

# Grundbegriffe der Theoretischen Informatik

Sommersemester 2018 - Thomas Schwentick

## Teil B: Kontextfreie Sprachen

### 10: Pumping-Lemma, Algorithmen und Abschlusseigenschaften

Version von: 22. Mai 2018 (12:11)

# Einleitung

- In diesem Kapitel werden wir sehen, welche der angenehmen Eigenschaften der regulären Sprachen auch für die kontextfreien Sprachen gelten, und welche nicht
- Methoden zum Nachweis, dass eine Sprache nicht kontextfrei ist:
  - Es gibt ein Pumping-Lemma für kontextfreie Sprachen, das nur wenig komplizierter als das für reguläre Sprachen ist
  - Zum Satz von Nerode korrespondierende Resultate haben wir aber nicht
- Algorithmen:
  - Einige algorithmische Probleme für kontextfreie Sprachen lassen sich effizient lösen
  - Andere gar (!) nicht
- Abschlusseigenschaften:
  - Die Klasse der kontextfreien Sprachen ist unter weniger Operationen abgeschlossen als die Klasse der regulären Sprachen
  - Die Klasse der deterministisch kontextfreien Sprachen hat andere Abschlusseigenschaften als beide Klassen

# Inhalt

## ▷ 10.1 Das Pumping-Lemma für kontextfreie Sprachen

10.2 Algorithmen für kontextfreie Sprachen

10.3 Abschlusseigenschaften der kontextfreien Sprachen

10.4 Deterministische Kellerautomaten

# Pumping-Lemma für kontextfreie Sprachen

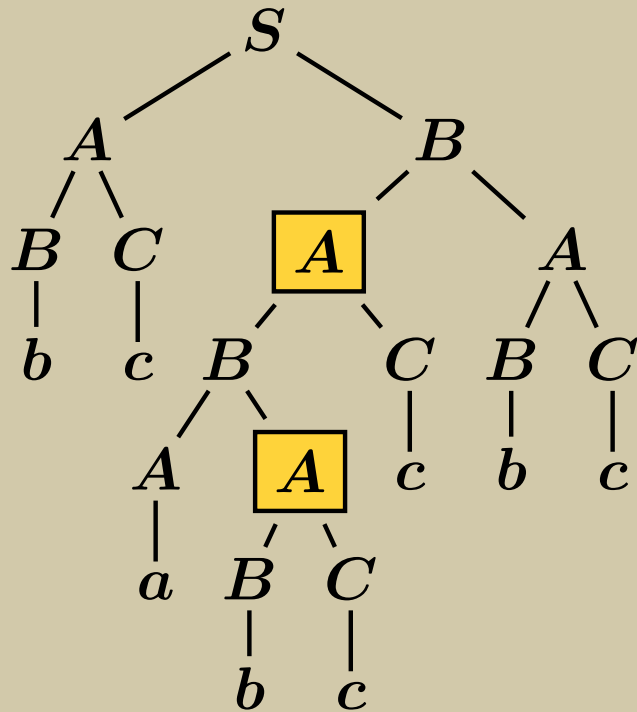
- Zur Erinnerung:
  - Das Pumping-Lemma für reguläre Sprachen beschreibt eine Abschlusseigenschaft, die jede reguläre Sprache hat
  - Es wird benutzt, um zu beweisen, dass eine gegebene Sprache **nicht** regulär ist
- Jetzt betrachten wir eine ähnliche Aussage für kontextfreie Sprachen
  - Der Beweis beruht darauf, dass „gleichartige Teile“ eines Ableitungsbaumes beliebig oft wiederholt werden können
  - Das ist ähnlich wie beim Pumping-Lemma für reguläre Sprachen, nur etwas komplizierter

# Pumping-Lemma: Vorbereitendes Beispiel

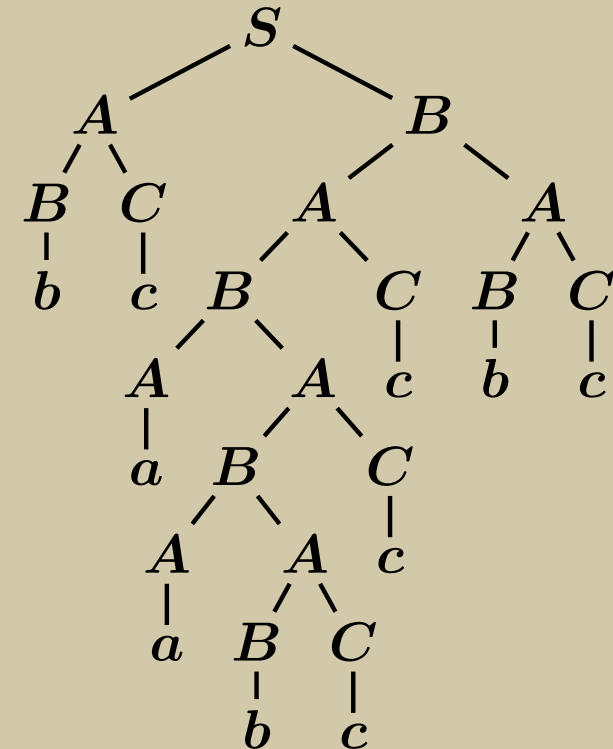
## Beispiel: Grammatik

$$\begin{aligned} S &\rightarrow AB & | & a \\ A &\rightarrow BC & | & a \\ B &\rightarrow AA & | & b \\ C &\rightarrow CB & | & c \end{aligned}$$

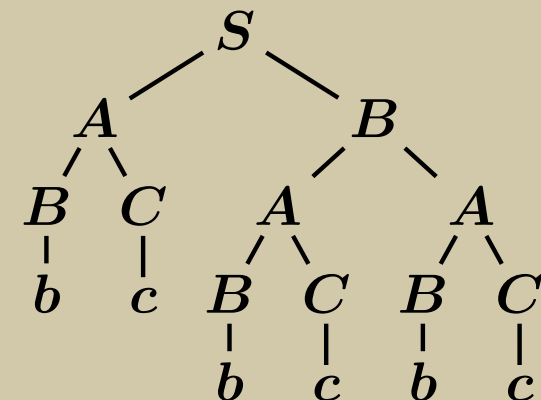
## Beispiel: Ableitung



## Beispiel: „Mittelteil“ wiederholen

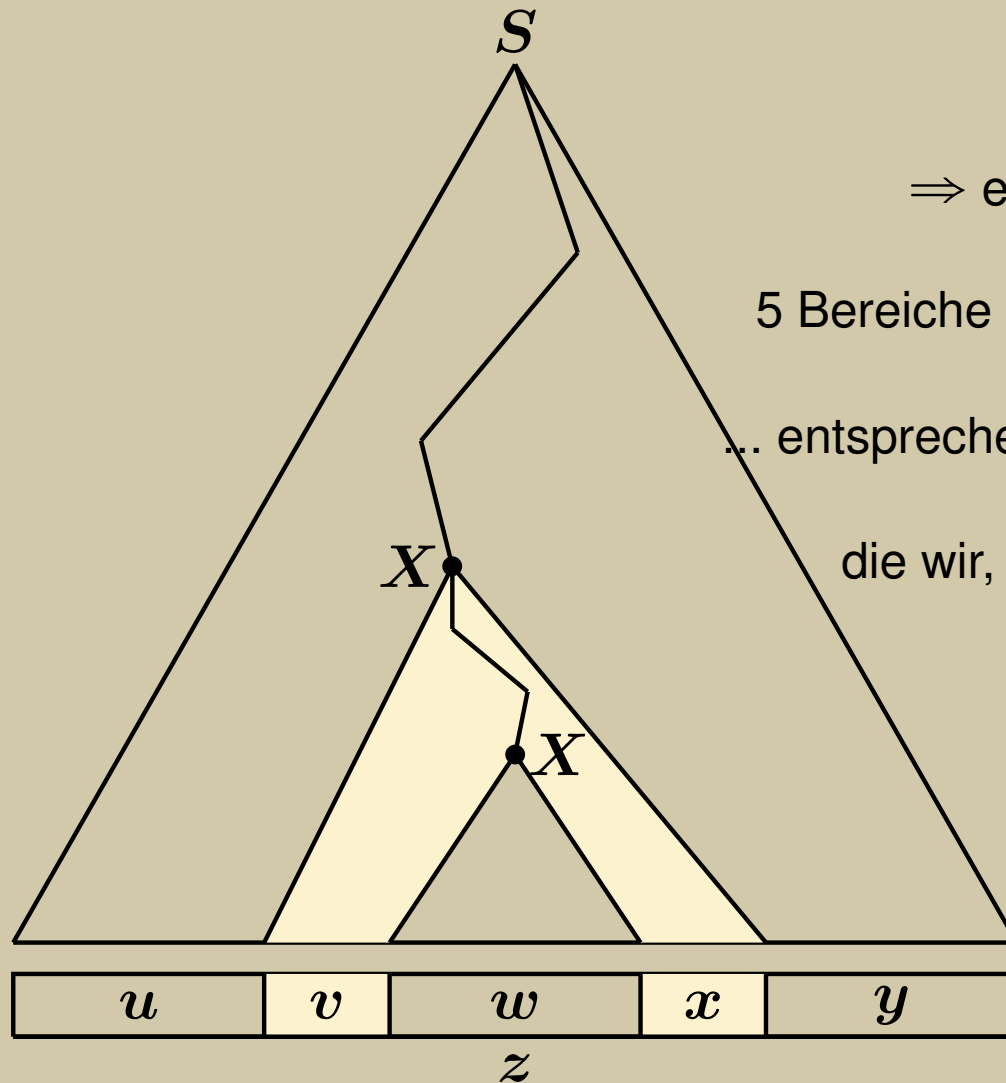


## Beispiel: „Mittelteil“ löschen



# Pumping-Lemma: Illustration (1/2)

Beispiel: Ableitungsbaum  $T$  für  $z$



Weg mit Länge  $> |V|$

$\Rightarrow$  eine Variable  $X$  doppelt

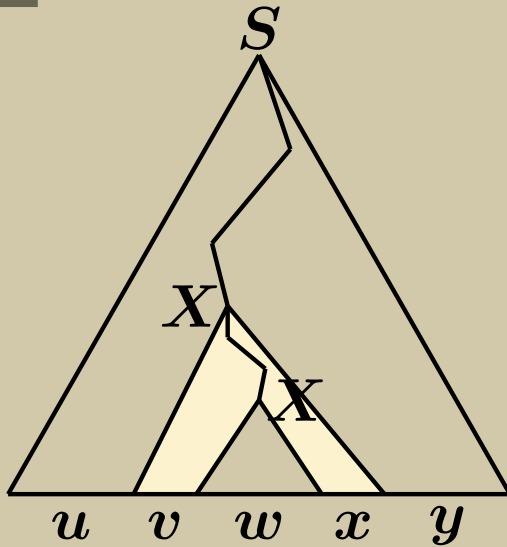
5 Bereiche des Ableitungsbaums...

... entsprechen 5 Abschnitten von  $z$ ,

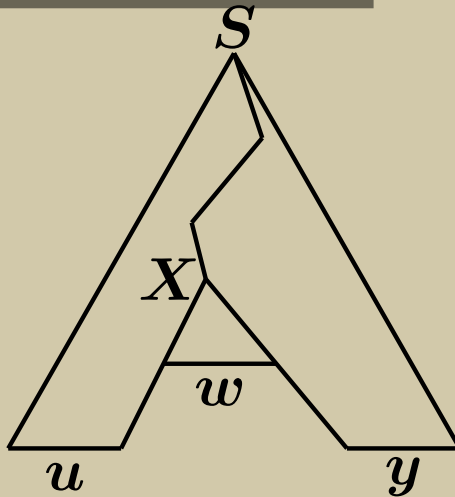
die wir,  $u, v, w, x, y$  nennen

## Pumping-Lemma: Illustration (2/2)

Beispiel:  $T$

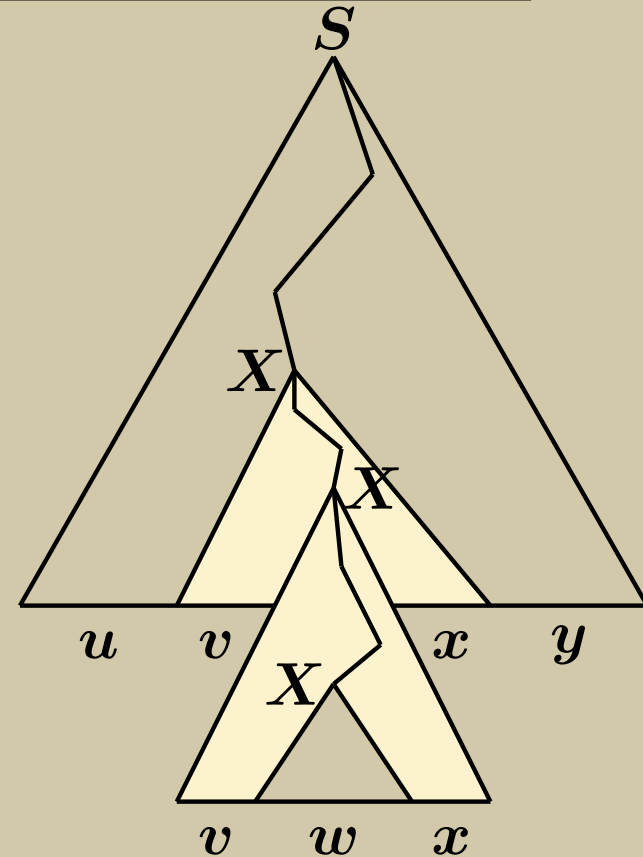


Beispiel:  $v$  und  $x$  entfernen



- Weglassen des „Mittelteils“ ergibt einen Ableitungsbaum für  $uwy$

Beispiel:  $v$  und  $x$  wiederholen



- Wiederholen des „Mittelteils“ ergibt einen Ableitungsbaum für  $uvvwxxy$

# Pumping-Lemma für kontextfreie Sprachen (1/2)

## Satz 10.1 (Pumping-Lemma)

- Ist  $L$  kontextfrei, so gibt es ein  $n \in \mathbb{N}$ , so dass sich jeder String  $z \in L$  mit  $|z| \geq n$  so in  $z = uvwxy$  zerlegen lässt, dass gelten:
  - (1)  $vx \neq \epsilon$ ,
  - (2)  $|vwx| \leq n$ ,
  - (3)  $uv^kwx^ky \in L$ , für alle  $k \geq 0$

## Beweisskizze

- Sei  $L$  eine kontextfreie Sprache
  - Sei  $G = (V, \Sigma, S, P)$  eine Grammatik für  $L$  in Chomsky-Normalform
  - Sei  $m \stackrel{\text{def}}{=} |V|$  die Anzahl der Variablen von  $G$
- Wir setzen  $n \stackrel{\text{def}}{=} 2^{m+1}$
- Sei  $z \in L$  beliebig mit  $|z| \geq n$
- Sei  $T$  ein Ableitungsbaum für  $z$



# Pumping-Lemma für kontextfreie Sprachen (2/2)

## Beweisskizze (Forts.)

- Da  $G$  in CNF ist, hat jeder innere Knoten von  $T$  höchstens zwei Kinder
  - ➔ Die Tiefe von  $T$  ist  $\geq m + 1$
- Sei  $W$  ein Weg maximaler Länge von der Wurzel von  $T$  zu einem Blatt von  $T$ 
  - ➔  $W$  enthält mindestens  $m + 1$  mit Variablen markierte Knoten
  - ➔ Unter den letzten  $m + 1$  dieser Knoten muss eine Variable  $X \in V$  doppelt vorkommen
  - ➔ Das aus dem oberen  $X$  abgeleitete Teilwort hat eine Länge  $\leq 2^{m+1}$ 
    - Und: Der obere  $X$ -Knoten hat 2 Kinder und erzeugt deshalb einen echt größeren String als der untere  $X$ -Knoten
- Also:  $S \Rightarrow^* uXy \Rightarrow^* uvXxy \Rightarrow^* uvwxy = z$  mit
  - $u, v, w, x, y \in \Sigma^*$
  - $v \neq \epsilon$  oder  $x \neq \epsilon$
  - $|vwx| \leq 2^{m+1} = n$

## Beweisskizze (Forts.)

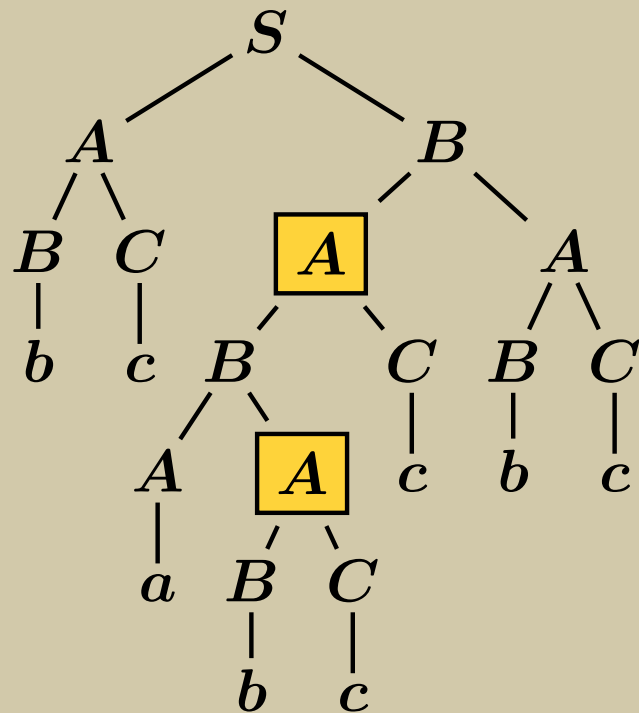
- Idee: der am oberen  $X$  hängende Teilbaum kann an der Stelle des unteren  $X$  eingefügt werden und umgekehrt
  - Das Einfügen des oberen Teilbaums am unteren  $X$  kann wiederholt ausgeführt werden
- Für den formalen Beweis nutzen wir aus, dass gelten:
  - $X \Rightarrow^* vXx$  und
  - $X \Rightarrow^* w$
- Also gilt auch:
  - $S \Rightarrow^* uXy \Rightarrow^* uwy$  und
  - für jedes  $k \geq 1$ :
 
$$S \Rightarrow^* uXy \Rightarrow^* uvXxy \Rightarrow^* \dots \Rightarrow^* uv^kXx^ky \Rightarrow^* uv^kwx^ky$$
- ➔  $uv^kwx^ky \in L$

# Pumping-Lemma: Beispiel

## Beispiel: Grammatik

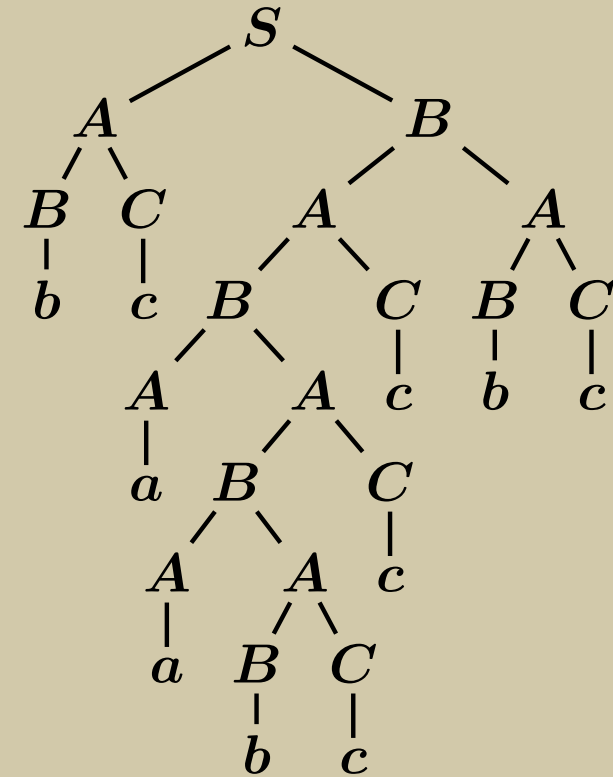
$$S \rightarrow AB \mid a$$
$$A \rightarrow BC \mid a$$
$$B \rightarrow AA \mid b$$
$$C \rightarrow CB \mid c$$

## Beispiel: Ableitung

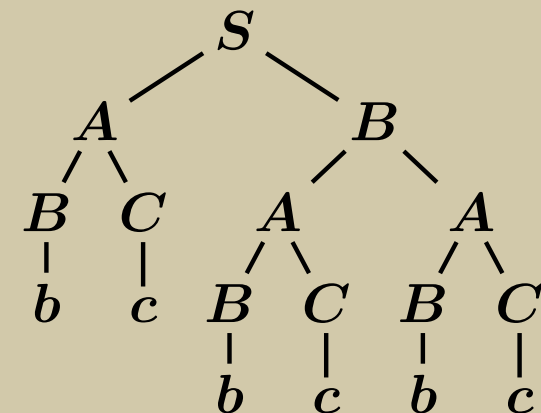


- $u = bc, w = bc, y = bc$
- $v = a, x = c$

## Beispiel: $v$ und $x$ wiederholen



## Beispiel: $v$ und $x$ löschen




# Pumping-Lemma: Anwendung (1/6)

- Wie beim regulären Pumping-Lemma betrachten wir eine Formulierung des Pumping-Lemmas, die sich besser zur Anwendung eignet
- Sie entsteht wieder einfach durch Kontraposition

## Korollar 10.2

- Sei  $L$  eine Sprache
- Angenommen, für jedes  $n > 0$  gibt es einen String  $z \in L$  mit  $|z| \geq n$  so dass für jede Zerlegung  $z = uvwxy$  mit
  - (1)  $vx \neq \epsilon$ ,
  - (2)  $|vwx| \leq n$ ,ein  $k \geq 0$  existiert mit  $uv^kwx^ky \notin L$
- Dann ist  $L$  nicht kontextfrei

- Bei der Anwendung des Pumping-Lemmas muss wieder darauf geachtet werden, an welchen Stellen im Beweis eine Wahl besteht und an welchen Stellen nicht:
  - $n$ : keine Wahl, das folgende Argument muss für beliebige  $n$  funktionieren
  - $z$  kann in Abhängigkeit von  $n$  frei in  $L$  gewählt werden  Aber:  $|z| \geq n$
  - Zerlegung  $z$  in  $uvwxy$ : hier besteht keine Wahl, das Argument muss für beliebige Zerlegungen gelten, die (1) und (2) erfüllen
  - $k$  kann frei gewählt werden
- Das ist analog zum regulären Pumping-Lemma
- Zu beachten:
  - Beim regulären Pumping-Lemma war  $uv$  immer ein Präfix des Strings  $w$
  - Beim kontextfreien Pumping-Lemma kann sich  $vwx$  irgendwo in  $z$  befinden

# Pumping-Lemma: Anwendung (2/6)

## Korollar 10.2

- Sei  $L$  eine Sprache
- Angenommen, für jedes  $n > 0$  gibt es einen String  $z \in L$  mit  $|z| \geq n$  so dass für jede Zerlegung  $z = uvwxy$  mit
  - (1)  $vx \neq \epsilon$ ,
  - (2)  $|vwx| \leq n$ ,ein  $k \geq 0$  existiert mit  $uv^kwx^ky \notin L$
- Dann ist  $L$  nicht kontextfrei

## Proposition 10.3

- Die beiden folgenden Sprachen sind nicht kontextfrei:
  - (a)  $L_{abc} = \{a^m b^m c^m \mid m \geq 1\}$
  - (b)  $L_{\text{doppel}} = \{ww \mid w \in \{a, b\}^*\}$

## Beweis für Proposition 10.3 (a)

- Sei  $n$  beliebig
- Wähle  $z = a^n b^n c^n$
- Sei  $uvwxy$  eine Zerlegung von  $z$  mit  $u, v, w, x, y \in \{a, b, c\}^*$ , die (1) und (2) erfüllt
- Wegen (2) kann  $vx$  nicht sowohl  $a$  als auch  $c$  enthalten
  - ➔  $\#_a(uwy) = \#_a(z)$  oder  $\#_c(uwy) = \#_c(z)$
  - ➔  $uwy \notin L_{abc}$ , da in  $uwy$  zumindest ein Zeichen weniger als  $n$  mal vorkommt, aber  $a$  oder  $c$  noch  $n$  mal vorkommen
  - ➔  $L_{abc}$  nicht kontextfrei

# Pumping-Lemma: Anwendung (3/6)

## Korollar 10.2

- Sei  $L$  eine Sprache
- Angenommen, für jedes  $n > 0$  gibt es einen String  $z \in L$  mit  $|z| \geq n$  so dass für jede Zerlegung  $z = uvwxy$  mit
  - (1)  $vx \neq \epsilon$ ,
  - (2)  $|vwx| \leq n$ ,ein  $k \geq 0$  existiert mit
$$uv^kwx^ky \notin L$$
- Dann ist  $L$  nicht kontextfrei

## Proposition 10.3

- Die beiden folgenden Sprachen sind nicht kontextfrei:
  - (a)  $L_{abc} = \{a^m b^m c^m \mid m \geq 1\}$
  - (b)  $L_{\text{doppel}} = \{ww \mid w \in \{a, b\}^*\}$

## Beweis für Proposition 10.3 (b)

- Sei  $n$  beliebig
  - Wähle  $z = a^n b^n a^n b^n$  ☞ 4 „Blöcke“
  - Sei  $uvwxy$  eine Zerlegung von  $z$  mit  $u, v, w, x, y \in \{a, b\}^*$ , die (1) und (2) erfüllt
  - Klar: falls  $|vx|$  ungerade ist, ist  $|uwy|$  auch ungerade, und deshalb  $uwy \notin L_{\text{doppel}}$
- ☞ Im Rest des Beweises sei  $|vx|$  also gerade

# Pumping-Lemma: Anwendung (4/6)

## Korollar 10.2

- Sei  $L$  eine Sprache
- Angenommen, für jedes  $n > 0$  gibt es einen String  $z \in L$  mit  $|z| \geq n$  so dass für jede Zerlegung  $z = uvwxy$  mit
  - (1)  $vx \neq \epsilon$ ,
  - (2)  $|vwx| \leq n$ ,ein  $k \geq 0$  existiert mit
$$uv^kwx^ky \notin L$$
- Dann ist  $L$  nicht kontextfrei

## Proposition 10.3

- Die beiden folgenden Sprachen sind nicht kontextfrei:
  - (a)  $L_{abc} = \{a^m b^m c^m \mid m \geq 1\}$
  - (b)  $L_{\text{doppel}} = \{ww \mid w \in \{a, b\}^*\}$

## Beweis für Proposition 10.3 (b) (Forts.)

- Wir unterscheiden vier Fälle:
  - (1)  $vx$  enthält  $a$ 's aus dem ersten Block  
☞ möglicherweise aber auch andere Zeichen
  - (2)  $vx$  enthält  $b$ 's aus dem zweiten Block, aber keine  $a$ 's aus dem ersten Block  
☞ möglicherweise aber noch andere Zeichen
  - (3)  $vx$  enthält  $a$ 's aus dem dritten Block aber keine Zeichen aus den ersten zwei Blöcken  
☞ vielleicht aber Zeichen aus dem vierten Block
  - (4)  $vx$  enthält nur  $b$ 's aus dem vierten Block aber keine Zeichen aus anderen Blöcken

- In allen vier Fällen zeigen wir:
$$1^{\text{st}}(uwy) \neq 2^{\text{nd}}(uwy)$$
und damit  $uwy \notin L_{\text{doppel}}$   
✎ Zur Erinnerung:  $1^{\text{st}}(w)$  bezeichnet die erste Hälfte eines Strings  $w$  und  $2^{\text{nd}}(w)$  bezeichnet die zweite Hälfte

# Pumping-Lemma: Anwendung (5/6)

## Korollar 10.2

- Sei  $L$  eine Sprache
- Angenommen, für jedes  $n > 0$  gibt es einen String  $z \in L$  mit  $|z| \geq n$  so dass für jede Zerlegung  $z = uvwxy$  mit
  - (1)  $vx \neq \epsilon$ ,
  - (2)  $|vwx| \leq n$ ,ein  $k \geq 0$  existiert mit
$$uv^kwx^ky \notin L$$
- Dann ist  $L$  nicht kontextfrei

## Proposition 10.3

- Die beiden folgenden Sprachen sind nicht kontextfrei:
  - (a)  $L_{abc} = \{a^m b^m c^m \mid m \geq 1\}$
  - (b)  $L_{\text{doppel}} = \{ww \mid w \in \{a, b\}^*\}$

## Beweis für Proposition 10.3 (b)

- Zur Erinnerung:  $z = a^n b^n a^n b^n$  (4 „Blöcke“)

- (1)  $vx$  enthält  $a$ 's aus dem ersten Block
  - Möglicherweise enthält  $vx$  auch Zeichen aus dem zweiten Block
  - Da  $|vwx| \leq n$  enthält  $vx$  keine Zeichen aus den letzten beiden Blöcken
  - ➡  $uwy$  ist von der Form  $a^i b^j a^n b^n$  mit
    - \*  $i < n$  und  $j \leq n$
  - Da  $|vx| \leq n$  gilt:  $3n \leq |uwy| < 4n$
  - ➡ Das letzte Zeichen von  $1^{\text{st}}(uwy)$  ist ein  $a$  (aus dem dritten Block), aber das letzte Zeichen von  $2^{\text{nd}}(uwy)$  ein  $b$
  - ➡  $uwy \notin L_{\text{doppel}}$

## Pumping-Lemma: Anwendung (6/6)

### Beweis für Proposition 10.3 (b) (Forts.)

- Zur Erinnerung:  $z = a^n b^n a^n b^n$

☞ 4 „Blöcke“

(2)  $vx$  enthält  $b$ 's aus dem zweiten Block, aber keine  $a$ 's aus dem ersten Block

➡ Da  $|vwx| \leq n$  enthält  $vx$  keine  $b$ 's aus dem vierten Block

➡  $uwy$  ist von der Form  $a^n b^i a^j b^n$  mit  
\*  $i < n$  und  $j \leq n$  und  
\*  $i + j \geq n$  ☞  $|vwx| \leq n$

➡  $2^{\text{nd}}(uwy)$  endet mit einem Block der Form  $b^n$ , aber  $1^{\text{st}}(uwy)$  enthält weniger als  $n$   $b$ 's

➡  $uwy \notin L_{\text{doppel}}$

### Beweis für Proposition 10.3 (b) (Forts.)

(3)  $vx$  enthält  $a$ 's aus dem dritten Block aber keine Zeichen aus den ersten zwei Blöcken

➡  $uwy = a^n b^n a^i b^j$

➡  $a^n$ -Block in  $1^{\text{st}}(uwy)$ , aber nicht in  $2^{\text{nd}}(uwy)$

➡  $uwy \notin L_{\text{doppel}}$

(4)  $vx$  enthält nur  $b$ 's aus dem vierten Block aber keine Zeichen aus anderen Blöcken

➡  $uwy = a^n b^n a^n b^i$

➡  $1^{\text{st}}(uwy)$  beginnt mit  $a$ ,  $2^{\text{nd}}(uwy)$  mit  $b$

➡  $uwy \notin L_{\text{doppel}}$



# Inhalt

10.1 Das Pumping-Lemma für kontextfreie Sprachen

## **10.2 Algorithmen für kontextfreie Sprachen**

### ▷ **10.2.1 Umwandlungsalgorithmen**

10.2.2 Analyse-Algorithmen für kontextfreie Sprachen

10.3 Abschlusseigenschaften der kontextfreien Sprachen

10.4 Deterministische Kellerautomaten

# Kontextfreie Sprachen: Umwandlungsalgorithmen


- Die folgenden Umwandlungen sind in linearer Zeit möglich und erzeugen Objekte linearer Größe:

- kontextfreie Grammatik  $\rightarrow$  Kellerautomat
- Kellerautomat mit leerem Keller  $\rightarrow$  Kellerautomat mit akzeptierenden Zuständen
- Kellerautomat mit akzeptierenden Zuständen  $\rightarrow$  Kellerautomat mit leerem Keller

- Die Umwandlung eines Kellerautomaten  $\mathcal{A}$  in eine Grammatik ist in Zeit  $\mathcal{O}(|\mathcal{A}|^4)$  möglich

- Dabei bezeichnet  $|\mathcal{A}|$  die Größe der Transitionsfunktion, wobei jede Transition  $1 +$  Länge des neuen Kellerstrings beiträgt

- Die Umwandlung einer kontextfreien Grammatik in eine CNF-Grammatik ist in Zeit  $\mathcal{O}(|G|^2)$  möglich

 wir hatten nur:  $\mathcal{O}(|G|^4)$

- Bei der Umwandlung in Greibach-Normalform ist Vorsicht geboten:
  - Der Original-Algorithmus von Greibach kann im schlimmsten Fall eine Grammatik exponentieller Größe erzeugen
  - Es gibt Algorithmen, die immer eine Grammatik in GNF der Größe  $\mathcal{O}(|G|^3)$  erzeugen

[Rosenkrantz 67; Blum, Koch 98]

# Inhalt

10.1 Das Pumping-Lemma für kontextfreie Sprachen

## **10.2 Algorithmen für kontextfreie Sprachen**

10.2.1 Umwandlungsalgorithmen

### ▷ **10.2.2 Analyse-Algorithmen für kontextfreie Sprachen**

10.3 Abschlusseigenschaften der kontextfreien Sprachen

10.4 Deterministische Kellerautomaten

# Leerheitstests für kontextfreie Sprachen

Def.: Leerheitsproblem für CFGs

**Gegeben:** Kontextfreie Grammatik  $G$

**Frage:** Ist  $L(G) \neq \emptyset$ ?

Def.: Leerheitsproblem für PDAs

**Gegeben:** Kellerautomat  $\mathcal{A}$

**Frage:** Ist  $L(\mathcal{A}) \neq \emptyset$ ?

Satz 10.4

- Zu einer gegebenen kontextfreien Grammatik  $G$  lässt sich in linearer Zeit  $\mathcal{O}(|G|)$  entscheiden, ob  $L(G) \neq \emptyset$  gilt

Beweisidee

- $L(G) \neq \emptyset$  gilt genau dann, wenn das Startsymbol  $S$  erzeugend ist
- Das lässt sich bei geschickter Implementierung in Zeit  $\mathcal{O}(|G|)$  testen

[Beeri, Bernstein 79]

- Leerheitstest für Kellerautomaten:
  - Wandle Kellerautomat in Grammatik um, dann Leerheitstest für Grammatik
  - Ein effizienterer „direkter“ Algorithmus ist (mir) nicht bekannt

# Endlichkeitstest für kontextfreie Sprachen

Def.: Endlichkeitsproblem für KFGs

**Gegeben:** Grammatik  $G$

**Frage:** Ist  $|L(G)| < \infty$ ?

## Satz 10.5

- Das Endlichkeitsproblem für kontextfreie Grammatiken in CNF kann in linearer Zeit entschieden werden

## Beweis

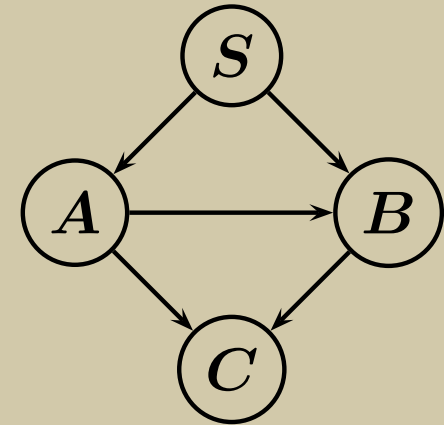
- Da  $G$  in CNF ist, hat  $G$  insbesondere keine nutzlosen Symbole
- Wir definieren zu  $G$  einen gerichteten Graphen  $H(G)$  (wie in CNF1):
  - Die Knoten von  $H(G)$  sind die Variablen von  $G$
  - Ist  $X \rightarrow YZ$  eine Regel von  $G$ , so hat  $H(G)$  die Kanten  $(X, Y)$  und  $(X, Z)$
- Behauptung:**  $L(G)$  ist genau dann unendlich, wenn  $H(G)$  einen gerichteten Kreis enthält

## Beispiel

$G_1$ :

$$\begin{array}{l|l} S \rightarrow AB & \\ A \rightarrow BC & a \\ B \rightarrow CC & b \\ C \rightarrow a & \end{array}$$

$H(G_1)$ :

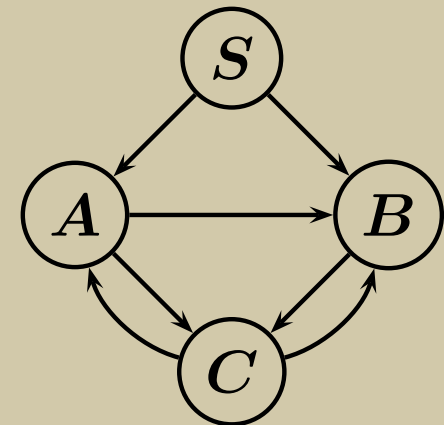


## Beispiel

$G_2$ :



$$\begin{array}{l|l} S \rightarrow AB & \\ A \rightarrow BC & a \\ B \rightarrow CC & b \\ C \rightarrow AB & a \end{array}$$

$H(G_2)$ :



 Beweisdetails im Anhang

# Andere Tests für kontextfreie Sprachen

- Es gibt natürlich noch weitere Algorithmen für kontextfreie Grammatiken
- Es existieren aber auch Probleme, für die es keine Algorithmen gibt!
- Die folgenden algorithmischen Probleme für kontextfreie Grammatiken können nicht von Algorithmen gelöst werden:
  - (1) Ist  $L(G)$  eindeutig oder inhärent mehrdeutig?
  - (2) Ist  $G$  eindeutig?
  - (3) Ist  $L(G)$  deterministisch kontextfrei?  siehe 10.4
  - (4) Ist  $L(G)$  regulär?
  - (5) Ist  $L(G_1) \cap L(G_2) = \emptyset$ ?
  - (6) Ist  $L(G_1) \cap L(G_2)$  kontextfrei?
  - (7) Ist  $L(G_1) \subseteq L(G_2)$ ?
  - (8) Ist  $L(G_1) = L(G_2)$ ?
  - (9) Ist  $L(G) = \Sigma^*$ ?
-  Die genaue Bedeutung von „können nicht von Algorithmen gelöst werden“ werden wir in Teil C der Vorlesung kennen lernen
  - Dort werden die Aussagen dann auch bewiesen

# Inhalt

10.1 Das Pumping-Lemma für kontextfreie Sprachen

10.2 Algorithmen für kontextfreie Sprachen

▷ **10.3 Abschlusseigenschaften der kontextfreien Sprachen**

10.4 Deterministische Kellerautomaten

# Abschlusseigenschaften der kontextfreien Sprachen

- Die Klasse der kontextfreien Sprachen ist unter vielen Operationen abgeschlossen
  - ...aber nicht unter ganz so vielen wie die regulären Sprachen

## Satz 10.6

- Die Klasse der kontextfreien Sprachen ist abgeschlossen unter
  - (a) Vereinigung,
  - (b) Konkatination,
  - (c) dem  $*$ -Operator und
  - (d) dem  $+$ -Operator,

## Beweis

- Seien  $L_1$  und  $L_2$  kontextfreie Sprachen mit Grammatiken
  - $G_1 = (V_1, \Sigma, S_1, P_1)$  und
  - $G_2 = (V_2, \Sigma, S_2, P_2)$mit  $V_1 \cap V_2 = \emptyset$
- Dann können wir Grammatiken konstruieren für:
  - (a)  $L_1 \cup L_2$ : durch Vereinigung der beiden Grammatiken und Hinzunahme einer neuen Startvariablen  $S$  mit  $S \rightarrow S_1 \mid S_2$
  - (b)  $L_1 L_2$ : durch Vereinigung der beiden Grammatiken und Hinzunahme einer neuen Startvariablen  $S$  mit  $S \rightarrow S_1 S_2$
  - (c)  $L_1^*$ : durch Hinzunahme einer neuen Startvariablen  $S$  mit  $S \rightarrow S_1 S \mid \epsilon$
  - (d)  $L_1^+$ : durch Hinzunahme einer neuen Startvariablen  $S$  mit  $S \rightarrow S_1 S \mid S_1$




# Weitere Abschlusseigenschaften

## Satz 10.7

- (a) Ist  $L$  kontextfrei, so auch  $\{w^R \mid w \in L\}$
- (b) Ist  $L \subseteq \Sigma^*$  kontextfrei,  $h : \Sigma^* \rightarrow \Gamma^*$  ein Homomorphismus, so ist auch  $h(L)$  kontextfrei
- (c) Ist  $L \subseteq \Sigma^*$  kontextfrei,  $h : \Gamma^* \rightarrow \Sigma^*$  ein Homomorphismus, so ist auch  $h^{-1}(L)$  kontextfrei

## Beweisidee

- (a) Idee: Drehe die Regeln um
  - Sei  $G$  CNF-Grammatik für  $L$
  - Ersetze  $X \rightarrow YZ$  jeweils durch
$$X \rightarrow ZY$$
- (b) Ersetze in der Grammatik für  $L$  jedes Vorkommen eines Terminalsymbols  $\sigma \in \Sigma$  durch  $h(\sigma)$
- (c) Konstruktion eines Kellerautomaten  $\mathcal{B}$  für  $h^{-1}(L)$ :
  -  analog zum Beweis für endliche Automaten
    - Sei  $\mathcal{A}$  Kellerautomat für  $L$
    - Wenn  $\mathcal{B}$  das Zeichen  $\sigma$  liest, simuliert er das Verhalten von  $\mathcal{A}$  bei Eingabe  $h(\sigma)$
  - Neue Zustände,  $\epsilon$ -Übergänge müssen beachtet werden

# Fehlende Abschlusseigenschaften

## Satz 10.8

- Die Klasse der kontextfreien Sprachen ist **nicht** abgeschlossen unter
  - (a) Durchschnitt,
  - (b) Komplement,
  - (c) Mengendifferenz

## Beweis

- (a) •  $L_1 \stackrel{\text{def}}{=} \{a^n b^n c^m \mid n, m \geq 1\}$   
•  $L_2 \stackrel{\text{def}}{=} \{a^m b^n c^n \mid n, m \geq 1\}$   
•  $L_1$  und  $L_2$  sind kontextfrei  
•  $L_1 \cap L_2 = L_{abc} = \{a^n b^n c^n \mid n \geq 1\}$   
ist **nicht** kontextfrei

☞ Proposition 12.3 (a)

- (b) Sonst ließe sich der Durchschnitt durch Kombination von Vereinigung und Komplement ausdrücken
- (c) Sonst ließe sich das Komplement darstellen als:  $\Sigma^* - L$

- Unter Durchschnittsbildung sind die kontextfreien Sprachen also nicht abgeschlossen
  - Insbesondere gibt es also keine Produktautomatenkonstruktion für zwei Kellerautomaten

☞ Warum eigentlich?

- Es gilt aber eine schwächere Abschlusseigenschaft:

## Satz 10.9

- Ist  $L_1$  kontextfrei und  $L_2$  regulär, so ist  $L_1 \cap L_2$  kontextfrei

## Beweisidee

- Sei  $\mathcal{A}_1$  ein Kellerautomat für  $L_1$
- Sei  $\mathcal{A}_2$  ein DFA für  $L_2$
- $\mathcal{B}$  sei der „Produktautomat“ von  $\mathcal{A}_1$  und  $\mathcal{A}_2$
- $\mathcal{A}_2$  wirkt sich nur auf die Zustände aus, nicht auf den Kellerinhalt

# Inhalt


10.1 Das Pumping-Lemma für kontextfreie Sprachen

10.2 Algorithmen für kontextfreie Sprachen

10.3 Abschlusseigenschaften der kontextfreien Sprachen

▷ **10.4 Deterministische Kellerautomaten**

# Deterministische Kellerautomaten: Motivation

- Ein Nachteil von PDAs:
  - Algorithmisch zu entscheiden, ob ein gegebener PDA  $\mathcal{A}$  für einen gegebenen String  $w$  eine akzeptierende Berechnung hat, ist nicht so leicht
  - Es kann exponentiell viele verschiedene Berechnungen geben...
  - Der nahe liegende Algorithmus verwendet Backtracking und kann zu exponentiellem Aufwand führen  Kapitel 11

- **Frage:** Gibt es zu jedem PDA einen äquivalenten deterministischen Kellerautomaten (DPDA)?
  - Dazu müssen wir zuerst definieren, wann Kellerautomaten deterministisch sind

- PDAs haben zwei Quellen für Nichtdeterminismus:

- Für einen Zustand  $p$ , ein gelesenes Eingabezeichen  $\sigma$ , und ein Kellersymbol  $\tau$  kann es in  $\delta$  mehrere Transitionen geben:
  - $(p, \sigma, \tau, q_1, w_1)$
  - $(p, \sigma, \tau, q_2, w_2)$

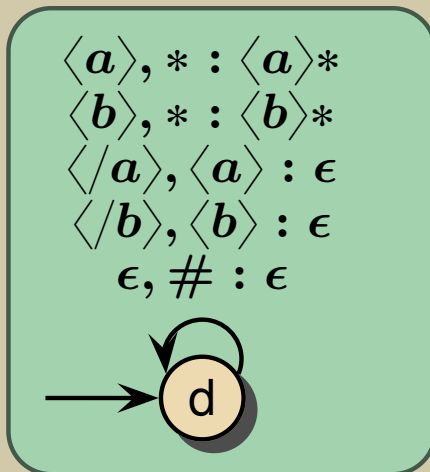
➡ Klar: in deterministischen Kellerautomaten darf es für jede Kombination von  $p, \sigma, \tau$  nur eine Transition  $(p, \sigma, \tau, q, w)$  in  $\delta$  geben

- PDAs können außerdem  $\epsilon$ -Übergänge haben
  - Wir erlauben  $\epsilon$ -Übergänge auch in DPDAs
  - Aber: es darf keine zwei verschiedenen Transitionen
    - \*  $(p, \epsilon, \tau, q_1, w_1)$
    - \*  $(p, \epsilon, \tau, q_2, w_2)$geben
  - Und: wenn es eine Transition  $(p, \epsilon, \tau, q, w)$  gibt, so darf es für kein  $\sigma$  eine Transition  $(p, \sigma, \tau, q', w')$  geben

# Deterministische Kellerautomaten: Beispiele

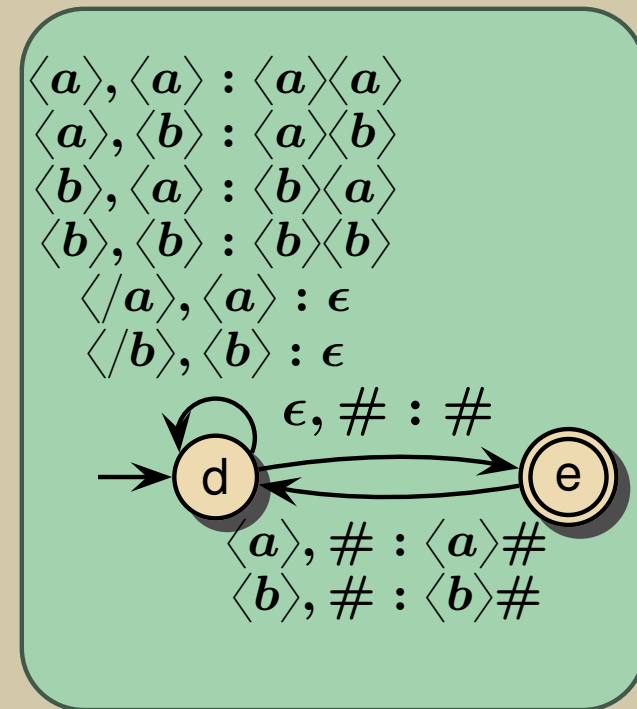
## Beispiel

- Zur Erinnerung: der folgende Kellerautomat entscheidet die Sprache der wohlgeformten Klammerausdrücke über  $\{\langle a \rangle, \langle /a \rangle, \langle b \rangle, \langle /b \rangle\}$ :



- Der Automat akzeptiert mit leerem Keller
- Aber der Automat ist nicht deterministisch:
  - $(d, \langle b \rangle, \#, d, \langle b \rangle \#) \in \delta$  und
  - $(d, \epsilon, \#, d, \epsilon) \in \delta$
- Ein kritischer String:
  - $\langle b \rangle \langle a \rangle \langle /a \rangle \langle /b \rangle \langle b \rangle \langle /b \rangle$
  - Wie soll es nach Lesen von  $\langle b \rangle \langle a \rangle \langle /a \rangle \langle /b \rangle$  weitergehen?

## Beispiel



- Dies ist ein deterministischer Kellerautomat mit akzeptierenden Zuständen für dieselbe Sprache
- Nach Lesen von  $\langle b \rangle \langle a \rangle \langle /a \rangle \langle /b \rangle$  geht er in einen akzeptierenden Zustand
- Wenn der String noch nicht zu Ende ist, kann er dann aber auch noch weitere Zeichen lesen

# Deterministische Kellerautomaten: Definition

- Wir verwenden in der Definition von DPDAs die folgende Notation:

$$- \underline{\delta(p, \sigma, \tau)} \stackrel{\text{def}}{=} \{(q, z) \mid (p, \sigma, \tau, q, z) \in \delta\}$$

$$- \underline{\delta(p, \epsilon, \tau)} \stackrel{\text{def}}{=} \{(q, z) \mid (p, \epsilon, \tau, q, z) \in \delta\}$$

## Definition

- Ein Kellerautomat  $\mathcal{A} = (Q, \Sigma, \Gamma, \delta, s, \tau_0, F)$  heißt deterministisch, falls für alle  $p \in Q, \sigma \in \Sigma, \tau \in \Gamma$  gilt:
  - $|\delta(p, \sigma, \tau)| + |\delta(p, \epsilon, \tau)| \leq 1$
- Eine Sprache heißt deterministisch kontextfrei, wenn sie von einem deterministischen Kellerautomaten entschieden wird

# Deterministisch kontextfreie Sprachen: Akzeptiermechanismen

## Satz 10.10

- (a) Zu jedem DPDA  $\mathcal{A}$ , der mit leerem Keller akzeptiert, gibt es es einen DPDA  $\mathcal{B}$ , der mit akzeptierenden Zuständen akzeptiert und  $L(\mathcal{B}) = L(\mathcal{A})$  erfüllt
- (b) Die Umkehrung gilt nicht

## Beweisidee

- (a) Die Konstruktion aus Satz 9.1 (a) erzeugt aus einem DPDA wieder einen DPDA
  - (b) Von einem DPDA mit leerem Keller akzeptierte Sprachen sind **präfixfrei**:
    - Sie enthalten keine Strings  $u$  und  $v$ , für die  $u$  echtes Präfix von  $v$  ist
      - Denn: nach dem Lesen von  $u$  ist der Keller leer, der Automat kann den Rest von  $v$  also gar nicht mehr lesen
- 
- DPDAs mit leerem Keller gibt es also nicht einmal für jede reguläre Sprache  
👉 z.B.:  $\Sigma^*$
  - Es gilt aber:
    - wenn  $L$  präfixfrei ist und einen DPDA mit akzeptierenden Zuständen hat
    - dann hat  $L$  auch einen DPDA mit leerem Keller

# Det. kontextfreie Sprachen: Komplementabschluss (1/2)

## Satz 10.11


- Die Klasse der deterministisch kontextfreien Sprachen ist abgeschlossen unter Komplementbildung

## Beweisidee

- Sei  $\mathcal{A}$  ein deterministischer Kellerautomat für Sprache  $L$
- Grundidee: Der Automat  $\mathcal{B}$  für  $\overline{L}$  entstehe durch Vertauschung akzeptierender und ablehnender Zustände in  $\mathcal{A}$

## Beweisidee (Forts.)


### • **Komplikationen:**

- $\mathcal{A}$  könnte anhalten, ohne die Eingabe ganz zu lesen,
  - weil  $\mathcal{A}$  in einer Konfiguration keine Transition hat,
  - weil vorzeitig eine Konfiguration mit leerem Keller erreicht wird, oder
  - weil nur noch eine (unendliche) Folge von  $\epsilon$ -Übergängen möglich ist➡ In all diesen Fällen muss  $\mathcal{B}$  die Eingabe akzeptieren
- $\mathcal{A}$  könnte akzeptieren mit einer Berechnung der Art:
$$(s, w, \tau_0) \vdash_{\mathcal{A}}^* (q_1, \epsilon, \alpha) \vdash_{\mathcal{A}}^* (q_2, \epsilon, \beta)$$
mit  $q_1 \in F$  und  $q_2 \notin F$   oder umgekehrt  
➡ In diesem Fall darf  $\mathcal{B}$  **nicht** akzeptieren, obwohl am Ende der Zustand  $q_2 \notin F$  erreicht wird!




# Det. kontextfreie Sprachen: Komplementabschluss (2/2)

## Beweisidee (Forts.)

- (1) Für das Problem von Berechnungen, die nicht die ganze Eingabe lesen, hat  $\mathcal{B}$  einen zusätzlichen Zustand  $q_+$ , in dem der Rest der Eingabe gelesen und dann akzeptiert wird
  - Die Frage ist nur: wie erkennt  $\mathcal{B}$ , dass er in den Zustand  $q_+$  übergehen muss?
- (1a) Hat  $\mathcal{A}$  für Zustand  $p$ , Eingabezeichen  $\sigma$  und Kellersymbol  $\tau$  keine Transition, so geht  $\mathcal{B}$  in den Zustand  $q_+$  über
- (1b) Um Konfigurationen mit leerem Keller zu vermeiden (und zu erkennen), verwendet  $\mathcal{B}$  ein neues unterstes Kellersymbol
  -  Ähnlich der Umwandlung von Kellerautomaten in Satz 9.1 (a) und 9.1 (b)
- (1c) Die Menge der Paare  $(p, \tau)$ , die zu unendlichen Folgen von  $\epsilon$ -Transitionen führen, lässt sich berechnen
  - \*  $\mathcal{B}$  geht für diese Paare in  $q_+$  über

## Beweisidee (Forts.)

- (2)  $\mathcal{B}$  merkt sich immer, ob seit der letzten Nicht- $\epsilon$ -Transition schon ein akzeptierender Zustand von  $\mathcal{A}$  gesehen wurde
- Mit diesen Ideen lässt sich ein korrekter Automat für das Komplement von  $L(\mathcal{A})$  konstruieren
-  Details finden sich beispielsweise im Buch von Ingo Wegener

# Det. kontextfreie Sprachen: Vereinigung und Durchschnitt

## Satz 10.12

- Die Klasse der deterministisch kontextfreien Sprachen ist **nicht** abgeschlossen unter Vereinigung und Durchschnitt

## Beweis

- Seien wieder
  - $L_1 := \{a^n b^n c^m \mid n, m \geq 1\}$  und
  - $L_2 := \{a^m b^n c^n \mid n, m \geq 1\}$
- $L_1$  und  $L_2$  sind sogar deterministisch kontextfrei
- Aber:  $L_1 \cap L_2$  ist noch nicht einmal kontextfrei
- ➔ Die deterministisch kontextfreien Sprachen sind nicht unter Durchschnitt abgeschlossen
- ➔ Die deterministisch kontextfreien Sprachen sind nicht unter Vereinigung abgeschlossen
  - Sonst wären sie wegen De Morgan und dem Komplementabschluss auch unter Durchschnitt abgeschlossen

# Nicht alle kontextfreien Sprachen haben einen DPDA

- Die Klasse der kontextfreien Sprachen hat also andere Abschlusseigenschaften als die Klasse der deterministisch kontextfreien Sprachen
  - ➔ die beiden Klassen sind verschieden
  - ➔ die deterministisch kontextfreien Sprachen bilden eine echte Teilklasse der Klasse der kontextfreien Sprachen
- Wir betrachten nun ein Beispiel einer kontextfreien Sprache, die nicht deterministisch kontextfrei ist
- Zur Erinnerung:
  - $L_{\text{diff}} = \{w \in \{a, b\}^* \mid 1^{\text{st}}(w) \neq 2^{\text{nd}}(w)\}$  ist kontextfrei
  - $L_{\text{doppel}} = \{ww \mid w \in \{a, b\}^*\}$  ist nicht kontextfrei
- Sei  $L_{\text{undoppel}}$  das Komplement von  $L_{\text{doppel}}$
- Also:  $L_{\text{undoppel}}$  enthält alle Strings ungerader Länge sowie alle Strings aus  $L_{\text{diff}}$

## Proposition 10.13

- $L_{\text{undoppel}}$  ist kontextfrei aber nicht deterministisch kontextfrei

## Beweisskizze

- Dass  $L_{\text{doppel}}$  nicht kontextfrei ist, haben wir in Proposition 12.3 (b) schon bewiesen
- Also kann  $L_{\text{undoppel}}$  nicht deterministisch kontextfrei sein, sonst wäre es ja auch  $L_{\text{doppel}}$ 
  - ☞ Komplementabschluss
- Andererseits ist  $L_{\text{undoppel}}$  die Vereinigung
  - einer regulären Sprache (Strings ungerader Länge) und
  - der kontextfreien Sprache  $L_{\text{diff}}$und damit kontextfrei

# Verhältnis zu anderen Klassen

## Satz 10.14

- (a) Jede reguläre Sprache ist deterministisch kontextfrei
- (b) Es gibt deterministisch kontextfreie Sprachen, die nicht regulär sind
- (c) Es gibt kontextfreie Sprachen, die nicht deterministisch kontextfrei sind

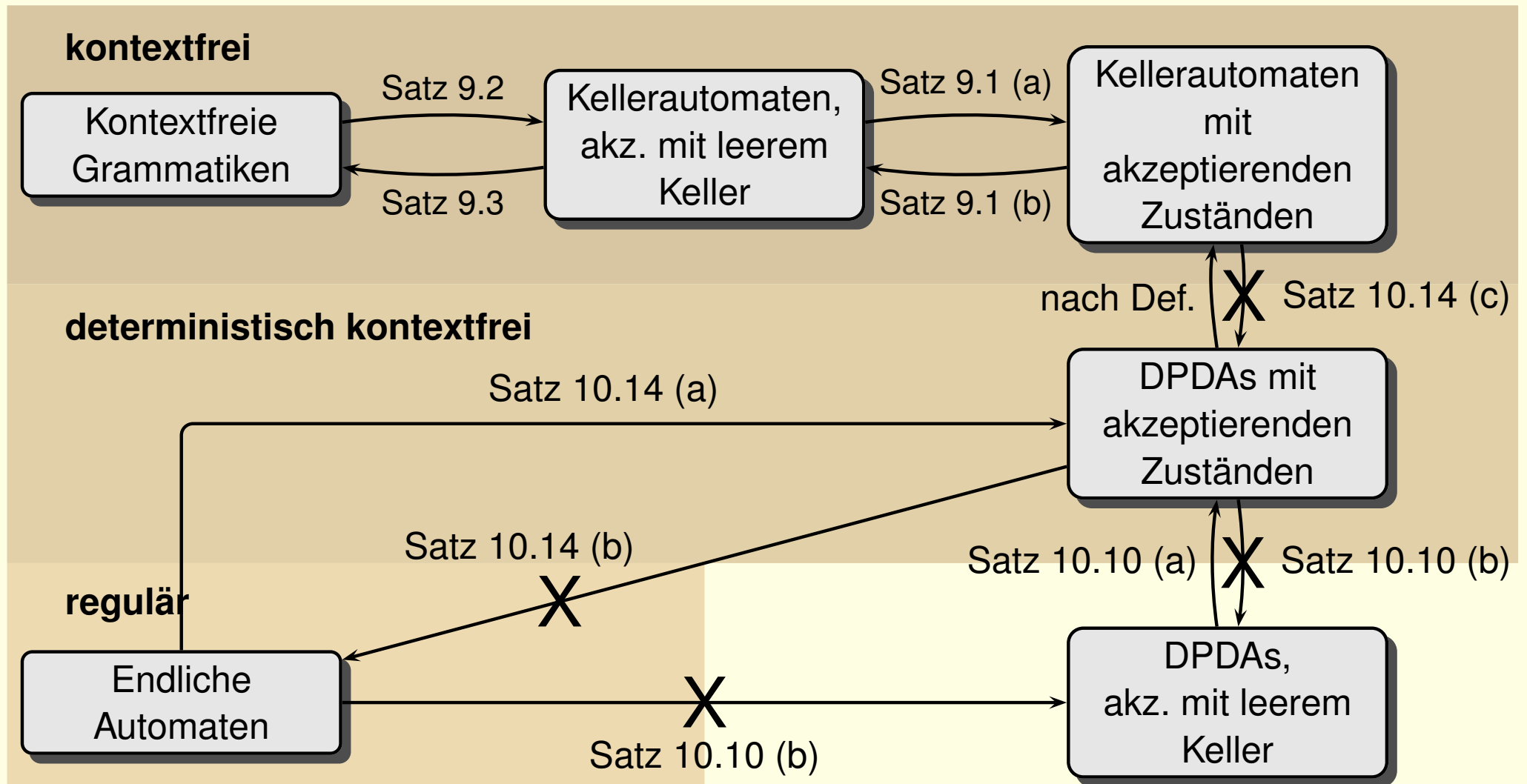
## Beweisidee

- (a) Jeder DFA kann als deterministischer PDA mit akzeptierenden Zuständen interpretiert werden, der seinen Keller nicht verwendet
- (b) Beispiel:  $\{a^n b^n \mid n \geq 0\}$
- (c) Beispiele:
  - $L_{\text{undoppel}}$
  - $L_{\text{rev}}$

☞ Proposition 10.13

☞ Beweis etwas komplizierter

# Verhältnis der Modelle



# Zusammenfassung

- Das Pumping-Lemma für kontextfreie Sprachen ist ein Hilfsmittel um nachzuweisen, dass eine gegebene Sprache nicht kontextfrei ist
- Es gibt einige stärkere Versionen des Pumping-Lemmas, wie zum Beispiel **Ogden's Lemma**

👉 siehe Buch von Ingo Wegener

- Die kontextfreien Sprachen haben etwas weniger günstige algorithmische und Abschlusseigenschaften als die regulären Sprachen
- Insbesondere gibt es Fragen, die sich algorithmisch gar nicht lösen lassen
- Deterministische Kellerautomaten sind echt schwächer als Kellerautomaten und echt stärker als endliche Automaten

# Literatur für dieses Kapitel

- **Effiziente Umwandlung in GNF:**

- Daniel J. Rosenkrantz. Matrix equations and normal forms for context-free grammars. *J. ACM*, 14(3):501–507, 1967
- Norbert Blum and Robert Koch. Greibach normal form transformation revisited. *Inf. Comput.*, 150(1):112–118, 1999

- **Leerheitstest für kontextfreie Grammatiken:**

- Catriel Beeri and Philip A. Bernstein. Computational problems related to the design of normal form relational schemas. *ACM Trans. Database Syst.*, 4(1):30–59, 1979

- **Pumping-Lemma:**

- Y. Bar-Hillel, M. Perles, and E. Shamir. On formal properties of simple phrase structure grammars. *Z. Phonetik, Sprachwiss. Kommunikationsforsch.*, 14:143–172, 1961

# Endlichkeitstest für kontextfreie Sprachen: Beweisdetails

## Beweis (Forts.)

- Zu zeigen:  
 $L(G)$  unendlich  $\iff H(G)$  hat Kreis
  - Sei  $G = (V, \Sigma, S, P)$
  - Wir zeigen zuerst: „ $\Leftarrow$ “
  - Da  $H(G)$  einen Kreis hat, gibt es eine Variable  $X$  mit  $X \Rightarrow_G^* vXx$ , für gewisse  $v, x \in \Sigma^*$  mit  $vx \neq \epsilon$ 
    - \* Denn: die Anwendung der Regeln, die die Kanten des Kreises ergeben haben, liefert  $X \Rightarrow_G^* \alpha X \beta$  für gewisse  $\alpha, \beta \in V^*$
    - \* Alle in  $\alpha, \beta$  vorkommenden Variablen lassen sich zu nichtleeren Strings ableiten
  - Da  $X$  nützlich ist, gilt
    - \*  $X \Rightarrow_G^* w$  für ein  $w \in \Sigma^*$
    - \*  $S \Rightarrow_G^* uXy$ , für gewisse  $u, y \in \Sigma^*$
- ➡ alle (unendlich vielen) Strings der Form  $uv^kwx^ky$ ,  $k \geq 0$  sind in  $L$

## Beweis (Forts.)

- Wir zeigen jetzt: „ $\Rightarrow$ “
- Sei  $m \stackrel{\text{def}}{=} |V|$
- Wenn  $L(G)$  unendlich viele Strings enthält, enthält  $L(G)$  insbesondere einen String  $w$  mit  $|w| > 2^{m+1}$
- Wie im Beweis des Pumping-Lemmas muss deshalb auf einem Weg eines Blattes des Ableitungsbaumes zu  $w$  eine Variable  $X$  mehrfach vorkommen
- ➡  $X$  liegt in  $H(G)$  auf einem Kreis
- ➡ Behauptung