

---

# Formale Sprachen und Automaten

---

## Reguläre und kontextfreie Grammatiken, Kellerautomaten

## Grammatik und Automat

- Automaten sind Konzepte, die eine Sprache  $L$  dadurch charakterisieren, dass sie  $L$  akzeptieren.
- Grammatiken sind Konzepte, die eine Sprache  $L$  dadurch charakterisieren, dass sie  $L$  generieren.

## Grammatiken

- Eine Grammatik besteht im wesentlichen aus einer **endlichen** Zahl von Regeln.
- Diese Regeln werden nach einem bestimmten Verfahren eingesetzt, um die Wörter einer Sprache  $L \subseteq \Sigma^*$  über einem Alphabet  $\Sigma$  zu erzeugen.
- Jede dieser Regeln besteht
  1. aus einem Pfeil  $\rightarrow$
  2. aus seiner **linken Seite** (lS)
  3. aus seiner **rechten Seite** (rS)
- Auf lS und rS einer Regel stehen eine Reihe von Symbolen.

## Grammatiken 2

- Bei den Symbolen unterscheidet man
  1. **Terminalsymbole**
  2. **Nonterminalsymbole**
- Terminalsymbole
  1. sind Elemente aus  $\Sigma$
  2. werden nach Konvention klein geschrieben
- Nonterminalsymbole
  1. sind Hilfssymbole (auch **syntaktische Variablen** genannt)
  2. sind nicht aus  $\Sigma$
  3. werden groß geschrieben
- Wenigstens eines der Symbole auf der lS jeder Regel ist ein Nonterminalsymbol.

## Grammatiken 3

- Beispiel Regel:  $xYz \rightarrow uVw$
- $Y, V$  sind Nonterminalsymbole,  $x, y, u, w$  sind Terminalsymbole.
- Beispiel Grammatik  $G$ :
  1.  $S \rightarrow aMb$
  2.  $M \rightarrow aM$
  3.  $M \rightarrow bM$
  4.  $M \rightarrow \epsilon$
- $S$  ist das **Startsymbol** (eine Nichtterminalsymbol).
- $G$  erzeugt die reguläre Sprache  $a(a \cup b)^*b$ .

## Grammatiken 4

- Prozedur zum Generieren:
  1. Beginne mit dem Startsymbol  $S$ .
  2. Bilde die Kette  $K$  dadurch, dass du  $S$  durch die  $rS$  einer Regel, deren  $lS$   $S$  ist, ersetzt.
  3. Wiederhole diesen Ersetzungsprozess für jedes Nichtterminalsymbol in  $K$  und bilde dadurch jeweils eine neue Kette  $K'$ .
  4. Verfahre so, bis in  $K'$  keine Ersetzungen mehr möglich sind ( $K'$  enthält keine Nonterminale mehr).
- Das Abarbeiten dieser Prozedur nennt man **Derivation** oder **Ableitung**.

## Kontextfreie Grammatiken 1

- Beispielderivationen mit  $G$ :
  1.  $S \Rightarrow aMb \Rightarrow a\epsilon b$
  2.  $S \Rightarrow aMb \Rightarrow aaMb \Rightarrow aa\epsilon b$
  3.  $S \Rightarrow aMb \Rightarrow aaMb \Rightarrow aabMb \Rightarrow aab\epsilon b$
  4. ...
- Beachte: Beim ersetzen eines Nonterminals  $M$  spielt es keine Rolle, in welchem **Kontext** auf der rS  $M$  erscheint.
- $S \Rightarrow aMb \Rightarrow aaMb \Rightarrow aabMb \Rightarrow aab\epsilon b$ 
  1. Kontext von  $M$  in  $aMb$  ist  $a\_b$
  2. Kontext von  $M$  in  $aaMb$  ist  $aa\_b$
  3. Kontext von  $M$  in  $aabMb$  ist  $aab\_b$
- Man ersetzt  $M$  jedesmal, ohne auf den Kontext Bezug zu nehmen.

## Kontextfreie Grammatiken 2

- Daher nennt man eine solche Grammatik **kontextfrei**.
- Die Regeln einer kontextfreien Grammatik (**KfG**) haben
  1. genau ein Nonterminalsymbol auf der lS
  2. eine Kette von Terminal- oder Nonterminalsymbolen auf der rS

## Rekursion

- Beachte: Das Nonterminal  $M$  erscheint in Regeln 2. und 3. sowohl auf der lS als auch auf der rS. Dies nennt man **Rekursion**.
- $G$ :
  1.  $S \rightarrow aMb$
  2.  $M \rightarrow aM$
  3.  $M \rightarrow bM$
  4.  $M \rightarrow \epsilon$
- Durch Rekursion können Grammatiken unendliche Sprachen (mit endlichen Mitteln) erzeugen.

## Rekursion 2

- Rekursion kann unendliche Sprachen erzeugen, weil eine rekursive Regel  $R$  in einer Derivation beliebig oft angewandt werden kann.
- Da jede Anwendung von  $R$  die Kette  $K$  erweitert, erhält man beliebig viele Ketten (Wörter) verschiedener Länge
- Drei verwandte Konzepte
  1. Rekursion (Grammatik)
  2. Kleene-Stern (regulärer Ausdruck)
  3. Schleife (Automat)

## Formale Spezifikation von KfGn

- Eine kontextfreie Grammatik  $G$  ist ein 4-Tupel  $(V, \Sigma, R, S)$ , wobei
  1.  $V$  ein Alphabet ist
  2.  $\Sigma \subseteq V$  das Alphabet der Terminalsymbole ist
  3.  $R \subseteq (V - \Sigma) \times V^*$  eine endliche Menge von Regeln ist
  4.  $S \in (V - \Sigma)$  das Startsymbol ist
- Für jedes  $A \in (V - \Sigma)$  und  $u \in V^*$  bedeutet  $A \rightarrow_G u$ , dass  $(A, u) \in R$ .

## Formale Spezifikation von KfGn 2

- Für beliebige Ketten  $u, v \in V^*$  bedeutet  $u \Rightarrow_G v$ , dass es  $x, y \in V^*$  und ein  $A \in (V - \Sigma)$  gibt, so dass  $u = xAy$  und  $A \rightarrow_G v'$  und  $v = xv'y$ .
- $\Rightarrow_G^*$  ist der reflexiv transitive Abschluss von  $\Rightarrow_G$ .
- $L(G) =$  die Sprache, die von  $G$  **generiert** wird  $= \{w \in \Sigma^* \mid S \Rightarrow_G^* w\}$ .  $G$  generiert jede Kette in  $L(G)$ .
- Eine Sprache  $L$  ist eine kontextfreie Sprache, gdw.  $L = L(G)$  für eine kontextfreie Grammatik  $G$ .

## Formale Spezifikation von kontextfreien Derivationen

- Jede Sequenz der Form  $w_0 \Rightarrow_G w_1 \Rightarrow_G \dots \Rightarrow_G w_n$  ist eine Derivation von  $w_n$  in  $G$  von  $w_0$  aus.
- $n$  ist die **Länge** der Derivation. Man sagt auch, die Derivation hat  $n$  Schritte.
- Eine Derivation kann die Länge 0 haben.

## Beispielgrammatiken

- Grammatik  $G_1$  generiert  $\{a^n b^n \mid n \geq 0\}$ 
  1.  $G_1 = \{V, \Sigma, R, S\}$  mit
  2.  $V = \{S, a, b\}$
  3.  $\Sigma = \{a, b\}$
  4.  $R = \{S \rightarrow aSb, S \rightarrow \epsilon\}$
- Grammatik  $G_2$  generiert  $\{w \mid w \in \{a, b\}^* \text{ und } w = w^R\}$  (die Sprache der Palindrome)
  1.  $G_2 = \{V, \Sigma, R, S\}$  mit
  2.  $V = \{S, a, b\}$
  3.  $\Sigma = \{a, b\}$
  4.  $R = \{S \rightarrow aSa, S \rightarrow bSb, S \rightarrow a, S \rightarrow b, S \rightarrow \epsilon\}$

## Beispielgrammatiken 2

- Grammatik  $G_3$  generiert  $\{a^i b^j \mid i > j\}$ 
  1.  $G_3 = \{V, \Sigma, R, S\}$  mit
  2.  $V = \{S, a, b\}$
  3.  $\Sigma = \{a, b\}$
  4.  $R = \{S \rightarrow aSb, S \rightarrow aS, S \rightarrow a\}$
- Grammatik  $G_4$  generiert die Sprache der **ausbalancierten Klammern**.
  1.  $G_4 = \{V, \Sigma, R, S\}$  mit
  2.  $V = \{S, (, )\}$
  3.  $\Sigma = \{(, )\}$
  4.  $R = \{S \rightarrow \epsilon, S \rightarrow SS, S \rightarrow (S)\}$

## Ambiguität

- Beachte: Das Wort  $()(())$  kann mit  $G_4$  auf verschiedenen Wegen generiert werden.
  1.  $S \Rightarrow SS \Rightarrow S(S) \Rightarrow S((S)) \Rightarrow S(()) \Rightarrow (S)(()) \Rightarrow ()(())$
  2.  $S \Rightarrow SS \Rightarrow (S)S \Rightarrow ()S \Rightarrow ()(S) \Rightarrow ()((S)) \Rightarrow ()(())$
  3.  $S \Rightarrow SS \Rightarrow (S)S \Rightarrow (S)(S) \Rightarrow ()(S) \Rightarrow ()((S)) \Rightarrow ()(())$
- Es gibt noch mehr Wege  $()(())$  mit  $G_4$  abzuleiten.

## Kontextfreie und Reguläre Sprachen

- Theorem: Die Menge der regulären Sprachen ist echt enthalten in der Menge der kontextfreien Sprachen.
- Anders: Jeder reguläre Sprache ist auch kontextfrei, aber nicht jede kontextfreie Sprache ist regulär.
- Betrachte die reguläre Sprache  $L$ , die von einem DEA  $M = \{K, \Sigma, \delta, s, F\}$  akzeptiert wird.
- Dieselbe Sprache  $L$  wird dann von einer Grammatik  $G_M$  generiert, mit  $G_M = \{V, \Sigma, R, S\}$ , wobei
  1.  $V = K \cup \Sigma$
  2.  $S = s$
  3.  $R = \{q \rightarrow ap \mid \delta(q, a) = p\} \cup \{q \rightarrow \epsilon \mid q \in F\}$
- Jede rS der Regeln von  $G_M$  besteht aus einer Folge von Terminalsymbolen, gefolgt von **höchstens einem** Nichtterminalsymbol. Man nennt eine solche Grammatik **regulär**.

## Kontextfreie und Reguläre Sprachen 2

- Reguläre Grammatiken haben nur Regeln von der folgenden Form ( $A, B \in (V - \Sigma)$  und  $\alpha \in \Sigma^*$ ):
  1.  $A \rightarrow \alpha B$
  2.  $A \rightarrow \alpha$oder
  3.  $A \rightarrow B\alpha$
  4.  $A \rightarrow \alpha$
- Eine Grammatik mit Regeln der Art 1. und 2. heißt **rechtslinear**, eine Grammatik mit Regeln der Art 3. und 4. **linkslinear** (je nachdem, ob das Nonterminal auf der rS ganz links oder ganz rechts auftaucht).
- Wenn  $A = B$  spricht man von **Endrekursion** (auch, wenn  $B$  am Beginn der rS steht).
- Rechtslineare und linkslineare Grammatiken sind äquivalent.

## Reguläre Sprachen und Grammatiken

- Theorem: Jede reguläre Sprache wird von einer regulären Grammatik erzeugt und jede Sprache, die von einer regulären Grammatik erzeugt wird, ist regulär.
- Der erste Teil wurde schon diskutiert. Zu klären: wie konstruiert man für beliebige Sprache  $L$ , die von einer regulären Grammatik erzeugt wird, einen endlichen Automaten, der  $L$  akzeptiert.
- Wir beschränken uns auf eine rechtslineare Grammatik  $G = \{V, \Sigma, R, S\}$ .
- Es ist praktisch, einen NDEA zu konstruieren und dabei Übergänge zu betrachten, die ganze Symbolketten auf einmal einlesen (anstatt einzelner Symbole).
- Im Prinzip ist so ein Automat äquivalent zu einem DEA, der einzelne Symbole einliest.

## Reguläre Sprachen und Grammatiken 2

- Konstruiere den NDEA  $M = \{K, \Sigma, \Delta, s, F\}$  mit
  1.  $K = (V - \Sigma) \cup \{final\}$
  2.  $s = S$
  3.  $F = \{final\}$
- Jede Regel  $r \in R$  ist von der Form 1. oder 2.:
  1.  $q \rightarrow w, w \in \Sigma^*$
  2.  $q \rightarrow wA, w \in \Sigma^*, A \in (V - \Sigma)$
- Erweitere  $\Delta$  für jede Regel von Typ 1 um den Übergang  $(q, w, final)$ .
- Erweitere  $\Delta$  für jede Regel von Typ 2 um den Übergang  $(q, w, A)$ .

## Äquivalente Derivationen

- Erinnerung:  $L(G_4) =$  Sprache der **ausbalancierten Klammern**.

1.  $G_4 = \{V, \Sigma, R, S\}$  mit
2.  $V = \{S, (, )\}$
3.  $\Sigma = \{(, )\}$
4.  $R = \{S \rightarrow \epsilon, S \rightarrow SS, S \rightarrow (S)\}$

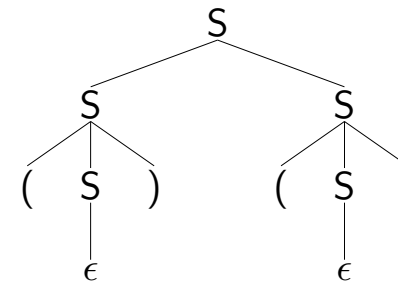
- Es gibt wenigstens zwei verschiedene, aber **äquivalente** Derivationen, die  $()()$  ableiten.

1.  $S \Rightarrow SS \Rightarrow (S)S \Rightarrow ()S \Rightarrow ()(S) \Rightarrow ()()$
2.  $S \Rightarrow SS \Rightarrow S(S) \Rightarrow S() \Rightarrow (S)() \Rightarrow ()()$

- Die äquivalenten Derivationen sind demselben **Derivations-** oder **Ableitungsbaum** zugeordnet.

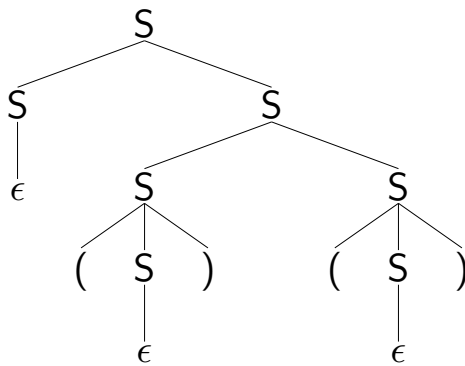
## Ableitungsäume

- Der Ableitungsbaum für den Ausdruck  $()()$ :



## Ableitungsbäume 2

- Es gibt **unendlich** viele **nicht äquivalente** Ableitungen für den Ausdruck  $()()$ .
- Der Ableitungsbaum für eine solche Ableitung ist folgender.



## Ableitungsbäume 3

- Ableitungsbäume bestehen aus **Knoten**, die wiederum durch **Kanten** miteinander verbunden sind.
- Die Knoten tragen **Etiketten (Labels)**.
- Der oberste Knoten des Baumes ist die **Wurzel**.
- Die untersten Knoten des Baumes sind die **Blätter**.
- Blätter werden nur von Nichtterminalsymbolen etikettiert.
- Wenn man die Blätter des Baumes von links nach rechts **konkateniert** (hintereinanderhängt), dann erhält man die abgeleitete Terminalkette (das Wort, auch **Yield**) genannt.

## Linsrandinge und Rechtsrandige Ableitungen

- Innerhalb einer Klasse von äquivalenten Derivationen kann man unterscheiden zwischen
  1. einer **linksrandigen** Derivation
  2. einer **rechtsrandigen** Derivation
- Eine linksrandige Derivation ersetzt in einem Ableitungsbaum immer das Nonterminalsymbol, welches am weitesten links steht. Entsprechend verhält es sich mit der rechtsrandigen Derivation.
- Linksrandige Derivation für  $()()$ , basierend auf dem ersten Ableitungsbaum:  $S \Rightarrow SS \Rightarrow (S)S \Rightarrow ()S \Rightarrow ()(S) \Rightarrow ()()$
- Rechtsrandige Derivation für  $()()$ , basierend auf dem ersten Ableitungsbaum:  $S \Rightarrow SS \Rightarrow S(S) \Rightarrow S() \Rightarrow (S)() \Rightarrow ()()$

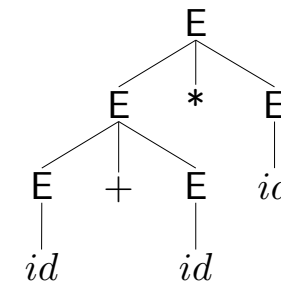
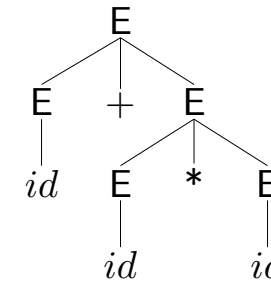
## Linsrandinge und Rechtsrandige Ableitungen 2

- $x \xRightarrow{L} y$  bedeutet, dass  $x \Rightarrow y$  Teil einer linksrandigen Derivation ist.
- $x \xRightarrow{L} y$  gdw.  $x = wA\beta, y = w\alpha\beta$ , wobei  $w \in \Sigma^*, \alpha, \beta \in V^*, A \in (V - \Sigma)$  und  $A \rightarrow \alpha$  ist eine Regel in  $G$ .
- Theorem: Sei  $G = \{V, \Sigma, R, S\}$  ein kontextfreie Grammatik,  $A \in (V - \Sigma)$  und  $w \in \Sigma^*$ . Die folgenden Aussagen sind dann äquivalent.
  1.  $A \Rightarrow^* w$
  2. Es gibt einen Ableitungsbaum mit Wurzel  $A$  und Terminalkette  $w$ .
  3. Es gibt eine linksrandige Derivation  $A \xRightarrow{L}^* w$
  4. Es gibt eine rechtsrandige Derivation  $A \xRightarrow{R}^* w$

## Ambiguität

- Eine Kette nennt man **ambig** gdw. es verschiedene Ableitungsbäume dafür gibt (und **nicht**, wenn es verschiedene Ableitungen dafür gibt!)
- **Ambiguität** ist wichtig, da sie die Bedeutung von Wörtern ändern kann.
- Eine Grammatik  $G$  ist **ambig**, wenn sie ambige Ketten generiert.
- $G_4$  war ambig. Weiteres Beispiel für eine ambige Grammatik ist  $G_5 = \{V, \Sigma, R, S\}$  mit
  1.  $V = \{E, (, ), +, *, id\}$
  2.  $\Sigma = \{(, ), +, *, id\}$
  3.  $R = \{E \rightarrow E + E, E \rightarrow E * E, E \rightarrow (E), E \rightarrow id\}$
  4.  $S = E$
- Die Kette  $id + id * id \in L(G_5)$  ist ambig.

## Ambiguität



- $2 + (5 * 10) = 100 \neq 70 = (2 + 5) * 10$

## Kellerautomaten

- Ein **Kellerautomat** (engl. **Push Down Automaton, PDA**) sind wie ein endlicher Automat, nur dass er einen **Speicher** hat.
- Der Speicher eines PDAs liegt in Form eines **Stapels** vor.
- Der Stapel kann beliebig hoch (oder tief) sein, aber der PDA kann immer nur das **höchste Element des Stapels** lesen.
- Damit gehorcht der Speicher des PDA dem Prinzip: Was zuletzt auf dem Stapel abgelegt wurde, wird auch zuerst wieder weggenommen (last in, first out).
- Der Kellerautomat hat ebenfalls ein Band, auf dem die Eingabe steht, und er liest mit seinem Lesekopf ein Eingabesymbol nach dem anderen (wie (N)DEAs).

## Kellerautomaten 2

- Die Operation,
  1. die etwas vom Stapel entfernt, nennt man **Pop**
  2. die etwas auf den Stapel schiebt, nennt man **Push**
- Der Stapel kann während eines Schritts, den der Automat macht, auch **unverändert** bleiben.
- Die PDAs, die wir hier betrachten, wechseln ihren Zustand **nichtdeterministisch** in Abhängigkeit von
  1. der Eingabe,
  2. des aktuellen Zustands
  3. dem obersten Element auf dem Stapel

## Nichtdeterministischer PDA: Formale Spezifikation

- Ein nichtdeterministischer Kellerautomat ist ein 7-Tupel  $(K, \Sigma, \Gamma, \Delta, s, F, \#)$ , wobei
  1.  $K$  die endliche Zustandsmenge ist
  2.  $\Sigma$  das Eingabealphabet ist
  3.  $\Gamma$  das **Stapelalphabet** ist
  4.  $s$  der Anfangszustand ist
  5.  $F \subseteq K$  die Menge der finalen Zustände ist
  6.  $\Delta$  die Übergangsrelation ist.  $\Delta$  ist eine endliche Teilmenge von  $(K \times \Sigma \times \Gamma) \times (K \times \Gamma^*)$
  7.  $\#$  das **Bodensymbol** des Stapels ist.

## Nichtdeterministischer PDA: Formale Spezifikation 2

- Diese mengentheoretische Beschreibung eines PDA kann nun als **Algorithmus** interpretiert werden (wie bei den endlichen Automaten).
- Annahme: Der Algorithmus kann pro Schritt
  1. nur **ein Eingabesymbol** lesen
  2. nur **ein Stapelsymbol** lesen
  3. aber keine Ketten lesen
- Achtung:  $\epsilon$  bezeichnet üblicherweise die **Kette der Länge 0**.
- Annahmen:
  1.  $\#$  liegt bei Beginn der Berechnung immer auf dem Stapel.
  2. Der Stapel gilt als leer, wenn er nur  $\#$  enthält.

## Nichtdeterministischer PDA: Formale Spezifikation 3

- Ein **Übergang** eines PDA  $M$  ist irgendein  $((p, a, Z), (q, \gamma)) \in \Delta$ , wobei  $p, q \in K, a \in \Sigma, Z \in \Gamma$  und  $\gamma \in \Gamma^*$ .
- Der Übergang  $((p, a, Z), (q, \gamma Z))$ 
  1. liebt das Eingabesymbol  $a$
  2. ersetzt das oberste Stapelsymbol  $Z$  durch die Kette  $\gamma Z \in \Gamma^*$  (**Push** von  $\gamma$ ).
- Der Übergang  $((p, a, Z), (q, Z))$ 
  1. liebt das Eingabesymbol  $a$
  2. ersetzt das oberste Stapelsymbol  $Z$  durch die einelementige Kette  $Z \in \Gamma^*$  (der Stapel bleibt unverändert; wie oben, nur dass  $\gamma = \epsilon$ )

## Nichtdeterministischer PDA: Formale Spezifikation 4

- Der Übergang  $((p, a, Z), (q, \epsilon))$ 
  1. liebt das Eingabesymbol  $a$
  2. ersetzt das oberste Stapelsymbol  $Z$  durch die leere Kette  $\epsilon \in \Gamma^*$  (**Pop** von  $Z$ ).
- Beachte: Man kann  $\epsilon$  nur an der letzten Stelle der Übergangsrelation verwenden ( $\epsilon \in \Gamma^*$ ), weil nur dort von Ketten die Rede ist ( $\epsilon$  **ist die Kette der Länge 0**).

## Nichtdeterministischer PDA: Konfigurationen

- Eine **Konfiguration** eines PDAs ist ein Element aus  $K \times \Sigma^* \times \Gamma^*$ .
- An jedem Punkt ist nur wichtig
  1. was die Eingabe ist
  2. was das oberste Element auf dem Stapel ist
  3. was der aktuelle Zustand ist
- Die Konfiguration  $(q, w, abc)$  bedeutet, dass
  1. der aktuelle Zustand  $q$  ist
  2. der Rest der Eingabe  $w$  ist
  3. der Stapel  $abc$  enthält

## Nichtdeterministischer PDA: Konfigurationen 2

- Die Relation  $\vdash_M$  verbindet zwei Konfigurationen eines PDA  $M$  in einem Schritt:
  1. Seien  $(p, x, \alpha)$  und  $(q, y, \zeta) \in K \times \Sigma^* \times \Gamma^*$
  2. Dann gilt  $(p, x, \alpha) \vdash_M (q, y, \zeta)$ , wenn es einen Übergang  $((p, a, Z), (q, \gamma)) \in \Delta$  gibt, so dass
    - (a)  $x = ay$
    - (b)  $\alpha = Z\xi$
    - (c)  $\zeta = \gamma\xi$für irgendein  $\xi \in \Gamma^*$ .
- Die Relation  $\vdash_M^*$  verbindet zwei Konfigurationen eines PDA  $M$  in beliebig vielen (inklusive 0) Schritten:  
 $(p, x, \alpha) \vdash_M^* (q, y, \zeta)$

## Nichtdeterministischer PDA: Konfigurationen 3

- $M$  akzeptiert  $w$ , gdw.  $(s, w, \epsilon) \vdash_M^* (q, \epsilon, \#)$ , für ein  $q \in F$ .
- Äquivalent:  $M$  akzeptiert  $w$ , gdw. es eine Sequenz  $C_0, C_1, \dots, C_n$  mit  $n \geq 0$  gibt, so dass  $C_0 \vdash_M C_1 \vdash_M \dots \vdash_M C_n$  und  $C_0 = (s, w, \epsilon)$  und  $C_n = (q, \epsilon, \#)$  für ein  $q \in F$ .
- Eine solche Sequenz nennt man eine **Berechnung** von  $M$  in  $n$  **Schritten** (von der **Länge**  $n$ ).
- Die Sprache  $L(M)$ , die von  $M$  akzeptiert wird, ist  $\{w \mid \text{Es gibt } q \in F \text{ mit } (s, w, \epsilon) \vdash_M^* (q, \epsilon, \#)\}$
- $M$  akzeptiert also ein Wort  $w$ , wenn
  1.  $M$  in einem finalen Zustand ist
  2.  $w$  komplett gelesen wurde
  3. der Stapel geleert ist (bis auf  $\#$ )

## Nichtdeterministischer PDA: Konfigurationen 4

- Man kann Akzeptanz auch definieren durch Ende der Eingabe und alternativ
  1. einen leeren Stapel oder
  2. einen finalen Zustand
- Alle drei Möglichkeiten (leerer Stapel, finaler Zustand oder beides) sind äquivalent.

## Nichtdeterministischer PDA: Konfigurationen 5

## Graphen für PDAs

- Ist das Akzeptanzkriterium
    1. der **leere Stapel**, dann muss der Algorithmus prüfen, ob der Stapel leer ist.
    2. die **Finalität** des Zustands, dann muss der Algorithmus prüfen, ob der aktuelle Zustand final ist.
    3. **beides**, dann muss der Algorithmus prüfen, ob der Stapel leer ist und, ob der aktuelle Zustand final ist.
  - In jedem Fall muss geprüft werden, ob die Eingabe zu Ende ist, sobald der Algorithmus anhält.
  - Der Algorithmus hält an, wenn er nicht mehr von einer Konfiguration in die nächste wechseln kann.
- Konvention für Automatengraph: Man kann  $((q_1, a, b), (q_2, c))$  repräsentieren durch
    1. einen Pfeil zwischen den Zuständen  $q_1$  und  $q_2$
    2. setzen das Label  $a, b \rightarrow c$
    3. und interpretieren dies als:
      - (a) beim Lesen des Eingabesymbols  $a$
      - (b) wird das oberste Stapelsymbol  $b$  durch  $c$  ersetzt
      - (c) der Automat geht von Zustand  $q_1$  nach  $q_2$

## Beispielautomaten

- $L(M_1) = \{a^n b^n \mid n \geq 0\}$ .  $M_1 = (K, \Sigma, \Gamma, \Delta, q_1, F, \#)$ , wobei

1.  $K = \{q_1, q_2\}$
2.  $\Sigma = \{a, b\}$
3.  $\Gamma = \{a, \#\}$
4.  $F = \{q_1, q_2\}$
5.  $\Delta = \{ ((q_1, a, \#), (q_1, a\#)), ((q_1, a, a), (q_1, a, aa)), ((q_1, b, a), (q_2, \epsilon)), ((q_2, b, a), (q_2, \epsilon)) \}$

## Beispielautomaten 2

- Kommentare:

1.  $M_1$  schiebt so lang  $as$  auf den Stapel, wie er  $as$  in der Eingabe liest ( $M_1$  kann in  $q_1$  keine  $bs$  lesen).
2. Liest  $M_1$  das erste  $b$ , dann wechselt er in Zustand  $q_2$  und löscht ein  $a$  vom Stapel.
3. Für jedes weitere  $b$  wird ein  $a$  vom Stapel gelöscht; der Zustand bleibt  $q_2$  ( $M_1$  kann in  $q_2$  keine  $as$  lesen).
4. Im Zustand  $q_1$  kann  $M_1$  also nur  $as$  lesen, im Zustand  $q_2$  nur  $bs$ .
5. Enthält die Eingabe weniger  $bs$  als  $as$ , dann wird der Stapel nicht leer.
6. Enthält die Eingabe mehr  $bs$  als  $as$ , dann wird die Eingabe nicht leer.
7. Das Symbol  $\#$  ist hier notwendig, weil man sonst **am Anfang** nicht über den leeren Stapel sprechen kann.

## Nichtdeterministische PDAs: eine Variante ohne #

- Man kann die Übergangsrelation auf Ketten generalisieren, so dass in einem Schritt
  1. eine **Kette** von Eingabesymbolen gelesen werden kann
  2. eine **Kette** von Stapelsymbolen gelesen (und ersetzt) werden kann
- Das verändert nicht die Leistungsfähigkeit des Automaten, macht aber die Beschreibung leichter.
- Insbesondere kann dann auf das Bodensymbol # verzichtet werden.

## Nichtdeterministische PDAs: eine Variante ohne # 2

- Ein nichtdeterministischer Kellerautomat ist ein 6-Tupel  $(K, \Sigma, \Gamma, \Delta, s, F)$ , wobei
  1.  $K$  die endliche Zustandsmenge ist
  2.  $\Sigma$  das Eingabealphabet ist
  3.  $\Gamma$  das **Stapelalphabet** ist
  4.  $s$  der Anfangszustand ist
  5.  $F \subseteq K$  die Menge der finalen Zustände ist
  6.  $\Delta$  die Übergangsrelation ist.  $\Delta$  ist eine endliche Teilmenge von  $(K \times \Sigma^* \times \Gamma^*) \times (K \times \Gamma^*)$

## Nichtdeterministische PDAs: eine Variante ohne # 3

- Ein **Übergang** eines PDA  $M$  ist irgendein  $((p, u, \alpha), (q, \beta)) \in \Delta$ , wobei  $p, q \in K, u \in \Sigma^*, \alpha, \beta \in \Gamma^*$ .
- Der Übergang  $((p, u, \alpha), (q, \gamma\alpha))$ 
  1. liebt das Eingabeprefix  $u$
  2. ersetzt das Stapelpräfix  $\alpha \in \Gamma^*$  durch die Kette  $\gamma\alpha \in \Gamma^*$  (**Push** von  $\gamma$ ).
- Der Übergang  $((p, \epsilon, \alpha), (q, \gamma\alpha))$ 
  1. liebt das leere Eingabeprefix  $\epsilon$  (= ignoriert die Eingabe)
  2. ersetzt das Stapelpräfix  $\alpha \in \Gamma^*$  durch die Kette  $\gamma\alpha \in \Gamma^*$  (**Push** von  $\gamma$ ).

## Nichtdeterministische PDAs: eine Variante ohne # 4

- Der Übergang  $((p, u, \epsilon), (q, \gamma))$ 
  1. liebt das Eingabeprefix  $u$
  2. ersetzt das Stapelpräfix  $\epsilon \in \Gamma^*$  durch die Kette  $\gamma \in \Gamma^*$  (**Push** von  $\gamma$ , wobei der Stapel ignoriert wird).
- Der Übergang  $((p, u, \alpha), (q, \epsilon))$ 
  1. liebt das Eingabeprefix  $u$
  2. ersetzt das Stapelpräfix  $\alpha \in \Gamma^*$  durch die leere Kette  $\epsilon \in \Gamma^*$  (**Pop** von  $\alpha$ ).
- Der Übergang  $((p, u, \epsilon), (q, \epsilon))$ 
  1. liebt das Eingabeprefix  $u$
  2. ersetzt das leere Stapelpräfix  $\epsilon \in \Gamma^*$  durch die leere Kette  $\epsilon \in \Gamma^*$  (der Stapel bleibt unverändert).

## Beispielautomaten 3

- $L(M'_1) = \{a^n b^n \mid n \geq 0\}$ .  $M'_1 = (K, \Sigma, \Gamma, \Delta, q_1, F)$ ,  
wobei

1.  $K = \{q_1, q_2\}$
2.  $\Sigma = \{a, b\}$
3.  $\Gamma = \{a\}$
4.  $F = \{q_1, q_2\}$
5.  $\Delta' = \{ ((q_1, a, \epsilon), (q_1, a)),$   
 $((q_1, b, a), (q_2, \epsilon)),$   
 $((q_2, b, a), (q_2, \epsilon)) \}$

## Beispielautomaten 4

- $L(M_2) = \{a^i b^j c^k \mid i, j, k \geq 0 \text{ und } i = j \text{ oder } i = k\}$   
und  $M_2 = (K, \Sigma, \Gamma, \Delta, q_1, F)$ , wobei

1.  $K = \{q_1, q_2, q_3, q_4, q_5, q_6, q_7\}$
2.  $\Sigma = \{a, b, c\}$
3.  $\Gamma = \{a, \$\}$
4.  $F = \{q_4, q_7\}$
5.  $\Delta = \{ ((q_1, \epsilon, \epsilon), (q_2, \$)), ((q_2, a, \epsilon), (q_2, a)),$   
 $((q_2, \epsilon, \epsilon), (q_3, \epsilon)), ((q_2, \epsilon, \epsilon), (q_5, \epsilon)),$   
 $((q_3, b, a), (q_3, \epsilon)), ((q_3, \epsilon, \$), (q_4, \epsilon)),$   
 $((q_4, c, \epsilon), (q_4, \epsilon)), ((q_5, b, \epsilon), (q_5, \epsilon)),$   
 $((q_5, \epsilon, \epsilon), (q_6, \epsilon)), ((q_6, c, a), (q_6, \epsilon)),$   
 $((q_6, \epsilon, \$), (q_7, \epsilon)) \}$

## Beispielautomaten 5

- $L(M_3) = \{ww^R \mid w \in \{a, b\}^*\}$  und  $M_3 = (K, \Sigma, \Gamma, \Delta, q_1, F)$ , wobei
  1.  $K = \{q_1, q_2, q_3, q_4\}$
  2.  $\Sigma = \{a, b\}$
  3.  $\Gamma = \{a, b, \$\}$
  4.  $F = \{q_1, q_4\}$
  5.  $\Delta = \{ ((q_1, \epsilon, \epsilon), (q_2, \$)), ((q_2, a, \epsilon), (q_2, a)), ((q_2, \epsilon, \epsilon), (q_3, \epsilon)), ((q_2, b, \epsilon), (q_2, b)), ((q_3, a, a), (q_3, \epsilon)), ((q_3, b, b), (q_3, \epsilon)), ((q_3, \epsilon, \$), (q_4, \epsilon)) \}$

## Endliche Automaten und PDAs

- Bereits gezeigt: Die regulären Sprachen sind eine echte Teilmenge der kontextfreien Sprachen.
- Neu: Die regulären Sprachen sind auch eine echte Teilmenge der Sprachen, die von PDAs akzeptiert werden.
  1. Ein PDA ist einfach ein EA mit Stapel.
  2. Ein EA ist ein PDA, der von seinem Stapel keinen Gebrauch macht.
- Umwandlung eines EAs in einen PDA:
  1. Sei  $M = \{K, \Sigma, \Delta, s, F\}$  ein NEA.
  2. Dann ist  $M' = \{K, \Sigma, \emptyset, \Delta', s, F\}$  ein äquivalenter PDA, wobei
  3.  $\Delta' = \{((p, u, \epsilon), (q, \epsilon)) \mid (p, u, q) \in \Delta\}$
- Erinnerung: Ein Übergang  $((p, u, \epsilon), (q, \epsilon))$  eines PDAs lässt den Stapel unverändert.

## Varianten des Akzeptierens

- Die Sprache, die durch
  1. Erreichen eines finalen Zustands akzeptiert wird:  
 $\{w \mid \text{Es gibt } p, \gamma, \text{ mit } p \in F \text{ und } \gamma \in \Gamma^* \text{ und } (s, w, \epsilon) \vdash_M^* (p, \epsilon, \gamma)\}$
  2. Erreichen des leeren Stapels akzeptiert wird:  
 $\{w \mid \text{Es gibt } p \in K \text{ und } (s, w, \epsilon) \vdash_M^* (p, \epsilon, \epsilon)\}$
- Theorem: Folgende Sprachenmengen sind gleich:
  1. Sprachen, die von einem PDA durch Erreichen eines **finalen Zustands** akzeptiert werden.
  2. Sprachen, die von einem PDA durch Erreichen des **leeren Stapels** erzeugt werden.
  3. Sprachen, die von einem PDA durch Erreichen eines **finalen Zustands** und des **leeren Stapels** akzeptiert werden.

## Varianten des Akzeptierens 2

- Theorem: Wenn  $L$  von einem PDA  $M$  durch Erreichen eines **finalen Zustandes** akzeptiert wird, dann gibt es einen PDA  $M_1$ , so dass  $L$  von  $M_1$  durch Erreichen des **leeren Stapels** akzeptiert wird.
- Idee:  $M_1$  **simuliert**  $M$  und hat die Option seinen Stapel zu leeren, sobald die Simulation von  $M$  in einem finalen Zustand ist.
- Strategie:
  1. Füge einen Zustand  $q_e$  zu  $M_1$  hinzu, um den Stapel zu leeren.
  2. Füge ein neues Stapelsymbol  $X_0$  zu  $M_1$ , welches den Boden des Stapels markiert.
  3. Damit wird verhindert, dass  $M_1$  versehentlich ein Wort  $w$  akzeptiert, weil  $M$  seinen Stapel geleert hat, ohne  $w$  zu akzeptieren.

## Varianten des Akzeptierens 3

- **Konstruktion:**
- Sei  $M = (K, \Sigma, \Gamma, \Delta, s, F)$ , mit  $Z_0$  als Stapelsymbol, welches den Boden des Stapels markiert.
- Sei  $M_1 = (K \cup \{q_e, s'\}, \Sigma, \Gamma \cup \{X_0\}, \Delta', s', \emptyset)$ , mit  $\Delta'$  wie folgt:
  1.  $((s', \epsilon, X_0), (s, \epsilon, Z_0X_0)) \in \Delta'$   
(\* $M_1$  geht in  $M$ s Anfangskonfiguration\*)
  2. Alle Übergänge mit  $q \in K, u \in \Sigma^*, \gamma \in \Gamma^*$ , die in  $\Delta$  sind, sind auch in  $\Delta'$ .  
(\* $M_1$  simuliert  $M$ \*)
  3. Für alle  $q \in F, \gamma \in \Gamma^* \cup \{X_0\}$ :  $((q, \epsilon, \gamma), (q_e, \epsilon)) \in \Delta'$ .
  4. Für alle  $\gamma \in \Gamma^* \cup \{X_0\}$ :  $((q_e, \epsilon, \gamma), (q_e, \epsilon)) \in \Delta'$ .  
(\*Wenn  $M$  in finalem Zustand, dann geht  $M_1$  in Zustand  $q_e$  und leert seinen Stapel\*)
  5. Sonst ist nichts in  $\Delta'$ .

## Varianten des Akzeptierens 4

- Theorem: Wenn  $L$  von einem PDA  $M$  durch Erreichen des **leeren Stapels** akzeptiert wird, dann gibt es einen PDA  $M_1$ , so dass  $L$  von  $M_1$  durch Erreichen eines **finalen Zustandes** akzeptiert wird.
- Idee:
  1.  $M_1$  **simuliert**  $M$  und erkennt, wenn  $M$  seinen Stapel geleert hat.
  2.  $M$  geht in einen finalen Zustand über genau dann, wenn dies passiert (sonst nie).

## Varianten des Akzeptierens 5

- **Konstruktion:**
- Sei  $M = (K, \Sigma, \Gamma, \Delta, s, \emptyset)$ , mit  $Z_0$  als Stapelsymbol, welches den Boden des Stapels markiert.
- Sei  $M_1 = (K \cup \{s', q_f\}, \Sigma, \Gamma \cup \{X_0\}, \Delta', s', \{q_f\})$ , mit  $X_0$  als Bodensymbol des Stapels und mit  $\Delta'$  wie folgt:
  1.  $((s', \epsilon, X_0), (s, \epsilon, Z_0X_0)) \in \Delta'$   
(\* $M_1$  geht in  $M$ s Anfangskonfiguration\*)
  2. Alle Übergänge mit  $q \in K, u \in \Sigma^*, \gamma \in \Gamma^*$ , die in  $\Delta$  sind, sind auch in  $\Delta'$ .  
(\* $M_1$  simuliert  $M$ \*)
  3. Für alle  $q \in K$ :  $((q, \epsilon, X_0), (q_f, \epsilon)) \in \Delta'$ .  
(\*Wenn  $M$  seinen Stapel geleert hat, dann geht  $M_1$  in seinen finalen Zustand  $q_f$ \*)
  4. Sonst ist nichts in  $\Delta'$ .

## Varianten des Akzeptierens 6

- Theorem:  $L$  wird von einem PDA  $M$  durch Erreichen eines **finalen Zustandes** akzeptiert genau dann, wenn es es einen PDA  $M_1$  gibt, so dass  $L$  von  $M_1$  durch Erreichen des **leeren Stapels** und eines **finalen Zustandes** akzeptiert wird.
- Theorem:  $L$  wird von einem PDA  $M$  durch Erreichen des **leeren Stapels** akzeptiert genau dann, wenn es einen PDA  $M_1$  gibt, so dass  $L$  von  $M_1$  durch Erreichen des **leeren Stapels** und eines **finalen Zustandes** akzeptiert wird.

## PDA's und ktf Grammatiken

- Theorem: Eine Sprache  $L$  ist kontextfrei genau dann, wenn es einen PDA gibt, der  $L$  akzeptiert.
- Teil  $\Rightarrow$ : Wenn eine Sprache  $L$  kontextfrei ist, dann gibt es einen PDA, der  $L$  akzeptiert.
- Es wird ein Verfahren vorgestellt, um aus einer kontextfreien Grammatik  $G$  einen PDA  $M$  zu machen, so dass  $L(G) = L(M)$ .
- $M$  wird dabei eine linksrandige Derivation von  $G$  simulieren.

## KfG nach PDA

- Prozedur des Automaten:
- Lege das Bodensymbol  $\#$  und das Startsymbol der Grammatik auf den Stapel.
- Wiederhole:
  1. Wenn das oberste Symbol auf dem Stapel ein Nichtterminal  $A$  ist, dann ersetze  $A$  durch die rS einer Regel, von der  $A$  die lS ist, wobei die Regel **nichtdeterministisch** geraten wurde.
  2. Wenn das oberste Symbol auf dem Stapel ein Terminal  $a$  ist, dann entferne  $a$  vom Stapel.
  3. Wenn das oberste Symbol auf dem Stapel  $\#$  ist, dann gehe in den finalen Zustand.
  4. Die Eingabe ist akzeptiert, wenn sie vollständig gelesen ist und der Automat in einem finalen Zustand ist.

## KfG nach PDA 2

- Sei  $G = \{V, \Sigma, R, S\}$ .
- Sei  $M = (K, \Sigma, \Gamma, \Delta, s, F)$  ein PDA, wobei
  1.  $K = \{q_0, q_1, q_2\}$
  2.  $\Gamma = \{\#\} \cup V$
  3.  $s = q_0, F = \{q_2\}$
  4.  $\Delta$  ist definiert wie folgt:
    - (a)  $\Delta$  enthält den Übergang  $((q_0, \epsilon, \epsilon), (q_1, S\#))$ .
    - (b) Für jede Regel  $A \rightarrow \varphi$  in  $G$  gibt es einen Übergang  $((q_1, \epsilon, A), (q_1, \varphi))$  in  $M$ .
    - (c) Für jedes  $a \in \Sigma$  in  $G$  gibt es einen Übergang  $((q_1, a, a), (q_1, \epsilon))$  in  $M$ .
    - (d)  $\Delta$  enthält den Übergang  $((q_1, \epsilon, \#), (q_2, \epsilon))$ .

## PDA nach KfG

- Teil  $\Leftarrow$ : Wenn ein PDA eine Sprache  $L$  akzeptiert, dann ist  $L$  kontextfrei.
- Es wird ein Verfahren angegeben, das aus einem PDA  $M$  eine kontextfreie Grammatik  $G$  konstruiert, so dass  $L(M) = L(G)$ .
- Wandle  $M$  um in einen PDA  $M'$ , so dass  $M'$  nur bei leerem Stapel akzeptiert.
- Für jedes Paar von Zuständen  $p, q$  aus  $M'$  und jedes  $A \in \Gamma$  besitzt  $G$  ein Nichtterminalsymbol  $[p, A, q]$ .
- $[p, A, q]$  soll alle Ketten  $x$  generieren für die gilt:  
 $(p, x, A) \vdash_{M'}^* (q, \epsilon, \epsilon)$

## PDA nach KfG 2

- Sei  $M = (K, \Sigma, \Gamma, \Delta, q_0, F)$  und sei  $\#$  das Bodensymbol des Stapels (d.h.,  $\# \in \Gamma$ ).
- Dann ist  $G = \{V, \Sigma, R, S\}$ , wobei
  1.  $V = \{[q, A, p] \mid p, q \in K \text{ und } A \in \Gamma\} \cup \{S\}$
  2.  $R$  ist wie folgt definiert:
    - (a) Für jedes  $p \in K$  gibt es eine Regel  $S \rightarrow [q_0, \#, p]$ .
    - (b) Wenn  $((q, a, A), (q_1, \gamma)) \in \Delta$  wobei  $a \in \Sigma^*$ ,  $\gamma = B_1 B_2 \dots B_m$ , mit  $B_i \in \Gamma$  für  $1 \leq i \leq m$  und  $q, q_1 \dots q_{m+1} \in K$ , dann ist  $[q, A, q_{m+1}] \rightarrow a[q_1, B_1, q_2][q_2, B_2, q_3] \dots [q_m, B_m, q_{m+1}]$  eine Regel in  $R$ .
    - (c) Gilt  $m = 0$ , dann lautet die Regel  $[q, A, q_1] \rightarrow a$ .
- Mann müsste jetzt noch beweisen:  $[p, A, q] \Rightarrow^* w \iff (p, w, A) \vdash^* (q, \epsilon, \epsilon)$   
Das tun wir hier nicht.

## Beispielrechnung

- Sei  $M = (\{q_0, q_1\}, \{0, 1\}, \{X, \#\}, \Delta, q_0, \emptyset)$  und sei  $\#$  das Bodensymbol des Stapels.
- $\Delta$  ist folgendermaßen definiert:
 
$$\Delta = \left\{ \begin{array}{ll} ((q_0, 0, \#), (q_0, X\#)), & ((q_0, 0, X), (q_0, XX)), \\ ((q_0, 1, X), (q_1, \epsilon)), & ((q_1, 1, X), (q_1, \epsilon)), \\ ((q_1, \epsilon, X), (q_1, \epsilon)), & ((q_1, \epsilon, \#), (q_1, \epsilon)) \end{array} \right\}$$
- Dann kann man eine kfG  $G = (V, \Sigma, R, S)$  erzeugen, so dass  $L(G) = L(M)$ , wie folgt.
  1.  $V = \{S, [q_0, X, q_0], [q_0, X, q_1], [q_1, X, q_0], [q_1, X, q_1], [q_0, \#, q_0], [q_0, \#, q_1], [q_1, \#, q_0], [q_1, \#, q_1]\}$

## Beispielrechnung 2

- Es macht Sinn, nur die Regeln hinzuschreiben, deren IS-Symbol in einer mit  $S$  beginnenden Ableitung auftauchen kann.

2. Regeln für Startvariable  $S$ :

(a)  $S \rightarrow [q_0, \#, q_0]$

(b)  $S \rightarrow [q_0, \#, q_1]$

3. Regeln für Variable  $[q_0, \#, q_0]$  (ergeben sich durch Übergang  $((q_0, 0, \#), (q_0, X\#))$ ):

(a)  $[q_0, \#, q_0] \rightarrow 0[q_0, X, q_0][q_0, \#, q_0]$

(b)  $[q_0, \#, q_0] \rightarrow 0[q_0, X, q_1][q_1, \#, q_0]$

4. Regeln für Variable  $[q_0, \#, q_1]$  (ergeben sich ebenfalls durch Übergang  $((q_0, 0, \#), (q_0, X\#))$ ):

(a)  $[q_0, \#, q_1] \rightarrow 0[q_0, X, q_0][q_0, \#, q_1]$

(b)  $[q_0, \#, q_1] \rightarrow 0[q_0, X, q_1][q_1, \#, q_1]$

## Beispielrechnung 3

5. Regeln für Variablen  $[q_0, X, q_0]$  und  $[q_0, X, q_1]$  (ergeben sich durch Übergang  $((q_0, 0, X), (q_0, XX))$ ):

(a)  $[q_0, X, q_0] \rightarrow 0[q_0, X, q_0][q_0, X, q_0]$

(b)  $[q_0, X, q_0] \rightarrow 0[q_0, X, q_1][q_1, X, q_0]$

(c)  $[q_0, X, q_1] \rightarrow 0[q_0, X, q_0][q_0, X, q_1]$

(d)  $[q_0, X, q_1] \rightarrow 0[q_0, X, q_1][q_1, X, q_1]$

6. Regel für Variable  $[q_0, X, q_1]$  (ergibt sich durch Übergang  $((q_0, 1, X), (q_1, \epsilon))$ ):  $[q_0, X, q_1] \rightarrow 1$

7. Regel für Variable  $[q_1, \#, q_1]$  (ergibt sich durch Übergang  $((q_1, \epsilon, \#), (q_1, \epsilon))$ ):  $[q_1, \#, q_1] \rightarrow \epsilon$

8. Regel für Variable  $[q_1, X, q_1]$  (ergibt sich durch Übergang  $((q_1, \epsilon, X), (q_1, \epsilon))$ ):  $[q_1, X, q_1] \rightarrow \epsilon$

9. Regel für Variable  $[q_1, X, q_1]$  (ergibt sich durch Übergang  $((q_1, 1, X), (q_1, \epsilon))$ ):  $[q_1, X, q_1] \rightarrow 1$

## Beispielrechnung 4

- Beachte: Für die Variablen  $[q_1, X, q_0]$  und  $[q_1, \#, q_0]$  gibt es keine Regeln.
- Da die Regeln von  $[q_0, X, q_0]$  und  $[q_0, \#, q_0]$  jeweils  $[q_1, X, q_0]$  oder  $[q_1, \#, q_0]$  auf der rS haben, können diese Regeln keine Terminalketten erzeugen.
- Man kann alle Produktionen löschen, die eine dieser vier Variablen auf der lS oder rS haben und verbleibt dann mit folgenden Regeln:
  1.  $S \rightarrow [q_0, \#, q_1]$
  2.  $[q_0, \#, q_1] \rightarrow 0[q_0, X, q_1][q_1, \#, q_1]$
  3.  $[q_0, X, q_1] \rightarrow 0[q_0, X, q_1][q_1, X, q_1]$
  4.  $[q_0, X, q_1] \rightarrow 1$
  5.  $[q_1, X, q_1] \rightarrow \epsilon$
  6.  $[q_1, \#, q_1] \rightarrow \epsilon$
  7.  $[q_1, X, q_1] \rightarrow 1$

## Abschlusseigenschaften von kontextfreien Sprachen

- Die kontextfreien Sprachen sind abgeschlossen unter Vereinigung, Verkettung und Kleene-Stern.
- Seien  $G_1 = \{V_1, \Sigma_1, R_1, S_1\}$  und  $G_2 = \{V_2, \Sigma_2, R_2, S_2\}$  zwei kontextfreie Grammatiken, wobei  $(V_1 - \Sigma_1) \cap (V_2 - \Sigma_2) = \emptyset$ .
  1. **Vereinigung:** Sei  $G = \{V_1 \cup V_2 \cup \{S\}, \Sigma_1 \cup \Sigma_2, R, S\}$ , wobei  $R = R_1 \cup R_2 \cup \{S \rightarrow S_1, S \rightarrow S_2\}$ . Dann ist  $L(G) = L(G_1) \cup L(G_2)$ .
  2. **Verkettung:** Sei  $G = \{V_1 \cup V_2 \cup \{S\}, \Sigma_1 \cup \Sigma_2, R, S\}$ , wobei  $R = R_1 \cup R_2 \cup \{S \rightarrow S_1 S_2\}$ . Dann ist  $L(G) = L(G_1)L(G_2)$ .
  3. **Kleene-Stern:** Sei  $G = \{V_1 \cup \{S\}, \Sigma_1, R, S\}$ , wobei  $R = R_1 \cup \{S \rightarrow S S_1, S \rightarrow \epsilon\}$ . Dann ist  $L(G) = L(G_1)^*$ .

## Abschlusseigenschaften von kontextfreien Sprachen 2

- Theorem: Die kontextfreien Sprachen sind abgeschlossen unter Schnitt mit den regulären Sprachen, d.h., wenn  $L_1$  eine kfS ist und  $L_2$  eine reguläre Sprache, dann ist  $L_1 \cap L_2$  eine kfS.
- Konstruktion eines PDAs  $M$  aus einem PDA  $M_1 = (K_1, \Sigma, \Gamma_1, \Delta_1, s_1, F_1)$  und einem DEA  $M_2 = (K_2, \Sigma, \delta, s_2, F_2)$ , so dass  $L(M) = L(M_1) \cap L(M_2)$ .
- Sei  $M = (K, \Sigma, \Gamma, \Delta, s, F)$ , wobei
  1.  $K = K_1 \times K_2, \Gamma = \Gamma_1, s = (s_1, s_2), F = F_1 \times F_2$
  2.  $\Delta$  definiert ist wie folgt:
    - (a) Für jeden Übergang in  $M_1, ((q_1, a, \beta), (p_1, \gamma)) \in \Delta_1$  und für jeden  $q_2 \in K_2$ , füge zu  $\Delta$   $((q_1, q_2), a, \beta), ((p_1, \delta(q_2, a)), \gamma)$ .
    - (b) Für jeden Übergang in  $M_1, ((q_1, \epsilon, \beta), (p_1, \gamma)) \in \Delta_1$  und jeden  $q_2 \in K_2$ , füge zu  $\Delta$   $((q_1, q_2), \epsilon, \beta), ((p_1, q_2), \gamma)$ .

## Abschlusseigenschaften von kontextfreien Sprachen 3

- Anwendung: Man kann mit den Abschlusseigenschaften zeigen, dass eine Sprache kontextfrei sein muss.
- Beispiel:  $L = \{w \mid w \text{ enthält eine gerade Zahl von } as \text{ und } bs \text{ und enthält nicht die Teilkette } abaa\}$
- Bekannt:  $L_1 = \{w \mid w \text{ enthält eine gerade Zahl von } as \text{ und } bs\}$  ist kontextfrei.
- Bekannt:  $L_2 = \{w \mid w \text{ enthält nicht die Teilkette } abaa\}$  ist eine reguläre Sprache.
- Es folgt, dass  $L = L_1 \cap L_2$  eine kontextfreie Sprache ist.

## Abschlusseigenschaften von kontextfreien Sprachen 4

- KfSn sind nicht abgeschlossen unter Schnitt und Komplementbildung, d.h., wenn  $L_1$  und  $L_2$  kfSn sind, dann ist es nicht garantiert, dass  $L_1 \cap L_2$  oder  $\overline{L_1}$  eine kfS ist.
- Ein Gegenbeispiel bzgl. Schnitt:
  1. Gesehen:  $L_1 = \{a^i b^j c^k \mid i, j, k \geq 0 \text{ mit } i = j \text{ oder } i = k\}$  ist kontextfrei.
  2. Es folgt:
    - (a)  $L_2 = \{a^i b^j c^k \mid i, j, k \geq 0 \text{ und } i = j\}$  ist kf
    - (b)  $L_3 = \{a^i b^j c^k \mid i, j, k \geq 0 \text{ und } i = k\}$  ist kf
  3. Man kann zeigen, dass  $L_4 = \{a^i b^j c^k \mid i, j, k \geq 0 \text{ und } i = j = k\}$  **nicht** kontextfrei ist.
  4. Da  $L_4 = L_2 \cap L_3$  folgt, dass die kontextfreien Sprachen unter Schnitt nicht abgeschlossen sind.
- Nichtabgeschlossenheit unter Komplementbildung folgt aus den DeMorganschen Gesetzen.

## Determinismus

- Ein PDA  $M$  ist **deterministisch** genau dann, wenn es für jede Konfiguration  $K_1$  **höchstens** eine Konfiguration  $K_2$  gibt, so dass  $K_2$  auf  $K_1$  in einer Berechnung von  $M$  folgen kann.
- Frage: Sind deterministische und nichtdeterministische PDAs äquivalent?
- Antwort: Nein.
- Ein deterministischer PDA kann die Sprache  $\{ww^R \mid w \in \{a, b\}^*\}$  nicht erkennen.
- Allerdings kann die Sprache  $\{w c w^R \mid w \in \{a, b\}^*\}$  von einem deterministischen PDA erkannt werden.

## Deterministische kontextfreie Sprachen

- Eine Sprache  $L$  ist eine deterministische kontextfreie Sprache (**dkfS**) genau dann, wenn es einen deterministischen PDA (**DPDA**)  $M$  gibt, mit  $L(M) = L$ .
- Theorem: Die dkfSn sind abgeschlossen unter Komplementbildung, d.h., wenn  $L$  eine dkfS ist, dann ist  $\overline{L}$  auch eine dkfS.
- Wir benennen die Mengen
  1.  $DCFL = \{L \mid L \text{ ist eine dkfS} \}$
  2.  $CFL = \{L \mid L \text{ ist eine kfS} \}$

## Deterministische kontextfreie Sprachen 2

- Theorem:  $DCFL \subset CFL$ . Das heißt, alle dkfSn sind kfSn, aber es gibt kfSn, die nicht deterministisch sind.
- Nichtdeterminismus gibt hier ein echtes Plus an Ausdrucksstärke.
- Beweis: durch Widerspruch.
  1. Bekannt:  $L = \{a^i b^j c^k \mid i, j, k \geq 0 \text{ und } i \neq j \text{ oder } i \neq k\}$  ist eine kfS, da
    - (a)  $L_1 = \{a^i b^j c^k \mid i, j, k \geq 0 \text{ und } i \neq j\}$  kf ist,
    - (b)  $L_2 = \{a^i b^j c^k \mid i, j, k \geq 0 \text{ und } i \neq k\}$  kf ist,
    - (c)  $L = L_1 \cup L_2$
    - (d) Es folgt, dass  $L$  eine kfS ist, da die kfS unter Vereinigung abgeschlossen sind.

## Deterministische kontextfreie Sprachen 3

- Fortsetzung Beweis:
  2. Angenommen  $L$  sei eine dkfS.
  3. Dann muss auch  $\bar{L}$  eine dkfS sein.
  4.  $\bar{L} = (\Sigma^* - a^*b^*c^*) \cup \{a^n b^n c^n | n \geq 0\}$ 
    - (a)  $\{a^n b^n c^n | n \geq 0\}$  ist Komplement des Teils von  $L$ , der durch Beschränkung der Indizes ( $i \neq j$  oder  $i \neq k$ ) gegeben ist.
    - (b)  $(\Sigma^* - a^*b^*c^*)$  ist Komplement des Teils von  $L$ , der durch Reihenfolge  $as$  vor  $bs$  vor  $cs$  gegeben ist.
  5.  $\{a^n b^n c^n | n \geq 0\} = \bar{L} \cap a^*b^*c^*$ 
    - (a) Das ist so, da, wie Thomas Goldammer gewusst hat, gilt:  $(X \cup Y) \cap Z = (X \cap Z) \cup (Y \cap Z)$ .
    - (b) Daher:
$$\begin{aligned} & ((\Sigma^* - a^*b^*c^*) \cup \{a^n b^n c^n | n \geq 0\}) \cap a^*b^*c^* = \\ & ((\Sigma^* - a^*b^*c^*) \cap a^*b^*c^*) \cup \\ & (\{a^n b^n c^n | n \geq 0\} \cap a^*b^*c^*) = \\ & \emptyset \cup \{a^n b^n c^n | n \geq 0\} = \{a^n b^n c^n | n \geq 0\}. \end{aligned}$$

## Deterministische kontextfreie Sprachen 4

- Fortsetzung Beweis:
  6. Die kfSn sind abgeschlossen bzgl. Schnitt mit den regulären Sprachen. Dann muss  $\{a^n b^n c^n | n \geq 0\}$  eine kfS sein. Dies ist ein Widerspruch.
  7. Also war die Annahme falsch und  $L$  ist keine dkfS.

## Deterministische kontextfreie Sprachen 5

- Deterministische und nichtdeterministische kfSn.
  1.  $L_1 = \{a^m b^n c^m \mid m \geq 0, n > 0\}$  ist dkf
  2.  $L_2 = \{a^m b^n c^m \mid m > 0, n \geq 0\}$  ist dkf
  3.  $L_3 = \{a^m b^n c^m \mid m, n \geq 0\}$  ist dkf
  4.  $L_4 = \{a^m b^n c^m \mid m, n > 0\}$  ist dkf
  5.  $L_5 = \{c^n w w^R c^n \mid w \in \{a, b\}^*, n > 0\}$  ist nicht dkf
  6.  $L_6 = \{w c^n c^n w^R \mid w \in \{a, b\}^*, n > 0\}$  ist dkf
  7.  $L_7 = \{w w^R \mid w \in a^* b^+\}$  ist dkf
  8.  $L_8 = \{a^m b^n \mid m \geq n\}$  ist dkf
  9.  $L_9 = \{a^m b^n c^p \mid m = n \text{ oder } m = p\}$  ist nicht dkf
  10.  $L_{10} = \{a^m b^n c^p \mid m \geq n \text{ oder } m \geq p\}$  ist nicht dkf

## Appendix: Deterministische kontextfreie Sprachen 2

- Oben wurde vorausgesetzt, dass
  1.  $L_1 = \{a^i b^j c^k \mid i, j, k \geq 0 \text{ und } i \neq j\}$  kf ist,
  2.  $L_2 = \{a^i b^j c^k \mid i, j, k \geq 0 \text{ und } i \neq k\}$  kf ist,
- PDA  $M$  mit  $L(M) = L_1$ :
 
$$M = (\{q_0, q_1, q_2, q_3, q_4, q_5, q'_0, q'_1, q'_2, q'_3, q_s\}, \{a, b, c\}, \{a, \#\}, \Delta, q_s, \{q'_1, q'_2, q'_3\}); \# \text{ sei Bod.symb.}$$

$$\Delta = \left\{ \begin{array}{ll} ((q_s, \epsilon, \epsilon), (q_0, \epsilon)), & ((q_s, \epsilon, \epsilon), (q'_0, \epsilon)), \\ ((q_0, a, \#), (q_1, a\#)), & ((q_0, b, \#), (q_2, \#)), \\ ((q_1, a, a), (q_1, aa)), & ((q_1, c, a), (q_5, \epsilon)) \\ ((q_1, b, a), (q_4, a)), & ((q_1, c, \#), (q_3, \#)) \\ ((q_2, c, \epsilon), (q_3, \epsilon)), & ((q_2, b, \epsilon), (q_2, \epsilon)) \\ ((q_3, c, \epsilon), (q_3, \epsilon)), & ((q_5, c, a), (q_5, \epsilon)) \\ ((q_5, c, \#), (q_3, \#)), & ((q_4, b, \epsilon), (q_4, \epsilon)) \\ ((q_4, c, a), (q_5, \epsilon)), & ((q'_0, a, \#), (q'_1, a\#)) \\ ((q'_1, b, a), (q'_1, a)), & ((q'_1, a, a), (q'_2, aa)) \\ ((q'_2, c, a), (q'_2, \epsilon)), & ((q'_2, c, \#), (q'_3, \#)) \end{array} \right\}$$

## Das Pumping-Lemma für kontextfreie Sprachen

- Sei  $L$  eine **unendliche** kontextfreie Sprache und  $w \in L$  mit  $|w| > p$ , wobei die Zahl  $p$  die **Pumpzahl** ist.
- Dann kann  $w$  zerlegt werden in  $w = uvxyz$  so dass die folgenden Bedingungen erfüllt sind:
  1. Für alle  $i \geq 0 : uv^i xy^i z \in L$  (Bedingung 1)
  2.  $|vy| \geq 1$  ( $v, y$  sind nicht beide gleichzeitig leer) (Bedingung 2)
  3.  $|vxy| \leq p$  (Bedingung 3)

## Das Pumping-Lemma für kontextfreie Sprachen 2

- Beweisidee:
  1. Nimm ein **sehr langes** Wort  $w \in L$ .
  2. Da  $w \in L$  und da  $L$  eine kfS, gilt:  $S \Rightarrow^* w$ , d.h., es gibt einen Ableitungsbaum von  $w$ .
  3. Da  $w$  sehr lang ist, ist dieser Baum **sehr hoch/tief**.
  4. Der Baum muss einen **sehr langen Pfad** von der Wurzel zu einem Terminalsymbol enthalten.
  5. Da der Pfad sehr lang ist, da er ausschließlich aus Nichtterminalen besteht, und da die Anzahl der Nichtterminale einer Grammatik begrenzt ist, wird wenigstens ein Nichtterminal **mehrmals** auf dem Pfad liegen.
  6. Dieses Nichtterminal sei  $A$ .
  7.  $S$  **dominiert** also  $w$ , das obere  $A$  dominiert  $vxy$ , das untere  $A$  dominiert  $x$ .

## Das Pumping-Lemma für kontextfreie Sprachen 3

- Fortsetzung Beweisidee:
  7. Man kann den Teilbaum, dessen Wurzel das höhere  $A$  ist, ersetzen durch den Teilbaum, dessen Wurzel das tiefere  $A$  ist, und man erhält wieder einen legitimen Ableitungsbaum (man ersetzt  $vxy$  durch  $x$ , pumpt also **abwärts**).
  8. Man kann auch den Teil zwischen den beiden  $A$ s wiederholen (beliebig oft) und man erhält wieder jeweils einen legitimen Ableitungsbaum (man ersetzt  $vxy$  durch  $vvxyy$ , pumpt also **aufwärts**).

## Das Pumping-Lemma für kontextfreie Sprachen 4

- Beweis:
  1. Sei  $G$  eine kfG für die kfS  $L$ .
  2. Sei  $b$  die maximale Anzahl von Symbolen auf der rS der Regeln von  $G$  (angenommen  $b \geq 2$ ).
  3. Die maximale Länge einer Kette, die mit einem Baum der Höhe  $h$  assoziiert ist, ist dann  $b^h$ .
  4. Sei  $|(\Sigma - V)|$  die Anzahl der Nichtterminale von  $G$ .
  5. Sei  $p = b^{|\Sigma - V| + 2}$ .
  6. Da  $b \geq 2$  folgt, dass  $p > b^{|\Sigma - V| + 1}$ .
  7. Jeder Baum, der mit einer Kette assoziiert ist, die mindestens die Länge  $p$  hat, muss wenigstens die Höhe  $|\Sigma - V| + 2$  haben.

## Das Pumping-Lemma für kontextfreie Sprachen 5

- Fortsetzung Beweis:
  8. Sei  $w \in L$  und  $|w| \geq p$ .
  9. Sei  $\tau$  ein Ableitungsbaum von  $w$  mit minimaler Anzahl an Knoten.
  10. Da  $|w| \geq p$ , folgt, dass  $\tau$  mindestens die Höhe  $|(\Sigma - V)| + 2$  hat, d.h., der längste Pfad in  $\tau$  hat mindestens Länge  $|(\Sigma - V)| + 2$ .
  11. Nur das Blatt auf diesem Pfad ist ein Nichtterminal.
  12. Da es nur  $|(\Sigma - V)|$  Nichtterminale gibt, und da der Pfad  $|(\Sigma - V)| + 1$  Nichtterminale enthält, muss eines dieser Nichtterminale zweimal vorkommen:  $A$ .

## Das Pumping-Lemma für kontextfreie Sprachen 6

- Fortsetzung Beweis:
  13.  $w$  kann zerlegt werden in  $w = uvxyz$  (so dass das obere  $A$  ausschließlich die Kette  $vxy$  dominiert).
  14.  $v$  und  $y$  können entweder wiederholt oder ausgelassen werden. Das leitet Bedingung 1 ab.
  15. Wären  $v$  und  $y$  beide  $\epsilon$ , dann gäbe es eine Ableitungsfolge  $A \Rightarrow \dots \Rightarrow A$ , ohne Terminale zu generieren.
  16. Dann hätte man aber gleich beim ersten  $A$  bleiben können;  $\tau$  wäre dann aber nicht minimal, entgegen der Annahme. Daher können  $v$  und  $y$  nicht beide leer sein (Bedingung 2).
  17. Das höhere  $A$  generiert  $vxy$  und der Teilbaum, dessen Wurzel dieses  $A$  ist, hat maximal Höhe  $|(\Sigma - V)| + 2$ .
  18. Ein Baum dieser Höhe kann mit einer Kette von maximaler Länge  $b^{|\Sigma - V| + 2} = p$  assoziiert sein (Bedingung 3).

## Pumping-Lemma Anwendung

- $L_1 = \{a^n b^n c^n \mid n \geq 0\}$  ist nicht kontextfrei.
  1. Angenommen  $L_1$  wäre kontextfrei.
  2. Sei  $p$  die Pumpzahl.
  3. Sei  $w = uvxyz = a^p b^p c^p \in L_1$ .
  4. Wegen Bedingung 2. können  $v$  und  $y$  nicht beide leer sein.
- Es gibt zwei Fälle:
  1.  $v$  und  $y$  bestehen jeweils aus verschiedenen Symbolen.
    - (a)  $v$  kann nicht aus  $as$  und  $bs$  bestehen, weil das Aufpumpen von  $v$  dann Ketten erzeugt, bei denen  $bs$  **vor**  $as$  stehen (Ketten, die nicht in  $L_1$  sind).
    - (b) Das gleiche gilt, wenn  $v$  aus  $bs$  und  $cs$  besteht. Und dasselbe gilt auch für die andere Pumpvariable,  $y$ .

## Pumping-Lemma Anwendung 2

- Fortsetzung:
  1.  $v$  besteht nur aus  $as$ , nur aus  $bs$  oder nur aus  $cs$ . Egal, wie man dann  $y$  wählt, führt das simultane Aufpumpen von  $v$  und  $y$  zu Ketten, die nicht in  $L_1$  sind
    - (a)  $v$  besteht aus  $as$  und  $y$  aus  $bs$ . Dann führt Pumpen zu mehr  $as$  und  $bs$ , aber die  $cs$  bleiben gleich.
    - (b)  $v$  besteht aus  $bs$  und  $y$  aus  $cs$ . Dann führt Pumpen zu mehr  $bs$  und  $cs$ , aber die  $as$  bleiben gleich.
- Dies steht im Widerspruch zum Pumping-Lemma, also kann  $L_1$  nicht kontextfrei sein.

## Pumping-Lemma Anwendung 3

- $L_2 = \{a^i b^i c^j \mid j \geq i\}$  ist nicht kontextfrei.
  1. Angenommen  $L_2$  wäre kontextfrei.
  2. Sei  $p$  die Pumpzahl.
  3. Sei  $w = uvxyz = a^p b^p c^p \in L_2$ .
  4. Bed. 2.:  $|vy| \neq \epsilon$ ; Bed. 3.:  $|vxy| \leq p$
- Angenommen  $v$  und  $y$  bestehen nur aus  $as$  (oder nur aus  $bs$ ). Aufpumpen führt dann zu Ketten mit mehr  $as$  als  $bs$  (oder umgekehrt).
- Angenommen  $v$  und  $y$  bestehen nur aus  $cs$ . **Abwärtspumpen** führt dann zu Ketten mit **weniger**  $cs$  als  $bs$ .

## Pumping-Lemma Anwendung 4

- Fortsetzung.
- Bestehen  $v$  und  $y$  aus verschiedenen Symbolen, dann müssen dies aufeinanderfolgende Symbole sein ( $a, b$  oder  $b, c$ ), siehe Bed. 3.
  1. Besteht  $v$  aus  $as$  und  $y$  aus  $bs$ , dann führt das Aufpumpen zu mehr  $as/bs$  als  $cs$ .
  2. Besteht  $v$  aus  $bs$  und  $y$  aus  $cs$ , dann führt das Aufpumpen zu mehr  $bs$  als  $as$ .
- Andere Kombinationen lassen sich ebenfalls nicht aufpumpen (z.B. wenn  $v$  aus verschiedenen Symbolen besteht, siehe Diskussion  $L_1$ ).
- Dies steht im Widerspruch zum Pumping-Lemma. Daher ist  $L_2$  nicht kontextfrei.

## Pumping-Lemma Anwendung 5

- $L_3 = \{a^i b^j c^i d^j \mid j, i \geq 1\}$  ist nicht kontextfrei.
  1. Angenommen  $L_3$  wäre kontextfrei.
  2. Sei  $p$  die Pumpzahl.
  3. Sei  $w = uvxyz = a^p b^p c^p d^p \in L_3$ .
  4. Bed. 2.:  $|vy| \neq \epsilon$ ; Bed. 3.:  $|vxy| \leq p$
- $vy$  kann höchstens aus zwei verschiedenen Symbolen bestehen; diese müssen aufeinanderfolgend sein.
- Besteht  $vy$  nur aus  $as$ , dann ist  $uxz$  (abwärts gepumpt) nicht in  $L_3$ , da es weniger  $as$  als  $cs$  gibt. Dasselbe gilt für  $b, c$  und  $d$ .
- Besteht  $vy$  aus  $as$  und  $bs$ , dann ist  $uxz \notin L_3$  (abwärts gepumpt): weniger  $as$  als  $cs$  oder weniger  $bs$  als  $ds$ . Dasselbe für  $b$  und  $c$  und für  $c$  und  $d$ .
- Da dies alle Möglichkeiten sind, entsteht ein Widerspruch zum Pumping-Lemma. Also kann  $L_3$  nicht kontextfrei sein.

## Pumping-Lemma Anwendung 6

- $L_4 = \{a^i b^j c^k \mid i \leq j \leq k\}$  ist nicht kontextfrei.
  1. Angenommen  $L_4$  wäre kontextfrei.
  2. Sei  $p$  die Pumpzahl.
  3. Sei  $w = uvxyz = a^p b^p c^p \in L_4$ .
  4. Bed. 2.:  $|vy| \neq \epsilon$ ; Bed. 3.:  $|vxy| \leq p$
- Fall 1:  $vy$  besteht nur aus einer Symbolart.
  1. Wenn dies  $a$  ist, dann führt **aufwärtspumpen** zu Ketten, die nicht in  $L_4$  sind.
  2. Wenn dies  $b$  oder  $c$  ist, dann führt **abwärtspumpen** zu Ketten, die nicht in  $L_4$  sind.

## Pumping-Lemma Anwendung 7

- Fortsetzung
- Fall 2:  $vy$  besteht aus mehr als nur einer Art von Symbol. Wegen Bed. 3. können dies höchstens zwei Arten sein, die aufeinanderfolgend sein müssen.
  1.  $v$  besteht nur aus  $as$  und  $y$  nur aus  $bs$ . Dann führt das Aufpumpen zu Ketten, die nicht in  $L_4$  sind (mehr  $as/bs$  als  $cs$ ).
  2.  $v$  besteht nur aus  $bs$  und  $y$  nur aus  $cs$ . Dann führt das Abpumpen zu Ketten, die nicht in  $L_4$  sind (weniger  $cs$  als  $as$ ).
  3. Wenigstens  $v$  oder  $y$  bestehen aus mehr als nur einer Art Symbol. Dann sind die Symbole in der Kette  $uv^2xy^2z$  nicht in der richtigen Reihenfolge.
- Da dies alle Möglichkeiten sind, entsteht ein Widerspruch zum Pumping-Lemma. Also kann  $L_4$  nicht kontextfrei sein.

## Entscheidbarkeit

- Für eine beliebige kontextfreie Grammatik  $G$  und ein Wort  $w$  gibt es ein Verfahren, um zu entscheiden, ob  $w \in L(G)$ .
- Verfahren:
  1. Durchlaufe Ableitungen von  $G$  in einer systematischen Weise.
  2. Wenn die abgeleitete Kette länger ist als  $|w|$ , dann brich ab.
  3. Dies ist eine endliche Anzahl von Ableitungen.
  4. Wenn  $w$  von einer dieser Ableitungen generiert wird, dann ist  $w \in L$ , sonst nicht.

## Entscheidbarkeit 2

- Probleme des Verfahrens:
  1. Wenn  $G$  Regeln der Form  $A \rightarrow \epsilon$  enthält, dann könnte eine Derivation, die eine Kette generiert hat, die länger ist als  $|w|$ , wieder auf Länge  $|w|$  schrumpfen.
  2. Wenn  $G$  Regelgruppen der Art  $A \rightarrow B, B \rightarrow C, C \rightarrow A$  enthält, dann könnte eine Ableitung unendlich lange fortschreiten, d.h., das Verfahren hält nicht an.
- Lösung: Wandle  $G$  in eine äquivalente Grammatik  $G'$  um, die keine Regeln dieser Art enthält.

## Entscheidbarkeit 3

- Umwandlung:
- Für jede Regel  $R$  der Form  $A \rightarrow \epsilon$ ,
  1. wenn  $A$  auf der rS irgendeiner Regel  $R'$  erscheint, füge analoge Regeln hinzu, bei denen  $A$  nicht auf der rS vorkommt. Nämlich,
    - (a) für  $R' \rightarrow uAv$  füge  $R' \rightarrow uv$  hinzu
    - (b) für  $R' \rightarrow uAvAw$  füge  $R' \rightarrow uAvw, R' \rightarrow uvAw$  und  $R' \rightarrow uvw$  hinzu
  2. Entferne die Regel  $A \rightarrow \epsilon$ , wenn  $A \neq S$ .
- $S \rightarrow \epsilon$  (falls vorhanden) kann nicht entfernt werden, da die Sprache sonst nicht mehr das leere Wort generieren kann.
- Sie muss aber auch nicht entfernt werden, da jede nichtleere Kette, die durch Anwendung von  $S \rightarrow \epsilon$  erzeugt wurde, auch ohne deren Anwendung erzeugt werden kann (nach Anwendung des Verfahrens).

## Entscheidbarkeit 4

- Fortsetzung Umwandlung:
- Für jede Regel  $R$  der Form  $A \rightarrow B$ 
  1. nimm eine Regel  $R' A \rightarrow w$ , mit  $w \notin (V - \Sigma)$ , und
  2. für jedes Nichtterminal  $C \neq A$ , stelle fest, ob  $C \Rightarrow_G^* A$  (kann in endlicher Anzahl von Schritten festgestellt werden)
  3. Wenn  $C \Rightarrow_G^* A$ , dann füge Regel  $C \rightarrow w$  hinzu.
  4. Wiederhole dies für alle Regeln  $A \rightarrow w$  ( $w \notin (V - \Sigma)$ ) und alle  $C \neq A$ .
  5. Dann entferne die Regel  $A \rightarrow B$ .

## Entscheidbarkeit 5

- Es ist entscheidbar, ob die Sprache, die von einer ktf Grammatik  $G$  generiert wird
  1. leer ist
  2. unendlich ist
  3. endlich ist
- Generiert  $G$  keine Ketten der Länge  $< p$ ,  $p$  die Pumpzahl für  $G$ , dann ist  $L(G)$  leer.
- Generiert  $G$  keine Ketten der Länge  $> p$  und  $< 2p$ ,  $p$  die Pumpzahl für  $G$ , dann ist  $L(G)$  endlich.
- Generiert  $G$  Ketten der Länge  $\geq p$  und  $< 2p$ ,  $p$  die Pumpzahl für  $G$ , dann ist  $L(G)$  unendlich.