

Grundlagen der Theoretischen Informatik

4. Kellerautomaten und kontextfreie Sprachen (III)

20.05.2021

Viorica Sofronie-Stokkermans

e-mail: sofronie@uni-koblenz.de

Übersicht

1. Motivation
2. Terminologie
3. Endliche Automaten und reguläre Sprachen
4. Kellerautomaten und kontextfreie Sprachen
5. Turingmaschinen und rekursiv aufzählbare Sprachen
6. Berechenbarkeit, (Un-)Entscheidbarkeit
7. Komplexitätsklassen P und NP

Bis Jetzt

- **Umformung von Grammatiken**

Startsymbol nur links

Keine nutzlose Symbole

- **Normalform für Regeln**

Theorem (Normalform)

Zu jeder Grammatik G (beliebigen Typs) existiert eine äquivalente Grammatik G' , bei der für alle Regeln $P \rightarrow Q \in R'$ gilt:

- $Q \in V^*$ und P beliebig
- $Q \in T$ und $P \in V$

Für alle Typen außer den linearen hat G' denselben Typ wie G .

Bis Jetzt

- Elimination von ε -Regeln
- Elimination von Kettenproduktionen

Theorem. Zu jeder cf-Grammatik existiert eine äquivalente cf-Grammatik

- ohne ε -Regeln
(bis auf $S \rightarrow \varepsilon$, falls ε zur Sprache gehört;
in diesem Fall darf S in keiner Regelconclusio vorkommen),
- ohne nutzlose Symbole,
- ohne Kettenproduktionen,
- so dass für jede Regel $P \rightarrow Q$ gilt: entweder $Q \in V^*$ oder $Q \in T$.

Chomsky-Normalform

Definition. Eine cf-Grammatik $G = (V, T, R, S)$ ist in **Chomsky-Normalform (CNF)**, wenn gilt:

- G hat nur Regeln der Form

$$A \rightarrow BC \quad \text{mit } A, B, C \in V \text{ und}$$

$$A \rightarrow a \quad \text{mit } A \in V, a \in T \quad (\text{nicht } \varepsilon!)$$

- Ist $\varepsilon \in L(G)$, so darf G zusätzlich die Regel $S \rightarrow \varepsilon$ enthalten. In diesem Fall darf S in keiner Regelconclusio vorkommen.
- G enthält keine nutzlosen Symbole.

Greibach-Normalform

Definition.

Eine cf-Grammatik $G = (V, T, R, S)$ ist in **Greibach-Normalform (GNF)**, wenn gilt:

- G hat nur Regeln der Form
$$A \rightarrow a\alpha \text{ mit } A \in V \text{ und } a \in T \text{ und } \alpha \in V^*$$
- Ist $\varepsilon \in L(G)$, so darf G zusätzlich die Regel $S \rightarrow \varepsilon$ enthalten. In diesem Fall darf S in keiner Regelconclusio vorkommen.
- G enthält keine nutzlosen Symbole.

Pumping-Lemma für kontextfreie Sprachen

Wiederholung

Theorem (Pumping-Lemma für \mathcal{L}_3 -Sprachen)

Sei $L \in \mathbf{RAT}$.

Dann existiert ein $n \in \mathbb{N}$, so dass:

Für alle

$$x \in L \quad \text{mit} \quad |x| \geq n$$

existiert eine Zerlegung

$$x = uvw \quad u, v, w \in \Sigma^*$$

mit

- $|v| \geq 1$
- $|v| < n$
- $uv^m w \in L$ für alle $m \in \mathbb{N}$

Pumping-Lemma für kontextfreie Sprachen

Theorem (Pumping-Lemma für kontextfreie Sprachen)

Sei L kontextfrei.

Dann existiert ein $n \in \mathbb{N}$, so dass:

Für alle

$$z \in L \quad \text{mit} \quad |z| \geq n$$

existiert eine Zerlegung

$$z = uvwxy \quad u, v, w, x, y \in \Sigma^*$$

mit

- $|vx| \geq 1$
- $|vwx| < n$
- $uv^mwx^my \in L$ für alle $m \in \mathbb{N}$

Pumping-Lemma für kontextfreie Sprachen

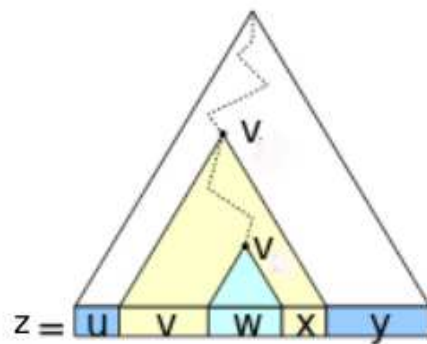
Beweisidee:

Bei der Ableitung eines hinreichend langen Wortes muss es eine Variable geben, die mehr als einmal auftaucht.

Dies führt zu einer Schleife in der Ableitung, die aufgepumpt werden kann.

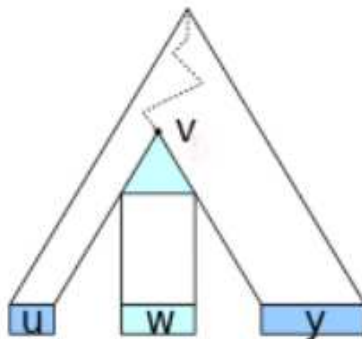
Pumping-Lemma für kontextfreie Sprachen

Beweisidee:

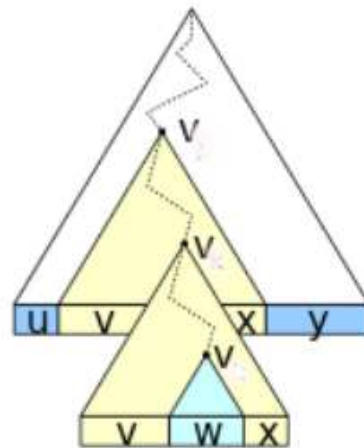


} h

Gegeben: Wort $z \in L$
mit $|z| \geq n$
Ableitungsbaum T für z
mit Höhe $h \geq N$



Erzeugen von uv^0wx^0y



Erzeugen von uv^2wx^2y

Pumping-Lemma für kontextfreie Sprachen

Idee: Untersuchung der Form und Größe der Ableitungsbäumen.

↪ Einfache Form (binär), wenn Grammatik in Chomsky Normalform ist.

Theorem: Zu jeder cf-Grammatik existiert eine äquivalente cf-Grammatik in Chomsky-Normalform.

Definition $G = (V, T, R, S)$ ist in Chomsky Normalform, wenn folgendes gilt:

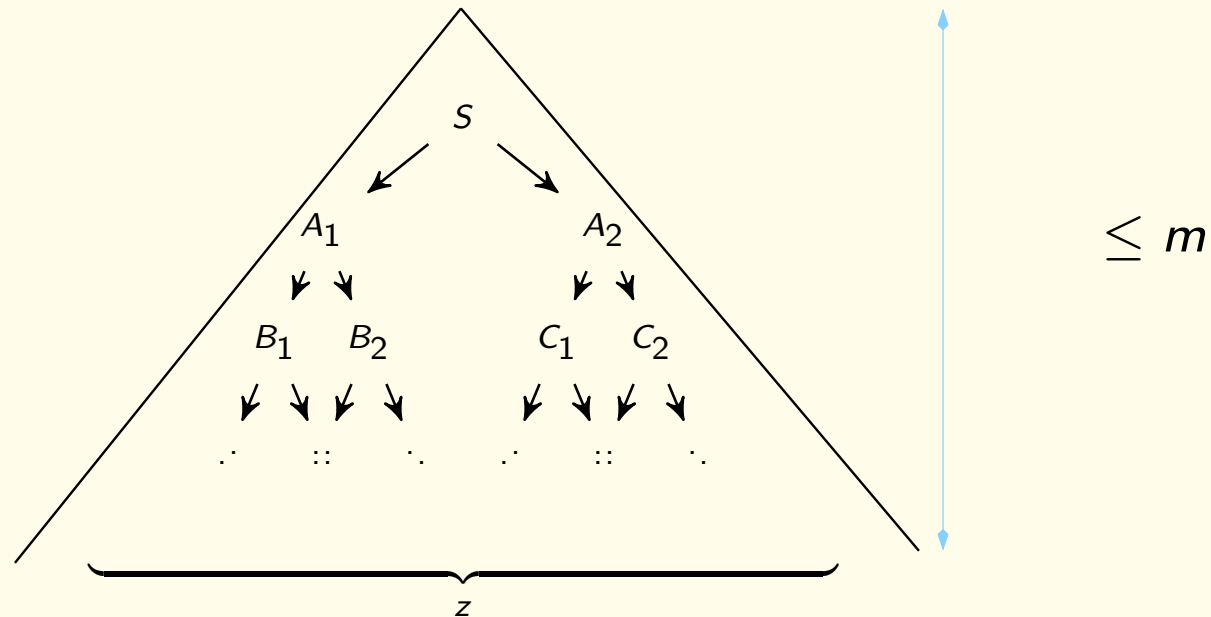
- G hat nur Regeln der Form
 $A \rightarrow BC$ mit $A, B, C \in V$ und
 $A \rightarrow a$ mit $A \in V, a \in T$ (nicht ε !)
- $S \rightarrow \varepsilon$ erlaubt nur wenn $\varepsilon \in L(G)$.
In diesem Fall darf S in keiner Regelconclusio vorkommen.
- G enthält keine nutzlosen Symbole

Pumping-Lemma für kontextfreie Sprachen

$G = (V, T, R, S)$ in Chomsky-Normalform

$S \Rightarrow^* z$

Ableitungsbaum:



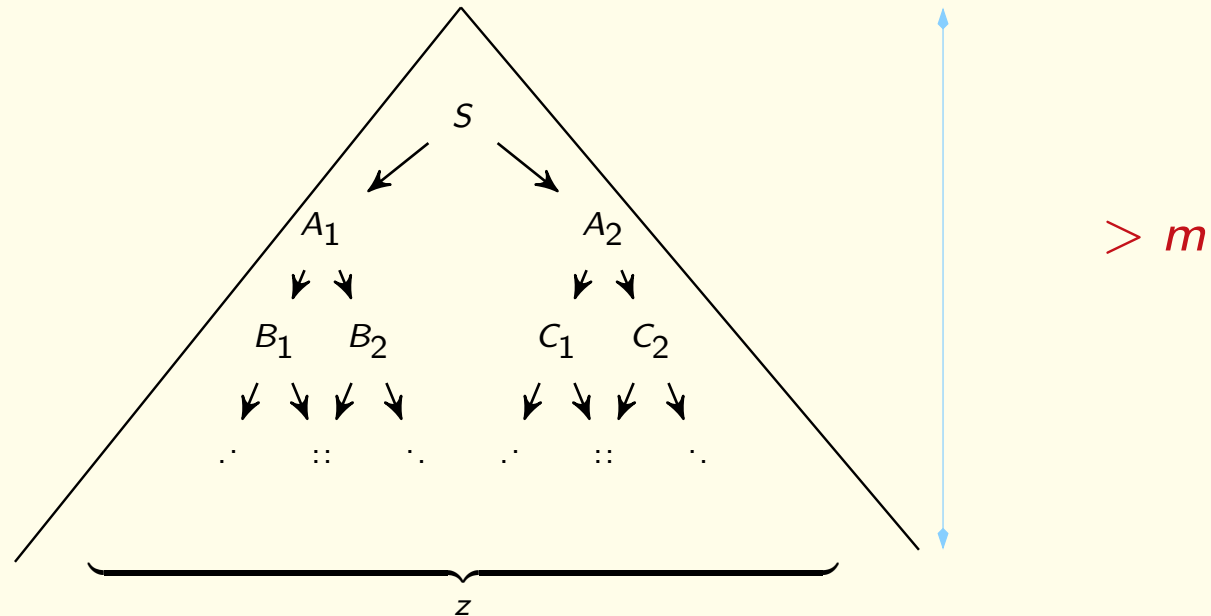
Wenn der längste Pfad Länge $\leq m$ hat, dann gilt $|z| \leq 2^{m-1}$.

Pumping-Lemma für kontextfreie Sprachen

$G = (V, T, R, S)$ in Chomsky-Normalform

$S \Rightarrow^* z$

Ableitungsbaum:



Wenn der längste Pfad Länge $\leq m$ hat, dann gilt $|z| \leq 2^{m-1}$.

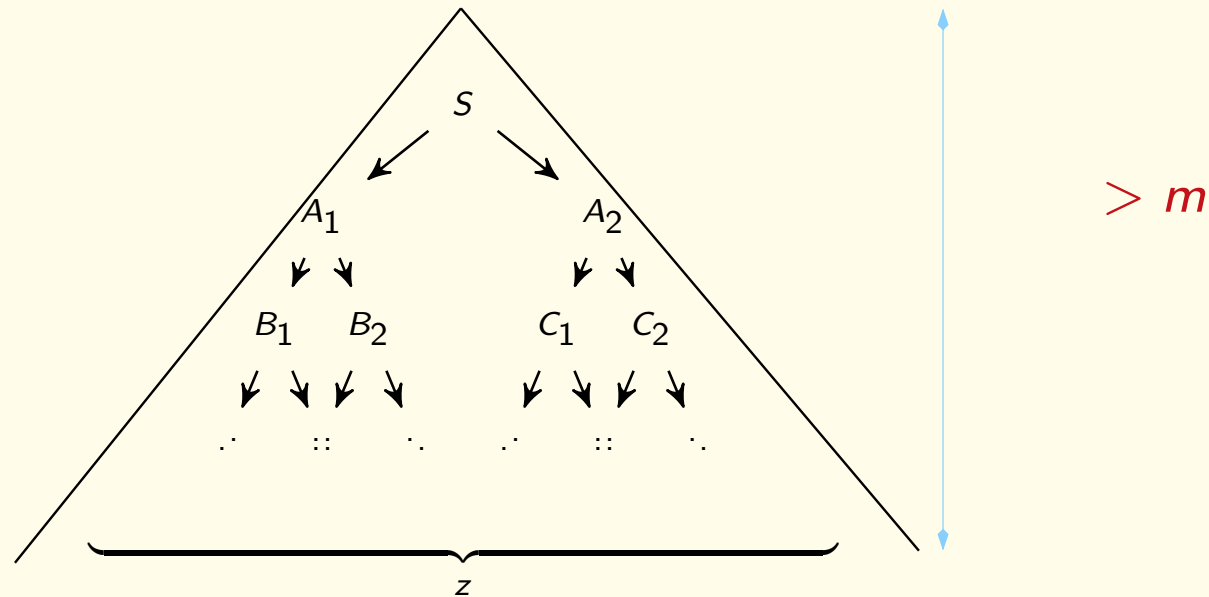
Wenn $|z| > 2^{m-1}$, muss der längste Pfad Länge $> m$ haben

Pumping-Lemma für kontextfreie Sprachen

$G = (V, T, R, S)$ in Chomsky-Normalform

$S \Rightarrow^* z$

Ableitungsbaum:



Wenn der längste Pfad Länge $\leq m$ hat, dann gilt $|z| \leq 2^{m-1}$.

Wenn $|z| > 2^{m-1}$, muss der längste Pfad P Länge $> m$ haben

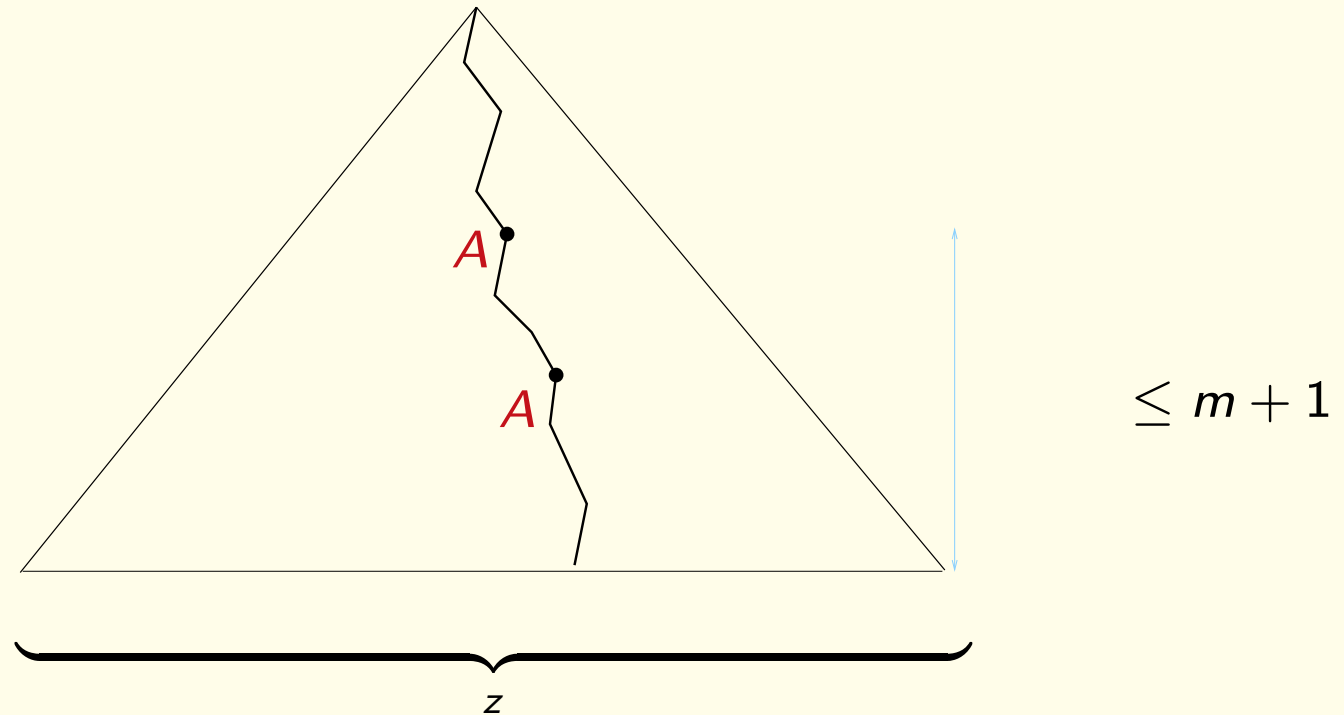
Sei $m = |V|$.

Wenn $|z| > 2^{m-1}$, muss es sich bei mindestens zwei der letzten $m + 1$ Variablen auf dem Pfad um dieselben Variablen handeln.

Pumping-Lemma für kontextfreie Sprachen

$S \Rightarrow^* z$,

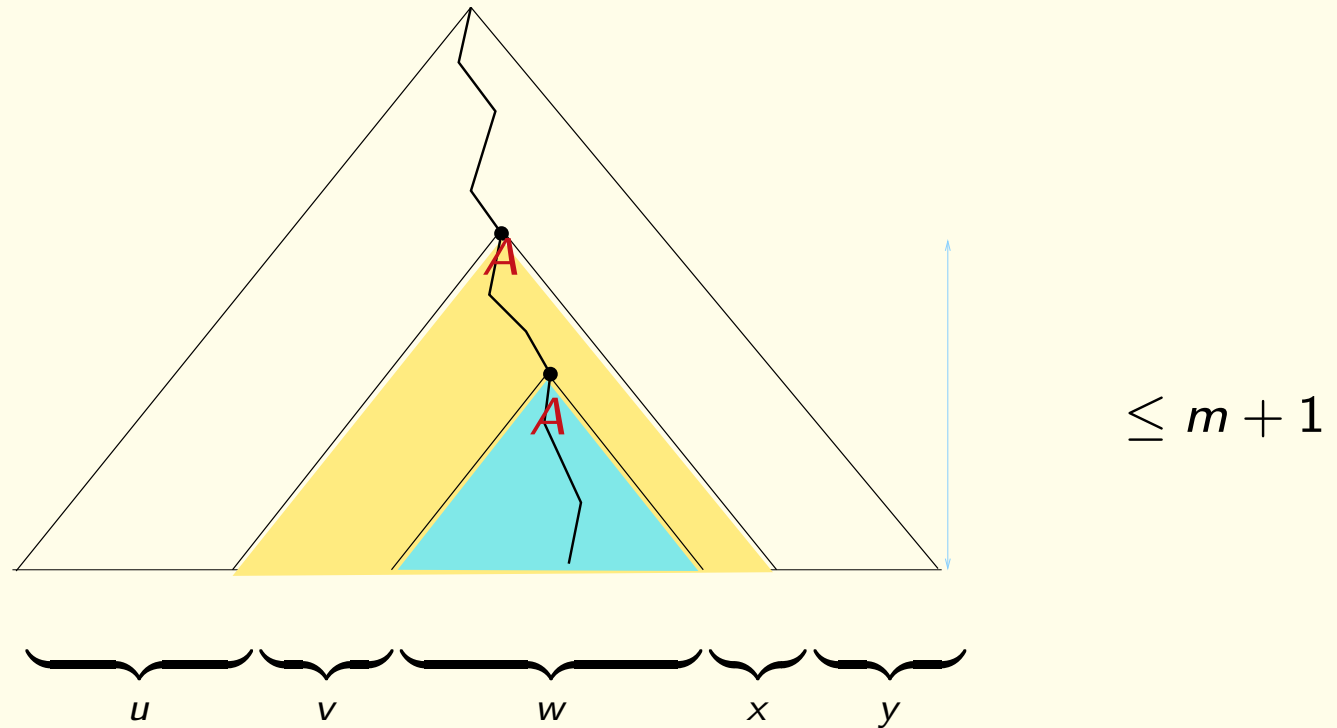
$$|z| \geq n = 2^m + 1 > 2^{m-1}, \quad m = |V|$$



Pumping-Lemma für kontextfreie Sprachen

$S \Rightarrow^* z$,

$$|z| \geq n = 2^m + 1 > 2^{m-1}, \quad m = |V|$$



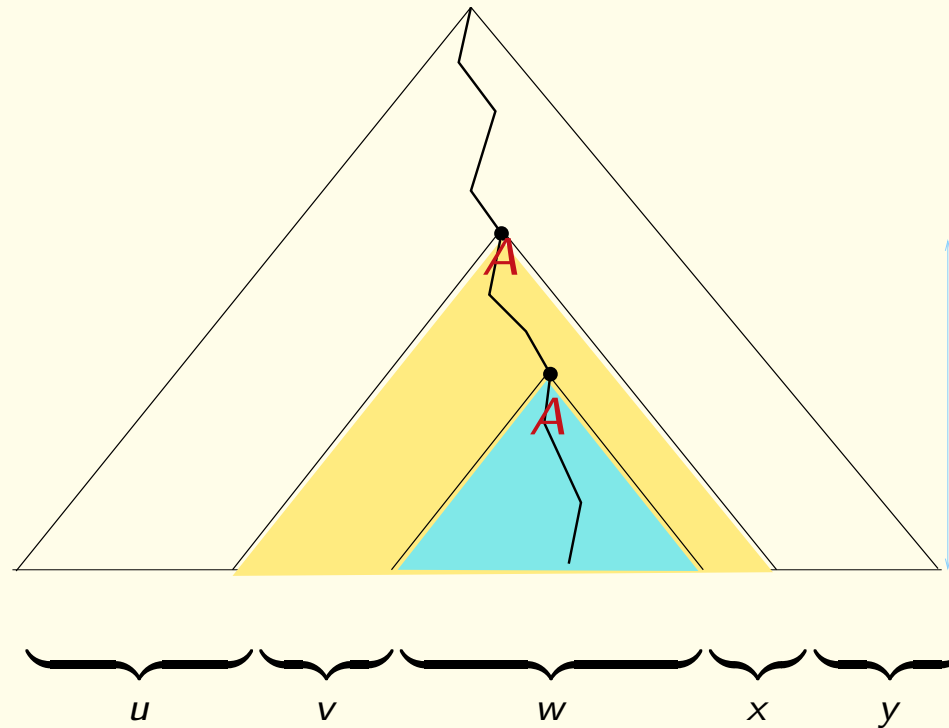
Pumping-Lemma für kontextfreie Sprachen

$$S \Rightarrow^* z,$$

$$|z| \geq n = 2^m + 1 > 2^{m-1}, \quad m = |V|$$

$$|vx| > 0$$

$$|vwx| \leq 2^m < n$$

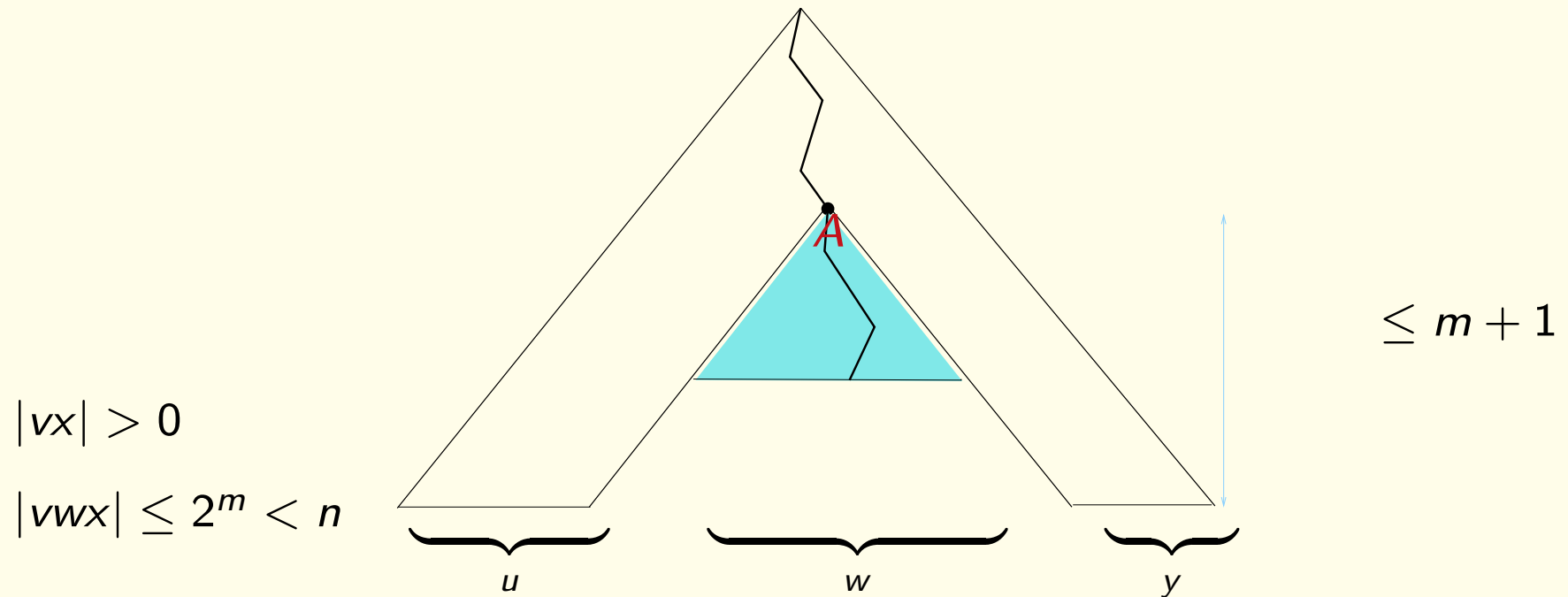


$$\leq m + 1$$

Pumping-Lemma für kontextfreie Sprachen

$S \Rightarrow^* z,$

$|z| \geq n = 2^m + 1 > 2^{m-1}, \quad m = |V|$



$uv^k wx^k y \in L$ für alle $k \in \mathbb{N}$

$k = 0: uwy \in L$

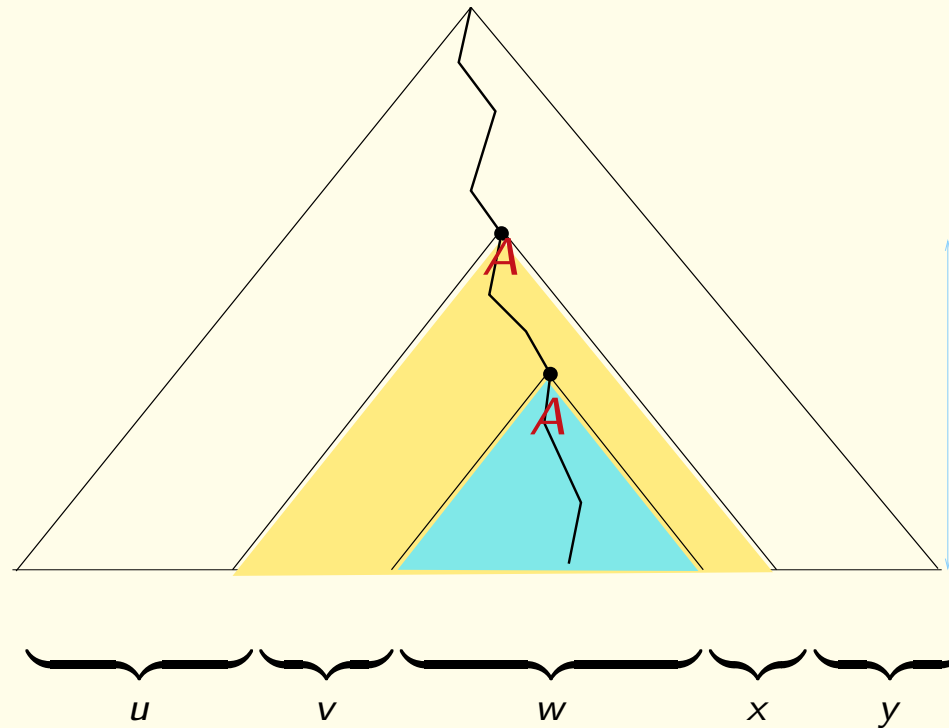
Pumping-Lemma für kontextfreie Sprachen

$$S \Rightarrow^* z,$$

$$|z| \geq n = 2^m + 1 > 2^{m-1}, \quad m = |V|$$

$$|vx| > 0$$

$$|vwx| \leq 2^m < n$$



$$\leq m + 1$$

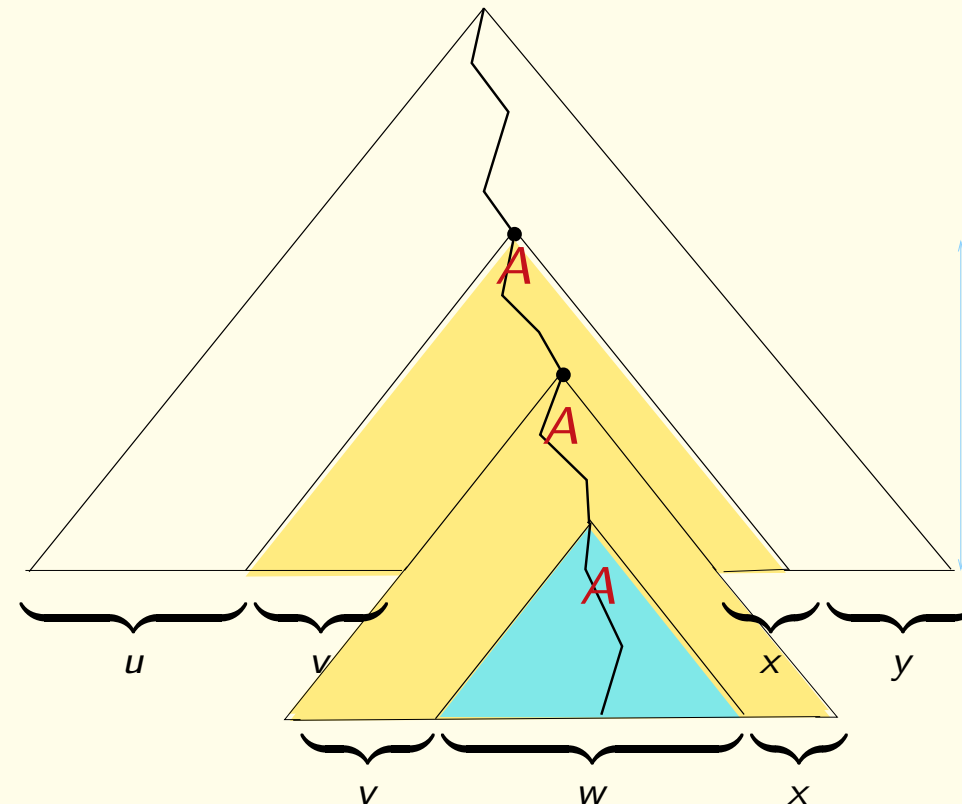
Pumping-Lemma für kontextfreie Sprachen

$S \Rightarrow^* z,$

$|z| \geq n = 2^m + 1 > 2^{m-1}, \quad m = |V|$

$|vx| > 0$

$|vwx| \leq 2^m < n$



$\leq m + 1$

$uv^k wx^k y \in L$ für alle $k \in \mathbb{N}$

$k = 2: uv^2 wx^2 y \in L$

Pumping-Lemma für kontextfreie Sprachen

Theorem (Pumping-Lemma für kontextfreie Sprachen)

Sei L kontextfrei.

Dann existiert ein $n \in \mathbb{N}$, so dass:

Für alle

$$z \in L \quad \text{mit} \quad |z| \geq n$$

existiert eine Zerlegung

$$z = uvwxy \quad u, v, w, x, y \in \Sigma^*$$

mit

- $|vx| \geq 1$
- $|vwx| < n$
- $uv^kwx^ky \in L$ für alle $k \in \mathbb{N}$

Pumping-Lemma für kontextfreie Sprachen

Theorem (Pumping-Lemma für kontextfreie Sprachen)

Sei L kontextfrei.

Dann existiert ein $n \in \mathbb{N}$, so dass:

$$n = 2|V| + 1,$$

V Menge der Variablen einer Grammatik G in CNF mit $L = L(G)$

Für alle

$$z \in L \quad \text{mit} \quad |z| \geq n$$

existiert eine Zerlegung

$$z = uvwxy \quad u, v, w, x, y \in \Sigma^*$$

mit

- $|vx| \geq 1$
- $|vwx| < n$
- $uv^kwx^ky \in L$ für alle $k \in \mathbb{N}$

Pumping-Lemma für kontextfreie Sprachen

Anwendung des Pumping-Lemmas für cf-Sprachen

Wenn das cf-Pumping-Lemma für eine Sprache nicht gilt,
dann kann sie nicht kontextfrei sein.

Pumping-Lemma für kontextfreie Sprachen

Anwendung des Pumping-Lemmas für cf-Sprachen

Wenn das cf-Pumping-Lemma für eine Sprache nicht gilt, dann kann sie nicht kontextfrei sein.

Beispiel (Sprachen, die nicht kontextfrei sind)

Für folgende Sprachen kann man mit Hilfe des cf-Pumping-Lemmas zeigen, dass sie nicht kontextfrei sind:

- $\{a^p \mid p \text{ prim}\}$
- $\{a^n b^n c^n \mid n \in \mathbb{N}\}$
- $\{zzz \mid z \in \{a, b\}^*\}$.

Pumping-Lemma für kontextfreie Sprachen

$L_1 = \{a^p \mid p \text{ prim}\}$ ist nicht kontextfrei.

Beweis: Wir nehmen an, L_1 wäre kontextfrei.

Sei dann n die zugehörige Konstante aus dem Pumping-Lemma.

Wir betrachten das Wort $z = a^p$, wobei p prim und $p \geq n + 2$.

Es muss dann eine Zerlegung $z = uvwxy$ geben, so dass:

$|vx| \geq 1$, $|vwx| < n$, $uv^iwx^iy \in L_1$ für alle $i \geq 0$.

Dann $u = a^{i_1}$, $v = a^{i_2}$, $w = a^{i_3}$, $x = a^{i_4}$, $y = a^{i_5}$ mit

- $i_1 + i_2 + i_3 + i_4 + i_5 = p$
- $i_2 + i_4 \geq 1$, $i_2 + i_3 + i_4 < n$
- $i_1 + mi_2 + i_3 + mi_4 + i_5$ prim für alle $m \geq 0$.

Wir wählen $m = i_1 + i_3 + i_5$.

Dann kann $i_1 + mi_2 + i_3 + mi_4 + i_5 = (i_1 + i_3 + i_5)(1 + i_2 + i_4)$ nicht prim sein, da $i_1 + i_3 + i_5 = p - (i_2 + i_4) \geq p - n \geq 2$ und $1 + i_2 + i_4 \geq 2$.

Also $uv^mwx^my \notin L_1$. Widerspruch.

Also kann L_1 nicht kontextfrei sein.

Pumping-Lemma für kontextfreie Sprachen

$L_2 = \{a^m b^m c^m \mid m \in \mathbb{N}\}$ ist nicht kontextfrei

Beweis:

Wir nehmen an, L_2 wäre kontextfrei.

Sei dann n die zugehörige Konstante aus dem Pumping-Lemma.

Wir betrachten das Wort $z = a^n b^n c^n$.

Es muss dann eine Zerlegung $z = uvwxy$ geben, so dass:

$|vx| \geq 1$, $|vwx| < n$, $uv^i wx^i y \in L_2$ für alle $i \geq 0$.

Da $|vwx| < n$, enthält das Wort vwx höchstens zwei verschiedene Buchstaben.

Da $|vx| \geq 1$, kann $uv^2 wx^2 y$ nicht von allen drei Buchstaben gleich viele enthalten.

Also $uv^2 wx^2 y \notin L_2$. Widerspruch.

Also kann L_2 nicht kontextfrei sein.

Pumping-Lemma für kontextfreie Sprachen

$L_3 = \{zzz \mid z \in \{a, b\}^*\}$ ist nicht kontextfrei.

Beweis: Wir nehmen an, L_3 wäre kontextfrei. Sei dann n die zugehörige Konstante aus dem Pumping-Lemma.

Wir betrachten das Wort $z = a^n ba^n ba^n b$.

Es muss dann eine Zerlegung $z = uvwxy$ geben, so dass:

$|vx| \geq 1$, $|vwx| < n$, $uv^i wx^i y \in L_3$ für alle $i \geq 0$.

Da $|vwx| < n$, enthält das Wort vwx höchstens ein b .

Fall 1: $vwx = a^k$. Dann wird durch aufpumpen von v, x eine a Sequenz länger als die anderen, so $uv^2 wx^2 y \notin L_3$.

Fall 2: $vwx = a^k ba^m$.

Fall 2.1: b kommt nicht in v oder x vor. Dann sind in $uv^2 wx^2 y$ zwei a Sequenzen länger als die dritte, so $uv^2 wx^2 y \notin L_3$.

Fall 2.2: b kommt in v oder x vor. Dann enthält $uv^0 wx^0 y$ nur zwei b 's, d.h. $uv^0 wx^0 y \notin L_3$.

Also kann L_3 nicht kontextfrei sein.

Pushdown-Automaten (PDAs)

Grammatiken vs. Automaten

Erzeugende Grammatiken– akzeptierende Automaten

Erinnerung: Reguläre Sprachen

- werden erzeugt von rechtslinearen Grammatiken
- werden akzeptiert von endlichen Automaten

Grammatiken vs. Automaten

Erzeugende Grammatiken– akzeptierende Automaten

Erinnerung: Reguläre Sprachen

- werden erzeugt von rechtslinearen Grammatiken
- werden akzeptiert von endlichen Automaten

Jetzt: Kontextfreie Sprachen

- werden erzeugt von kontextfreien Grammatiken
- werden akzeptiert von **Pushdown-Automaten**

Idee des Push-Down-Automaten

Beispiel

Die „prototypische“ cf-Sprache

$$\{a^n b^n \mid n \in \mathbb{N}\}$$

- Endliche Automaten reichen nicht aus.
- Sie können sich nicht merken, wie oft sie einen Zustand durchlaufen haben.
- Für $a^n b^n$ muss man aber **mitzählen**.

Idee des Push-Down-Automaten

Idee: Wie kann man diese Sprache akzeptieren?

- Weitere Informationen auf dem **Stack** sichern
- Später **zurückholen**
- Ähnlich einem “Prozeduraufruf”
- Grammatikregel wie $S \rightarrow aAb$ entspricht Aufruf einer Prozedur für das A .

Idee des Push-Down-Automaten

Idee: Wie kann man diese Sprache akzeptieren?

- Weitere Informationen auf dem **Stack** sichern
- Später **zurückholen**
- Ähnlich einem “Prozeduraufruf”
- Grammatikregel wie $S \rightarrow aAb$ entspricht Aufruf einer Prozedur für das A .

Stack, Stapel, Keller

- Last in, first out
- Zuletzt gespeicherte Information liegt immer “obenauf”
- Beliebig viel Information kann gespeichert werden (Aber kein beliebiger Zugriff!)

Push-Down-Automat

Push-Down-Automat (PDA): Informell

- Wie endlicher Automat, aber **zusätzlich** einen Stack
- Übergangsrelation bezieht das oberste Stacksymbol in den Übergang ein
- Bei Zustandsübergang: lesen und schreiben auf Stack

Push-Down-Automat

Definition(Push-Down-Automat)

Ein **Push-Down-Automat (PDA)** ist ein Tupel

$$\mathcal{M} = (K, \Sigma, \Gamma, \Delta, s_0, Z_0, F)$$

Dabei ist:

K eine endliche Menge von Zuständen

Σ das Eingabealphabet

Γ das Stack- oder Kellularphabet

$s_0 \in K$ der Startzustand

$Z_0 \in \Gamma$ das Anfangssymbol im Keller

$F \subseteq K$ eine Menge von finalen Zuständen

Δ die Zustandsübergangsrelation,
eine endliche Relation: $\Delta \subset (K \times (\Sigma \cup \{\varepsilon\}) \times \Gamma) \times (K \times \Gamma^*)$

Push-Down-Automat

Arbeitsschritt eines PDA

In Abhängigkeit

- vom aktuellen Zustand
- vom nächsten Eingabezeichen (oder auch unabhängig davon)
- vom obersten Kellersymbol

geschieht folgendes

- nächstes **Eingabezeichen** wird **gelesen oder nicht** (bei ε),
- das oberste **Kellersymbol** wird **entfernt**,
- der **Zustand** wird **geändert**,
- es werden null oder mehr **Zeichen auf den Keller** geschoben
Bei neuem Keller-Wort $\gamma = A_1 \dots A_n$ wird A_n zuerst auf den Keller geschoben usw., so dass am Schluss A_1 obenauf liegt.

Push-Down-Automat

Notation

- a, b, c für Buchstaben aus Σ
- u, v, w für Wörter aus Σ^*
- A, B für Stacksymbole aus Γ
- γ, η für Stackinhalte aus Γ^*

Push-Down-Automat: Konfiguration

Konfiguration eines PDA: Informell

- Konfiguration beschreibt die aktuelle Situation des PDA **komplett**
- Bestandteile:
 - **aktueller Zustand**
 - **noch zu lesendes Restwort**
 - **kompletter Stackinhalt**
- Für Konfigurationen C_1, C_2 bedeutet

$$C_1 \vdash C_2$$

dass der PDA in einem Schritt von C_1 nach C_2 gelangen kann.

Push-Down-Automat: Konfiguration

Definition (Konfiguration eines PDA)

Eine **Konfiguration** C eines PDA $\mathcal{M} = (K, \Sigma, \Gamma, \Delta, s_0, Z_0, F)$ ist ein Tripel

$$(q, w, \gamma) \in K \times \Sigma^* \times \Gamma^*.$$

- q ist der aktuelle Zustand
- w ist das noch zu lesende Restwort
- γ ist der komplette Stackinhalt

Push-Down-Automat: Konfiguration

Definition (Konfiguration eines PDA)

Eine **Konfiguration** C eines PDA $\mathcal{M} = (K, \Sigma, \Gamma, \Delta, s_0, Z_0, F)$ ist ein Tripel

$$(q, w, \gamma) \in K \times \Sigma^* \times \Gamma^*.$$

- q der aktuelle Zustand
- w der noch zu lesendes Restwort
- γ der komplette Stackinhalt

Definition (Startkonfiguration)

Bei Eingabewort w ist die **Startkonfiguration**:

$$(s_0, w, Z_0)$$

Push-Down-Automat: Konfiguration

Definition (Nachfolgekonfiguration)

$C_2 = (q_2, w_2, \gamma_2)$ heißt **Nachfolgekonfiguration** von $C_1 = (q_1, w_1, \gamma_1)$,

$$C_1 \vdash_{\mathcal{M}} C_2$$

falls

$$\exists a \in \Sigma \exists A \in \Gamma \exists w \in \Sigma^* \exists \gamma, \eta \in \Gamma^*$$

so dass

entweder $C_1 = (q_1, aw, A\gamma)$, $C_2 = (q_2, w, \eta\gamma)$, und $(q_1, a, A) \Delta (q_2, \eta)$,

oder $C_1 = (q_1, w, A\gamma)$, $C_2 = (q_2, w, \eta\gamma)$, und $(q_1, \varepsilon, A) \Delta (q_2, \eta)$.

Push-Down-Automat: Rechnung

Definition (Rechnung eines PDA)

Sei \mathcal{M} ein Push-Down-Automat.

$$C \vdash_{\mathcal{M}}^* C'$$

gdw es eine Reihe von Konfigurationen

$$C_0, C_1, \dots, C_n \quad (n \geq 0)$$

gibt, so dass

- $C = C_0$,
- $C' = C_n$,
- $C_i \vdash_{\mathcal{M}} C_{i+1}$ für alle $0 \leq i < n$

Dann heißt C_0, C_1, \dots, C_n eine **Rechnung** von \mathcal{M} .

Push-Down-Automat: Akzeptierte Sprache

Definition (von PDA akzeptierte Sprache)

Ein PDA \mathcal{M} kann auf zwei verschiedene Arten eine Sprache akzeptieren:

- über **finale Zustände**
- über **leeren Keller**

$$L_f(\mathcal{M}) = \{w \in \Sigma^* \mid \exists q \in F \exists \gamma \in \Gamma^* ((s_0, w, Z_0) \vdash_{\mathcal{M}}^* (q, \varepsilon, \gamma))\}$$

$$L_l(\mathcal{M}) = \{w \in \Sigma^* \mid \exists q \in K ((s_0, w, Z_0) \vdash_{\mathcal{M}}^* (q, \varepsilon, \varepsilon))\}$$

Push-Down-Automat: Akzeptierte Sprache

Definition (von PDA akzeptierte Sprache)

Ein PDA \mathcal{M} kann auf zwei verschiedene Arten eine Sprache akzeptieren:

- über **finale Zustände**
- über **leeren Keller**

$$L_f(\mathcal{M}) = \{w \in \Sigma^* \mid \exists q \in F \exists \gamma \in \Gamma^* ((s_0, w, Z_0) \vdash_{\mathcal{M}}^* (q, \varepsilon, \gamma))\}$$

$$L_l(\mathcal{M}) = \{w \in \Sigma^* \mid \exists q \in K ((s_0, w, Z_0) \vdash_{\mathcal{M}}^* (q, \varepsilon, \varepsilon))\}$$

Bemerkung

Das zu akzeptierende Wort w muss von \mathcal{M} ganz gelesen werden:

$$(s_0, w, Z_0) \vdash^* (q, \varepsilon, \cdot)$$

ist gefordert.

Push-Down-Automat

Bemerkung

- Das unterste Symbol im Keller kann gelöscht werden.
- Dann aber **hängt** der PDA.
- Er kann nicht mehr weiter rechnen.
- **Es gibt keine Nachfolgekonfiguration.**

Push-Down-Automat: Beispiel

Beispiel

Sprache der Palindrome über $\{a, b\}$:

$$L = \{w \in \{a, b\}^* \mid w = w^R\}$$

L wird über leeren Keller akzeptiert von dem PDA

$$\mathcal{M} := (\{s_0, s_1\}, \{a, b\}, \{Z_0, A, B\}, \Delta, s_0, Z_0, \emptyset)$$

mit ...

Push-Down-Automat: Beispiel

Beispiel (Forts.) Idee:

- Ein Palindrom $w = w^R$ hat die Form

$$vv^R \quad \text{oder} \quad vav^R \quad \text{oder} \quad vbv^R$$

für ein $v \in \{a, b\}^*$

- Der Automat \mathcal{M} liest v und merkt sich jeden Buchstaben.
- **Er rät indeterminiert die Wortmitte.**

Falls das Wort eine ungerade Anzahl von Buchstaben hat,
also $w = vav^R$ oder $w = vbv^R$,
dann muss dabei ein Buchstabe überlesen werden.

- Der Stack enthält nun v^R .
 \mathcal{M} muss jetzt nur noch jeden weiteren gelesenen Buchstaben mit dem jeweils obersten Kellersymbol vergleichen.

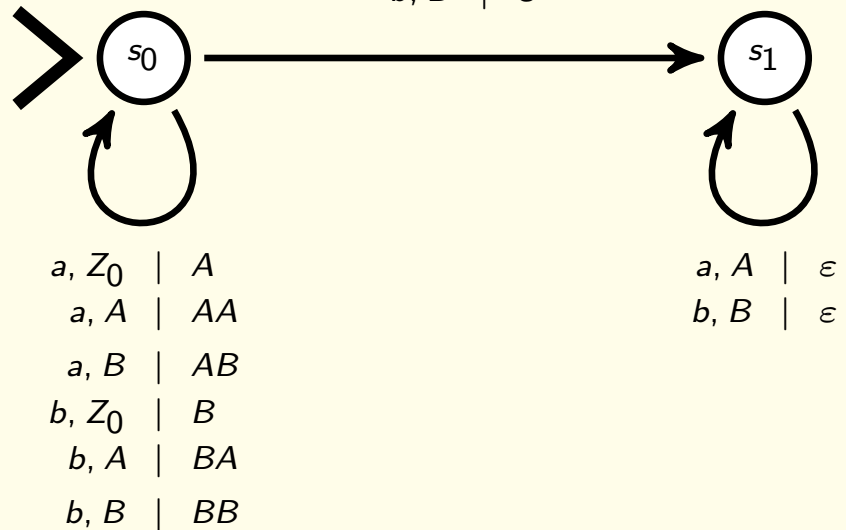
Push-Down-Automat: Beispiel

Beispiel. Sprache der Palindrome über $\{a, b\}$, $L = \{w \in \{a, b\}^* \mid w = w^R\}$

wird über leeren Keller akzeptiert von dem PDA

$\mathcal{M} := (\{s_0, s_1\}, \{a, b\}, \{Z_0, A, B\}, \Delta, s_0, Z_0, \emptyset)$:

ϵ, Z_0		ϵ
ϵ, A		ϵ
ϵ, B		ϵ
a, A		ϵ
b, B		ϵ



Push-Down-Automat: Beispiel

Beispiel

Sprache der Palindrome über $\{a, b\}$, $L = \{w \in \{a, b\}^* \mid w = w^R\}$

wird über leeren Keller akzeptiert von dem PDA

$\mathcal{M} := (\{s_0, s_1\}, \{a, b\}, \{Z_0, A, B\}, \Delta, s_0, Z_0, \emptyset)$ mit:

$(s_0, \varepsilon, Z_0) \Delta (s_1, \varepsilon)$ } ε akzeptieren

$(s_0, a, Z_0) \Delta (s_0, A)$
 $(s_0, a, A) \Delta (s_0, AA)$
 $(s_0, a, B) \Delta (s_0, AB)$
 $(s_0, b, Z_0) \Delta (s_0, B)$
 $(s_0, b, A) \Delta (s_0, BA)$
 $(s_0, b, B) \Delta (s_0, BB)$ } Stack aufbauen

Push-Down-Automat: Beispiel

Beispiel (Forts.)

$(s_0, \varepsilon, A) \Delta (s_1, \varepsilon)$ } Richtungswechsel für Palindrome
 $(s_0, \varepsilon, B) \Delta (s_1, \varepsilon)$ } mit ungerader Buchstabenanzahl

$(s_0, a, A) \Delta (s_1, \varepsilon)$ } Richtungswechsel für Palindrome
 $(s_0, b, B) \Delta (s_1, \varepsilon)$ } mit gerader Buchstabenanzahl

$(s_1, a, A) \Delta (s_1, \varepsilon)$ } Stack abbauen
 $(s_1, b, B) \Delta (s_1, \varepsilon)$ }

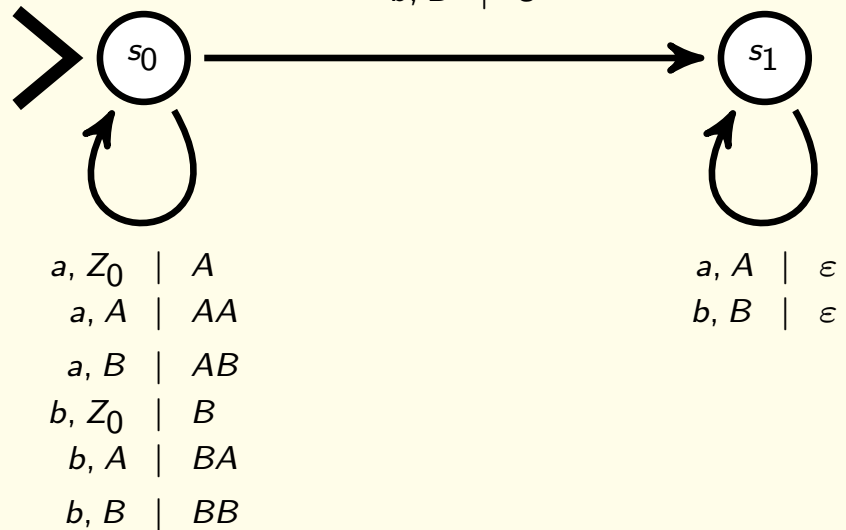
Push-Down-Automat: Beispiel

Beispiel. Sprache der Palindrome über $\{a, b\}$, $L = \{w \in \{a, b\}^* \mid w = w^R\}$

wird über leeren Keller akzeptiert von dem PDA

$\mathcal{M} := (\{s_0, s_1\}, \{a, b\}, \{Z_0, A, B\}, \Delta, s_0, Z_0, \emptyset)$:

ϵ, Z_0		ϵ
ϵ, A		ϵ
ϵ, B		ϵ
a, A		ϵ
b, B		ϵ



Push-Down-Automat: Beispiel

Beispiel (Forts.)

Für das Eingabewort *abbabba* rechnet \mathcal{M} so:

$$\begin{aligned} (s_0, \text{abbabba}, Z_0) \vdash (s_0, \text{bbabba}, A) \vdash (s_0, \text{babba}, BA) \vdash \\ (s_0, \text{abba}, BBA) \vdash (s_0, \text{bba}, ABBA) \vdash (s_1, \text{bba}, BBA) \vdash \\ (s_1, \text{ba}, BA) \quad \vdash \quad (s_1, \text{a}, A) \quad \vdash \quad (s_1, \varepsilon, \varepsilon) \end{aligned}$$

Push-Down-Automat: Beispiel

Die Sprache

$$L = \{w \in \{a, b\}^* \mid \#_a(w) = \#_b(w)\}$$

wird über finalen Zustand akzeptiert von dem PDA

$$\mathcal{M} = (\{s_0, s_1\}, \{a, b\}, \{Z_0, \underline{A}, A, \underline{B}, B\}, \Delta, s_0, Z_0, \{s_0\})$$

mit:

$$\begin{array}{ll} (s_0, a, Z_0) \Delta (s_1, \underline{A}) & (s_0, b, Z_0) \Delta (s_1, \underline{B}) \\ (s_1, a, \underline{A}) \Delta (s_1, A\underline{A}) & (s_1, b, \underline{B}) \Delta (s_1, B\underline{B}) \\ (s_1, a, A) \Delta (s_1, AA) & (s_1, b, B) \Delta (s_1, BB) \\ (s_1, a, \underline{B}) \Delta (s_0, Z_0) & (s_1, b, \underline{A}) \Delta (s_0, Z_0) \\ (s_1, a, B) \Delta (s_1, \varepsilon) & (s_1, b, A) \Delta (s_1, \varepsilon) \end{array}$$

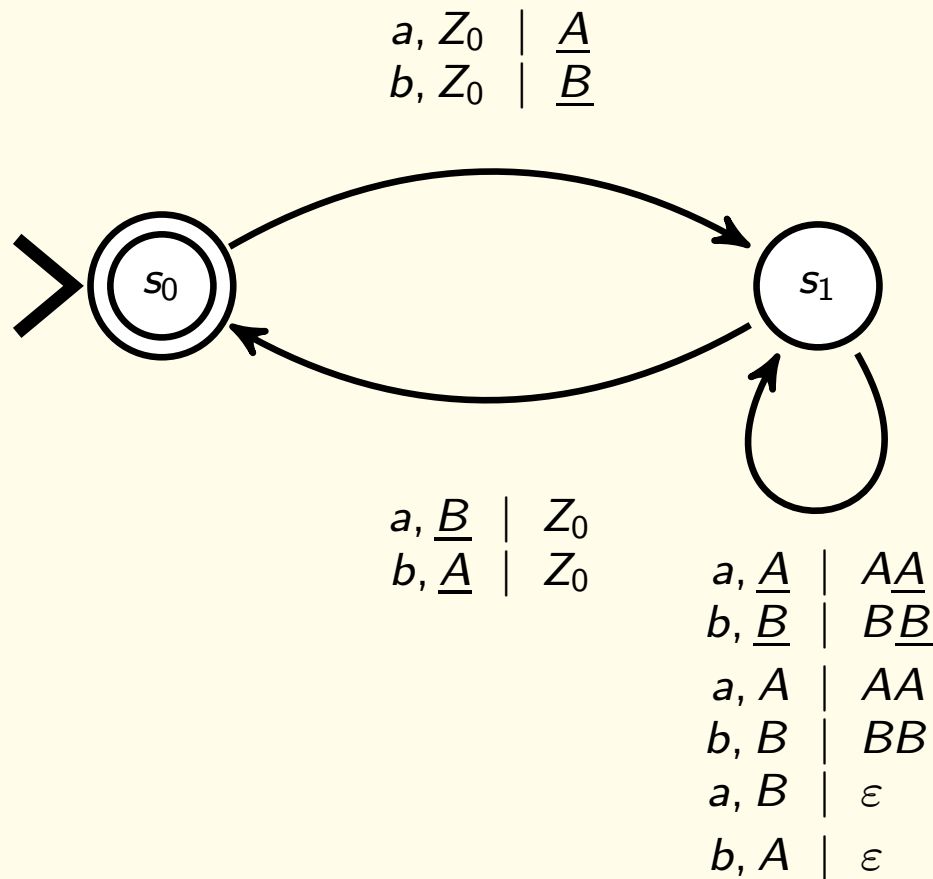
Push-Down-Automat: Beispiel

Idee:

- auf dem Stack mitzählen, wieviel A -Überhang oder B -Überhang momentan besteht
- Der Stack enthält zu jedem Zeitpunkt
 - entweder nur A/\underline{A} (A -Überhang)
 - oder nur B/\underline{B} (B -Überhang)
 - oder nur das Symbol Z_0 (Gleichstand).
- Das unterste A bzw. B auf dem Stack ist durch einen Unterstrich gekennzeichnet.

So weiß \mathcal{M} , wenn er dies Stacksymbol löscht, dass dann bis zu diesem Moment gleich viel as wie bs gelesen wurden.

Push-Down-Automat: Beispiel



Push-Down-Automat: Beispiel

Die Sprache

$$L = \{w \in \{a, b\}^* \mid \#_a(w) = \#_b(w)\}$$

wird über finalen Zustand akzeptiert von dem PDA

$$\mathcal{M} = (\{s_0, s_1\}, \{a, b\}, \{Z_0, \underline{A}, A, \underline{B}, B\}, \Delta, s_0, Z_0, \{s_0\})$$

mit:

$$\begin{array}{ll} (s_0, a, Z_0) \Delta (s_1, \underline{A}) & (s_0, b, Z_0) \Delta (s_1, \underline{B}) \\ (s_1, a, \underline{A}) \Delta (s_1, A\underline{A}) & (s_1, b, \underline{B}) \Delta (s_1, B\underline{B}) \\ (s_1, a, A) \Delta (s_1, AA) & (s_1, b, B) \Delta (s_1, BB) \\ (s_1, a, \underline{B}) \Delta (s_0, Z_0) & (s_1, b, \underline{A}) \Delta (s_0, Z_0) \\ (s_1, a, B) \Delta (s_1, \varepsilon) & (s_1, b, A) \Delta (s_1, \varepsilon) \end{array}$$

Übersicht

- Pumping Lemma für kontextfreie Sprachen
- Push-Down-Automat

Definition

Beispiele

Konfiguration eines PDA

Rechnung eines PDA

Akzeptierte Sprache

- über **finale Zustände**
- über **leeren Keller**