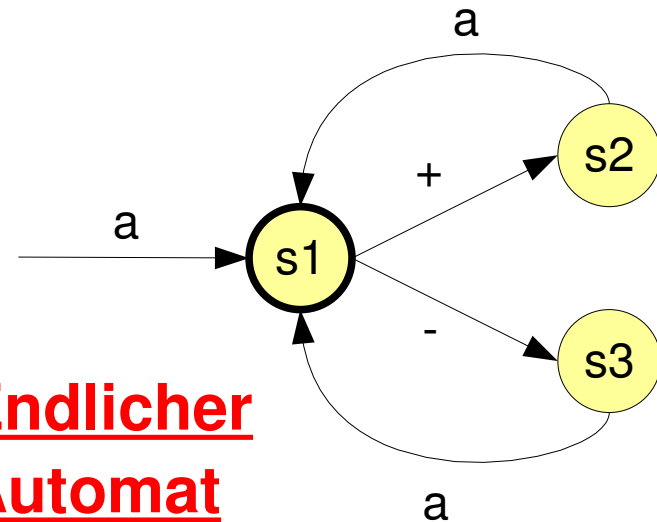
$L(a((+|-)a)^*) := \{ a+a, a-a, a+a+a, a+a-a, a-a+a, a-a-a, a+a+a+a, a+a+a-a, \dots \}$

(reguläre)
Sprache

Regulärer
Ausdruck



Syntax-
diagramm



Endlicher
Automat

Grenzen von regulären Sprachen

- Ein endlicher Automat kann sich nur endlich viele Zustände „merken“. D.h., Sprachen, bei denen man zählen muss und die Obergrenze nicht bekannt ist, lassen sich nicht darstellen.
- Beispiele:
 - $L = \{a^k b^k \mid k > 0\}$
 - $L = \{w \in \{a, b\}^* \mid |w_a| = |w_b|\}$ (gleichviele a's wie b's)
 - $L = \{a, a+a, (a+a), (a+(a+a)), \dots\}$
= Menge der korrekt geklammerten Ausdrücke

4.1 Kontextfreie Sprachen

- Mit einer kontextfreien Sprache lassen sich die regulären Sprachen und eine weitere Menge von Sprachen ausdrücken.
- Eine kontextfreie Sprache kann z.B. durch eine kontextfreie Grammatik beschrieben werden.

Grammatik

- Definition: Grammatik

Eine Grammatik ist ein Tupel $G=(\Sigma, N, P, S)$ mit

- Σ = Terminal-Alphabet
- N = Nonterminal-Alphabet
- P = Produktionenmenge (Regelmenge)
- S = Startsymbol

Beispiel

- $G = (\{a,b,c\}, \{S,A,B\}, P, S)$

mit $P = \{ S \rightarrow AB, A \rightarrow ab, B \rightarrow c \}$

- Die Sprache $L(G)$ enthält nur ein Wort:

$S \Rightarrow AB \Rightarrow abB \Rightarrow abc$

$S \Rightarrow AB \Rightarrow Ac \Rightarrow abc$

(2 Ableitungen)

Kontextfreie Grammatik

- Definition: Kontextfreie Grammatik

Eine Grammatik $G=(\Sigma, N, P, S)$ heißt *kontextfrei*, falls die Regelmengemenge

$$P \subseteq N \times (\Sigma \cup N)^* \quad \text{mit} \quad |P| < \infty$$

ist.

Beispiel

- Gesucht: Grammatik, die gleich viele a's wie b's produziert.

- Teste: $G = (\{a,b\}, \{S\}, P, S)$ mit $P = \{ S \rightarrow aSb \}$

$S \Rightarrow aSb \Rightarrow aaSbb \Rightarrow aaaSbbb \Rightarrow \dots$

- Zur Terminierung: $S \rightarrow \varepsilon$

Also: $G = (\{a,b\}, \{S\}, \{ S \rightarrow aSb, S \rightarrow \varepsilon \}, S)$

bzw. $G = (\{a,b\}, \{S\}, \{ S \rightarrow aSb \mid \varepsilon \}, S)$

Kontextfreie Sprache

- Definition: Kontextfreie Sprache

Eine Sprache L heißt kontextfrei, wenn es eine kontextfreie Grammatik G mit $L(G)=L$ gibt.

- Kontextfreie Sprachen (Grammatiken) heißen auch Typ-2 Sprachen (Grammatiken).

Beispiel

- Gesucht: Grammatik, die alle arithmetischen Ausdrücke erzeugt.

Beispiele:

a

$a+a$

a^*a

$a+(a+a)$

$a^*((a-a)/a)-((a+a)^*a)$

- a ist Hilfssymbol für Zahlen oder Variablen

Beispiel

- Terminalsymbole: $a + - * / ()$
- Nichtterminalsymbole: E (Expression)
 O (Operator)
- Regeln: $E \rightarrow a$
 $E \rightarrow EOE$
 $O \rightarrow + \mid - \mid * \mid /$
 $E \rightarrow (E)$

Beispiel

- Damit ergibt sich die Grammatik zu:

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

mit der Regelmenge

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

Ableitung

- Eine Ableitung beschreibt, wie man vom Startsymbol zu einem Wort der Sprache L kommt.
- Ein Ableitungsschritt „ \Rightarrow “ ist hierbei die Ersetzung *eines* Nichtterminalsymbols durch eines oder mehrere Terminal- oder Nichtterminalsymbole.
- Die Ableitung endet, wenn keine Nichtterminalsymbole mehr auftreten.

Beispiel

- Regelmenge

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

- Ableitung:

$$E \Rightarrow EOE \Rightarrow aOE \Rightarrow a+E \Rightarrow a+a$$

$$E \Rightarrow EOE \Rightarrow aOE \Rightarrow a+E \Rightarrow a+(E)$$

$$\Rightarrow a+(EOE) \Rightarrow a+(aOE) \Rightarrow a+(a-E) \Rightarrow a+(a-a)$$

Transitive Hülle

- Die transitive Hülle des Ableitungsschritts \Rightarrow^* bezeichnet die gesamte Ableitung bis zu einem Wort:

$$E \Rightarrow^* a+a$$

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

- Es gilt also: $L(G) = \{ w \in \Sigma^* \mid S \Rightarrow^* w \}$

Linksableitung

- Sei $G = \{ \{ a,b \}, \{ S,A \}, P, S \}$

mit der Regelmengemenge

$$P = \{ S \rightarrow aAS \mid a, \\ A \rightarrow SbA \mid SS \mid ba \}$$

- Eine Ableitung ist eine *Linksableitung*, falls in jedem Ableitungsschritt das am weitesten links stehende Nichtterminal ersetzt wird.
- $S \Rightarrow aAS \Rightarrow aSbAS \Rightarrow aabAS \Rightarrow aabbaS \Rightarrow aabbaa$

Rechtsableitung

- Sei $G = \{ \{ a,b \}, \{ S,A \}, P, S \}$

mit der Regelmeng

$$P = \{ S \rightarrow aAS \mid a, \\ A \rightarrow SbA \mid SS \mid ba \}$$

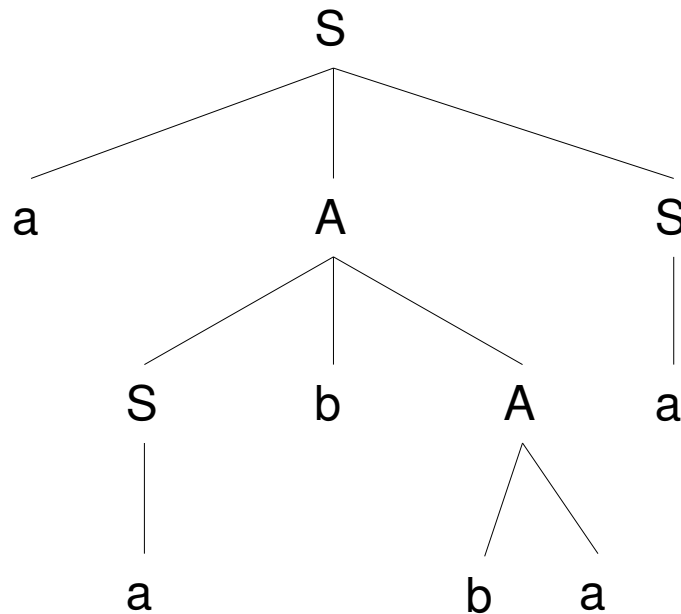
- Eine Ableitung ist eine *Rechtsableitung*, falls in jedem Ableitungsschritt das am weitesten rechts stehende Nichtterminal ersetzt wird.
- $S \Rightarrow aAS \Rightarrow aAa \Rightarrow aSbAa \Rightarrow aSbbaa \Rightarrow aabbaa$

Ableitungsbäume

- Sei $G = (\Sigma, N, P, S)$ eine kontextfreie Grammatik.
- Ein Baum t heißt Ableitungsbaum (engl.: parse tree) über G , wenn folgendes gilt:
 - Die Wurzel ist mit S markiert
 - Jeder innere Knoten ist mit Nichtterminalen markiert
 - Jedes Blatt ist mit $N \cup \Sigma \cup \{\varepsilon\}$ markiert
 - Wenn ein innerer Knoten mit $A \in N$ markiert ist und seine Nachfolger von links nach rechts mit $X_1, \dots, X_n \in N \cup \Sigma \cup \{\varepsilon\}$, dann muss $A \rightarrow X_1 \dots X_n \in P$ sein.

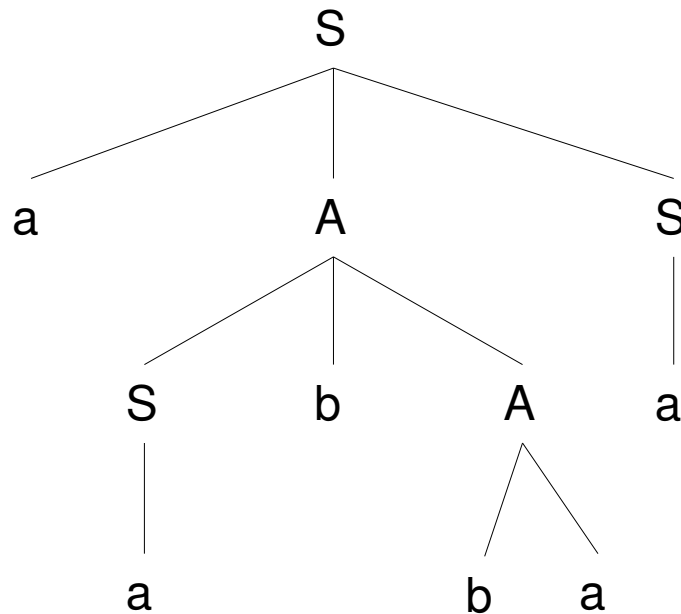
Beispiel

- Ableitungsbaum für die Linksableitung
 $S \Rightarrow aAS \Rightarrow aSbAS \Rightarrow aabAS \Rightarrow aabbaS \Rightarrow aabbaa$



Beispiel

- Ableitungsbaum für die Rechtsableitung
 $S \Rightarrow aAS \Rightarrow aAa \Rightarrow aSbAa \Rightarrow aSbbaa \Rightarrow aabbaa$



Die Ableitungsbäume sind identisch!

Gegenbeispiel

- Sei $G = \{ \{ a,b,c \}, \{ S,A,B \}, P, S \}$

mit der Regelmenge

$$P = \{ \begin{array}{l} S \rightarrow aB \mid Ac, \\ A \rightarrow ab, \\ B \rightarrow bc \end{array} \}$$

- $L(G)$ enthält nur das Wort abc .

Dies lässt sich aber auf zwei Arten ableiten:

$$S \Rightarrow Ac \Rightarrow abc$$

$$S \Rightarrow aB \Rightarrow abc$$

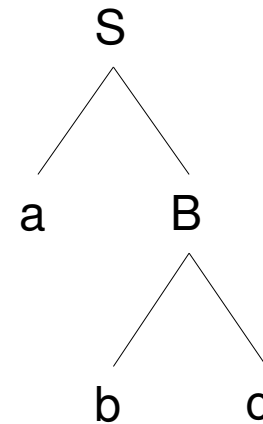
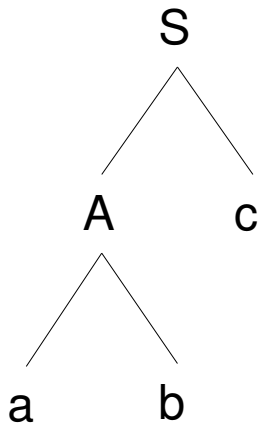
Gegenbeispiel

- Ableitungsbäume für

$S \Rightarrow Ac \Rightarrow abc$

und

$S \Rightarrow aB \Rightarrow abc$

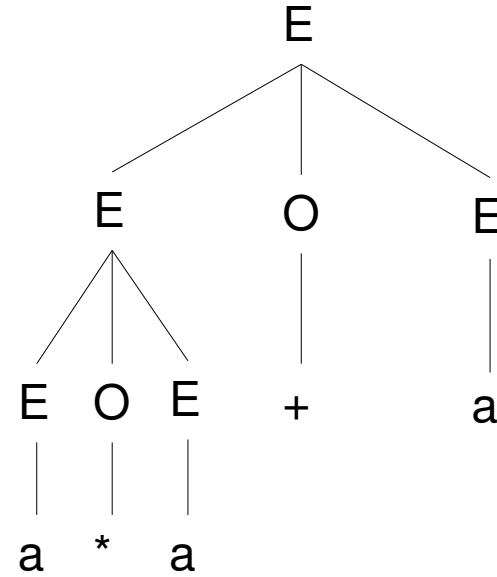
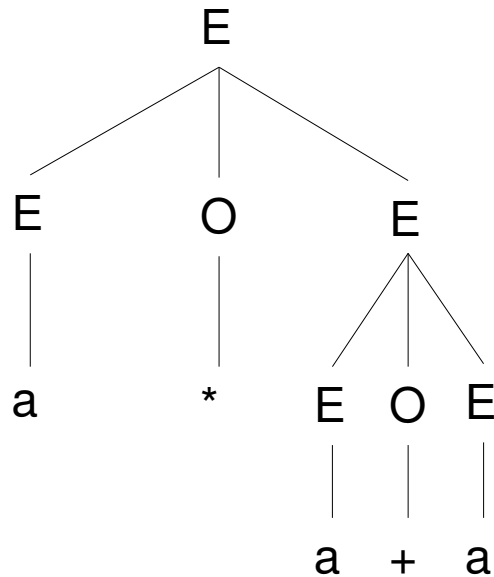


Mehrdeutigkeit

- Eine Kontextfreie Grammatik heißt *mehrdeutig*, falls es für mindestens ein Wort $w \in L(G)$ zwei (oder mehr) verschiedene Ableitungsbäume gibt.

Beispiel

- $E \Rightarrow EOE \Rightarrow aOE \Rightarrow a^*E \Rightarrow a^*EOE \Rightarrow a^*aOE \Rightarrow a^*a+E \Rightarrow a^*a+a$
 $E \Rightarrow EOE \Rightarrow EOEEOE \Rightarrow aOEEOE \Rightarrow a^*EOE \Rightarrow a^*aOE \Rightarrow a^*a+E \Rightarrow a^*a+a$



Vereinfachungen

- Eine Kontextfreie Grammatik lässt sich durch folgende Maßnahmen vereinfachen:
 - Eliminieren von ε -Regeln
 - Eliminieren nutzloser Symbole
 - Eliminieren von Kettenregeln

Eliminieren von ε -Regeln

- Satz:
Zu jeder kontextfreien Grammatik $G = (\Sigma, N, P, S)$ kann eine ε -freie kontextfreie Grammatik G' konstruiert werden, so dass gilt:

$$L(G') = L(G) - \{ \varepsilon \}$$

Eliminieren von ε -Regeln

- Verfahren in 5 Schritten:
 1. Bestimme alle Nichtterminale von G , die linke Seite einer ε -Regel sind.

$$N' = \{A \in N \mid A \rightarrow \varepsilon\}$$

Eliminieren von ε -Regeln

- Bestimme alle Nichtterminale, aus denen das leere Wort ableitbar ist, also

$$N'' = \{A \in N \mid A \Rightarrow^* \varepsilon\}$$

Gehe dazu von N' aus und bestimme so lange die Nichtterminale, aus denen Wörtern aus N' abgeleitet werden können, bis sich die N'' nicht mehr ändert.

wiederhole

$$N'' := N'$$

$$N' := N' \cup \{A \in N \mid A \rightarrow w, w \in N''^*\}$$

bis $N' = N''$

Eliminieren von ε -Regeln

3. Für jede Regel, deren rechte Seite ein Nichtterminal aus N'' enthält, fügen wir eine Regel hinzu ohne dieses Nichtterminal.

$P' := P$

wiederhole

$P'' := P'$

für jede Regel $A \rightarrow wBw'$ aus P'' mit $B \in N''$

$P' := P' \cup \{A \rightarrow ww'\}$

bis $N' = N''$

Eliminieren von ε -Regeln

4. Eliminiere alle ε -Regeln:

$$P'' = P' - \{ A \rightarrow \varepsilon \mid \text{für alle } A \}$$

5. Die gesuchte Grammatik ist dann

$$G' = (\Sigma, N, P'', S)$$

Eliminieren nutzloser Symbole

- Definition:

Sei $G = (\Sigma, N, P, S)$ eine kontextfreie Grammatik.

Ein Nichtterminal $A \in N$ heißt *nützlich* für G , wenn mit seiner Hilfe mindestens ein Terminalwort erzeugt werden kann, wenn also gilt:

$$S \Rightarrow^* uAv \Rightarrow^* w$$

mit $u, v \in (N \cup \Sigma)^*$ und $w \in \Sigma^*$.

Eliminieren nutzloser Symbole

- Satz:
Zu jeder kontextfreien Grammatik $G = (\Sigma, N, P, S)$ existiert eine *reduzierte* kontextfreie Grammatik G' (ohne nutzlose Symbole).

(Beweis ähnlich wie oben durch Konstruktion.)

Eliminieren von Kettenregeln

- Kettenregeln sind Regeln der Form
 $A \rightarrow B$ mit $A, B \in N$
- Diese tragen zu Erzeugung eines Wortes nichts bei.

Eliminieren von Kettenregeln

- Eliminierung in 3 Schritten:

1. Entfernen von Zyklen:

Gibt es Nichtterminale $B_1 \dots B_k \in N$, $k > 1$ und Regeln der Form $B_i \rightarrow B_{i+1}$ mit $1 \leq i \leq k-1$, und $B_k \rightarrow B_1$, dann entfernen wir alle B_i aus N und fügen ein neues Nichtterminal B hinzu. In allen Regeln in P ersetzen wir dann jedes B_i durch B .

Eliminieren von Kettenregeln

2. Umnummerierung

N habe nun n Elemente.

Wir bezeichnen diese mit $A_1 \dots A_n$, so dass gilt:

Wenn $A_i \rightarrow A_j$ dann ist $i < j$.

(Dies ist möglich, da G keine Zyklen mehr enthält!)

Eliminieren von Kettenregeln

3. Ersetzen von Kettenregeln

Gibt es noch Regeln $A_i \rightarrow A_j$

dann kann jede Regel $A_j \rightarrow w$ ersetzt werden durch

$A_i \rightarrow w$.

für alle $i := s-1$ to 1 tue

 für alle $i < j$ tue

 streiche $A_i \rightarrow A_j$

 wenn $A_j \rightarrow w$ dann ersetze durch $A_i \rightarrow w$

Chomsky Normalform

- Definition:

Sei $G = (\Sigma, N, P, S)$ eine kontextfreie Grammatik.
 G ist in *Chomsky-Normalform (CNF)*, wenn gilt:

$$P \subseteq N \times ((N \circ N) \cup \Sigma)$$

Die Regeln einer CNF haben also folgende Form:

$$A \rightarrow BC \quad \text{oder} \quad A \rightarrow a$$

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

Chomsky Normalform

- Satz:

Zu jeder kontextfreien Grammatik $G = (\Sigma, N, P, S)$ mit $\varepsilon \notin L(G)$ existiert eine äquivalente Grammatik G' in Chomsky-Normalform.

Chomsky Normalform

- Beweis:

G sei ε -frei, reduziert und ohne Kettenregeln.

Betrachte die Regeln, die nicht in CNF sind:

$$A \rightarrow X_1 X_2 \dots X_m \text{ mit } m \geq 2 \text{ und } X_i \in N \cup \Sigma$$

Diese Regeln formen wir jeweils in 2 Schritten in CNF-Regeln um:

Chomsky Normalform

1. Ist in der Regel $A \rightarrow X_1 X_2 \dots X_m$ $X_i \in \Sigma$ ein Terminal, so ersetzen wir dieses durch ein neues Nichtterminal C_{X_i} und fügen die Regel $C_{X_i} \rightarrow X_i$ zu P und C_{X_i} zu N hinzu.

Chomsky Normalform

2. Alle Regeln, die jetzt noch nicht in CNF sind, haben die Form $A \rightarrow B_1 B_2 \dots B_m$ mit $B_i \in N$ und $m \geq 3$.

Jede Regel dieser Art ersetzen wir durch folgende Regelmengemenge:

$$\begin{aligned} A &\rightarrow B_1 D_1 \\ D_1 &\rightarrow B_2 D_2 \\ &\dots \\ D_{m-3} &\rightarrow B_{m-2} D_{m-2} \\ D_{m-2} &\rightarrow B_{m-1} B_m \end{aligned}$$

Einschub:

Anwendungsbeispiel XML

- XML = Extensible Markup Language
- Sprache für Datenaustausch und Dokumentenmarkierung
- ASCII Datei

Einschub:

Anwendungsbeispiel XML

- Ein XML-Dokument besteht aus *Elementen*.
- Ein Element ist Text, der in zueinander passende öffnende und schließende *Tags* eingeschlossen ist.

Einschub:

Anwendungsbeispiel XML

- Innerhalb eines Elements kann gewöhnlicher Text vorkommen oder wieder Elemente oder beides durcheinander.
- Die Tags markieren die Struktur des Dokuments, der Text stellt den Inhalt dar.

Einschub: Anwendungsbeispiel XML

```
<bibliographie>
<buch>
  <autor>
    S. Abiteboul
  </autor>
  <autor>
    R. Hull
  </autor>
  <titel>
    Foundation of Databases
  </titel>
  <verlag>
    Addison-Wesey
  </verlag>
  <jahr>
    1985
  </jahr>
</buch>
```

```
<artikel>
  <autor>
    E.F. Codd
  </autor>
  <titel>
    A Relational Model of Data Banks
  </titel>
  <journal>
    Communications of the ACM
  </journal>
  <jahr>
    1970
  </jahr>
</artikel>
</bibliographie>
```

Einschub:

Anwendungsbeispiel XML

- Die Gesamtstruktur ist immer ein Baum.
- Welche Knoten welche Unterknoten haben können wird durch eine separate Datei definiert:

Document Type Definition DTD

- Die DTD ist im wesentlichen eine kontextfreie Grammatik!

Einschub:

Anwendungsbeispiel XML

- Beispiel: Bisherige Schreibweise:

bibliographie	→ (buch artikel)*
buch	→ autoren titel verlag jahr
artikel	→ autoren titel journal jahr
autoren	→ autor autor autoren
autor	→ text
titel	→ text
journal	→ text
jahr	→ text
verlag	→ text

Einschub: Anwendungsbeispiel XML

- Beispiel: DTD Schreibweise:

```
<!ELEMENT bibliographie      (buch | artikel)+ >
```

```
<!ELEMENT buch               (autor+,titel,verlag,jahr)>
```

```
<!ELEMENT artikel           (autor+,titel,journal,jahr)>
```

```
<!ELEMENT autor             PCDATA>
```

```
<!ELEMENT titel             PCDATA>
```

```
<!ELEMENT journal           PCDATA>
```

```
<!ELEMENT jahr              PCDATA>
```

```
<!ELEMENT verlag            PCDATA>
```

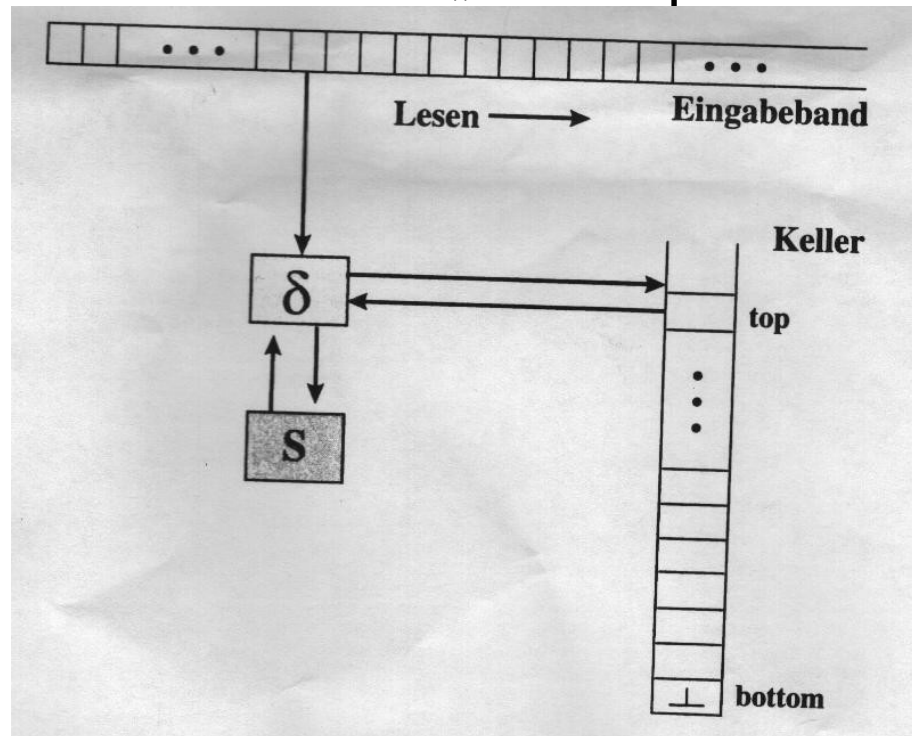
Einschub:

Anwendungsbeispiel XML

- Das heißt:
 - Die DTD ist die kontextfreie Grammatik, die eine Sprache definiert (nämlich die erlaubten XML Dateien).
 - Eine konkrete XML Datei ist dann EIN mögliches Wort aus dieser Sprache, mit einem bestimmten Ableitungsbaum.
- Wie kann ein Parser erkennen, ob eine XML-Datei einer DTD entspricht?

4.2 Kellerautomaten

- Ein endlicher Automat hatte als „Gedächtnis“ nur den Speicher für den aktuellen Zustand.
- Wir erweitern dieses Modell um einen „Kellerspeicher“ = Stack



Nichtdeterministische Kellerautomaten

- Definition: Kellerautomat (PDA = „push down automata“)

Ein (nichtdeterministischer) Kellerautomat K ist ein Tupel

$K = (\Sigma, S, T, \delta, s_0, \perp, F)$ mit

Σ = Eingabealphabet

S = endliche Zustandsmenge

T = Kelleralphabet

δ = Zustandsüberführung $\delta : S \times (\Sigma \cup \{\varepsilon\}) \times T \rightarrow \underline{P}(S \times T^*)$

s_0 = Startzustand $s_0 \in S$

\perp = Keller-Bottomsymbol $\perp \in T$

F = Endzustandsmenge

Zustandsüberführung

- Zustandsüberführung $\delta : S \times (\Sigma \cup \{\varepsilon\}) \times T \rightarrow \underline{P}(S \times T^*)$

d.h. $\delta(s, a, A) = \{(s', B_1 \dots B_k), (s'', C_1 \dots C_m), \dots, (s''', N_1 \dots N_n)\}$

- $(s', B_1 \dots B_k) \in \delta(s, a, A)$ bedeutet:

Wenn sich K im Zustand s befindet,
gerade das Zeichen a liest,
und A das oberste Kellersymbol ist,

dann kann K in den Zustand s' wechseln,
und dabei A durch das Wort $B_1 \dots B_k$ ersetzen

Zustandsüberführung

- Festlegung: B_1 ist das neue oberste Kellersymbol
- Spezialfälle:
 - Ist $a=\varepsilon$, wird kein Eingabesymbol gelesen, sondern nur der Keller verändert.
 - Ist bei $B_1 \dots B_k$ $k=0$, also $B_1 \dots B_k = \varepsilon$, so wird A durch das leere Wort ersetzt, also gelöscht.

Zustandsüberführung

- Einfachere Schreibweise:

$$\begin{aligned} & (s, a, A, s', B_1 \dots B_k), \\ & (s, a, A, s'', C_1 \dots C_m) \\ & (s, a, A, s''', N_1 \dots N_n) \end{aligned} \}$$

- Bedeutung wie oben:

Wenn sich K im Zustand s befindet,
gerade das Zeichen a liest,
und A das oberste Kellersymbol ist,
dann kann K in den Zustand s' wechseln,
und dabei A durch das Wort $B_1 \dots B_k$ ersetzen
oder in den Zustand s'' wechseln,...

Konfiguration

- Die *Konfiguration* des Kellerautomaten K enthält
 - den aktuellen Zustand s
 - das noch zu verarbeitende Suffix w des Eingabewortes
 - den aktuellen Kellerinhalt α .
- Formal: Konfiguration $k=(s,w,\alpha) \in S \times \Sigma^* \times T^*$

Konfigurationsübergänge

- Ein *Konfigurationsübergang* ist festgelegt durch die Relation

$$(S \times \Sigma^* \times T^*) \times (S \times \Sigma^* \times T^*)$$

- Wenn also gilt: $(s, a, A, s', \beta) \in \delta$

Dann folgt auch: $(s, av, A\alpha) \rightarrow (s', v, \beta\alpha)$

- Durch einen Konfigurationsübergang wird also das oberste Kellersymbol (A) gelöscht und durch ein neues Wort (β) ersetzt.

Akzeptierte Sprache

- Sei $K = (\Sigma, S, T, \delta, s_0, \perp, F)$ ein Kellerautomat.

Dann ist folgende Sprache die *von K akzeptierte Sprache*:

$$L(K) = \{ w \in \Sigma \mid (s_0, w, \perp) \rightarrow^* (s_f, \varepsilon, \gamma), s_f \in F, \gamma \in T^* \}$$

Beispiel

- Gesucht: Kellerautomat für $L = \{ a^n b^n \mid n \geq 0 \}$
- Idee: Wir legen für jedes gelesene „a“ ein „a“ im Keller ab. Für jedes gelesene „b“ löschen wir ein „a“ im Keller. Ist das Eingabewort leer und alle „a“s gelöscht, wird das Eingabewort akzeptiert.

Beispiel

- $K = (\Sigma, S, T, \delta, s_0, \perp, F)$ mit
 - Σ = Eingabealphabet = { a, b }
 - S = endliche Zustandsmenge = { s_0, s_1, s_f }
 - T = Kellularphabet = { a, \perp }
 - δ = Zustandsüberführung = (todo)
 - s_0 = Startzustand = s_0
 - \perp = Keller-Bottomsymbol = \perp
 - F = Endzustandsmenge = { s_f }

Beispiel

- $\delta = \{$
 $(s_0, \varepsilon, \perp, s_f, \varepsilon),$
 $(s_0, a, \perp, s_0, a\perp),$
 $(s_0, a, a, s_0, aa),$
 $(s_0, b, a, s_1, \varepsilon),$
 $(s_1, b, a, s_1, \varepsilon),$
 $(s_1, \varepsilon, \perp, s_f, \varepsilon)$ $\}$

Beweis

- Konfigurationsübergänge für a^3b^3 :

$$\begin{aligned}(s_0, a^3b^3, \perp) &\rightarrow (s_0, a^2b^3, a\perp) \rightarrow (s_0, ab^3, aa\perp) \\ &\rightarrow (s_0, b^3, aaa\perp) \rightarrow (s_1, b^2, aa\perp) \\ &\rightarrow (s_1, b, a\perp) \quad \rightarrow (s_1, \varepsilon, \perp) \rightarrow (s_f, \varepsilon, \varepsilon)\end{aligned}$$

- Also gilt: $(s_0, a^3b^3, \perp) \rightarrow^* (s_f, \varepsilon, \varepsilon)$

- D.h., a^3b^3 ist ein Wort der Sprache von K , $a^3b^3 \in L(K)$

Beispiel 2

- Gesucht: Kellerautomat für $L = \{ w c \text{ sp}(w) \mid w \in \{a,b\}^* \}$
- Die Sprache L enthält Wörter bestehend aus
 - einer beliebigen Kombination von „a“s und „b“s
 - gefolgt von einem „c“
 - gefolgt von der a-b-Kombination gespiegelt.
- Idee: Wir legen die Buchstaben von w auf den Keller, nachdem c gelesen wurde werden die folgenden Buchstaben mit dem jeweils obersten Kellersymbol verglichen. Wenn Eingabeband und Keller leer sind wird akzeptiert.

Beispiel 2

- $K = (\Sigma, S, T, \delta, s_0, \perp, F)$ mit
 - Σ = Eingabealphabet = { a, b, c }
 - S = endliche Zustandsmenge = { s_0, s_c }
 - T = Kellularphabet = { a, b, \perp }
 - δ = Zustandsüberführung = (todo)
 - s_0 = Startzustand = s_0
 - \perp = Keller-Bottomsymbol = \perp
 - F = Endzustandsmenge = { s_c }

Beispiel 2

- $\delta = \{$

$(s_0, c, \perp, s_c, \varepsilon),$	
$(s_0, a, \perp, s_0, a\perp),$	$(s_0, b, \perp, s_0, b\perp),$
$(s_0, a, a, s_0, aa),$	$(s_0, b, b, s_0, bb),$
$(s_0, a, b, s_0, ab),$	$(s_0, b, a, s_0, ba),$
$(s_0, c, a, s_c, a),$	$(s_0, c, b, s_c, b),$
$(s_c, a, a, s_c, \varepsilon),$	$(s_c, b, b, s_c, \varepsilon),$
$(s_c, \varepsilon, \perp, s_c, \varepsilon)$	$\}$

Beweis 2

- Konfigurationsübergänge für „abbcbbba“:

$$\begin{aligned}(s_0, \text{abbcbbba}, \perp) &\rightarrow (s_0, \text{bbcbba}, a\perp) \rightarrow (s_0, \text{bcbba}, ba\perp) \\ &\rightarrow (s_0, \text{cbba}, bba\perp) \rightarrow (s_c, \text{bba}, bba\perp) \\ &\rightarrow (s_c, \text{ba}, ba\perp) \quad \rightarrow (s_c, a, a\perp) \\ &\rightarrow (s_c, \varepsilon, \perp) \quad \rightarrow (s_c, \varepsilon, \varepsilon)\end{aligned}$$

- Also gilt: $(s_0, \text{abbcbbba}, \perp) \rightarrow^* (s_c, \varepsilon, \varepsilon)$

- D.h., $\text{abbcbbba} \in L(K)$

Beispiel 3

- Gesucht: Kellerautomat für $L = \{ w \text{ sp}(w) \mid w \in \{a,b\}^* \}$
- Die Sprache L enthält Wörter bestehend aus
 - einer beliebigen Kombination von „a“s und „b“s
 - gefolgt von der a-b-Kombination gespiegelt.
 - (Ohne Trennzeichen in der Mitte !)
- Problem: Wir wissen nicht, wann die gespiegelte Hälfte beginnt!
- Idee: Nichtdeterministischer Automat:
Wir legen die Buchstaben von w auf den Keller. Bei jedem zweiten Buchstaben eines Paares kann ganz normal abgelegt werden ODER mit dem Test auf Spiegelbild begonnen werden.

Beispiel 3

- $K = (\Sigma, S, T, \delta, s_0, \perp, F)$ mit
 - Σ = Eingabealphabet = $\{ a, b \}$
 - S = endliche Zustandsmenge = $\{ s_0, s_c, s_f \}$
 - T = Kellularphabet = $\{ a, b, \perp \}$
 - δ = Zustandsüberführung = (todo)
 - s_0 = Startzustand = s_0
 - \perp = Keller-Bottomsymbol = \perp
 - F = Endzustandsmenge = $\{ s_f \}$

Beispiel 3

- $\delta = \{$

$(s_0, \varepsilon, \perp, s_c, \varepsilon),$	(leeres Wort akzeptieren)
$(s_0, a, \perp, s_0, a\perp),$	(erstes a, auf Keller)
$(s_0, b, \perp, s_0, b\perp),$	(erstes b, auf Keller)
$(s_0, a, b, s_0, ab),$	(a nach b, auf Keller)
$(s_0, b, a, s_0, ba),$	(b nach a, auf Keller)
$(s_0, a, a, \{ (s_0, aa),$	(a nach a, auf Keller oder
$(s_c, \varepsilon) \})$	erster Abgleich mit a)
$(s_0, b, b, \{ (s_0, bb),$	(b nach b, auf Keller oder
$(s_c, \varepsilon) \})$	erster Abgleich mit b)
$(s_c, a, a, s_c, \varepsilon),$	(weiterer Abgleich mit a)
$(s_c, b, b, s_c, \varepsilon),$	(weiterer Abgleich mit b)
$(s_c, \varepsilon, \perp, s_f, \varepsilon)$	}

Beweis 3

- Konfigurationsübergänge für „abbbba“:
- Versuch 1:
 $(s_0, abbbba, \perp) \rightarrow (s_0, bbbba, a\perp) \rightarrow (s_0, bbba, ba\perp)$
 $\rightarrow (s_c, bba, a\perp)$ bricht ab!
- Versuch 2:
 $(s_0, abbbba, \perp) \rightarrow (s_0, bbbba, a\perp) \rightarrow (s_0, bbba, ba\perp)$
 $\rightarrow (s_0, bba, bba\perp) \rightarrow (s_c, ba, ba\perp)$
 $\rightarrow (s_c, a, a\perp) \rightarrow (s_c, \varepsilon, \perp) \rightarrow (s_f, \varepsilon, \varepsilon)$
- Also gilt: $(s_0, abbbba, \perp) \rightarrow^* (s_f, \varepsilon, \varepsilon)$

Beweis 3

- Falls ein Versuch in Leere führt, muss zurück gesetzt werden und die nächste Möglichkeit getestet werden.
- Backtracking ! (siehe Algorithmen)

Akzeptierte Sprache (2)

- Sei $K = (\Sigma, S, T, \delta, s_0, \perp, F)$ ein Kellerautomat.

- Die bisherige Definition sagte folgendes:
 - Für die von K akzeptierte Sprache $L(K)$ gilt:

$$L(K) = \{ w \in \Sigma \mid (s_0, w, \perp) \rightarrow^* (s_f, \varepsilon, \gamma), s_f \in F, \gamma \in T^* \}$$

„akzeptieren mit Endzustand“

- Äquivalent hierzu gilt (ohne Beweis):

$$L(K) = \{ w \in \Sigma \mid (s_0, w, \perp) \rightarrow^* (s, \varepsilon, \varepsilon) \}$$

„akzeptieren mit leerem Keller“

- Dann ist auch $K' = (\Sigma, S, T, \delta, s_0, \perp)$ ein Kellerautomat.

4.3 Äquivalenz von kontextfreien Grammatiken und Kellerautomaten

- Satz:
Zu jeder kontextfreien Grammatik $G = (\Sigma, N, P, S)$ lässt sich ein Kellerautomat K konstruieren mit $L(K) = L(G)$.
- Konstruktion:
Der Kellerautomat soll die Linksableitungen von G simulieren:
 - Ist die aktuelle Eingabe gleich dem obersten Keller-Symbol, so wird das Keller-Symbol gelöscht.
 - Ist oberste Keller-Symbol ein Nichtterminal, dann wird keine Eingabe gelesen und das Nichtterminal durch seine rechte Seite ersetzt.
 - Als Keller-Bottom wird das Startsymbol von G benutzt.
 - Der Automat hat nur einen Zustand.

Äquivalenz von kontextfreien Grammatiken und Kellerautomaten

- Formal:

Gegeben: $G = (\Sigma, N, P, S)$

- Die Konstruktion ergibt den Kellerautomaten

$$K = (\Sigma, \{q\}, \Sigma \cup N, \delta, q, S)$$

mit $\delta = \{ (q, a, a, q, \varepsilon) \mid a \in \Sigma \} \cup \{ (q, \varepsilon, A, q, \alpha) \mid A \rightarrow \alpha \in P \}$

Beispiel

- Gegeben: $G = (\{ a,b \}, \{ S \}, \{ S \rightarrow aSb \mid \varepsilon \}, S)$
- Konstruktion von K :

$$K = (\{ a,b \}, \{ q \}, \{ a, b, S \}, \delta, q, S)$$

mit $\delta = \{$

$$\begin{aligned} & (q,a,a,q,\varepsilon), \\ & (q,b,b,q,\varepsilon), \\ & (q,\varepsilon,S,q,aSb), \\ & (q,\varepsilon,S,q,\varepsilon) \} \end{aligned}$$

Beweis

- Konfigurationsübergänge für a^3b^3 :

$(q, a^3b^3, S) \rightarrow (q, a^3b^3, aSb) \rightarrow (q, a^2b^3, Sb)$
 $\rightarrow (q, a^2b^3, aSbb) \rightarrow (q, ab^3, Sbb)$
 $\rightarrow (q, ab^3, aSbbb) \rightarrow (q, b^3, Sbbb)$
 $\rightarrow (q, b^3, bbb) \rightarrow (q, b^2, bb)$
 $\rightarrow (q, b, b) \rightarrow (q, \varepsilon, \varepsilon)$

Parser

- Einen Kellerautomaten, der nach obigem Schema konstruiert wird, nennen wir *Parser*.
- Ein Parser dient in der Praxis zur Erkennung

Äquivalenz (2)

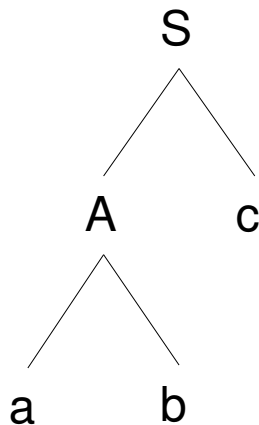
- Bereits gezeigt:
Grammatik $G \Rightarrow$ Kellerautomat K
- Damit Äquivalenz gilt ist noch zu zeigen:
Kellerautomat $K \Rightarrow$ Grammatik G
- Dies gilt tatsächlich (ohne Beweis),
also sind kontextfreie Grammatiken und Kellerautomaten
äquivalent

4.4 Praktische Anwendung: Compilerbau

- Wiederholung: Ableitungsbäume

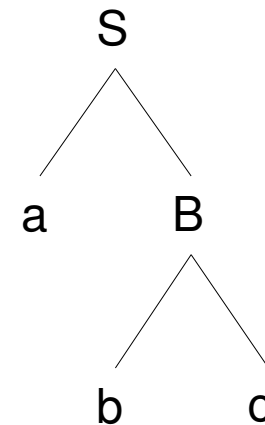
Es gibt *mehrdeutige Grammatiken*, d.h. ein Wort kann durch unterschiedliche Ableitungsbäume repräsentiert werden. Beispiel: $P = \{ S \rightarrow aB \mid Ac, A \rightarrow ab, B \rightarrow bc \}$

$S \Rightarrow Ac \Rightarrow abc$



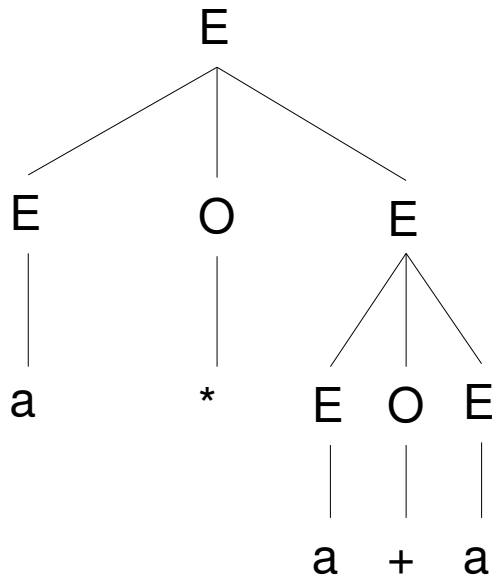
und

$S \Rightarrow aB \Rightarrow abc$

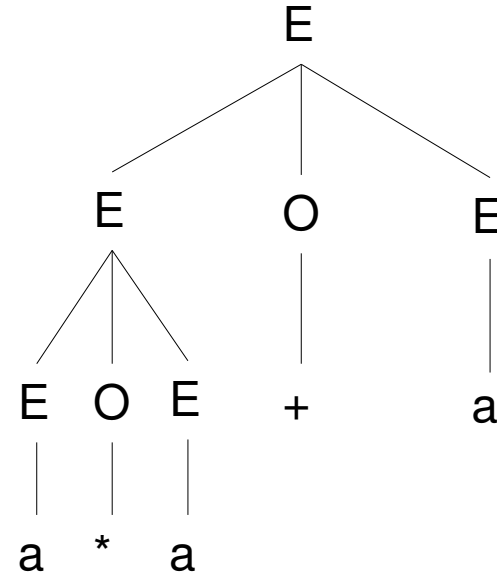


Beispiel

- $E \Rightarrow EOE \Rightarrow aOE \Rightarrow a^*E \Rightarrow a^*EOE \Rightarrow a^*aOE \Rightarrow a^*a+E \Rightarrow a^*a+a$
 $E \Rightarrow EOE \Rightarrow EOEOE \Rightarrow aOEOE \Rightarrow a^*EOE \Rightarrow a^*aOE \Rightarrow a^*a+E \Rightarrow a^*a+a$



„Punkt vor Strich“



„falsch“

Eindeutige Grammatiken

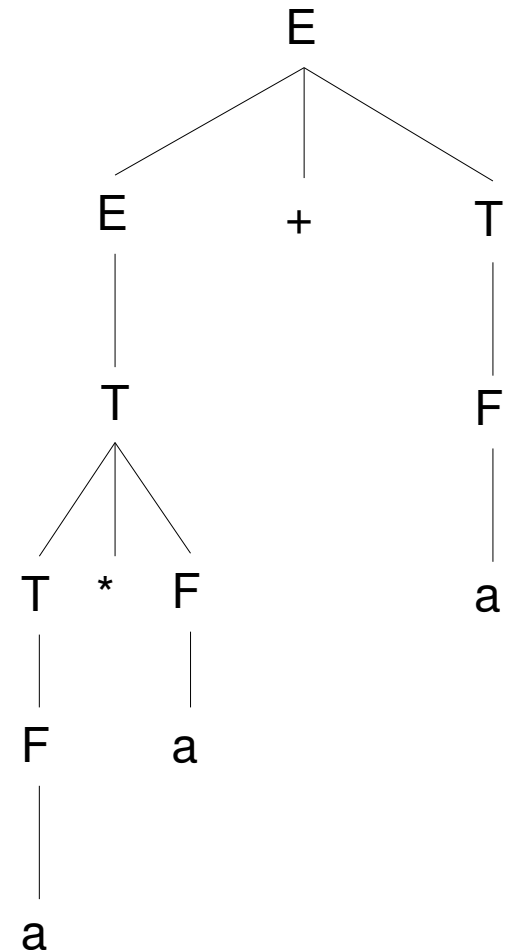
- Für reelle Parser sind mehrdeutige Grammatiken ungeeignet. Man beschränkt sich daher in der Praxis auf *eindeutige Grammatiken*.
- Beispiel:
Eindeutige Grammatik für Arithmetische Ausdrücke:
Wir verwenden 3 Hilfssymbole:
 - F (für Faktor) steht für „atomare Ausdrücke“, d.h. für einzelne Operanden oder geklammerte Ausdrücke
 - T (für Term) steht für „Punktoperationen“, also * und /
 - E (für Expression) steht für „Strichoperationen“ +, -

Eindeutige Grammatiken

- Regelmenge P:
 - $F \rightarrow a \mid (E)$
 - $E \rightarrow E+T \mid E-T \mid T$
 - $T \rightarrow T^*F \mid T/F \mid F$

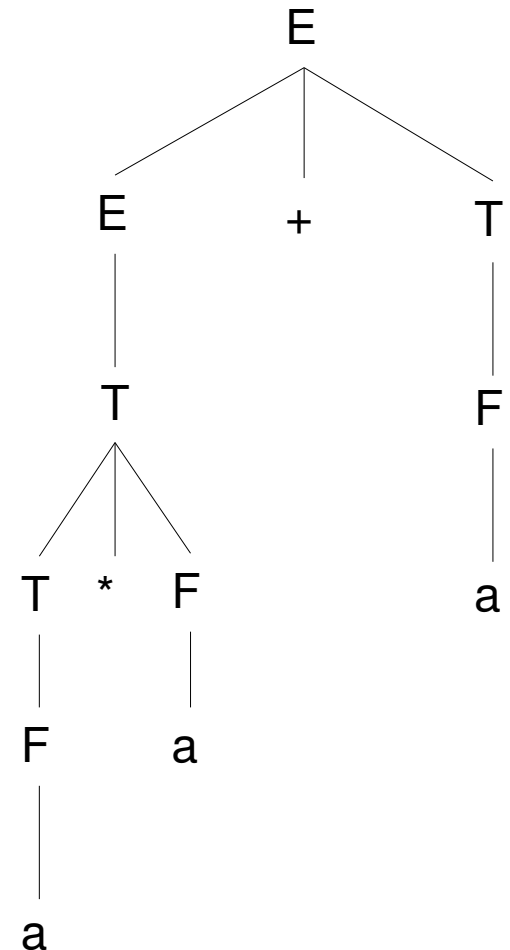
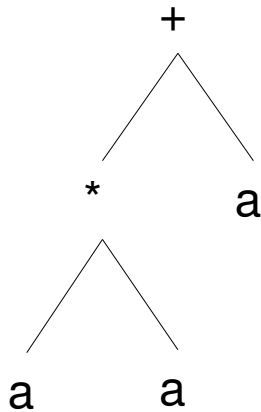
Eindeutige Grammatiken

- Ableitungsbaum für a^*a+a
- Regelmenge P:
 - $F \rightarrow a \mid (E)$
 - $E \rightarrow E+T \mid E-T \mid T$
 - $T \rightarrow T^*F \mid T/F \mid F$



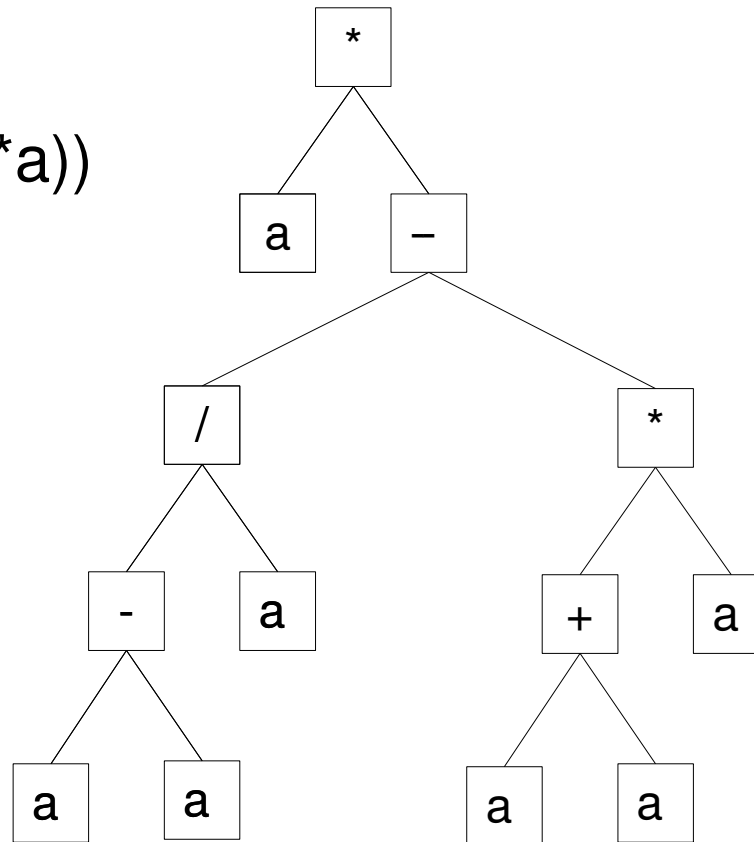
Termbaum (Syntaxbaum)

- Beim Termbaum wandern die Terminalsymbole nach oben und ersetzen die Nichtterminale



Beispiel

- Termbaum für $a^*(((a-a)/a)-((a+a)^*a))$



4.5 Abschlußbeispiel

- Ziel: Erkennen und Übersetzen von mathematischen Ausdrücken
- Gegeben: Grammatik für mathematische Ausdrücke
$$G = (\{ a, +, -, *, /, (,) \}, \{ F, E, T \}, P, E)$$
- Regelmenge P:
 - $F \rightarrow a \mid (E)$
 - $E \rightarrow E+T \mid E-T \mid T$
 - $T \rightarrow T*F \mid T/F \mid F$

Grammatik \rightarrow Kellerautomat

- Gegeben: $G = (\{ a, +, -, *, /, (,) \}, \{ F, E, T \}, P, E)$
 - $F \rightarrow a \mid (E)$
 - $E \rightarrow E+T \mid E-T \mid T$
 - $T \rightarrow T^*F \mid T/F \mid F$
- Konstruktion von K :

$$K = (\{ a, +, -, *, /, (,) \}, \{ q \}, \{ a, +, -, *, /, (,), F, E, T \}, \delta, q, E)$$

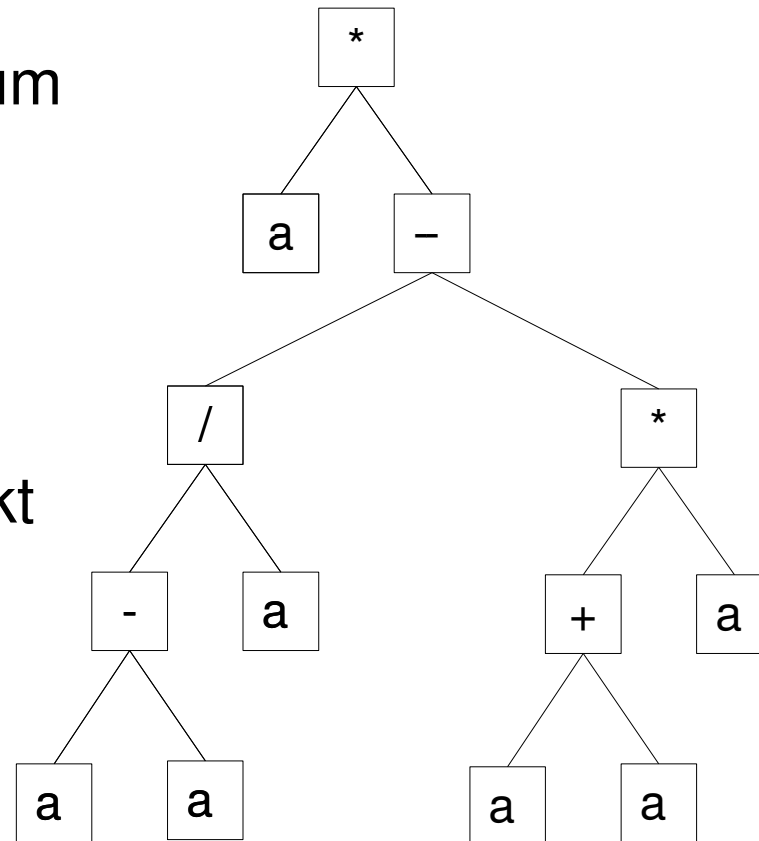
mit $\delta = \{ (q, a, a, q, \varepsilon), (q, +, +, q, \varepsilon), (q, -, -, q, \varepsilon), (q, *, *, q, \varepsilon),$
 $(q, /, /, q, \varepsilon), (q, (, (, q, \varepsilon), (q,),), q, \varepsilon),$
 $(q, \varepsilon, F, q, (E)), (q, \varepsilon, E, q, E+T), (q, \varepsilon, E, q, T),$
 $(q, \varepsilon, T, q, T^*F), (q, \varepsilon, T, q, T/F), (q, \varepsilon, T, q, F) \}$

Ableitungsbaum

- Jede Regel $A \rightarrow X_1 \dots X_n$ ergibt Unterknoten $X_1 \dots X_n$ im Ableitungsbaum
- Jede Regel $A \rightarrow a$ ergibt Unterknoten a
- Ableitungsbaum für $a^*(((a-a)/a)-((a+a)^*a))$
(siehe Blatt)

Termbaum

- Aus dem Ableitungsbaum wird der Termbaum für $a^*(((a-a)/a)-((a+a)^*a))$
- Termbaum ist Startpunkt für Assemblercode-Generierung



Assemblercode-Generierung

- ALU (Tafel)
- Variablen = symb. Namen für Speicherzellen
- Unendlich viele Register
- Zweiadress-Maschine
- Befehlssatz:
 - LOAD R_i, x
 - ADD R_i, R_j
 - SUB R_i, R_j
 - MULT R_i, R_j
 - DIV R_i, R_j
 - STORE R_i, x

Assemblercode-Generierung

- Schreibweisen für Algorithmus:
 - aktKnoten aktueller Knoten
 - vor(k) Vorgänger von k
 - linach(k) linker Nachfolger von k
 - renach(k) rechter Nachfolger von k
 - mark(k) Markierung von k
 - wurz(t) Wurzelknoten des Baumes t

Assemblercode-Generierung

- Algorithmus (siehe Blatt)
- Berechnung für
 - $a_1 = 5$
 - $a_2 = 19$
 - $a_3 = 3$
 - $a_4 = a_5 = a_7 = 2$
 - $a_6 = 1$