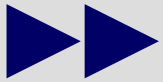
Theoretische Informatik
Kap 1: Formale Sprachen/Automatentheorie

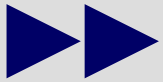


Gliederung der Vorlesung

0. Grundbegriffe
1. Formale Sprachen/Automatentheorie
    - 1.1. Grammatiken
    - 1.2. Reguläre Sprachen
    - **1.3. Kontextfreie Sprachen**

2. Berechnungstheorie
    - 2.1. Entscheidungsprobleme
    - 2.2. Berechenbarkeitsmodelle
    - 2.3. Die Churchsche These
    - 2.4. Unentscheidbarkeit
3. Komplexitätstheorie
    - 3.1. Nicht-deterministische Turing Maschinen
    - 3.1 Komplexitätsmaße
    - 3.2. Das P=NP? Problem

Theoretische Informatik
Kap 1: Formale Sprachen/Automatentheorie



Kontextfreie Sprachen

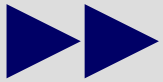
Reguläre Sprachen

- Beschreibung für Bezeichner in Programmiersprachen
- Beschreibung für „wild cards“ in Skriptsprachen
  - ?; [a-z]; [^begin]; *

Kontext-freie Sprachen

- Beschreibung der Syntax von Programmiersprachen
  - → Backus-Naur-Form (BNF)
  - → erweiterte Backus-Naur-Form (EBNF)

Theoretische Informatik
Kap 1: Formale Sprachen/Automatentheorie

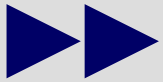


Kontextfreie Sprachen

BNF/EBNF

```
<Bezeichner> ::=  _<Name> | <Name>
<Name>   ::= <Buchstabe> | <Ziffer> | <Name><Buchstabe> |
                  <Name><Ziffer>
<Buchstabe>::= a | ... | z | A | ... | Z
<Ziffer> ::= 0 | ... | 9
```

Theoretische Informatik
Kap 1: Formale Sprachen/Automatentheorie

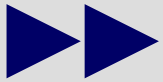


Kontextfreie Sprachen

BNF/EBNF

<Block> ::= <Anweisung> | begin <Anweisung> {<Anweisung>} end
<Anweisung> ::= <Zuweisung> | <Verzweigung> | <Schleife>
<Zuweisung> :: = <Bezeichner> = <Ausdruck>
<Verzweigung> ::= if <Bedingung> <Block> [else <Block>]
<Schleife> := while <Ausdruck> <Block>
...

Theoretische Informatik
Kap 1: Formale Sprachen/Automatentheorie



Kontextfreie Sprachen

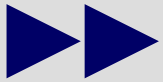
BNF/EBNF und kontextfreie Grammatiken

<Block> ::= <Anweisung> | begin <Anweisung> {<Anweisung>} end
<Anweisung> ::= <Zuweisung> | <Verzweigung> | <Schleife>
<Zuweisung> :: = <Bezeichner> = <Ausdruck>
<Verzweigung> ::= if <Bedingung> <Block> [else <Block>]
<Schleife> := while <Ausdruck> <Block>
...

... Bestimmungsstücke

Σ = { begin, end, if, else, while, ... }
V = { <Block>, <Anweisung>, <Zuweisung>, ..., H }
S = <Block>

Theoretische Informatik
Kap 1: Formale Sprachen/Automatentheorie



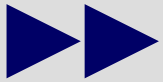
Kontextfreie Sprachen

BNF/EBNF und kontextfreie Grammatiken

<Block> ::= <Anweisung> | begin <Anweisung> {<Anweisung>} end
<Anweisung> ::= <Zuweisung> | <Verzweigung> | <Schleife>
<Zuweisung> :: = <Bezeichner> = <Ausdruck>
<Verzweigung> ::= if <Bedingung> <Block> [else <Block>]
<Schleife> := while <Ausdruck> <Block>
...

... Regeln

<Block> → <Anweisung> | begin H end
H → <Anweisung> | <Anweisung> H
...
<Verzweigung> → if <Bedingung> < Block>
<Verzweigung> → if <Bedingung> < Block> else <Block>



Kontextfreie Sprachen

konzeptioneller Aspekt

Beschreibung der Syntax von Programmiersprachen in EBNF

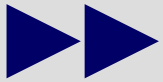
... Beschreibung der Syntax von Programmiersprachen mit Hilfe von kontextfreien Grammatiken

Hintergrund

es sei G eine kontextfreie Grammatik zur Beschreibung der Syntax einer Programmiersprache

jedes Wort in L(G) ist ein syntaktisch korrektes Programm

... und sollte deshalb von einem Compiler korrekt in Maschinen-Code übersetzt werden



Kontextfreie Sprachen

Relevante Fragestellungen

sind kontextfreie Grammatiken als Beschreibungsmittel

a) ausreichend ausdrucksfähig

b) adäquat

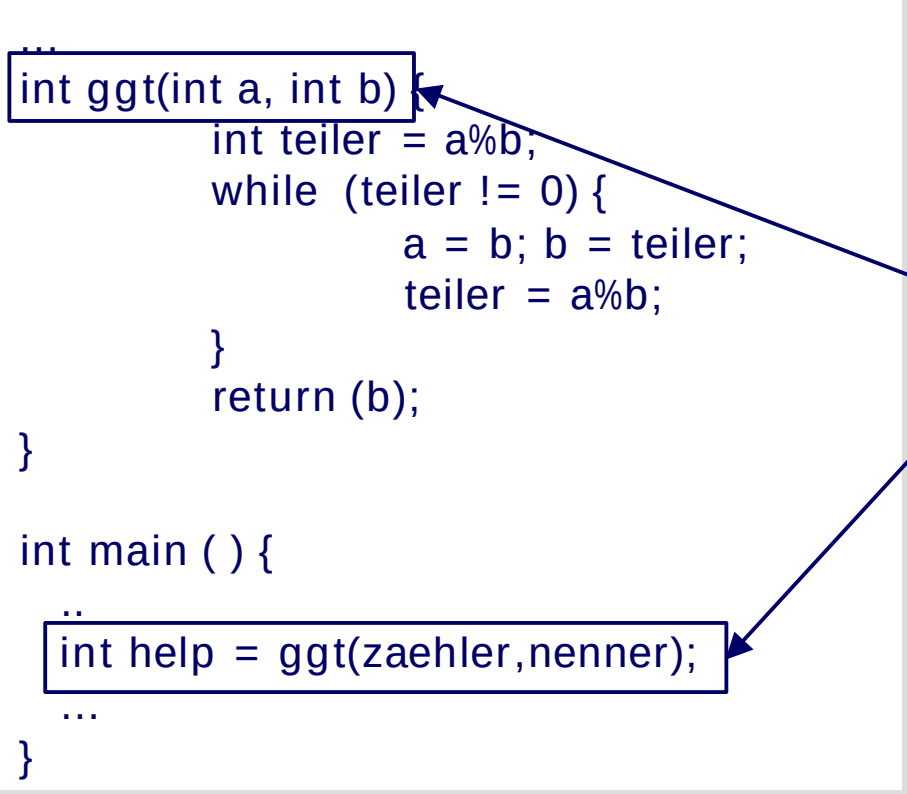
dabei fokussiert b) den algorithmischen Aspekt

- es geht um Algorithmen zur Lösung des Membership-Problems und deren Effizienz

... a) und b) sind im Zusammenhang zu sehen

Theoretische Informatik
Kap 1: Formale Sprachen/Automatentheorie

Kontextfreie Sprachen



solcherart Zusammenhänge lassen sich mit kontextfreien Grammatiken nicht beschreiben

... da hilft man sich auf andere Art und Weise

... für alles andere reichen kontextfreie Grammatiken

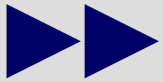
Kontextfreie Sprachen

es sei G = [Σ,V,S,R] eine Grammatik

G heißt kontextfreie Grammatik, falls für alle Regeln (l,r) ∈ R gilt:

- |l| ≤ |r|
- l = x mit x ∈ V

Eine Sprache L ⊆ Σ* heißt kontextfrei, falls es eine kontextfreie Grammatik G mit L(G) = L gibt.



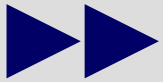
Kontextfreie Sprachen

Beispiel

L = { w$\underline{w}$ | w ∈ { a,b }*, $\underline{w}$ ist die gespiegelte Version von w }

Σ = { a, b }
V = { S, S‘ }
S

S → ε
S → S‘
S‘ → aa | bb
S‘ → aS‘a | bS‘b



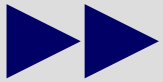
Kontextfreie Sprachen

Beispiel

$L = \{ a^n b^n \mid n \geq 1 \}$

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

$S \rightarrow ab$
$S \rightarrow aSb$



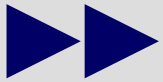
Kontextfreie Sprachen

Beispiel

$L = \{ a^i b^j c^k \mid i, j, k \geq 1, i = j \text{ oder } j = k \}$

$\Sigma = \{ a, b, c \}$
$V = \{ S, A, C, H, H' \}$
S

S → AH | H'C
A → a | aA
H → bc | bHc
C → c | cC
H' → ab | aH'b



Kontextfreie Sprachen

algorithmischer Aspekt

es sei G = [Σ,V,S,R] eine kontextfreie Grammatik

es sei w ein Wort aus Σ*

Frage: Gehört w zu L(G)?

m.a.W. ... kann man w – unter Verwendung der Regeln in R – in endlich vielen Schritten aus S ableiten?

Theoretische Informatik
Kap 1: Formale Sprachen/Automatentheorie

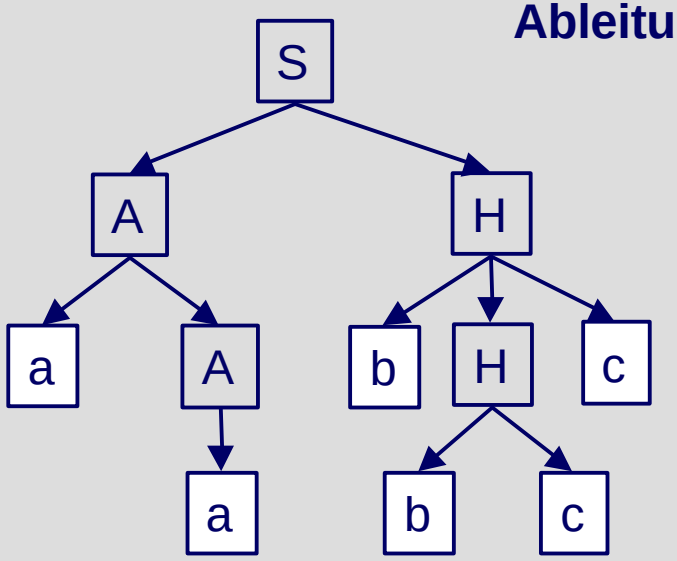
Kontextfreie Sprachen

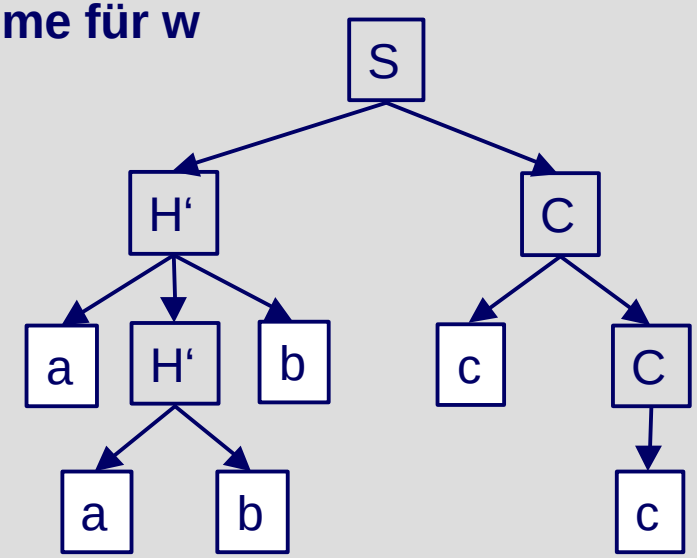
Σ = { a, b, c }
V = { S, A, C, H, H‘ }
S

S → AH | H‘C
A → a | aA
H → bc | bHc
C → c | cC
H‘ → ab | aH‘b

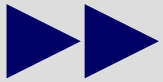
w = aabbcc

Ableitungsbäume für w





Theoretische Informatik
Kap 1: Formale Sprachen/Automatentheorie



Kontextfreie Sprachen

es sei G = [Σ,V,S,R] eine kontextfreie Grammatik

es sei w ein Wort aus L(G)

- für w kann es mehrere Ableitungsbäume geben

Ableitungsbaum für w

- Wurzel mit S markiert
- jeder innere Knoten ist mit einer Variablen markiert
- Söhne eines inneren Knotens mit der Variablen x ergeben von links nach rechts gelesen die rechte Seite einer Regel zum Ersetzen von x
- Blätter sind mit Buchstaben markiert (die Blätter – von links nach rechts gelesen – ergeben w)

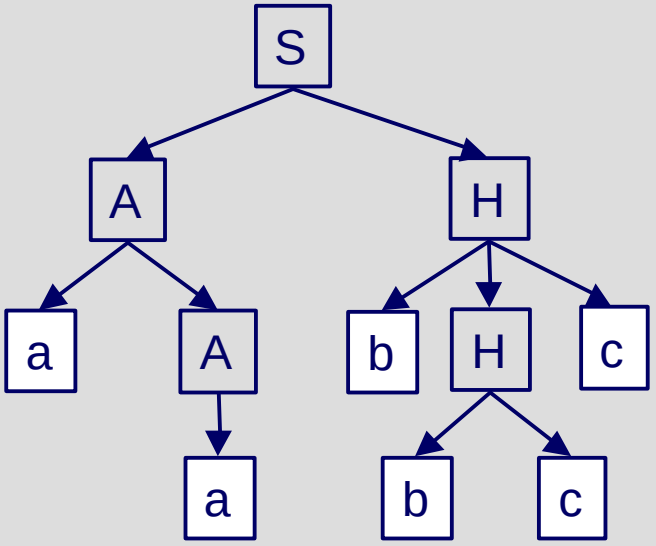
Theoretische Informatik
Kap 1: Formale Sprachen/Automatentheorie

Kontextfreie Sprachen

$\Sigma = \{ a, b, c \}$
$V = \{ S, A, C, H, H' \}$
S

$S \rightarrow AH \mid H'C$
$A \rightarrow a \mid aA$
$H \rightarrow bc \mid bHc$
$C \rightarrow c \mid cC$
$H' \rightarrow ab \mid aH'b$

$w = aabbcc$



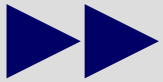
$S \rightarrow_G AH \rightarrow_G aAH \rightarrow_G aaH \rightarrow_G aabHc \rightarrow_G aabbcc$

$S \rightarrow_G AH \rightarrow_G AbHc \rightarrow_G aAbHc \rightarrow_G ... \rightarrow_G aabbcc$

...

$S \rightarrow_G AH \rightarrow_G AbH'c \rightarrow_G Abbcc \rightarrow_G aAbbcc$
$\rightarrow_G aabbcc$

Theoretische Informatik
Kap 1: Formale Sprachen/Automatentheorie



Kontextfreie Sprachen

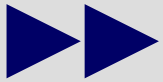
es sei G = [Σ,V,S,R] eine kontext-freie Grammatik

es sei w ein Wort aus L(G)

- für w gibt es im allgemeinen mehrere Ableitungsbäume
- jeder Ableitungsbaum repräsentiert mehrere Ableitungen
- es gibt je Ableitungsbaum eine eindeutig definierte Links- bzw. Rechtsableitung

Links- bzw. Rechtsableitung für w

- ersetze stets die am weitesten links bzw. rechts stehende Variable



Kontextfreie Sprachen

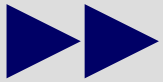
es sei G = [Σ,V,S,R] eine kontext-freie Grammatik

es sei w ein Wort aus Σ*

um herauszubekommen, ob w ∈ L(G) oder w ∉ L(G) gilt, versucht man eine Linksableitung für w zu konstruieren bzw. zu zeigen, daß es keine Linksableitung gibt

... wie man das macht, hängt von der gegebenen Grammatik ab

Theoretische Informatik
Kap 1: Formale Sprachen/Automatentheorie



Kontextfreie Sprachen

Ein Spezialfall (/* der eigentlich keiner ist ... */)

betrachten nur kontextfreie Grammatiken in Chomsky-Normalform

es sei G = [Σ,V,S,R] eine Grammatik

G ist in eine kontextfreie Grammatik in Chomsky-Normalform, falls für alle Regeln (l,r) ∈ R gilt:

- l = x mit x ∈ V
- r = a mit a ∈ Σ oder r = xy mit x, y ∈ V

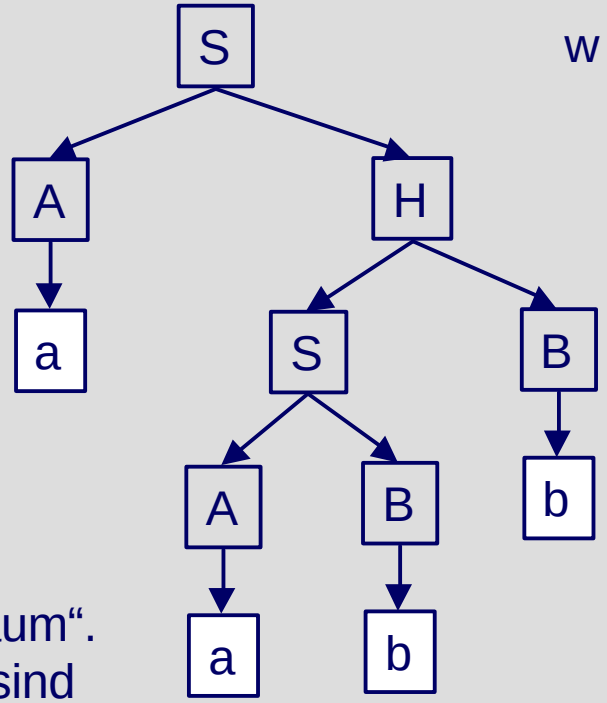
... dann sind die Ableitungsbäume strukturell einfacher

Kontextfreie Sprachen

Beispiel

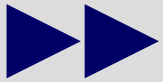
$\Sigma = \{ a, b \}$
$V = \{ S, A, B, H \}$
S

$S \rightarrow AH$
$S \rightarrow AB$
$H \rightarrow SB$
$A \rightarrow a$
$B \rightarrow b$



w = aabb

Jeder Ableitungsbaum ist ein „Binärbaum“.
Um ein Wort der Länge n abzuleiten, sind
|n| -1 + |n| Ableitungsschritte nötig.



Kontextfreie Sprachen

es sei G = [Σ,V,S,R] eine kontextfreie Grammatik in Chomsky-Normalform

es sei w = $a_1 \ldots a_n$ ein Wort aus Σ*

zugrunde liegende Beobachtung

für eine Variable x ∈ V gilt x $\rightarrow^*_G$ $a_1 \ldots a_n$, wenn

- n = 1 gilt und es eine Regel x → a_1 gibt
- n > 1 gilt und es ein i mit 1≤ i ≤ n-1 sowie eine Regel x → yz gibt, so daß gilt:
  - y $\rightarrow^*_G$ $a_1 \ldots a_i$
  - z $\rightarrow^*_G$ $a_{i+1} \ldots a_n$

Theoretische Informatik
Kap 1: Formale Sprachen/Automatentheorie

Kontextfreie Sprachen

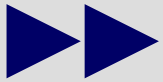
es sei G = [Σ,V,S,R] eine kontextfreie Grammatik in Chomsky-Normalform

es sei w = $a_1 ... a_n$ ein Wort aus Σ*

Bezeichnung: $t[i][k] = \{ x \mid x \in V, x \rightarrow^*_G a_i ... a_{i+k-1} \}$

$x \in t[i][1]$,	falls es eine Regel $x \rightarrow a_i$ gibt
$x \in t[i][k]$ $(k > 1)$,	falls es eine Regel $x \rightarrow yz$ und ein k' mit $0 \leq k' \leq k - 1$ gibt, so daß gilt: - $y \in t[i][k']$ - $z \in t[i+k'][k-k']$

... offenbar gilt $w \in L(G)$ gdw. $S \in t[1][n]$



Kontextfreie Sprachen

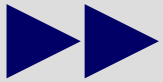
Beispiel

w = aabb

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

S → AH
S → AB
H → SB
A → a
B → b

i \ k	1	2	3	4
1				
2				
3				
4				



Kontextfreie Sprachen

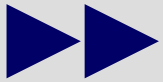
Beispiel

w = aabb

Σ = { a, b }
V = { S, A, B, H }
S

S → AH
S → AB
H → SB
A → a
B → b

i \ k	1	2	3	4
1	A			
2	A			
3	B			
4	B			



Kontextfreie Sprachen

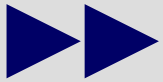
Beispiel

w = aabb

Σ = { a, b }
V = { S, A, B, H }
S

S → AH
S → AB
H → SB
A → a
B → b

i \ k	1	2	3	4
1	A			
2	A	S		
3	B			
4	B			



Kontextfreie Sprachen

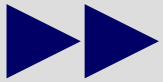
Beispiel

w = aabb

Σ = { a, b }
V = { S, A, B, H }
S

S → AH
S → AB
H → SB
A → a
B → b

i \ k	1	2	3	4
1	A			
2	A	S	H	
3	B			
4	B			



Kontextfreie Sprachen

Beispiel

w = aabb

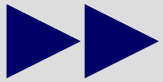
$\Sigma = \{ a, b \}$
$V = \{ S, A, B, H \}$
S

S → AH
S → AB
H → SB
A → a
B → b

i \ k	1	2	3	4
1	A			S
2	A	S	H	
3	B			
4	B			

$S \rightarrow_G AH \rightarrow_G aH \rightarrow_G aSB \rightarrow_G aABB \rightarrow_G aaBB \rightarrow_G aabB \rightarrow_G aabb$

Theoretische Informatik
Kap 1: Formale Sprachen/Automatentheorie



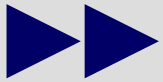
Kontextfreie Sprachen

es sei G = [Σ,V,S,R] eine kontextfreie Grammatik in Chomsky-Normalform

es sei w = $a_1 ... a_n$ ein Wort aus Σ*

CYK-Algorithmus

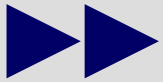
- für i = 1, ..., n: bestimme T[i][1]
- für k = 2, ..., n:
  - für i = 1, ..., n - k + 1:
    - setze T[i][k] = ∅
    - für k‘ = 1, ..., k - 1:
      - füge ein x ∈ V zu T[i][k] hinzu, falls es y, z ∈ V mit folgenden Eigenschaften gibt:
        - y ∈ T[i][k‘]
        - z ∈ T[i+k‘][k-k‘]
        - die Regel x → yz gehört zu R
- falls S ∈ T[1][n], gib „ja“ aus; sonst gib „nein“ aus



Kontextfreie Sprachen

Cocke-Younger-Kasami Algorithmus

- löst das Membership-Problem für kontextfreie Sprachen, für die es eine kontextfreie Grammatik G in Chomsky-Normalform gibt
- der CYK-Algorithmus benötigt $O(n^3)$ viele Schritte um für einem Wort der Länge n zu entscheiden, ob w zu L(G) gehört (/* die Größe von G ist im O „versteckt“ */)
- der CYK- Algorithmus realisiert das Prinzip der dynamischen Programmierung



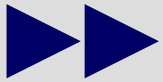
Kontextfreie Sprachen

Der Spezialfall ist kein Spezialfall

Es sei G eine kontextfreie Grammatik. Dann gibt es eine kontextfreie Grammatik G‘ in Chomsky-Normalform, so daß gilt: L(G‘) = L(G).

... der CYK-Algorithmus löst das Membership-Problem für alle kontextfreien Sprachen

... bevor man den CYK-Algorithmus starten kann, ist im allgemeinen ein Art „pre-processing“ erforderlich

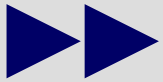


Kontextfreie Sprachen

Beispiel für das „pre-processing“

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

$S \rightarrow ab$
$S \rightarrow aSb$



Kontextfreie Sprachen

Beispiel für das „pre-processing“

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

$S \rightarrow ab$
$S \rightarrow aSb$

$\Sigma = \{ a, b \}$
$V = \{ S, H_a, H_b \}$
S

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

Kontextfreie Sprachen

Beispiel für das „pre-processing“

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

$S \rightarrow ab$
$S \rightarrow aSb$

⇒

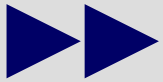
$\Sigma = \{ a, b \}$
$V = \{ S, H_a, H_b \}$
S

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

⇒

$\Sigma = \{ a, b \}$
$V = \{ S, H_a, H_b, H \}$
S

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



Kontextfreie Sprachen

allgemein (/* Teil 1 */)

falls G nur Regeln enthält, deren rechte Seiten aus mehr als zwei Zeichen (Buchstaben oder Variablen) bzw. aus genau einem Buchstaben bestehen, gehe wie folgt vor

Schritt 1:

...

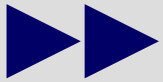
Schritt 2:

...

Schritt 3:

...

Bitte selber ergänzen!



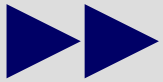
Kontextfreie Sprachen

Beispiel für das „pre-processing“

$\Sigma = \{ a, b \}$
$V = \{ S, S' \}$
S

$S \rightarrow ab$
$S \rightarrow aS'$
$S \rightarrow S'$
$S' \rightarrow aSb$

... Umgang mit Regeln der Form $x \rightarrow y$ für $x,y \in V$



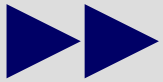
Kontextfreie Sprachen

Beispiel für das „pre-processing“

$\Sigma = \{ a, b \}$
$V = \{ S, S' \}$
S

$S \rightarrow ab$ $S \rightarrow aS'$ $S \rightarrow S'$ $S' \rightarrow aSb$	⇒	$S \rightarrow ab$ $S \rightarrow aS'$ $S \rightarrow aSb$ $S' \rightarrow aSb$

... Umgang mit Regeln der Form $x \rightarrow y$ für $x,y \in V$



Kontextfreie Sprachen

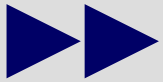
Beispiel für das „pre-processing“

$\Sigma = \{ a, b \}$
$V = \{ S, S' \}$
S

$S \rightarrow ab$ $S \rightarrow aS'$ $S \rightarrow S'$ $S' \rightarrow aSb$	⟹	$S \rightarrow ab$ $S \rightarrow aS'$ $S \rightarrow aSb$ $S' \rightarrow aSb$

und weiter wie zuvor !!!

... Umgang mit Regeln der Form $x \rightarrow y$ für $x,y \in V$



Kontextfreie Sprachen

Beispiel für das „pre-processing“

$\Sigma = \{ a, b \}$
$V = \{ S, S', S'' \}$
S

$S \rightarrow ab$
$S \rightarrow aS'$
$S \rightarrow S'$
$S' \rightarrow S''$
$S'' \rightarrow S$
$S' \rightarrow aSb$

... Umgang mit Regeln der Form $x \rightarrow y$ für $x,y \in V$

Kontextfreie Sprachen

Beispiel für das „pre-processing“

$\Sigma = \{ a, b \}$
$V = \{ S, S', S'' \}$
S

$S \rightarrow ab$	$S \rightarrow ab$
$S \rightarrow aS'$	$S \rightarrow aS'$
$S \rightarrow S'$	$S \rightarrow S''$
$S' \rightarrow S''$	$S \rightarrow aSb$
$S'' \rightarrow S$	$S' \rightarrow S''$
$S' \rightarrow aSb$	$S'' \rightarrow S$
	$S' \rightarrow aSb$

... Umgang mit Regeln der Form $x \rightarrow y$ für $x,y \in V$

Kontextfreie Sprachen

Beispiel für das „pre-processing“

$\Sigma = \{ a, b \}$
$V = \{ S, S', S'' \}$
S

S → ab	S → ab	S → ab
S → aS‘	S → aS‘	S → aS‘
S → S‘	S → S‘‘	S → S
S‘ → S‘‘	S → aSb	S → aSb
S‘‘ → S	S‘ → S‘‘	S‘ → S
S‘ → aSb	S‘‘ → S	S‘‘ → S
	S‘ → aSb	S‘ → aSb

... Umgang mit Regeln der Form $x \rightarrow y$ für $x,y \in V$

Kontextfreie Sprachen

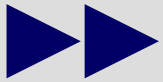
Beispiel für das „pre-processing“

$\Sigma = \{ a, b \}$
$V = \{ S, S', S'' \}$
S

S → ab S → aS‘ S → S‘ S‘ → S‘‘ S‘‘ → S S‘ → aSb	⇒	S → ab S → aS‘ S → S‘‘ S → aSb S‘ → S‘‘ S‘‘ → S S‘ → aSb	⇒	S → ab S → aS‘ S → S S → aSb S‘ → S S‘‘ → S S‘ → aSb	⇒	...

... Umgang mit Regeln der Form $x \rightarrow y$ für $x,y \in V$

... Termination/Aufblähen



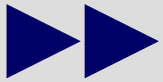
Kontextfreie Sprachen

Beispiel für das „pre-processing“

Σ = { a, b }
V = { S, S‘, S“, S“‘ }
S

S → ab
S → aS‘
S → S‘
S‘ → S‘‘
S‘‘ → S
S‘ → S‘‘‘
S‘ → aSb
S‘‘‘ → abS‘

Regeln der Form x → y separat analysieren ... Zyklen finden



Kontextfreie Sprachen

Beispiel für das „pre-processing“

$\Sigma = \{ a, b \}$
$V = \{ S, S', S'', S''' \}$
S

$S \rightarrow ab$
$S \rightarrow aS'$
$S \rightarrow S'$
$S' \rightarrow S''$
$S'' \rightarrow S$
$S' \rightarrow S'''$
$S' \rightarrow aSb$
$S''' \rightarrow abS'$

$S \rightarrow S'$
$S' \rightarrow S''$
$S'' \rightarrow S$
$S' \rightarrow S'''$

Regeln der Form x → y separat analysieren ... Zyklen finden

Kontextfreie Sprachen

Beispiel für das „pre-processing“

$\Sigma = \{ a, b \}$
$V = \{ S, S‘, S“, S“‘ \}$
S

$\Sigma = \{ a, b \}$
$V = \{ S, S“‘ \}$
S

S → ab
S → aS‘
S → S‘
S‘ → S‘‘
S‘‘ → S
S‘ → S‘‘‘
S‘ → aSb
S‘‘‘ → abS‘

S → S‘
S‘ → S‘‘
S‘‘ → S
S‘ → S‘‘‘

S → ab
S → aS
S → S‘‘‘
S → aSb
S‘‘‘ → abS

Regeln der Form x → y separat analysieren ... Zyklen finden

Theoretische Informatik
Kap 1: Formale Sprachen/Automatentheorie

Kontextfreie Sprachen

Beispiel für das „pre-processing“

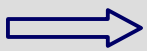
$\Sigma = \{ a, b \}$
$V = \{ S, S‘, S“, S“‘ \}$
S

$\Sigma = \{ a, b \}$
$V = \{ S, S“‘ \}$
S

$S \rightarrow ab$
$S \rightarrow aS‘$
$S \rightarrow S‘$
$S‘ \rightarrow S“$
$S“ \rightarrow S$
$S‘ \rightarrow S“‘$
$S‘ \rightarrow aSb$
$S“‘ \rightarrow abS‘$

$S \rightarrow S‘$
$S‘ \rightarrow S“$
$S“ \rightarrow S$
$S‘ \rightarrow S“‘$

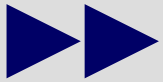
$S \rightarrow ab$
$S \rightarrow aS$
$S \rightarrow S“‘$
$S \rightarrow aSb$
$S“‘ \rightarrow abS$



$S \rightarrow ab$
$S \rightarrow aS$
$S \rightarrow abS$
$S \rightarrow aSb$
$S“‘ \rightarrow abS$

und weiter wie zuvor !!!

Regeln der Form x → y separat analysieren ... Zyklen finden



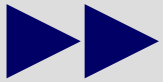
Kontextfreie Sprachen

allgemein (/* Teil 0 */)

falls G Regeln enthält, die auf der rechten Seite nur aus einer Variablen bestehen, so nimm alle Regeln diesen Typs her und gehe wie folgt vor

Schritt 1 (Zyklen finden + eliminieren):
suche Zyklen; falls Zyklus $x_1 \rightarrow x_2$; ...; $x_n \rightarrow x_1$ gefunden, so ersetze x_2, ..., x_n durch x_1

Schritt 2 (Regeln umformen):
numeriere die Variablen so durch, daß in allen Regeln vom Typ $x_i \rightarrow x_K$ stets $i < k$ gilt; ersetze in allen Regeln vom Typ $x_i \rightarrow x_k$ die Variable x_k durch alle rechten Seiten der Regeln vom Typ $x_k \rightarrow r$ mit $|r| > 1$
(/* beginne dabei mit dem größten i */)



Kontextfreie Sprachen

noch einmal allgemein

- zu jeder kontextfreien Grammatik G läßt sich eine kontextfreie Grammatik G‘ in Chomsky-Normalform mit L(G‘) = L(G) konstruieren
- anschließend kann man den CYK-Algorithmus verwenden, um das Membership-Problem für die Sprache L(G‘) (/* und damit auch für L(G) */) zu lösen