Kapitel 2: Formale Sprachen

Gliederung

Anmerkungen

- Es geht immer noch darum
  - formale Sprachen mithilfe von Regeln (Chomsky-Grammatiken) zu beschreiben und
  - zu verstehen, wie man das Wortproblem für so beschriebene Sprache lösen kann.

- Bisher haben wir uns mit regulären Grammatiken (Typ 3) und den durch sie beschriebenen Sprachen (regulären Sprachen) beschäftigt.

- Jetzt schauen wir uns einen „komplizierteren“ Typ von Grammatiken genauer an: die kontextfreien Grammatiken bzw. Typ-2-Grammatiken, sowie die durch sie beschriebenen Sprachen (kontextfreie Sprachen).
  .

Warum machen wir das?

- Es gibt Sprachen, die man nicht mit regulären Grammatiken beschreiben kann, z.B.
  - $L_1 = \{ 0^n 1^n \mid n \in N \}$
  - $L_2 = \{ w \in \Sigma^* \mid w$ enthält genauso viele Nullen wie Einsen $\}$

Wie hatten wir das eingesehen?

- Es geht nicht nur um solche Spielbeispiele: Die meisten Sprachen, die uns als Informatikern in der Praxis begegnen, sind kontextfreie Sprachen (zumindest „in ihrem Kern“), z.B. die Menge aller…
  - Programme einer Programmiersprache, die ein Compiler übersetzen soll;
  - Dokumente, die ein Internet-Browser darstellen soll;
  - Dokumente, die ein Textverarbeitungssystems verarbeiten soll.

- Wir könnten im Prinzip auch *kontextsensitive* Grammatiken (Typ 1) verwenden, um diese Sprachen zu beschreiben. Dann wäre aber unklar, wie man anhand einer solcher Grammatik einen effizienten Lösungsalgorithmus für das Wortproblem konstruiert.

Ein Beispiel: Programmiersprache 1

- Wir schauen uns eine einfache Programmiersprache an, die in der Theoretischen Informatik (genau gesagt in der Berechnungstheorie) eine wichtige Rolle spielt:
  - Für jede berechenbare Funktion (über der Menge der natürlichen Zahlen) kann man ein Programm in dieser Programmiersprache angeben, das diese Funktion berechnet.
- In dieser Programmiersprache gibt es nur die folgenden einfachen Sprachkonstrukte:
  - Variablen (x0,x1,x2, ...)
  - Konstanten (0,1,2, ...)
  - Wertzuweisungen (x0 := x1 + 0, x2:= x2 - 1, ...)
  - While-Schleifen (while x2 ≠ 0 do x0:= x0 + 1; x2 = x2 - 1 end)

Ein Beispiel: Programmiersprache 1 (weiter)

- Das folgende Programm in dieser Programmiersprache addiert zwei natürliche Zahlen (Diese werden den Variablen x1 und x2 vor dem Programmstart zugewiesen; das Ergebnis steht nach Ende der Programmausführung in der Variablen x0.)

```
x0 := x1 + 0;
while x2 ≠ 0 do
            x0 = x0 + 1; x2 = x2 - 1;
end
```

Ein Beispiel: Programmiersprache 1 (weiter)

- Mit der folgenden kontextfreien Grammatik kann man genau alle Programme dieser Programmiersprache erzeugen

- <P> → <A> | <A> ; <P>
- <A> → <V> := <V> + <K> | <V> := <V> − <K> |
  while <V> ≠ 0 do <P> end
- <V> → x<K>
- <K> → 0 | ... | 9 | 1<Z> | ... | 9<Z>
- <Z> → 0 | ... | 9 | <Z> <Z>

Anmerkungen:
Die Variablen haben hier die Form <Buchstabenfolge>.
<P> steht für Programm und ist die Startvariable;
<A> steht für Anweisung
<V> für Variable (im Programm) und <K> für Konstante
<Z> für Ziffernfolge, die auch nach 0 weitergehen darf

- Im wahren Leben …
  - … haben Programmiersprachen (dazu gehören auch C++ und JAVA) oft
    - *keine kontextfreie* Grammatik, mit der man genau alle Programme dieser Programmiersprache erzeugen kann.
  - *Man kann aber* eine kontextfreie Grammatik angeben, für die gilt:
    - Mit dieser Grammatik kann man eine echte Obermenge aller Programme dieser Programmiersprache erzeugen.
    - Man kann sehr effizient überprüfen, ob ein mit dieser Grammatik erzeugtes Wort ein Programm dieser Programmiersprache ist oder nicht.

Auf diesen Aspekt bezog sich der Hinweis, dass die meisten Sprachen, die uns als Informatiker begegnen, im Kern kontextfreie Sprache sind.

Ein Beispiel: Programmiersprache 2

- Anhand der folgenden Programmiersprache kann man den eben diskutierten Aspekt illustrieren.
  - Sie stammt auch aus der theoretischen Informatik und ist genauso leistungsfähig wie die zuvor betrachtete Programmiersprache 1.
  - In dieser Programmiersprache werden (wie in Assembler) die Anweisungen durchnummeriert.
  - Die zugehörigen Nummern werden verwendet, um die Programmabarbeitung zu steuern.

Ein Beispiel: Programmiersprache 2 (weiter)

- Das folgende Programm dieser Programmsprache kann benutzt werden, um zwei Zahlen zu addieren:

```
(a0): x0 := x1 + 0
(a1): if x2 = 0 goto (a5)
(a2): x0 := x0 + 1
(a3): x2 := x2 - 1
(a4): goto (a1)
(a5): end
```

Ein Beispiel: Programmiersprache 2 (weiter)

- Eine geeignete Obermenge aller Programme dieser Programmiersprache kann man mit folgender kontextfreien Grammatik erzeugen:

- <P> → <P′> <E>
- <E> → <AN>: end
- <P′> → <A> | <A> <P′>
- <A> → <AN>: <V> := <V> + <K> | <AN>: <V> := <V> − <K>
  | <AN>: goto <AN> | <AN>: if <V> = 0 goto <AN>
- <AN> → (a<K>)
- <V> → x<K>
- <K> → 0 | ... | 9 | 1<Z> | … | 9<Z>
- <Z> → 0 | ... | 9 | <Z><Z>

Ein Beispiel: Programmiersprache 2 (weiter)

- Mit der angegebenen kontextfreien Grammatik kann man aber auch folgendes Wort erzeugen:

```
(a3): x0 := x1
(a1): if x2 = 0 goto (a27)
(a0): x0 := x0 + 1
(a3): x2 := x2 - 1
(a4): goto (a0)
(a5): ende
```

- Dieses Wort ist kein „korrektes“ Programm:

  Allgemein sollten
  - die Anweisungen in aufsteigender Reihenfolge durchnummeriert sein, insbesondere soll es keine 2 Anweisungen mit gleicher Nummer geben,
  - Alle „Nummern“ hinter den *goto*’s als Anweisungsnummer vorkommen.

Anmerkungen

- Mit ähnlichen Problemen ist man konfrontiert, wenn man alle C++- bzw. Java-Programme mit einer kontextfreien Grammatik beschreiben will
- Diese Probleme ergeben sich, weil dort Variablen und Funktionen vor ihrer Verwendung *deklariert* worden sein müssen und nur typ- bzw. signaturkonform verwendet werden dürfen.

Vor der eigentlichen Übersetzung wird deshalb wie folgt vorgegangen:

- *Es wird geprüft, ob das „Programm“ mit Hilfe der zugrunde liegenden kontextfreien Grammatik erzeugt werden kann.*
- *Es werden alle verwendeten Variablen und Funktionen bestimmt.*
- *Es finden Prüfungen zur Typ- und Signaturkonformität statt.*

- Zur Erinnerung: Begriff kontextfreie Grammatik (Typ-2-Grammatik)

  - Es sei G = [Σ,V,S,R] eine Chomsky-Grammatik

G heißt kontextfreie Grammatik, falls für alle Regeln $l \rightarrow r$ von G gilt:

- $l \in V$
- $r \neq \varepsilon$ (Die Regeln sind also nicht verkürzend.)

Eine Sprache L heißt kontextfreie Sprache, falls es eine kontextfreie Grammatik G mit $L(G) = L \setminus \{ \varepsilon \}$ gibt.

Beispiel

- Palindrome ergeben von vorn und hinten gelesen dieselbe Zeichenkette.

  Natürlichsprachl. Beispiele: Reliefpfeiler
  Madam, I‘m Adam. (d.h. madamimadam)
  Trug Tim eine so helle Hose nie mit Gurt?
  Engage le jeu que je le gagne.
  たけやがやけた (Die Bambushütte brannte.)
  176-671 ()

- Sprache L aller Zeichenketten, die nur aus 0‘en und 1‘en bestehen und Palindrome sind.

  $\Sigma = \{ 0,1 \}$
  $V = \{ S \}$
  S

  $S \rightarrow 0 \mid 1 \mid 00 \mid 11$
  $S \rightarrow 0S0 \mid 1S1$

- zweites Beispiel

  - $\Sigma = \{ a,b,c \}$
  - $L = \{ a^i \circ b \circ c^k \mid i,k \in N \text{ und } i \leq k \}$
  - eine induktive Definition der Sprache L:
    - Das Wort abc gehört zur Sprache L.
    - Es sei w ein Wort aus L. Dann gehören auch die folgenden zwei Wörter zur Sprache L: $a \circ w \circ c$ und $w \circ c$.
  - eine kontextfreie Grammatik für die Sprache L, mit der Startvariablen S und einer weiteren Variablen A:

$$S \rightarrow A \mid Sc$$
$$A \rightarrow abc \mid aAc$$

- drittes Beispiel

  - $L = \{ 0^i 1^j 0^k \mid i, j, k \geq 1 \text{ und } (i = j \text{ oder } j = k) \}$

$\Sigma = \{ 0,1 \}$
$V = \{ S,A,B,C \}$
S

$S \rightarrow AB \mid CA$
$A \rightarrow 0 \mid 0A$
$B \rightarrow 10 \mid 1B0$
$C \rightarrow 01 \mid 0C1$

- .. und nun zum algorithmischen Aspekt

  - Sei $L \subseteq \Sigma^*$ mit $\varepsilon \notin L$,
  - und sei $G = [\Sigma, V, S, R]$ eine kontextfreie Grammatik mit $L(G) = L$.

  - Wir suchen einen Algorithmus, mit dem man für jedes Wort $w \in \Sigma^*$ entscheiden kann, ob $w \in L$ gilt.
    Und zwar soll er **schneller** sein als die bereits kennengelernte systematische Produktion aller Wörter bis zur Länge von w.

  - Dazu benötigen wir zwei neue Begriffe:
    - Ableitungsbaum und
    - Chomsky-Normalform.

... um den Fall, dass die Sprache L das leere Wort enthält, müssen wir uns keine Sorgen machen.

- zentraler Begriff: Ableitungsbaum

  - Es sei $G = [\Sigma,V,S,R]$ eine kontextfreie Grammatik.
  - Es sei $w \in L(G)$.

Ein Ableitungsbaum für w ist ein geordneter Baum, für den gilt:

- Seine **Wurzel** ist mit S markiert.
- Jeder **innere Knoten** (Elternknoten, Nicht-Blatt) ist mit einer Variablen $A \in V$ markiert, und
  seine Kinderknoten ergeben von links nach rechts gelesen die rechte Seite einer Regel zum Ersetzen von A.
- Die **Blätter** sind mit Buchstaben $x \in \Sigma$ markiert und ergeben – von links nach rechts gelesen – die Zeichenkette w.

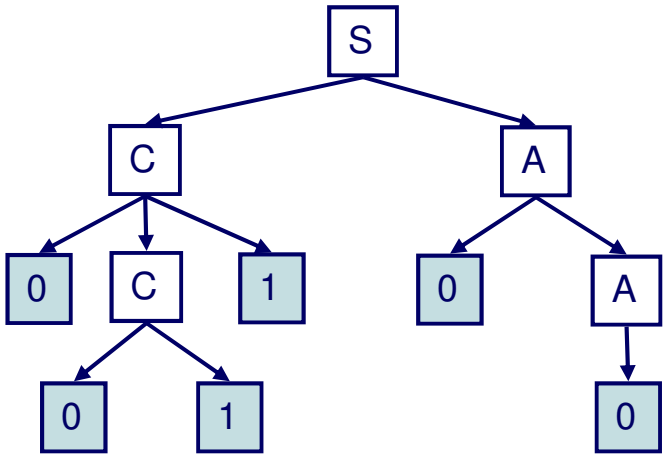
Beobachtung (Teil 1)

- Ist G = [Σ,V,S,R] eine kontextfreie Grammatik,
- w ∈ L(G) und B ein Ableitungsbaum für w,

$S \to AB \mid CA$
$A \to 0 \mid 0A$
$B \to 10 \mid 1B0$
$C \to 01 \mid 0C1$

... so „finden sich“ in B oft mehrere Ableitungen des Wortes w.

w = 001100



$S \to_G \mathbf{C}A \to_G 0\mathbf{C}1A \to_G 0011\mathbf{A} \to_G 00110\mathbf{A} \to_G 001100$

$S \to_G \mathbf{C}A \to_G 0C1\mathbf{A} \to_G 0C10\mathbf{A} \to_G 0\mathbf{C}100 \to_G 001100$

...

$S \to_G C\mathbf{A} \to_G C0\mathbf{A} \to_G \mathbf{C}00 \to_G 0\mathbf{C}100 \to_G 001100$

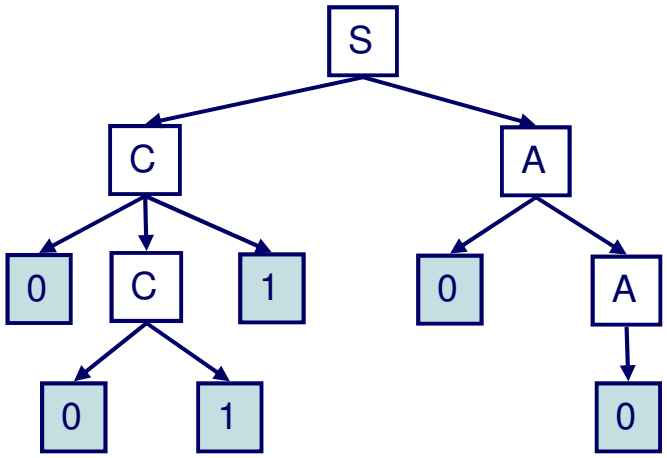
Beobachtung (Teil 2)

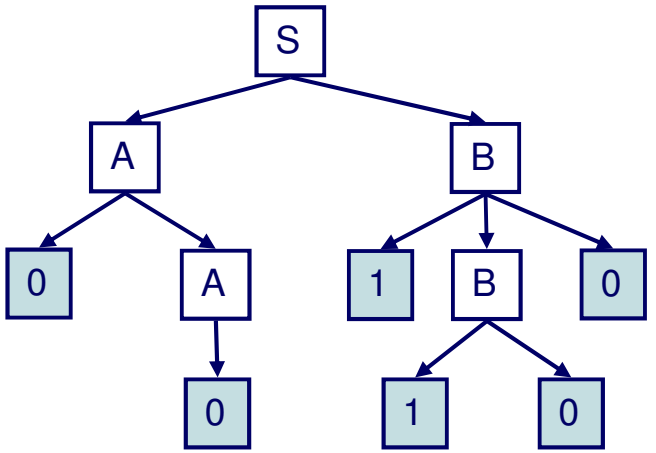
- Ist G = [Σ,V,S,R] eine kontextfreie Grammatik,
- und w ∈ L(G),

$S \rightarrow AB \mid CA$
$A \rightarrow 0 \mid 0A$
$B \rightarrow 10 \mid 1B0$
$C \rightarrow 01 \mid 0C1$

... so „finden sich“ oft mehrere Ableitungsbäume des Wortes w.

w = 001100





Beobachtung (Teil 1+2)

- Ist G = [Σ,V,S,R] eine kontextfreie Grammatik,
- w ∈ L(G) und B ein Ableitungsbaum für w,

- *so gibt es evtl. weitere Ableitungsbäume für w, und*
- *es „finden sich“ in B oft mehrere Ableitungen des Wortes w.*

Aber:

- *Zu jedem Ableitungsbaum „findet sich“ eine eindeutig festgelegte Linksableitung von w (Bildungsregel: es wird jeweils die am weitesten links stehende Variable ersetzt).*
- *Zu jedem Ableitungsbaum „findet sich“ eine eindeutig festgelegte Rechtsableitung von w (Bildungsregel: es wird jeweils die am weitesten rechts stehende Variable ersetzt).*

Rechts-/Linksableitung

- Um herauszukommen, ob ein gegebenes Wort w aus der Startvariablen einer gegebenen kontextfreien Grammatik G ableitbar ist (d.h. zur Sprache L(G) gehört), versucht man
  - eine Rechts- bzw. Linksableitung für das Wort w zu konstruieren

  oder
  - nachzuweisen, dass es keine Links- bzw. Rechtsableitung für das Wort w gibt.

Das geht sehr einfach, wenn die gegebene Grammatik bestimmte günstige Eigenschaften hat (nämlich in Chomsky-Normalform ist).

- Chomsky-Normalform für kontextfreie Grammatik

  - Es sei $G = [\Sigma,V,S,R]$ eine kontextfreie Grammatik.

  G ist in Chomsky-Normalform, falls für alle Regeln $l \to r$ von G gilt:

  - $l \in V$
  - $r = x$ mit $x \in \Sigma$ oder $r = AB$ mit $A,B \in V$

- Das bedeutet keine Einschränkung:

  Es sei H eine kontextfreie Grammatik. Dann gibt es eine Grammatik $G = [\Sigma,V,S,R]$ in Chomsky-Normalform, so dass $L(G) = L(H)$ gilt.

  … und das zeigen wir noch – später.

Vorteile von Grammatiken in Chomsky-Normalform (Teil 1)

- Es sei G = [Σ,V,S,R] eine Grammatik in Chomsky-Normalform.
- Es sei w ∈ L(G) mit | w | = n .

Jede Ableitung des Worts w benötigt genau 2n-1 viele Ableitungsschritte.

Hintergrund

Jeder Ableitungsbaum für w ist – mit Ausnahme der letzten Ebene – ein Binärbaum.

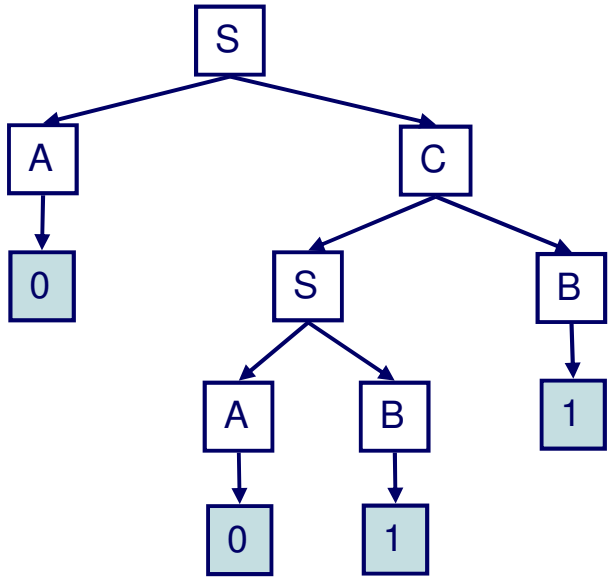
Induktionsbeweis → Ein Binärbaum mit n Blättern hat n-1 viele innere Knoten (Nicht-Blätter)

- Beispiel

$\Sigma = \{ 0,1 \}$
$V = \{ S,A,B,C \}$
S

$w = 0011$

$S \rightarrow AC$
$S \rightarrow AB$
$C \rightarrow SB$
$A \rightarrow 0$
$B \rightarrow 1$



Vorteile von Grammatiken in Chomsky-Normalform (Teil 2)

- Es sei $G = [\Sigma,V,S,R]$ eine Grammatik in Chomsky-Normalform.
- Es sei $w \in \Sigma^*$ mit $w = x_1...x_n$. $(n = |w|)$
- Es sei $A \in V$.

Das Wort w ist aus der Variablen A genau dann ableitbar (d.h. $A \rightarrow_G^* w$), wenn einer der folgenden beiden Fälle eintritt:

Fall 1: Ist $n = 1$, muss es die Regel $A \rightarrow x_1$ in G geben

Fall 2: Ist $n > 1$, muss es ein $z \in \{ 1,...,n-1 \}$, Variablen $B,C \in V$ und eine Regel $A \rightarrow BC$ in G geben, so dass gilt:

$$B \rightarrow_G^* x_1...x_z$$
$$C \rightarrow_G^* x_{z+1}...x_n$$

Dies kann man in einem effizienten Algorithmus zur Lösung des Wortproblems für L(G) ausnutzen: Anstatt alle möglichen Wörter von S aus abzuleiten, fragt man nach den möglichen Herkunfts-Variablen von Teilwörtern aufsteigender Länge.

Cocke-Younger-Kasami Algorithmus

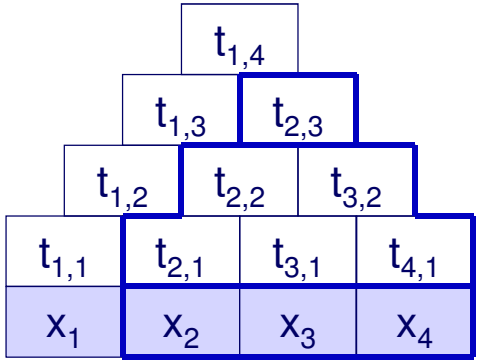
- Es sei $G = [\Sigma,V,S,R]$ eine Grammatik in Chomsky-Normalform.
- Es sei $w \in \Sigma^*$ mit $w = x_1 \ldots x_n$.

verwendete Datenstruktur:

- „halbe Tabelle“ der Größe $n \times n$, z.B. als „Pyramide“ angeordnet
- Zelle $t_{i,k}$ enthält Informationen, die das Teilwort $x_i x_{i+1} \ldots x_{i+k-1}$ betreffen, welches in w an der Position i beginnt und aus k Zeichen besteht:

  $t_{i,k}$ enthält die Variable $A \in V$ gdw. $A \rightarrow_{G^*} x_i x_{i+1} \ldots x_{i+k-1}$ gilt.

Schematisches Beispiel: $t_{2,3} \rightarrow x_2\, x_3\, x_4$



Cocke-Younger-Kasami Algorithmus (weiter)

- Es sei $G = [\Sigma,V,S,R]$ eine Grammatik in Chomsky-Normalform.
- Es sei $w \in \Sigma^*$ mit $w = x_1 ... x_n$.

verwendete Datenstruktur:

- „halbe Tabelle“ der Größe $n \times n$
- Zelle $t_{i,k}$ enthält Informationen, die das Teilwort $x_i x_{i+1} ... x_{i+k-1}$ betreffen, welches in w an der Position i beginnt und aus k Zeichen besteht:
  $t_{i,k}$ enthält die Variable $A \in V$, falls $A \rightarrow_{G^*} x_i x_{i+1} ... x_{i+k-1}$ gilt.

Berechnungsvorschrift:

(1) Wenn $k = 1$, so ist $A \in t_{i,k}$, falls es die Regel $A \rightarrow x_i$ in R gibt.

(2) Wenn $k > 1$, so ist $A \in t_{i,k}$, falls es eine Regel $A \rightarrow BC$ in R gibt und es ein $z \in \{ 1,...,k-1 \}$ mit $B \in t_{i,z}$ und $C \in t_{i+z,k-z}$ gibt.

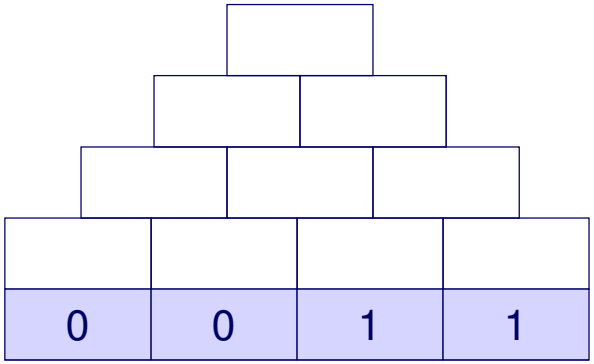
Nach der Berechnung enthält jede Zelle eine (evtl. leere) Menge von Variablen $A \in V$.

CYK-Beispiel

$\Sigma = \{ 0,1 \}$
$V = \{ S,A,B,C \}$
S

$w = 0011$

$S \rightarrow AC$
$S \rightarrow AB$
$C \rightarrow SB$
$A \rightarrow 0$
$B \rightarrow 1$

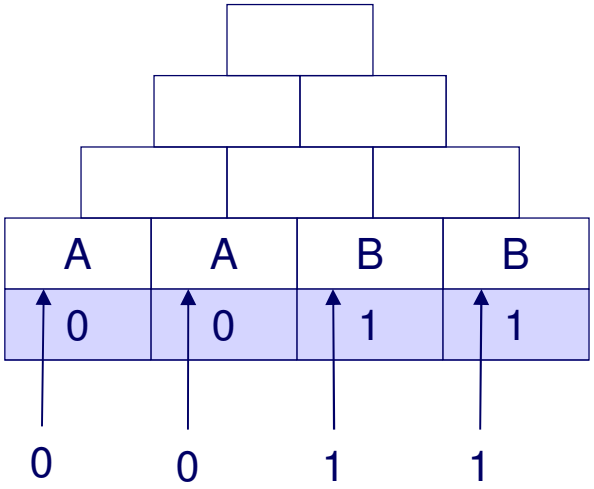


CYK-Beispiel (weiter)

$\Sigma = \{ 0,1 \}$
$V = \{ S,A,B,C \}$
S

$w = 0011$

$S \rightarrow AC$
$S \rightarrow AB$
$C \rightarrow SB$
$A \rightarrow 0$
$B \rightarrow 1$

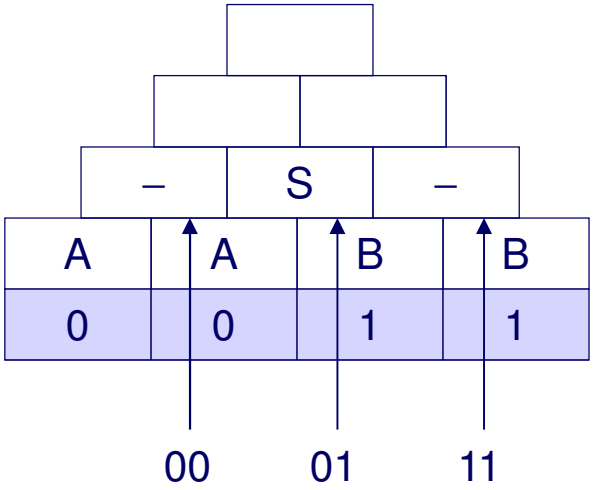


CYK-Beispiel (weiter)

$\Sigma = \{ 0,1 \}$
$V = \{ S,A,B,C \}$
S

$w = 0011$

$S \rightarrow AC$
$S \rightarrow AB$
$C \rightarrow SB$
$A \rightarrow 0$
$B \rightarrow 1$

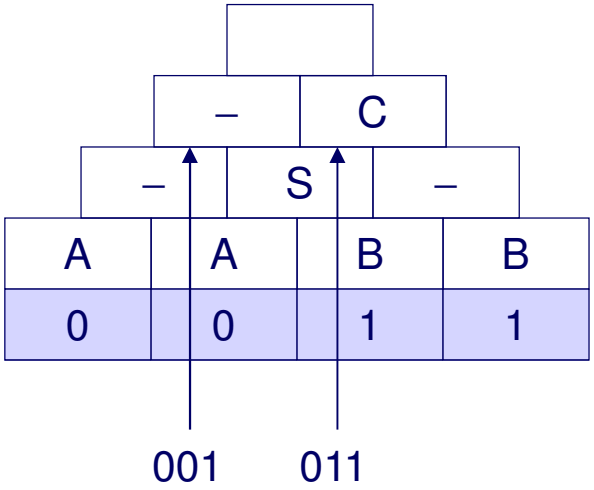


CYK-Beispiel (weiter)

$\Sigma = \{ 0,1 \}$
$V = \{ S,A,B,C \}$
S

$w = 0011$

$S \rightarrow AC$
$S \rightarrow AB$
$C \rightarrow SB$
$A \rightarrow 0$
$B \rightarrow 1$



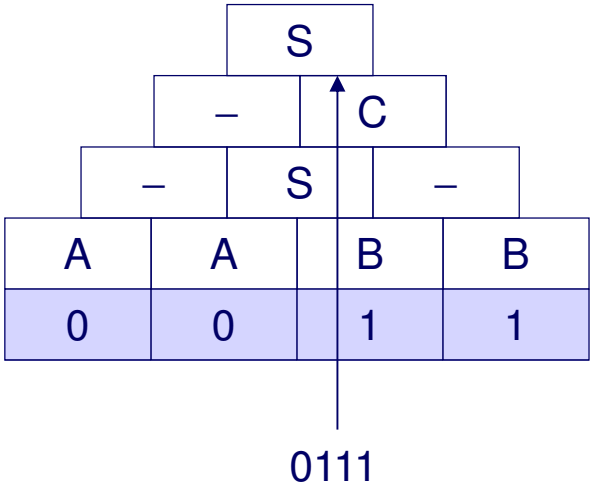
CYK-Beispiel (weiter)

*vgl. auch separates **animiertes Beispiel***

$\Sigma = \{ 0,1 \}$
$V = \{ S,A,B,C \}$
S

$w = 0011$

$S \rightarrow AC$
$S \rightarrow AB$
$C \rightarrow SB$
$A \rightarrow 0$
$B \rightarrow 1$



$S \rightarrow_G AC \rightarrow_G 0C \rightarrow_G 0SB \rightarrow_G 0ABB \rightarrow_G 00BB \rightarrow_G 001B \rightarrow_G 0011$

Cocke-Younger-Kasami Algorithmus in Pseudoprogrammiersprache

- Es sei $G = [\Sigma,V,S,R]$ eine Grammatik in Chomsky-Normalform.
- Es sei $w \in \Sigma^*$ mit $w = x_1 ... x_n$.

- Für i = 1,...,n: Setze $t_{i,1} = \{ A \mid A \rightarrow x_i \}$
- Für k = 2,...,n:
  für i = 1,...,n - k + 1:
  setze $t_{i,k} = \emptyset$;
  für z = 1,...,k - 1:
  füge ein $A \in V$ zu $t_{i,z}$ hinzu, falls es $B,C \in V$ mit folgenden Eigenschaften gibt:
  - $B \in t_{i,z}$
  - $C \in t_{i+z,k-z}$
  - die Regel $A \rightarrow BC$ gehört zu R
- Falls $S \in t_{1,n}$, gib „1“ aus; sonst gib „0“ aus.

Cocke-Younger-Kasami Algorithmus

- ... löst das Wortproblem für eine kontextfreie Sprache L = L(G), falls die Grammatik G in Chomsky-Normalform ist.
- Der CYK-Algorithmus benötigt $O(n^3)$ viele Schritte, um für ein Wort w der Länge n zu entscheiden, ob $w \in L(G)$ oder $w \notin L(G)$ gilt (Die Größe von G ist in der O-Notation „versteckt“.)

- Für welche Sprachen existiert eine Chomsky-Normalform-Grammatik und wie kommt man zu ihr?

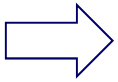
Es sei L eine kontextfreie Sprache. Dann gibt es eine Grammatik $G = [\Sigma,V,S,R]$ in Chomsky-Normalform, so dass $L(G) = L\setminus\{\varepsilon\}$ gilt.

- Ist $G = [\Sigma,V,S,R]$ eine kontextfreie Grammatik, so kann man aus G eine Grammatik $G^\circ = [\Sigma^\circ,V^\circ,S^\circ,R^\circ]$ in Chomsky-Normalform konstruieren, so dass $L(G^\circ) = L(G)$ gilt.

- Welche Regeln passen nicht zur Chomsky-Normalform?

$\Sigma = \{ a,b \}$
$V = \{ S,A,B,C \}$
S

$S \to A$
$S \to AB$
$S \to ABS$
$B \to bb$
$A \to a$



Die rechte Seite …

1. ist nur ein Zeichen, aber keine Konstante, sondern eine Variable; etwa die Regel $S \to A$
2. rechte Seite besteht aus einer Konstanten und noch weiteren Zeichen; etwa die Regel $B \to bb$
3. rechte Seite besteht aus Variablen, mehr als zweien; etwa die Regel $S \to ABS$

*Diese Liste der Problemtypen 1,2,3 ist **vollständig**:*
ohne Regeln dieser drei Arten ist die kontextfreie Grammatik in Chomsky-NF.

Umwandlung in Chomsky-Normalform

- Überführung einer kontextfreien Grammatik in Chomsky-Normalform

Schritt 0:
alle Regeln aus R zunächst in die Menge R° übernehmen

→ Probleme!
Beheben …

Schritt 1a:
alle Zyklen aus Regeln der Form $A \rightarrow B$ eliminieren
(Variablen identifizieren, Regeln der Form $A \rightarrow A$ streichen)

Schritt 1b:
alle Regeln der Form $A \rightarrow B$ modifizieren (B überspringen)

Typ 1

Schritt 2:
alle Regeln, die Konstanten enthalten und nicht die Form $A \rightarrow x$ haben, modifizieren (Zusatzvariablen für Konstanten)

Typ 2

Schritt 3:
alle Regeln, die rechts mehr als 2 Variablen enthalten, modifizieren (Zusatzvariablen für Suffixe)

Typ 3

Umwandlung in Chomsky-Normalform

Anforderungen

- Es ist sicher zu stellen, dass nach jedem Schritt für alle Wörter $w \in \Sigma^*$ gilt
  - $S \rightarrow_G^* w$ gdw. $S \rightarrow_{G^\circ}^* w$

... mit der Grammatik G° kann man nicht mehr und nicht weniger ableiten als mit Regeln der Grammatik G

Schritt 1a: Variablenzyklen beseitigen

- Zyklen innerhalb der Regeln der Form $A \rightarrow B$ identifizieren
- Beteiligte (zyklisch „gleichwertige“) Variablen in allen Regeln durch ein und dieselbe Variable ersetzen.
- Regeln der Form $A \rightarrow A$ streichen.

... solange, bis es keine Zyklen mehr gibt

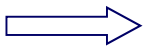
- Illustration (Schritt 1a)

$\Sigma = \Sigma° = \{ a,b \}$
$V = \{ S,A,B,C \}$
S

$V° = \{ S,C \}$
S

$S \to ab$
$S \to aA$
$S \to A$
$A \to B$
$B \to S$
$A \to C$
$A \to aBb$
$C \to abS$

$S \to A \to B \to S$ eliminieren,
„$S = A = B$“ :
Aus A und B wird S.



$S \to ab$
$S \to aS$
$S \to C$
$S \to aSb$
$C \to abS$

Zyklen eliminieren:
- dabei deren Variablen zusammenfassen
- und in allen Regeln die Variablen entsprechend ersetzen.

- Schritt 1b: Regeln „Variable → Variable“ durch Überspringen beseitigen

  - für jede Regel der Form A → B:
    für jede Regel der Form B → r:
    die rechte Seite B in „A → B“
    durch die rechte Seite r aus „B → r“ ersetzen.

... solange, bis es keine Regel der Form A → B mehr gibt

... geht gut wegen des vorherigen Schrittes 1.

Umwandlung in Chomsky-Normalform

- Illustration (Schritt 1b)

$\Sigma = \Sigma^\circ = \{ a,b \}$
$V = V^\circ = \{ S,A \}$
S

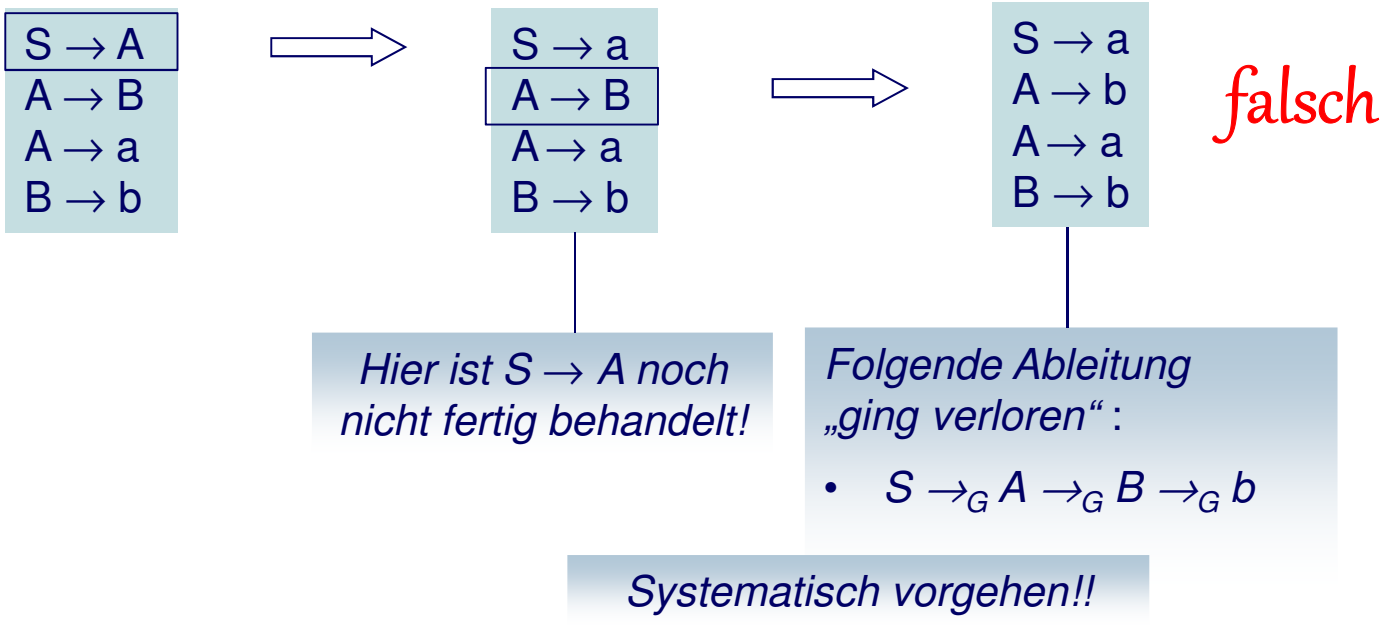
$S \rightarrow ab$ $S \rightarrow bA$ $S \rightarrow A$ $A \rightarrow aSb$	⇒	$S \rightarrow ab$ $S \rightarrow bA$ $S \rightarrow aSb$ $A \rightarrow aSb$

- *in Regeln der Form $A \rightarrow B$ die rechte Seite durch rechte Seiten von Regeln der Form $B \rightarrow r$ ersetzen (aber $B \rightarrow r$ nicht wegwerfen).*

- Illustration (Schritt 1b)

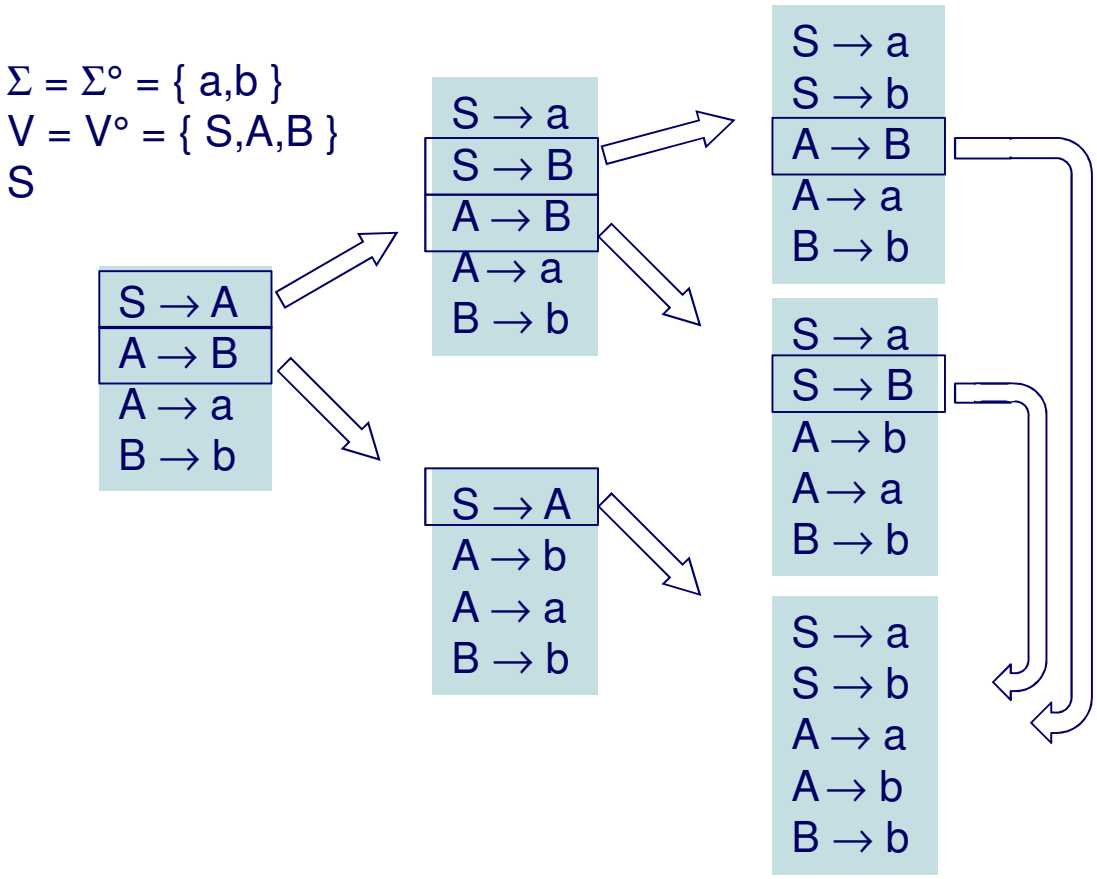
  Fehlerquelle: Mal hier, mal da „ein bisschen ersetzen“.

$\Sigma = \Sigma^\circ = \{ a,b \}$
$V = V^\circ = \{ S,A,B \}$
S



- Illustration (Schritt 1b)

  Systematisches Vorgehen führt (so oder so) zum Ziel.



Umwandlung in Chomsky-Normalform

- Schritt 2: Folgen mit Konstanten mithilfe von Hilfsvariablen beseitigen
  - in den Regeln der Form $A \rightarrow r$ mit $|r| \geq 2$ jede Konstante x durch eine Variable H_x ersetzen und die Regel $H_x \rightarrow x$ aufnehmen

    ... solange, bis alle Regeln der Form $A \rightarrow r$ mit $|r| \geq 2$ keine Konstanten mehr enthalten

- Schritt 3: Variablenfolgen mithilfe von Hilfsvariablen auf Länge 2 stutzen
  - die Regeln der Form $A \rightarrow Br'$ mit $|r'| \geq 2$ durch zwei Regeln $A \rightarrow BV$ und $V \rightarrow r'$ ersetzen. (V ist eine neue Variable.)

    ... solange, bis es keine Regel der Form $A \rightarrow r$ mit $|r| \geq 3$ mehr gibt

Umwandlung in Chomsky-Normalform

- Illustration (Schritte 2 und 3)

*vgl. auch separates **animiertes Beispiel***

$\Sigma = \Sigma^{\circ} = \Sigma^{\circ\circ} = \{ a,b \}$
$V = \{ S \}$
S

$V^{\circ} = \{ S, H_a, H_b \}$
S

$V^{\circ\circ} = \{ S, H_a, H_b, H \}$
S

$S \rightarrow a$
$S \rightarrow aSb$

⇒

$S \rightarrow a$
$S \rightarrow H_aSH_b$
$H_a \rightarrow a$
$H_b \rightarrow b$

⇒

$S \rightarrow a$
$S \rightarrow H_aH$
$H \rightarrow SH_b$
$H_a \rightarrow a$
$H_b \rightarrow b$

Schritt 3: Konstanten in „langen“ rechten Seiten durch Variablen ersetzen

Schritt 4: Regeln mit „langen“ rechten Seiten durch mehrere Regeln ersetzen