Karlsruher Institut für Technologie

Theoretische Grundlagen der Informatik Übung

8. Übungstermin · 24. Januar 2019

Guido Brückner

INSTITUT FÜR THEORETISCHE INFORMATIK · LEHRSTUHL ALGORITHMIK

Übersicht

Anmeldung zur Hauptklausur:

- **Anmeldeschluss**

  19.03.2019

- Eine Anmeldung zur Hauptklausur ist nach Anmeldeschluss nicht mehr möglich!
- Anmeldung erfolgt in der Regel online über das Studierendenportal.
- Bei Problemen bitte bei Ioana Gheta melden.

Inhalt

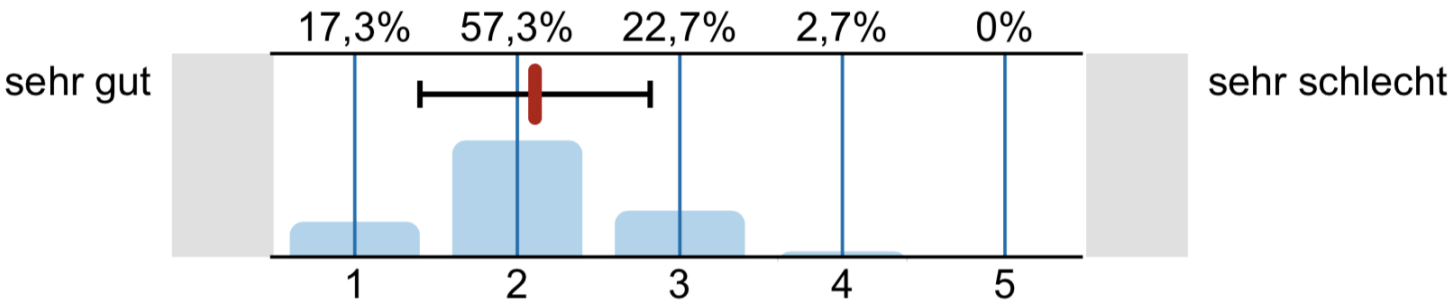
- Kontextfreie Grammatiken
  - Pumpinglemma
- Greibach-Normalform
- Kellerautomaten

Institut für Theoretische Informatik
Lehrstuhl Algorithmik

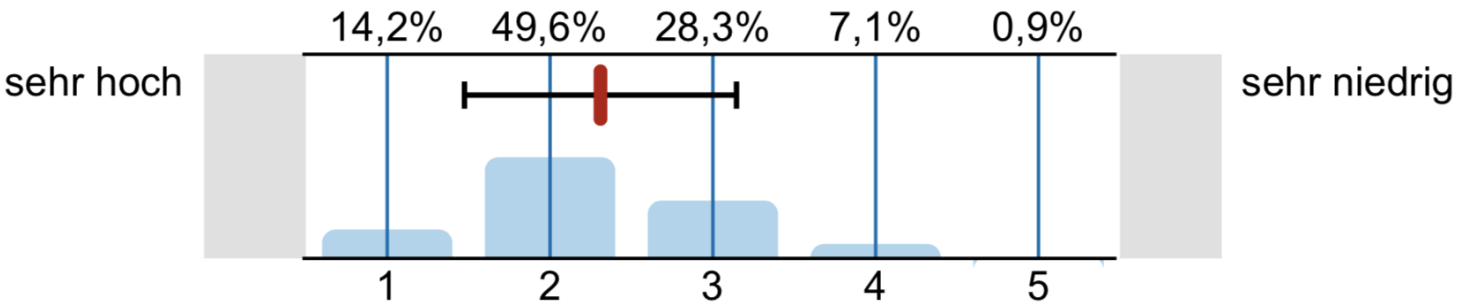
Evaluation

Karlsruher Institut für Technologie

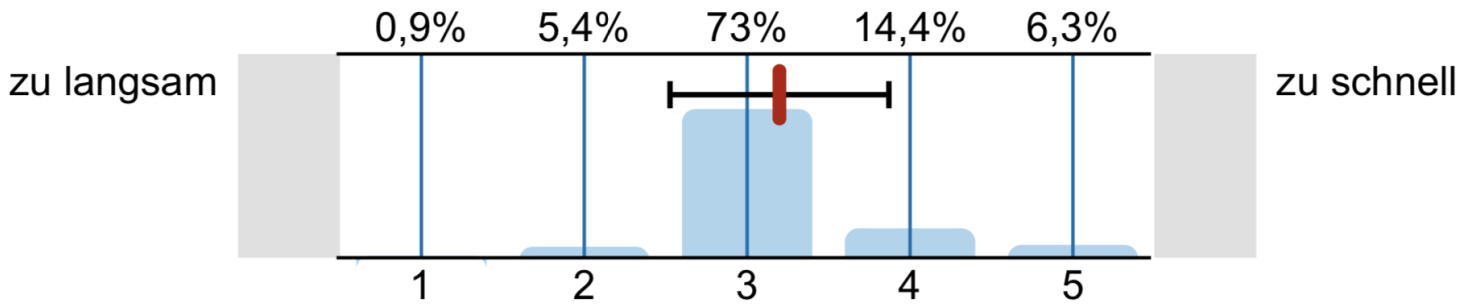
Insgesamt



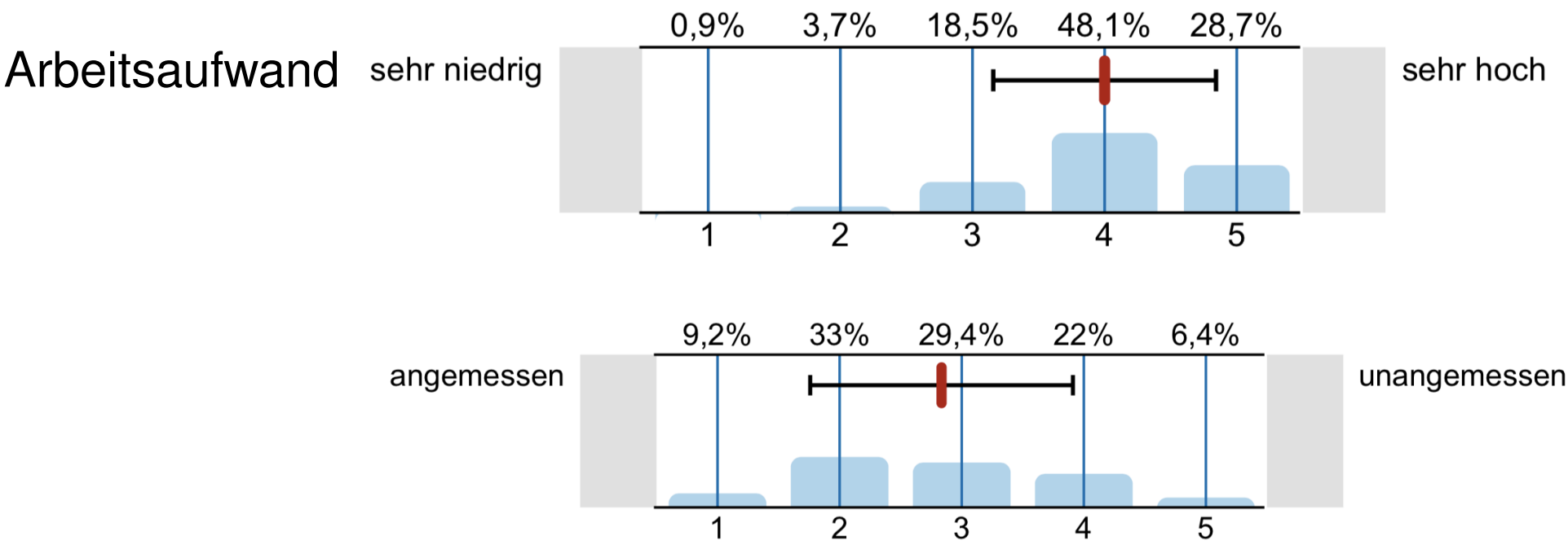
Lernzuwachs



Geschwindigkeit



Evaluation



Institut für Theoretische Informatik
Lehrstuhl Algorithmik

Evaluation

„Nicht gut gefallen hat mir:“

- Mikrofon zu leise
- Beamer unscharf
- Stufen im Hörsall
- Schwierigkeit der Beweise / Übungsblätter

„Gut gefallen hat mir:“

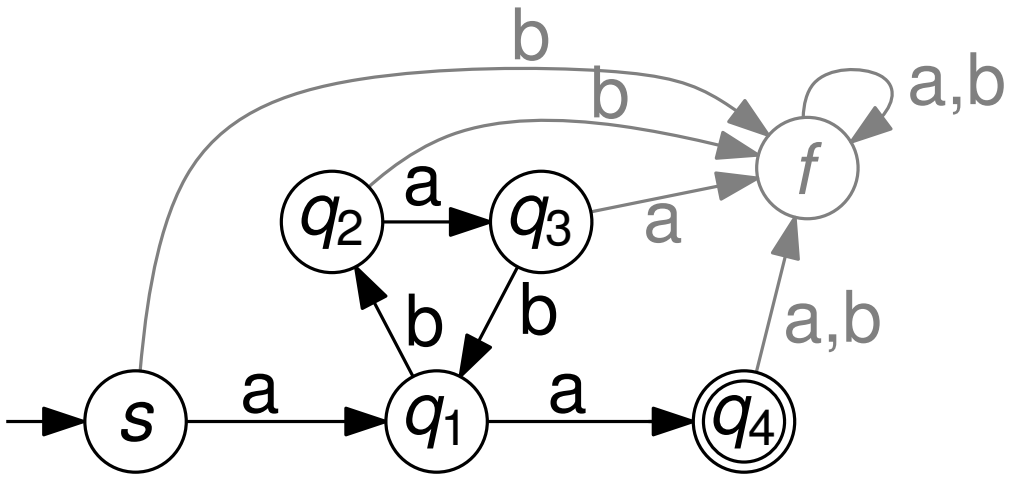
- Folien
- Beispiele
- step-by-step Konstruktionen
- Übungsblätter

Wiederholung: Pumpingl. für reguläre Sprachen

Sei L eine reguläre Sprache. Dann existiert eine Zahl $n \in \mathbb{N}$, so dass für jedes Wort $w \in L$ mit $|w| > n$ eine Darstellung

$$w = uvx \text{ mit } |uv| \leq n \text{ und } v \neq \varepsilon$$

existiert, bei der auch $uv^i x \in L$ ist für alle $i \in \mathbb{N}_0$.



Institut für Theoretische Informatik
Lehrstuhl Algorithmik

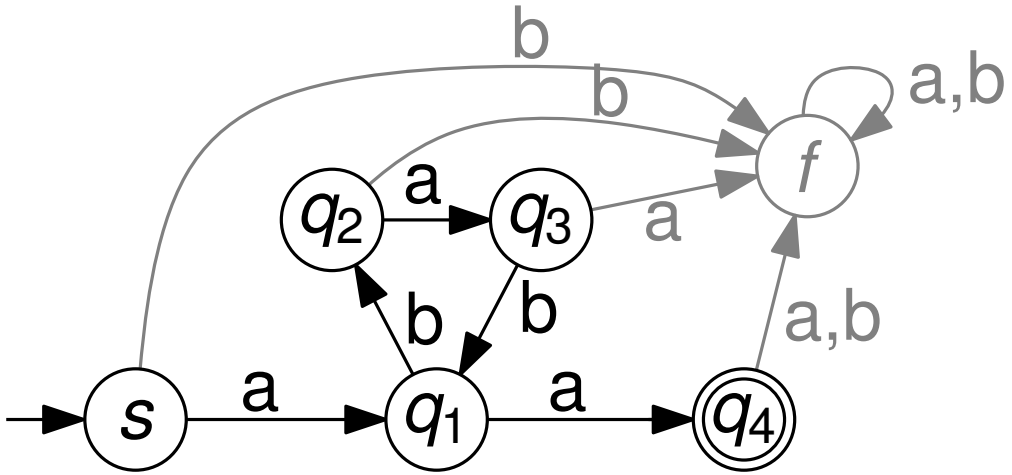
Wiederholung: Pumpingl. für reguläre Sprachen

Sei L eine reguläre Sprache. Dann existiert eine Zahl $n \in \mathbb{N}$, so dass für jedes Wort $w \in L$ mit $|w| > n$ eine Darstellung

$$w = uvx \text{ mit } |uv| \leq n \text{ und } v \neq \varepsilon$$

existiert, bei der auch $uv^i x \in L$ ist für alle $i \in \mathbb{N}_0$.

Erklärung: Sei L reguläre Sprache und $\mathcal{A}$ entsprechender endlicher Automat.



Institut für Theoretische Informatik
Lehrstuhl Algorithmik

Wiederholung: Pumpingl. für reguläre Sprachen

Sei L eine reguläre Sprache. Dann existiert eine Zahl $n \in \mathbb{N}$, so dass für jedes Wort $w \in L$ mit $|w| > n$ eine Darstellung

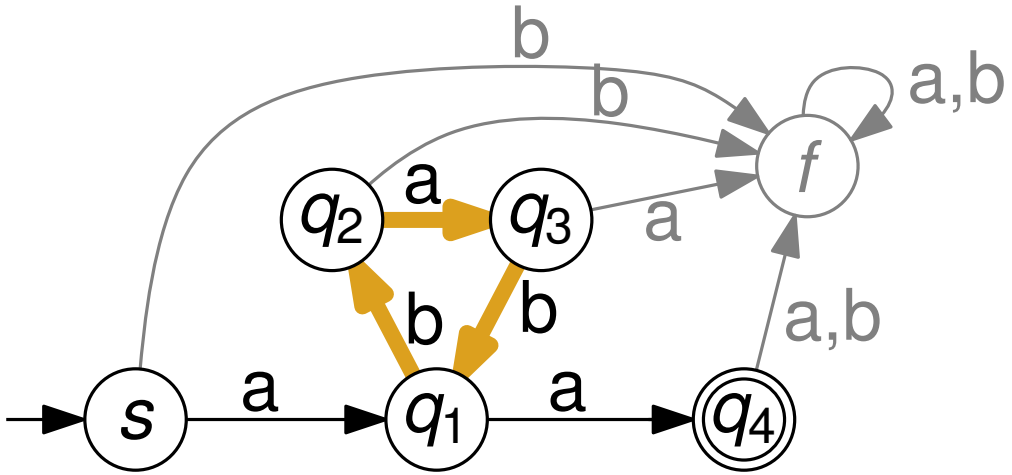
$$w = uvx \text{ mit } |uv| \leq n \text{ und } v \neq \varepsilon$$

existiert, bei der auch $uv^i x \in L$ ist für alle $i \in \mathbb{N}_0$.

Erklärung: Sei L reguläre Sprache und $\mathcal{A}$ entsprechender endlicher Automat.

Egal wie viele Zustände für $\mathcal{A}$ verwendet werden, für jedes Wort $w \in L$, das mehr Zeichen hat als $\mathcal{A}$ Zustände, gilt:

Während der Abarbeitung von w durchläuft man einen Zyklus $\mathcal{Z}$ in $\mathcal{A}$.



Institut für Theoretische Informatik
Lehrstuhl Algorithmik

Wiederholung: Pumpingl. für reguläre Sprachen

Sei L eine reguläre Sprache. Dann existiert eine Zahl $n \in \mathbb{N}$, so dass für jedes Wort $w \in L$ mit $|w| > n$ eine Darstellung

$$w = uvx \text{ mit } |uv| \leq n \text{ und } v \neq \varepsilon$$

existiert, bei der auch $uv^i x \in L$ ist für alle $i \in \mathbb{N}_0$.

Erklärung: Sei L reguläre Sprache und $\mathcal{A}$ entsprechender endlicher Automat.

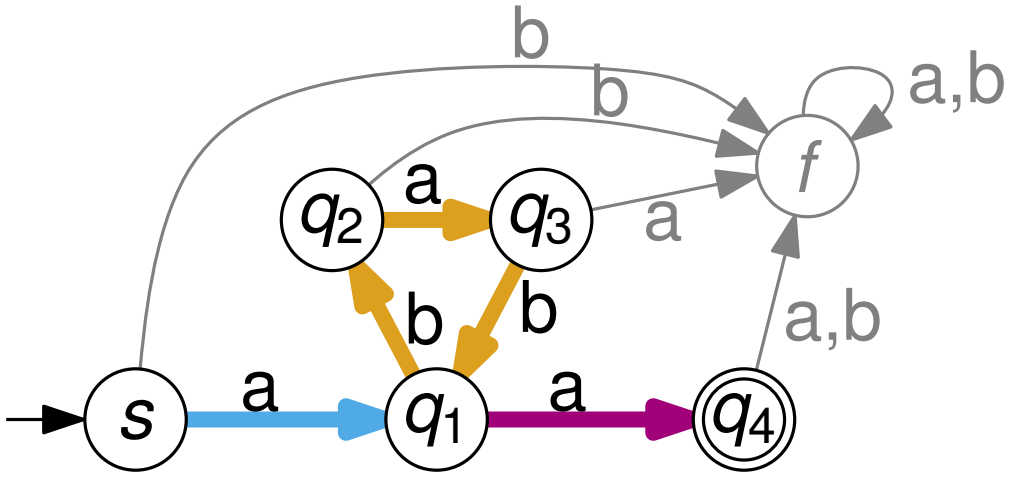
Egal wie viele Zustände für $\mathcal{A}$ verwendet werden, für jedes Wort $w \in L$, das mehr Zeichen hat als $\mathcal{A}$ Zustände, gilt:

Während der Abarbeitung von w durchläuft man einen Zyklus $\mathcal{Z}$ in $\mathcal{A}$.

Sei

- u das Teilwort von w, das vor $\mathcal{Z}$,
- v das Teilwort von w, das in $\mathcal{Z}$, und
- x das Teilwort von w, das nach $\mathcal{Z}$

abgearbeitet wird.



Institut für Theoretische Informatik
Lehrstuhl Algorithmik

Wiederholung: Pumpingl. für reguläre Sprachen

Sei L eine reguläre Sprache. Dann existiert eine Zahl $n \in \mathbb{N}$, so dass für jedes Wort $w \in L$ mit $|w| > n$ eine Darstellung

$$w = uvx \text{ mit } |uv| \leq n \text{ und } v \neq \varepsilon$$

existiert, bei der auch $uv^i x \in L$ ist für alle $i \in \mathbb{N}_0$.

Erklärung: Sei L reguläre Sprache und $\mathcal{A}$ entsprechender endlicher Automat.

Egal wie viele Zustände für $\mathcal{A}$ verwendet werden, für jedes Wort $w \in L$, das mehr Zeichen hat als $\mathcal{A}$ Zustände, gilt:

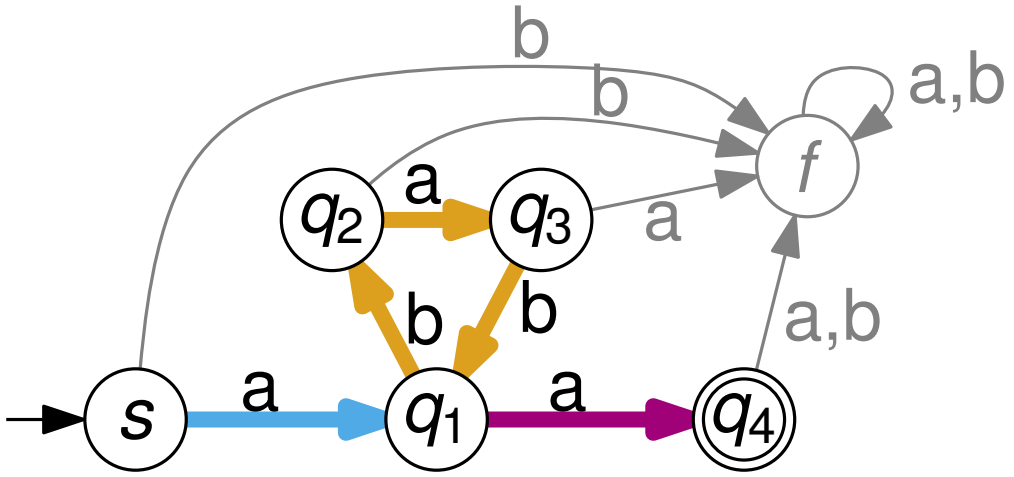
Während der Abarbeitung von w durchläuft man einen Zyklus $\mathcal{Z}$ in $\mathcal{A}$.

Sei

- u das Teilwort von w, das vor $\mathcal{Z}$,
- v das Teilwort von w, das in $\mathcal{Z}$, und
- x das Teilwort von w, das nach $\mathcal{Z}$

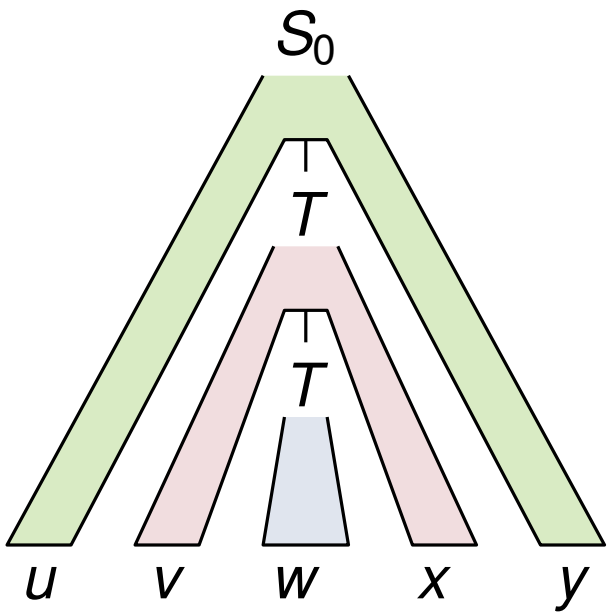
abgearbeitet wird.

→ $uv^i x$ (mit $i \in \mathbb{N}_0$) ist auch in L enthalten.
Durchlaufe $\mathcal{Z}$ entsprechend häufig.

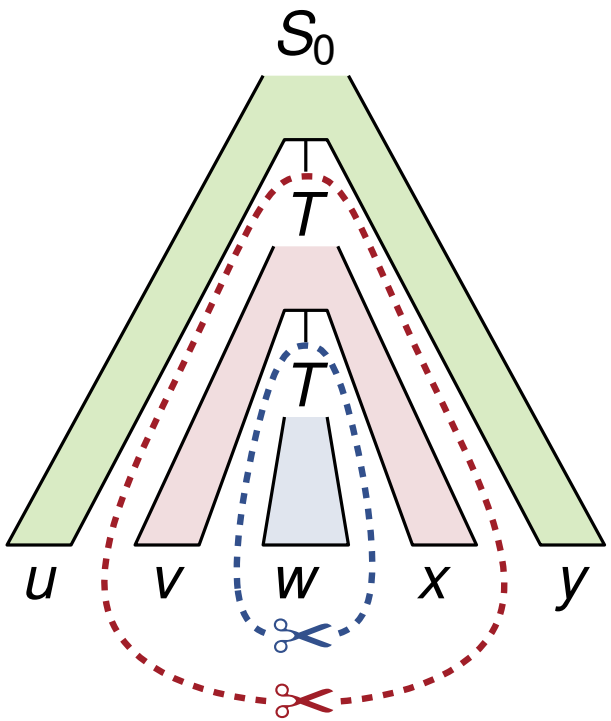


Institut für Theoretische Informatik
Lehrstuhl Algorithmik

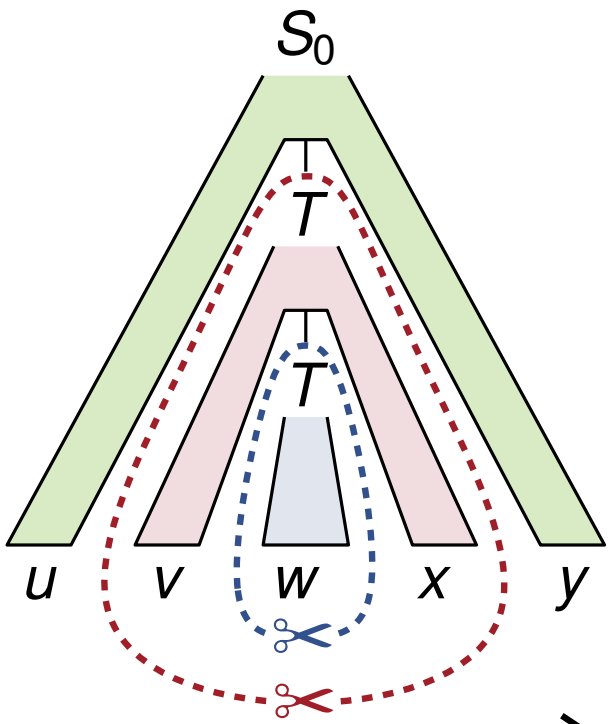
Pumpinglemma für kontextfreie Sprachen



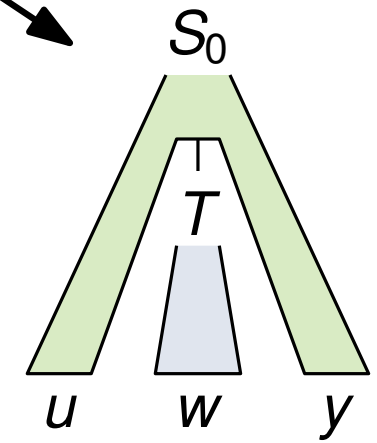
Pumpinglemma für kontextfreie Sprachen



Pumpinglemma für kontextfreie Sprachen



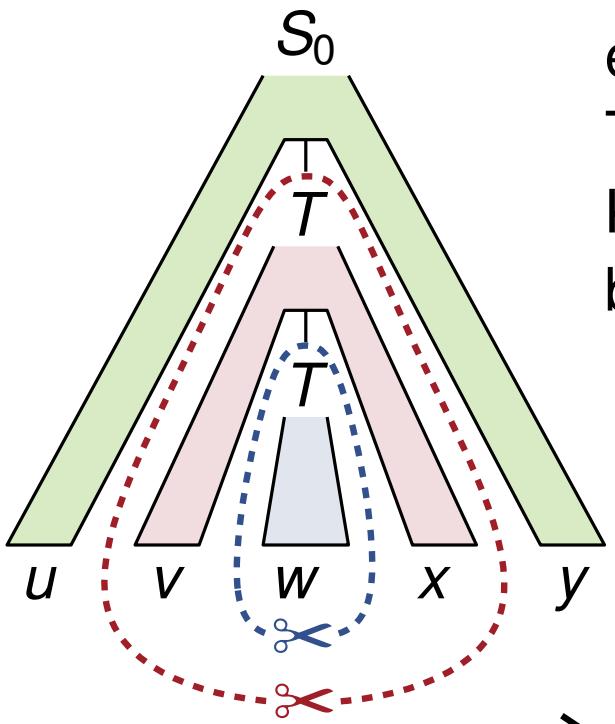
ersetze den roten Teilbaum durch den blauen Teilbaum



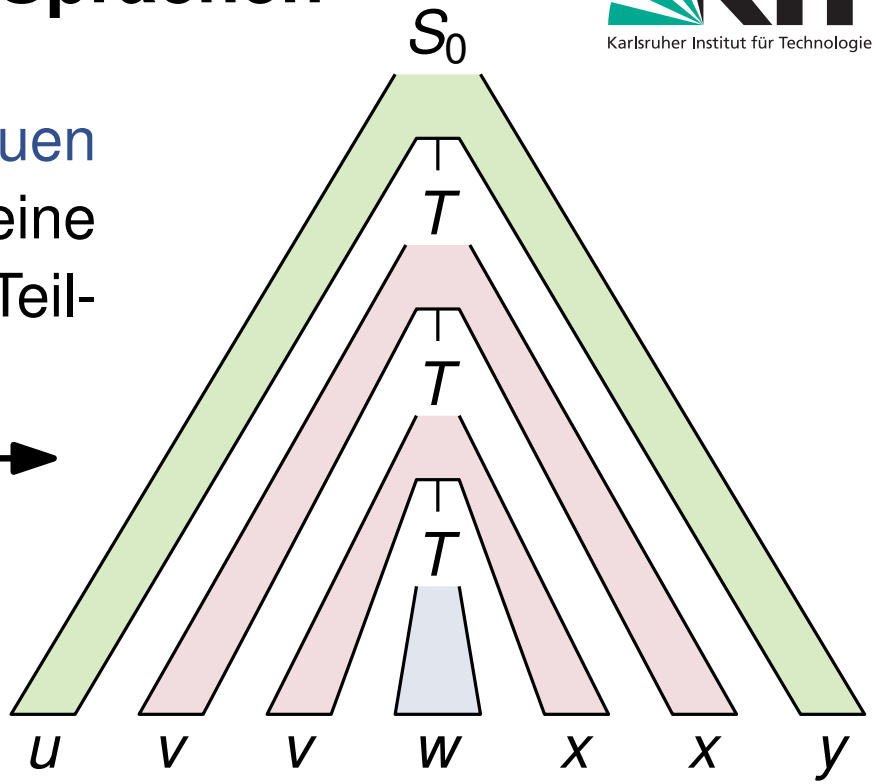
$(= uv^0wx^0y)$

Institut für Theoretische Informatik
Lehrstuhl Algorithmik

Pumpinglemma für kontextfreie Sprachen

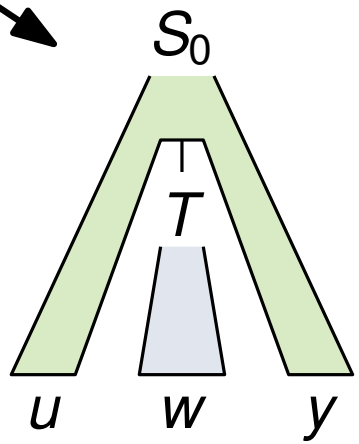


ersetze den blauen Teilbaum durch eine Kopie des roten Teilbaums



$(= uv^2wx^2y)$

ersetze den roten Teilbaum durch den blauen Teilbaum

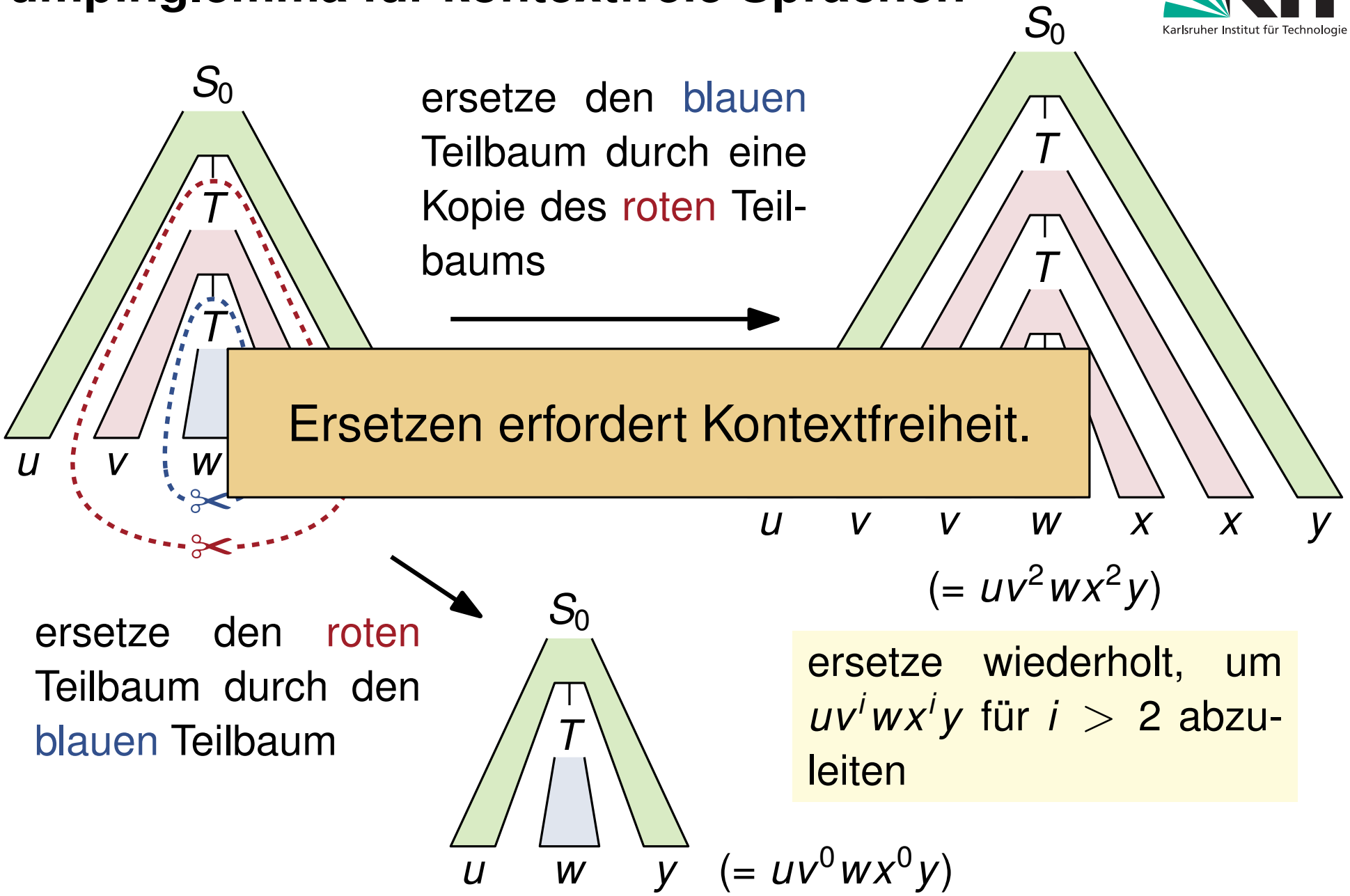


$(= uv^0wx^0y)$

ersetze wiederholt, um uv^iwx^iy für $i > 2$ abzuleiten

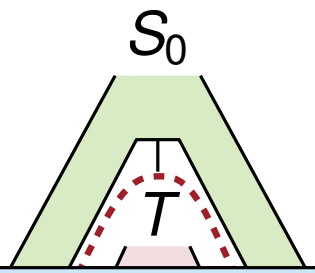
Institut für Theoretische Informatik
Lehrstuhl Algorithmik

Pumpinglemma für kontextfreie Sprachen

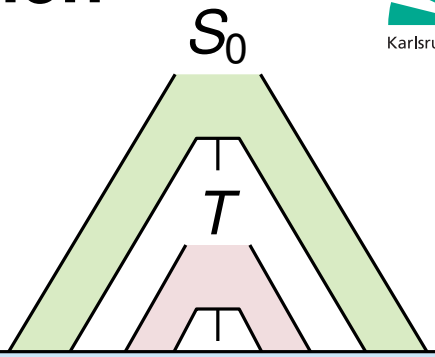


Institut für Theoretische Informatik
Lehrstuhl Algorithmik

Pumpinglemma für kontextfreie Sprachen



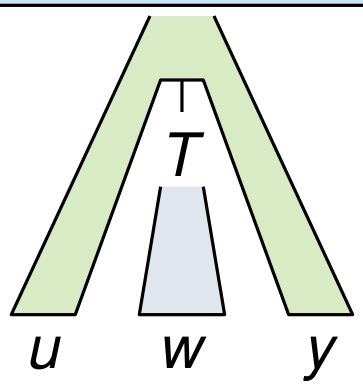
ersetze den blauen Teilbaum durch eine Kopie des roten Teil-



Für jede kontextfreie Sprache L gibt es eine Konstante $n \in \mathbb{N}$, so dass sich jedes Wort $z \in L$ mit $|z| \geq n$ so in $z = uvwxy$ zerlegen lässt, dass gilt:

- $|vx| \geq 1$,
- $|vwx| \leq n$ und
- $uv^i wx^i y \in L$ für alle $i \geq 0$.

ersetze den roten Teilbaum durch den blauen Teilbaum



$(= uv^0 wx^0 y)$

ersetze wiederholt, um $uv^i wx^i y$ für $i > 2$ abzuleiten

Institut für Theoretische Informatik
Lehrstuhl Algorithmik

Kontextfreiheit widerlegen $L_1 = \{a^k b a^{2k} b a^{3k} \in \{a,b\}^\star \mid k \geq 0\}$

Betrachte Zerlegung $a^n b a^{2n} b a^{3n} = uvwxy$

Fall 1: v oder x enthält ein b

Fall 2: vwx enthält kein b

Fall 3: vwx überspannt ein b.

Für jede kontextfreie Sprache L gibt es eine Konstante $n \in \mathbb{N}$, so dass sich jedes Wort $z \in L$ mit $|z| \geq n$ in

- $z = uvwxy$

zerlegen lässt und

- $|vx| \geq 1$,
- $|vwx| \leq n$ und
- $uv^i wx^i y \in L$ für alle $i \geq 0$.

Institut für Theoretische Informatik
Lehrstuhl Algorithmik

Kontextfreiheit widerlegen $L_1 = \{a^k b a^{2k} b a^{3k} \in \{a,b\}^\star \mid k \geq 0\}$

Betrachte Zerlegung $a^n b a^{2n} b a^{3n} = uvwxy$

Fall 1: v oder x enthält ein b

Betrachte $w' = uv^2wx^2y$

→ wegen $|vx| \geq 1$ muss w' mehr als zwei b's enthalten.

$uv^2wx^2y \in L$

Fall 2: vwx enthält kein b

Fall 3: vwx überspannt ein b.

so dass sich jedes Wort $z \in L$ mit $|z| \geq n$ in

- $z = uvwxy$

zerlegen lässt und

- $|vx| \geq 1$,
- $|vwx| \leq n$ und
- $uv^iwx^iy \in L$ für alle $i \geq 0$.

Institut für Theoretische Informatik
Lehrstuhl Algorithmik

Kontextfreiheit widerlegen $L_1 = \{a^k b a^{2k} b a^{3k} \in \{a,b\}^\star \mid k \geq 0\}$

Betrachte Zerlegung $a^n b a^{2n} b a^{3n} = uvwxy$

Fall 1: v oder x enthält ein b

Fall 2: vwx enthält kein b

Betrachte das Wort $w' = uv^0 wx^0 y$

Fall 2.1: vwx tritt vor vor dem ersten b auf.

→ wegen $|vx| \geq 1$ gilt $w' = a^m\ b\ a^{2n}\ b\ a^{3n}$ mit $m < n$ ↯ $w' \in L_1$

Fall 3: vwx überspannt ein b.

Für jede kontextfreie Sprache L gibt es eine Konstante $n \in \mathbb{N}$, so dass sich jedes Wort $z \in L$ mit $|z| \geq n$ in

- $z = uvwxy$

zerlegen lässt und

- $|vx| \geq 1$,
- $|vwx| \leq n$ und
- $uv^i wx^i y \in L$ für alle $i \geq 0$.

Institut für Theoretische Informatik
Lehrstuhl Algorithmik

Kontextfreiheit widerlegen $L_1 = \{a^k b a^{2k} b a^{3k} \in \{a,b\}^\star \mid k \geq 0\}$

Betrachte Zerlegung $a^n b a^{2n} b a^{3n} = uvwxy$

Fall 1: v oder x enthält ein b

Fall 2: vwx enthält kein b

Betrachte das Wort $w' = uv^0wx^0y$

Fall 2.1: vwx tritt vor vor dem ersten b auf.

→ wegen $|vx| \geq 1$ gilt $w' = a^m\ b\ a^{2n}\ b\ a^{3n}$ mit $m < n$ ↯ $w' \in L_1$

Fall 2.2: vwx liegt zwischen den b's.

→ wegen $|vx| \geq 1$ gilt $w' = a^n\ b\ a^m\ b\ a^{3n}$ mit $m < 2n$ ↯ $w' \in L_1$

Fall 3: vwx überspannt ein b.

Für jede kontextfreie Sprache L gibt es eine Konstante $n \in \mathbb{N}$, so dass sich jedes Wort $z \in L$ mit $|z| \geq n$ in

- $z = uvwxy$

zerlegen lässt und

- $|vx| \geq 1$,
- $|vwx| \leq n$ und
- $uv^iwx^iy \in L$ für alle $i \geq 0$.

Institut für Theoretische Informatik
Lehrstuhl Algorithmik

Kontextfreiheit widerlegen $L_1 = \{a^k b a^{2k} b a^{3k} \in \{a,b\}^\star \mid k \geq 0\}$

Betrachte Zerlegung $a^n b a^{2n} b a^{3n} = uvwxy$

Fall 1: v oder x enthält ein b

Fall 2: vwx enthält kein b

Betrachte das Wort $w' = uv^0wx^0y$

Fall 2.1: vwx tritt vor vor dem ersten b auf.

→ wegen $|vx| \geq 1$ gilt $w' = a^m\ b\ a^{2n}\ b\ a^{3n}$ mit $m < n$ ↯ $w' \in L_1$

Fall 2.2: vwx liegt zwischen den b's.

→ wegen $|vx| \geq 1$ gilt $w' = a^n\ b\ a^m\ b\ a^{3n}$ mit $m < 2n$ ↯ $w' \in L_1$

Fall 2.3: vwx liegt hinter dem zweiten b.

→ wegen $|vx| \geq 1$ gilt
$w' = a^n\ b\ a^{2n}\ b\ a^m$ mit $m < 3n$ ↯

Fall 3: vwx überspannt ein b.

Für jede kontextfreie Sprache L gibt es eine Konstante $n \in \mathbb{N}$, so dass sich jedes Wort $z \in L$ mit $|z| \geq n$ in

- $z = uvwxy$

zerlegen lässt und

- $|vx| \geq 1$,
- $|vwx| \leq n$ und
- $uv^iwx^iy \in L$ für alle $i \geq 0$.

Institut für Theoretische Informatik
Lehrstuhl Algorithmik

Kontextfreiheit widerlegen

$$L_2 = \{wcv \in \{a, b, c\}^\star \mid w \text{ ist ein Teilwort von } v \text{ mit } v \in \{a, b\}^\star\}$$

Betrachte Zerlegung $a^n b^n c a^n b^n = uvwxy$

Fall 1: v oder x enthält ein c
Fall 2: vwx tritt vor c auf

Betrachte das Wort $w' = uv^2wx^2y = w_1 c w_2$ für $w_1, w_2 \in \{a, b\}^\star$

Für jede kontextfreie Sprache L gibt es eine Konstante $n \in \mathbb{N}$, so dass sich jedes Wort $z \in L$ mit $|z| \geq n$ in

- $z = uvwxy$

zerlegen lässt und

- $|vx| \geq 1$,
- $|vwx| \leq n$ und
- $uv^iwx^iy \in L$ für alle $i \geq 0$.

Institut für Theoretische Informatik
Lehrstuhl Algorithmik

Kontextfreiheit widerlegen

$$L_2 = \{wcv \in \{a, b, c\}^\star \mid w \text{ ist ein Teilwort von } v \text{ mit } v \in \{a, b\}^\star\}$$

Betrachte Zerlegung $a^n b^n c a^n b^n = uvwxy$

Fall 1: v oder x enthält ein c

Betrachte $w' = uv^2wx^2y \in L_2$

→ wegen $|vx| \geq 1$ muss w' mehr als ein c enthalten.

Für jede kontextfreie Sprache L gibt es eine Konstante $n \in \mathbb{N}$, so dass sich jedes Wort $z \in L$ mit $|z| \geq n$ in

- $z = uvwxy$

zerlegen lässt und

- $|vx| \geq 1$,
- $|vwx| \leq n$ und
- $uv^iwx^iy \in L$ für alle $i \geq 0$.

Institut für Theoretische Informatik
Lehrstuhl Algorithmik

Kontextfreiheit widerlegen

$$L_2 = \{wcv \in \{a, b, c\}^\star \mid w \text{ ist ein Teilwort von } v \text{ mit } v \in \{a, b\}^\star\}$$

Betrachte Zerlegung $a^n b^n c a^n b^n = uvwxy$

Fall 1: v oder x enthält ein c
Fall 2: vwx tritt vor c auf

Betrachte das Wort $w' = uv^2wx^2y = w_1 c w_2$ für $w_1, w_2 \in \{a, b\}^\star$

→ wegen $|vx| \geq 1$ gilt $|w_1| > |w_2|$

→ w_1 ist nicht Teilwort von w_2 ↯ $w' \in L_2$

Für jede kontextfreie Sprache L gibt es eine Konstante $n \in \mathbb{N}$, so dass sich jedes Wort $z \in L$ mit $|z| \geq n$ in

- $z = uvwxy$

zerlegen lässt und

- $|vx| \geq 1$,
- $|vwx| \leq n$ und
- $uv^iwx^iy \in L$ für alle $i \geq 0$.

Institut für Theoretische Informatik
Lehrstuhl Algorithmik

Kontextfreiheit widerlegen

$$L_2 = \{wcv \in \{a, b, c\}^\star \mid w \text{ ist ein Teilwort von } v \text{ mit } v \in \{a, b\}^\star\}$$

Betrachte Zerlegung $a^n b^n c a^n b^n = uvwxy$

Fall 1: v oder x enthält ein c
Fall 2: vwx tritt vor c auf
Fall 3: vwx tritt nach c auf

Betrachte das Wort $w' = uv^0wx^0y = w_1cw_2$ für $w_1, w_2 \in \{a, b\}^\star$

→ wegen $|vx| \geq 1$ gilt $|w_1| > |w_2|$

→ w_1 ist nicht Teilwort von w_2 ↯ $w' \in L_2$

Für jede kontextfreie Sprache L gibt es eine Konstante $n \in \mathbb{N}$, so dass sich jedes Wort $z \in L$ mit $|z| \geq n$ in

- $z = uvwxy$

zerlegen lässt und

- $|vx| \geq 1$,
- $|vwx| \leq n$ und
- $uv^iwx^iy \in L$ für alle $i \geq 0$.

Institut für Theoretische Informatik
Lehrstuhl Algorithmik

Kontextfreiheit widerlegen

$$L_2 = \{wcv \in \{a, b, c\}^\star \mid w \text{ ist ein Teilwort von } v \text{ mit } v \in \{a, b\}^\star\}$$

Betrachte Zerlegung $a^n b^n c a^n b^n = uvwxy$

Fall 1: v oder x enthält ein c
Fall 2: vwx tritt vor c auf
Fall 3: vwx tritt nach c auf
Fall 4: w überspannt c

Fall 4.1: x beginnt mit einem a

Für jede kontextfreie Sprache L gibt es eine Konstante $n \in \mathbb{N}$, so dass sich jedes Wort $z \in L$ mit $|z| \geq n$ in

- $z = uvwxy$

zerlegen lässt und

- $|vx| \geq 1$,
- $|vwx| \leq n$ und
- $uv^i wx^i y \in L$ für alle $i \geq 0$.

Institut für Theoretische Informatik
Lehrstuhl Algorithmik

Kontextfreiheit widerlegen

$$L_2 = \{wcv \in \{a, b, c\}^\star \mid w \text{ ist ein Teilwort von } v \text{ mit } v \in \{a, b\}^\star\}$$

Betrachte Zerlegung $a^n b^n c a^n b^n = uvwxy$

Fall 1: v oder x enthält ein c
Fall 2: vwx tritt vor c auf
Fall 3: vwx tritt nach c auf
Fall 4: w überspannt c

Fall 4.1: x beginnt mit einem a

→ Betrachte das Wort $w' = uv^0wx^0y = w_1cw_2$ für $w_1, w_2 \in \{a, b\}^\star$

→ w_1 enthält ein a, während w_2 nur aus b's besteht.

→ w_1 ist kein Teilwort von w_2 ↯ $w' \in L_2$

Für jede kontextfreie Sprache L gibt es eine Konstante $n \in \mathbb{N}$, so dass sich jedes Wort $z \in L$ mit $|z| \geq n$ in

- $z = uvwxy$

zerlegen lässt und

- $|vx| \geq 1$,
- $|vwx| \leq n$ und
- $uv^iwx^iy \in L$ für alle $i \geq 0$.

Institut für Theoretische Informatik
Lehrstuhl Algorithmik

Kontextfreiheit widerlegen

$$L_2 = \{wcv \in \{a, b, c\}^\star \mid w \text{ ist ein Teilwort von } v \text{ mit } v \in \{a, b\}^\star\}$$

Betrachte Zerlegung $a^n b^n c a^n b^n = uvwxy$

Fall 1: v oder x enthält ein c
Fall 2: vwx tritt vor c auf
Fall 3: vwx tritt nach c auf
Fall 4: w überspannt c

Fall 4.1: x beginnt mit einem a
Fall 4.2: x beginnt mit einem b

Für jede kontextfreie Sprache L gibt es eine Konstante $n \in \mathbb{N}$, so dass sich jedes Wort $z \in L$ mit $|z| \geq n$ in

- $z = uvwxy$

zerlegen lässt und

- $|vx| \geq 1$,
- $|vwx| \leq n$ und
- $uv^i wx^i y \in L$ für alle $i \geq 0$.

Institut für Theoretische Informatik
Lehrstuhl Algorithmik

Kontextfreiheit widerlegen

$$L_2 = \{wcv \in \{a, b, c\}^\star \mid w \text{ ist ein Teilwort von } v \text{ mit } v \in \{a, b\}^\star\}$$

Betrachte Zerlegung $a^n b^n c a^n b^n = uvwxy$

Fall 1: v oder x enthält ein c
Fall 2: vwx tritt vor c auf
Fall 3: vwx tritt nach c auf
Fall 4: w überspannt c

Fall 4.1: x beginnt mit einem a
Fall 4.2: x beginnt mit einem b (Wegen $|vwx| \leq n$ nicht möglich)
Fall 4.3: $x = \varepsilon$

Für jede kontextfreie Sprache L gibt es eine Konstante $n \in \mathbb{N}$, so dass sich jedes Wort $z \in L$ mit $|z| \geq n$ in

- $z = uvwxy$

zerlegen lässt und

- $|vx| \geq 1$,
- $|vwx| \leq n$ und
- $uv^i wx^i y \in L$ für alle $i \geq 0$.

Institut für Theoretische Informatik
Lehrstuhl Algorithmik

Kontextfreiheit widerlegen

$$L_2 = \{wcv \in \{a, b, c\}^\star \mid w \text{ ist ein Teilwort von } v \text{ mit } v \in \{a, b\}^\star\}$$

Betrachte Zerlegung $a^n b^n c a^n b^n = uvwxy$

Fall 1: v oder x enthält ein c
Fall 2: vwx tritt vor c auf
Fall 3: vwx tritt nach c auf
Fall 4: w überspannt c

Fall 4.1: x beginnt mit einem a
Fall 4.2: x beginnt mit einem b (Wegen $|vwx| \leq n$ nicht möglich)
Fall 4.3: $x = \varepsilon$

Betrachte das Wort $w' = uv^2wx^2y = w_1 c w_2$ für $w_1, w_2 \in \{a, b\}^\star$

→ wegen $|vx| \geq 1$ gilt $|w_1| > |w_2|$

→ w_1 ist nicht Teilwort von w_2

Für jede kontextfreie Sprache L gibt es eine Konstante $n \in \mathbb{N}$, so dass sich jedes Wort $z \in L$ mit $|z| \geq n$ in

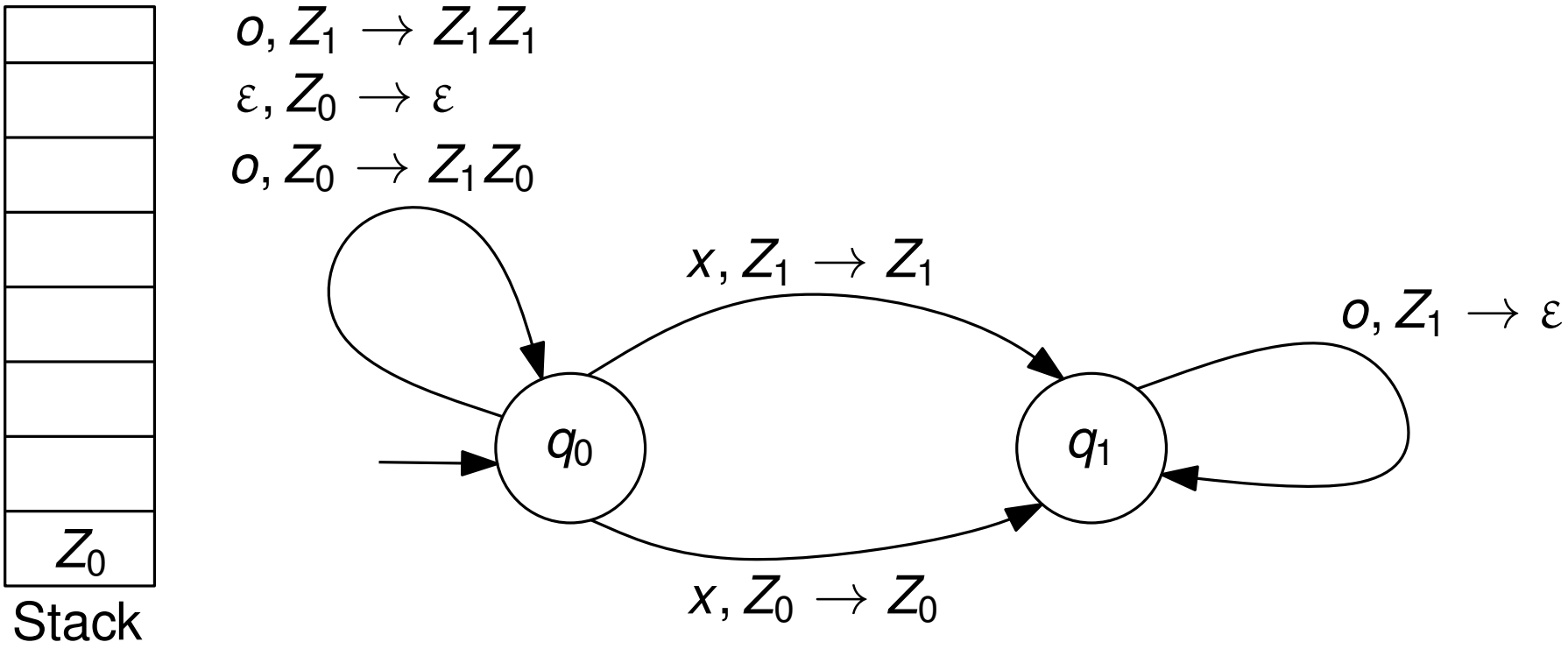
- $z = uvwxy$

zerlegen lässt und

- $|vx| \geq 1$,
- $|vwx| \leq n$ und
- $uv^iwx^iy \in L$ für alle $i \geq 0$.

Institut für Theoretische Informatik
Lehrstuhl Algorithmik

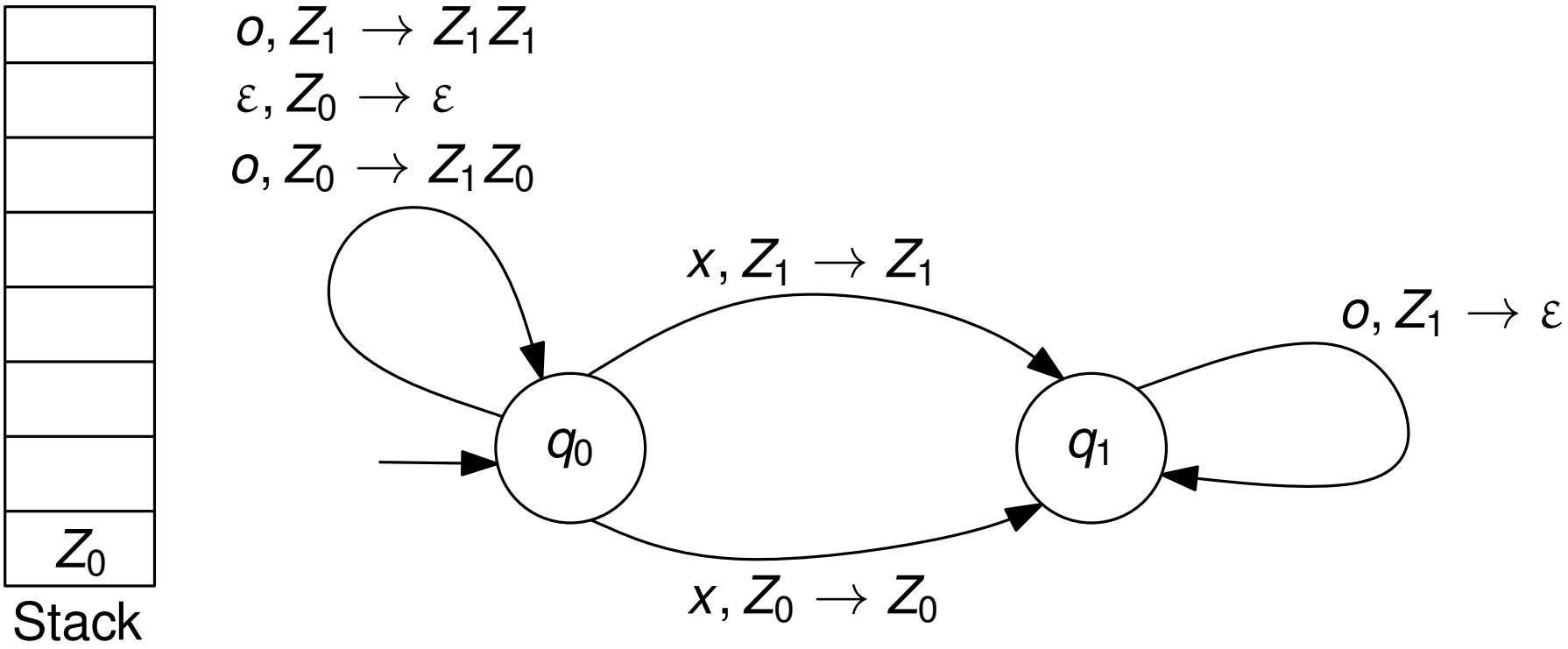
Kellerautomaten



Ein (nichtdet.) *Kellerautomat* $(Q, \Sigma, \Gamma, q_0, Z_0, \delta, F)$ besteht aus

Institut für Theoretische Informatik
Lehrstuhl Algorithmik

Kellerautomaten

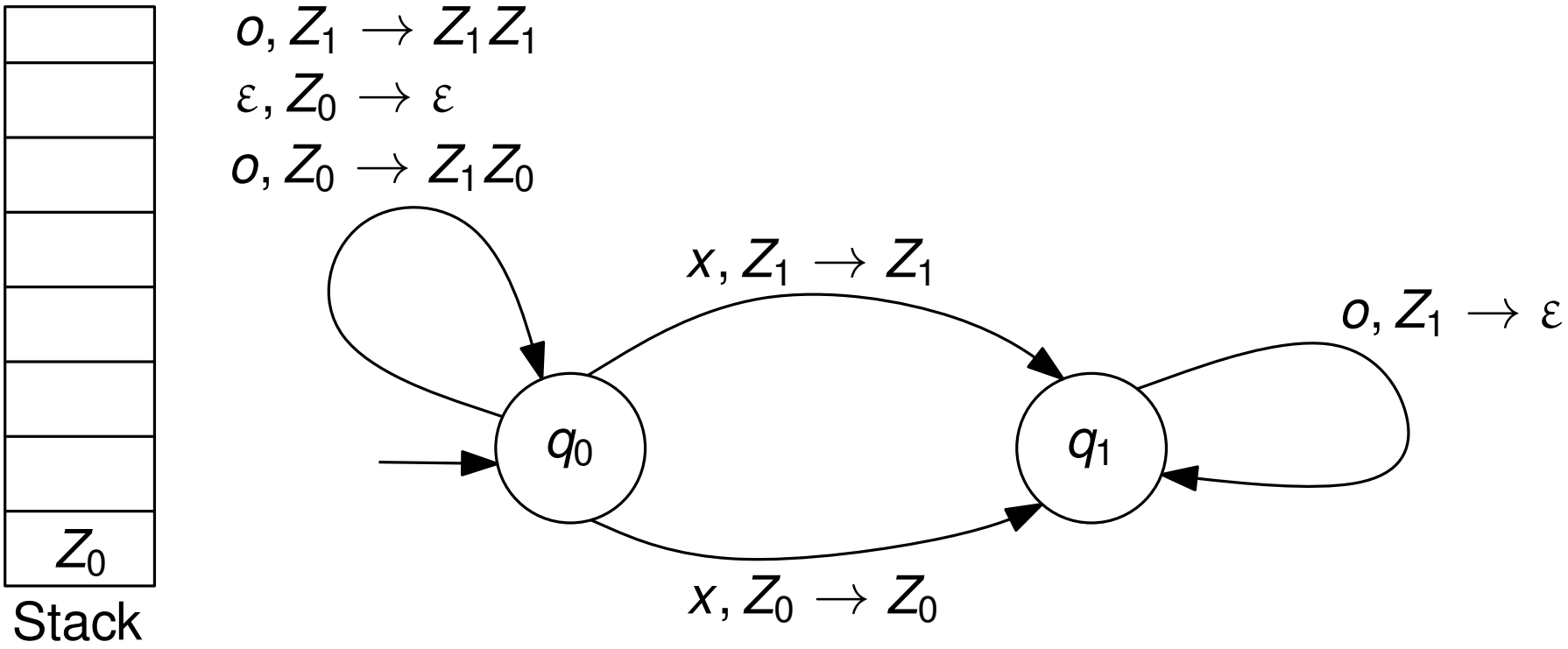


Ein (nichtdet.) *Kellerautomat* $(Q, \Sigma, \Gamma, q_0, Z_0, \delta, F)$ besteht aus

- Q endliche Zustandsmenge und Anfangszustand $q_0 \in Q$
- F Menge akzeptierender Zustände
- Σ endliches Eingabealphabet
- Γ endliches Stackalphabet und Stackinitialisierung $Z_0 \in \Gamma$

Institut für Theoretische Informatik
Lehrstuhl Algorithmik

Kellerautomaten

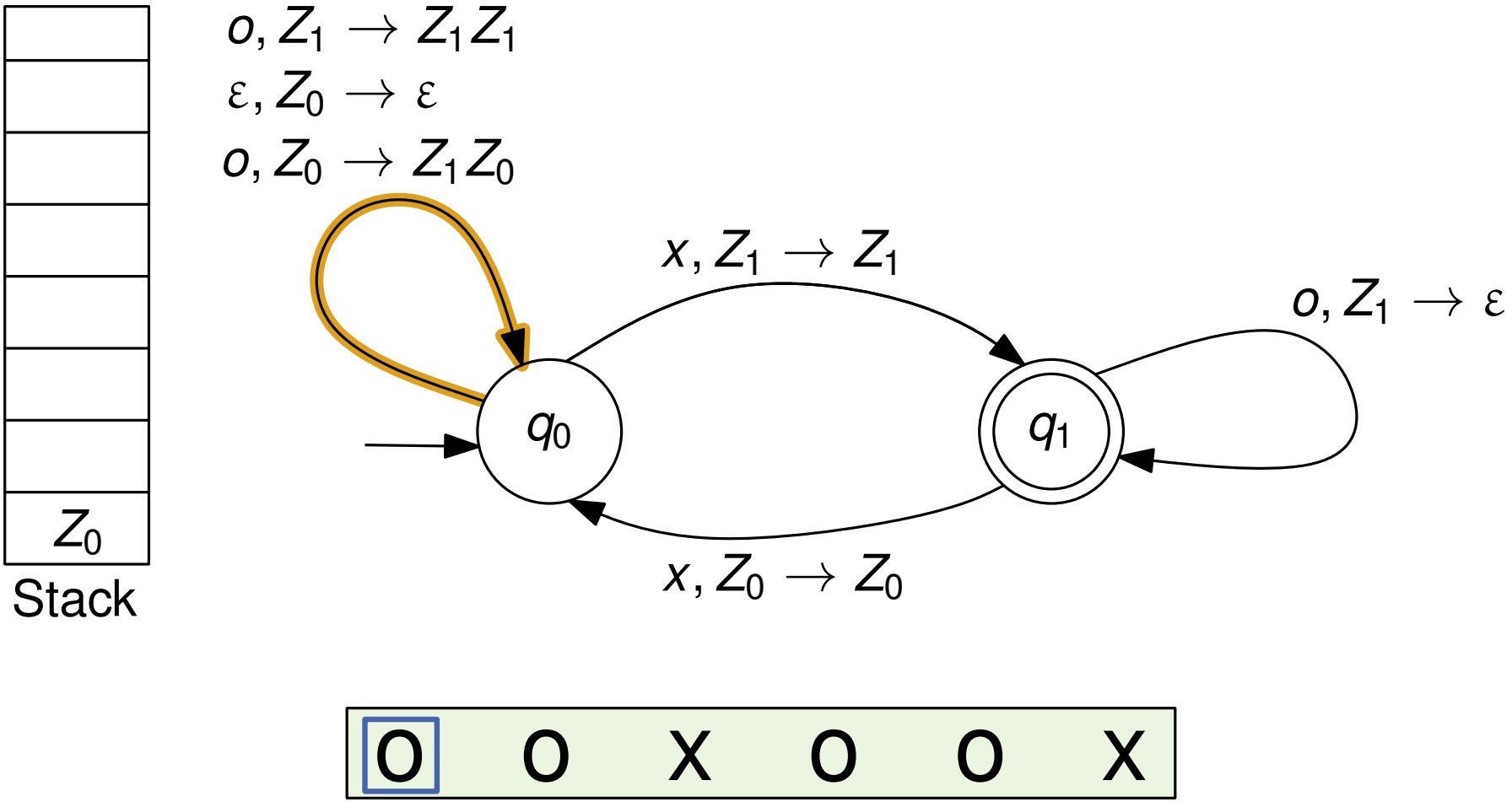


Ein (nichtdet.) *Kellerautomat* $(Q, \Sigma, \Gamma, q_0, Z_0, \delta, F)$ besteht aus

- $\delta : Q \times (\Sigma \cup \{\varepsilon\}) \times \Gamma \to 2^{Q \times \Gamma^\star}$
  - $\delta(q, a, Z) \subseteq \{(q, \gamma) : q \in Q, \gamma \in \Gamma^\star\}$
  - $\delta(q, \varepsilon, Z) \subseteq \{(q, \gamma) : q \in Q, \gamma \in \Gamma^\star\}$

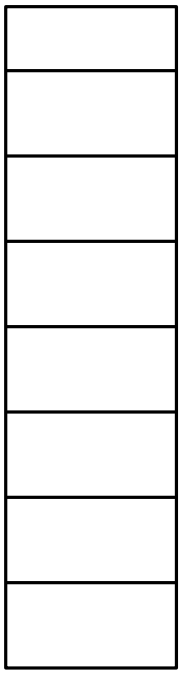
Institut für Theoretische Informatik
Lehrstuhl Algorithmik

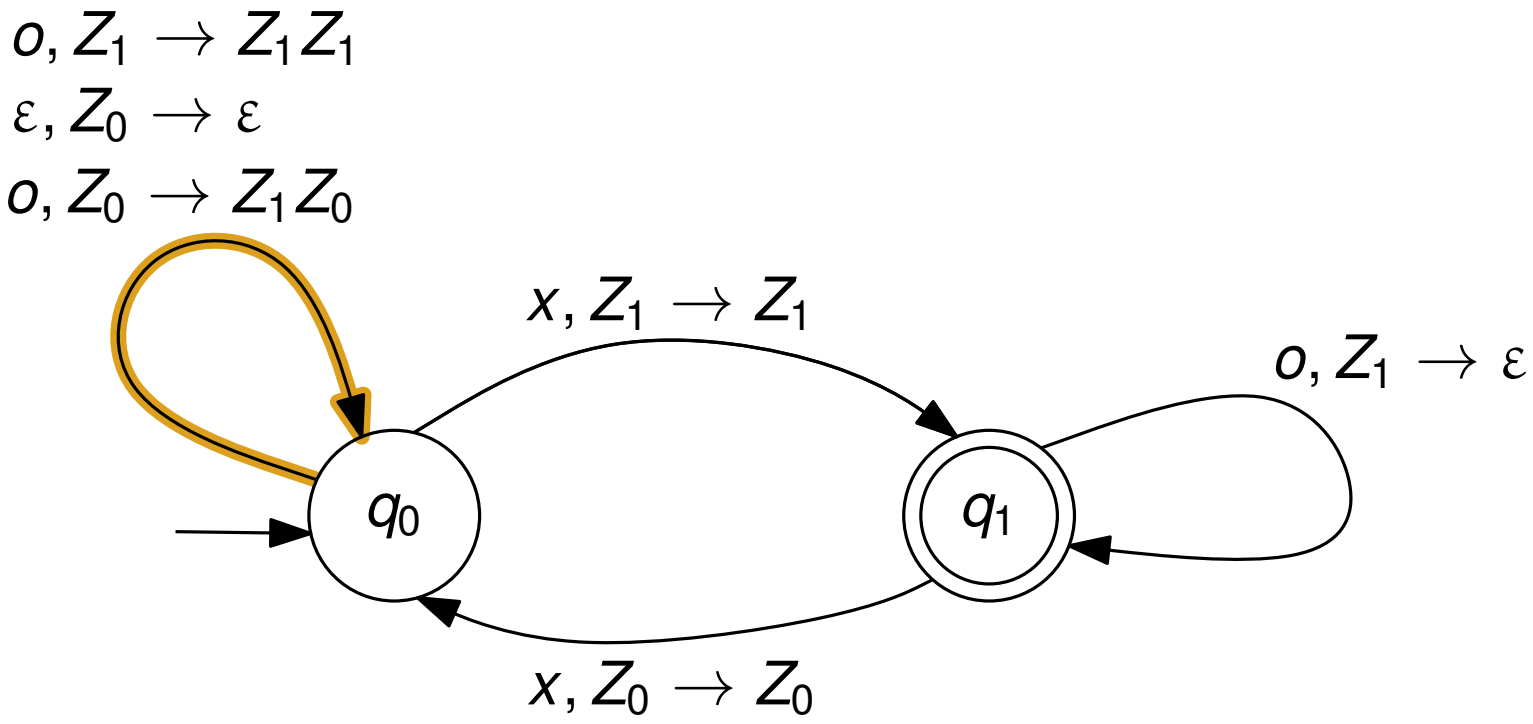
Abarbeitung eines Wortes



Institut für Theoretische Informatik
Lehrstuhl Algorithmik

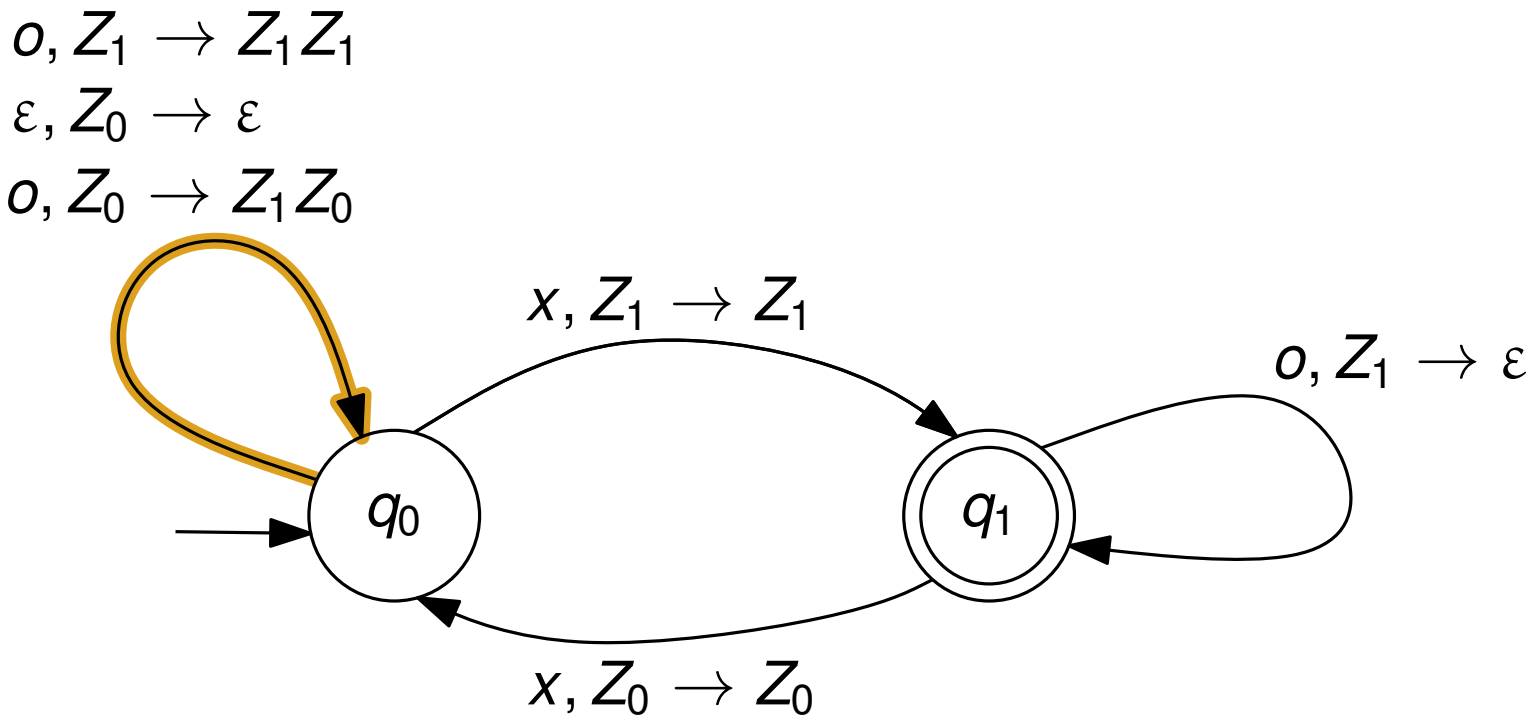
Abarbeitung eines Wortes





Institut für Theoretische Informatik
Lehrstuhl Algorithmik

Abarbeitung eines Wortes



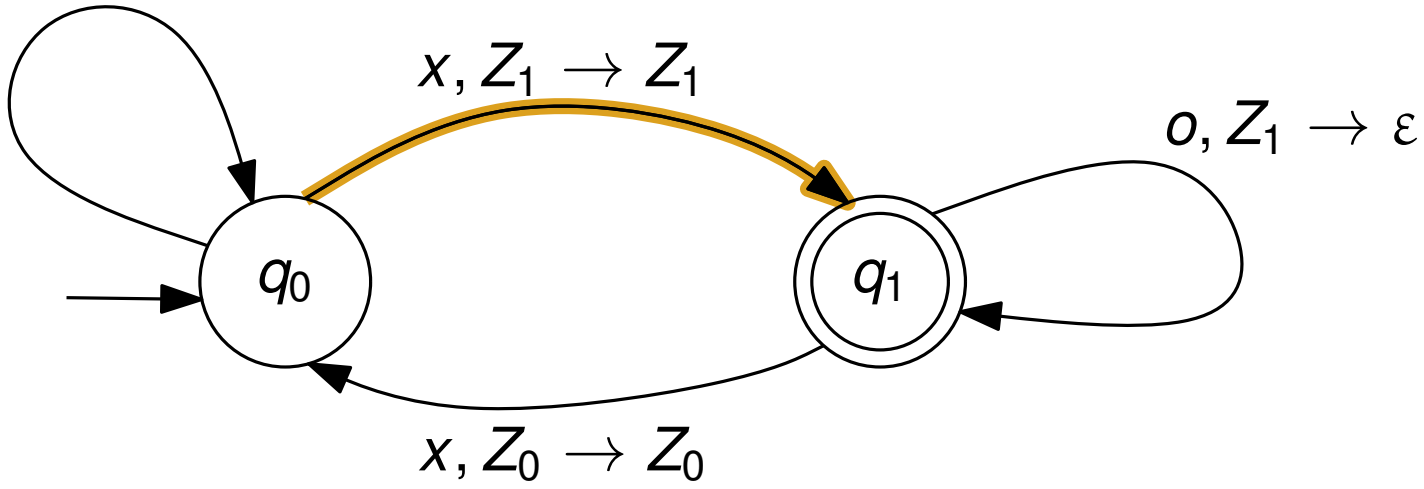
Institut für Theoretische Informatik
Lehrstuhl Algorithmik

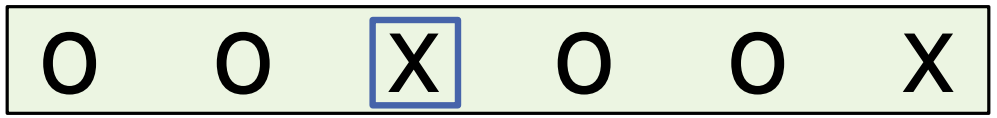
Abarbeitung eines Wortes

$o, Z_1 \to Z_1 Z_1$

$\varepsilon, Z_0 \to \varepsilon$

$o, Z_0 \to Z_1 Z_0$





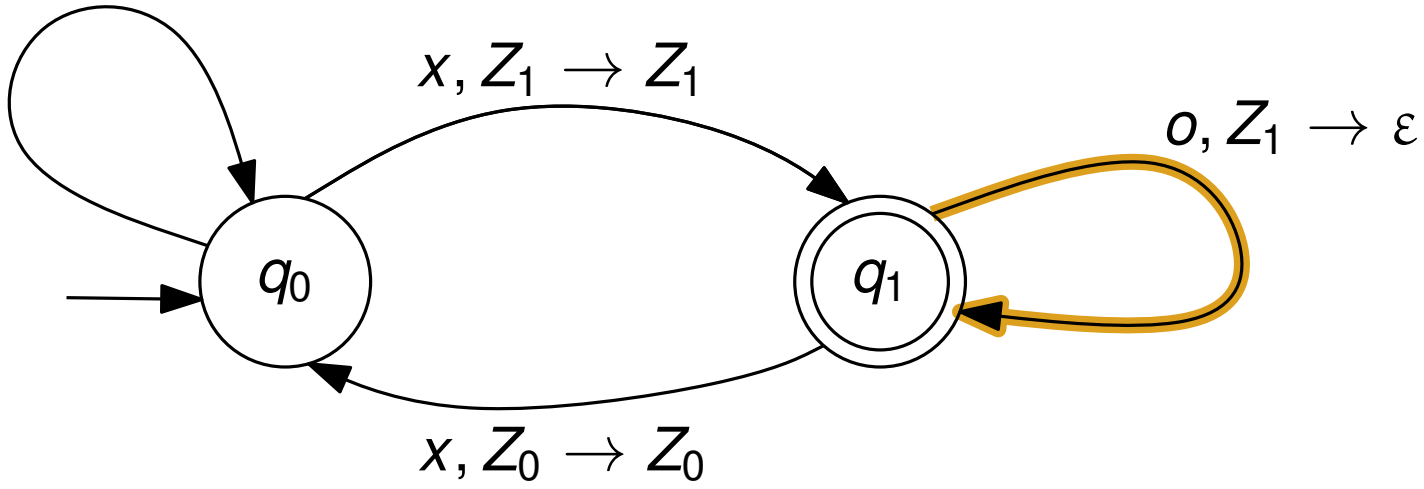
Institut für Theoretische Informatik
Lehrstuhl Algorithmik

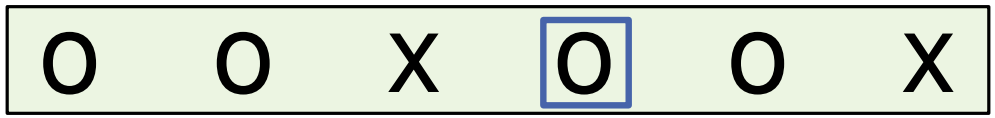
Abarbeitung eines Wortes

$o, Z_1 \to Z_1 Z_1$

$\varepsilon, Z_0 \to \varepsilon$

$o, Z_0 \to Z_1 Z_0$





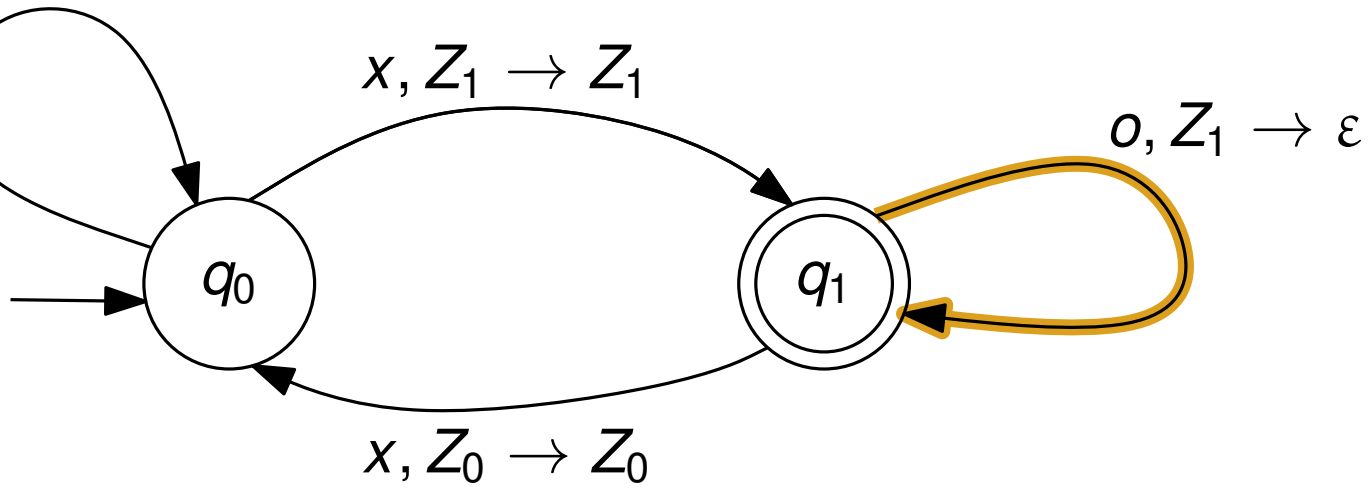
Institut für Theoretische Informatik
Lehrstuhl Algorithmik

Abarbeitung eines Wortes

$o, Z_1 \to Z_1 Z_1$

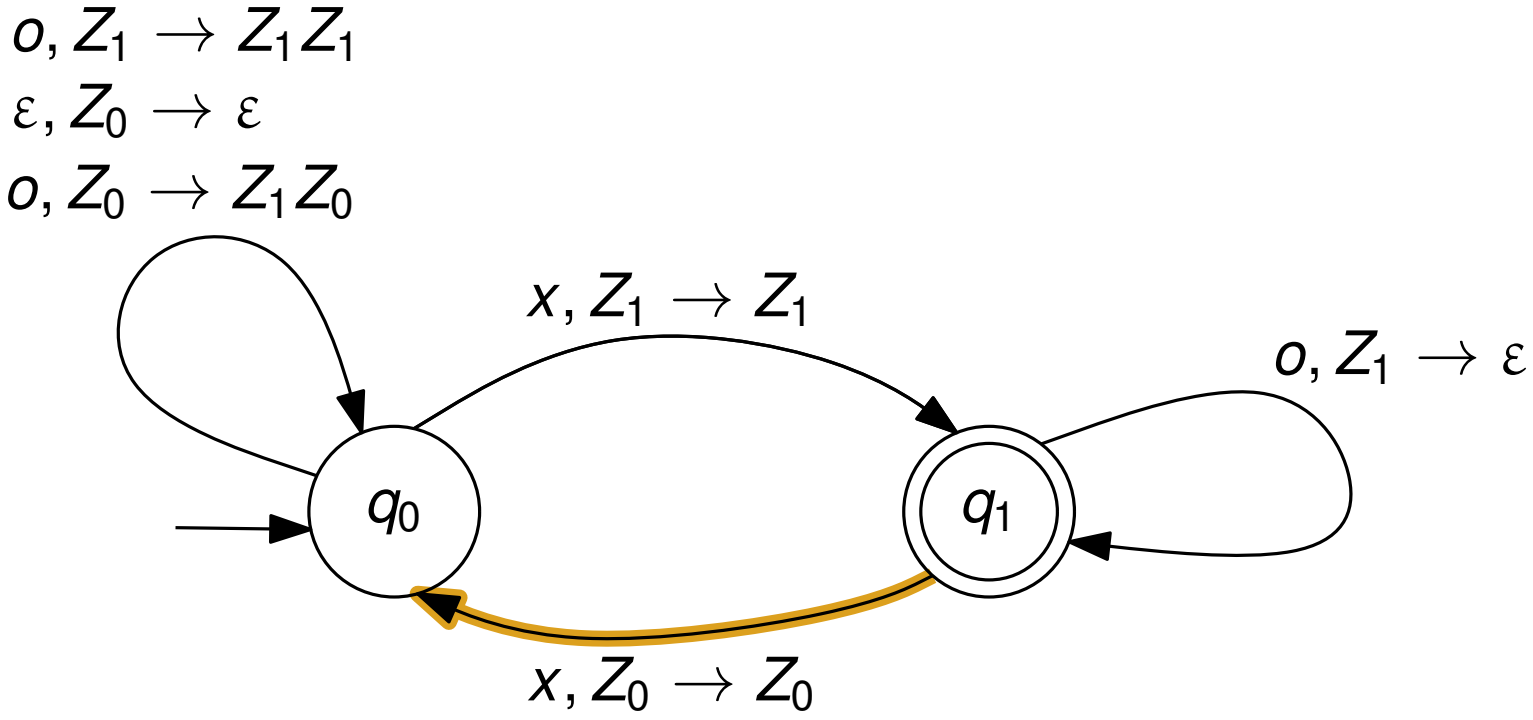
$\varepsilon, Z_0 \to \varepsilon$

$o, Z_0 \to Z_1 Z_0$

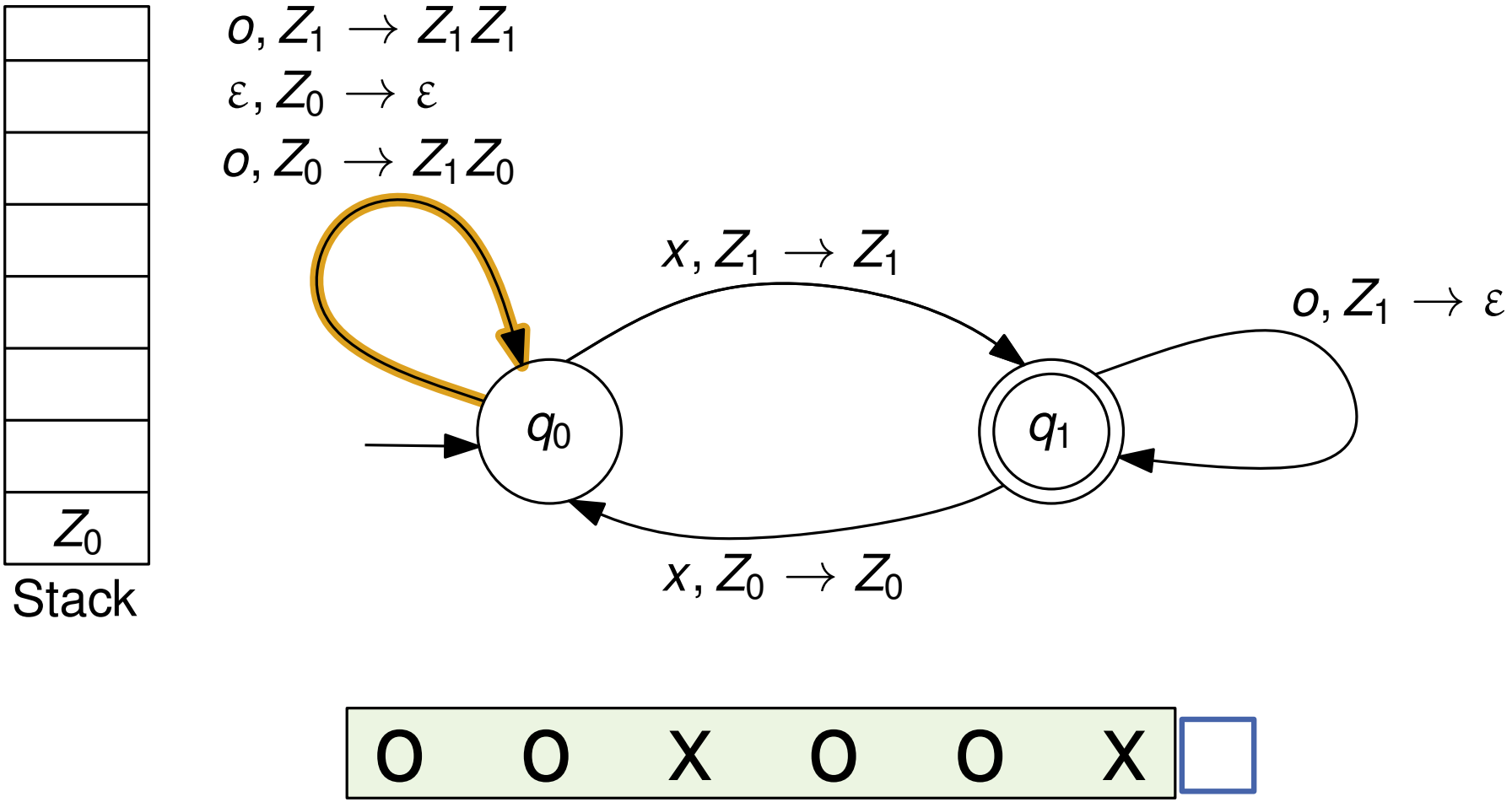


Institut für Theoretische Informatik
Lehrstuhl Algorithmik

Abarbeitung eines Wortes



Abarbeitung eines Wortes



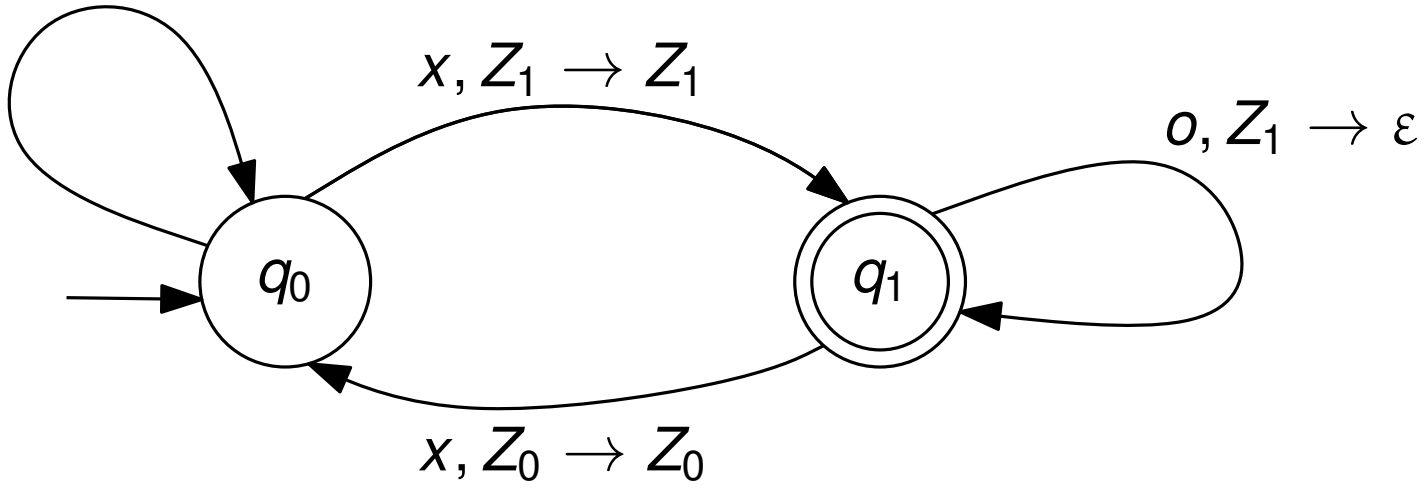
Institut für Theoretische Informatik
Lehrstuhl Algorithmik

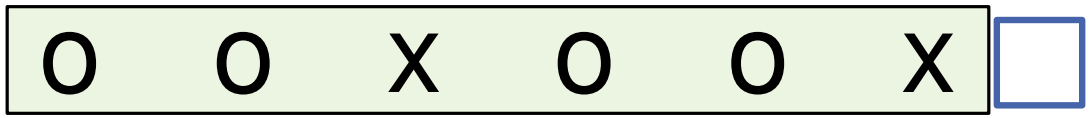
Abarbeitung eines Wortes

$o, Z_1 \to Z_1 Z_1$

$\varepsilon, Z_0 \to \varepsilon$

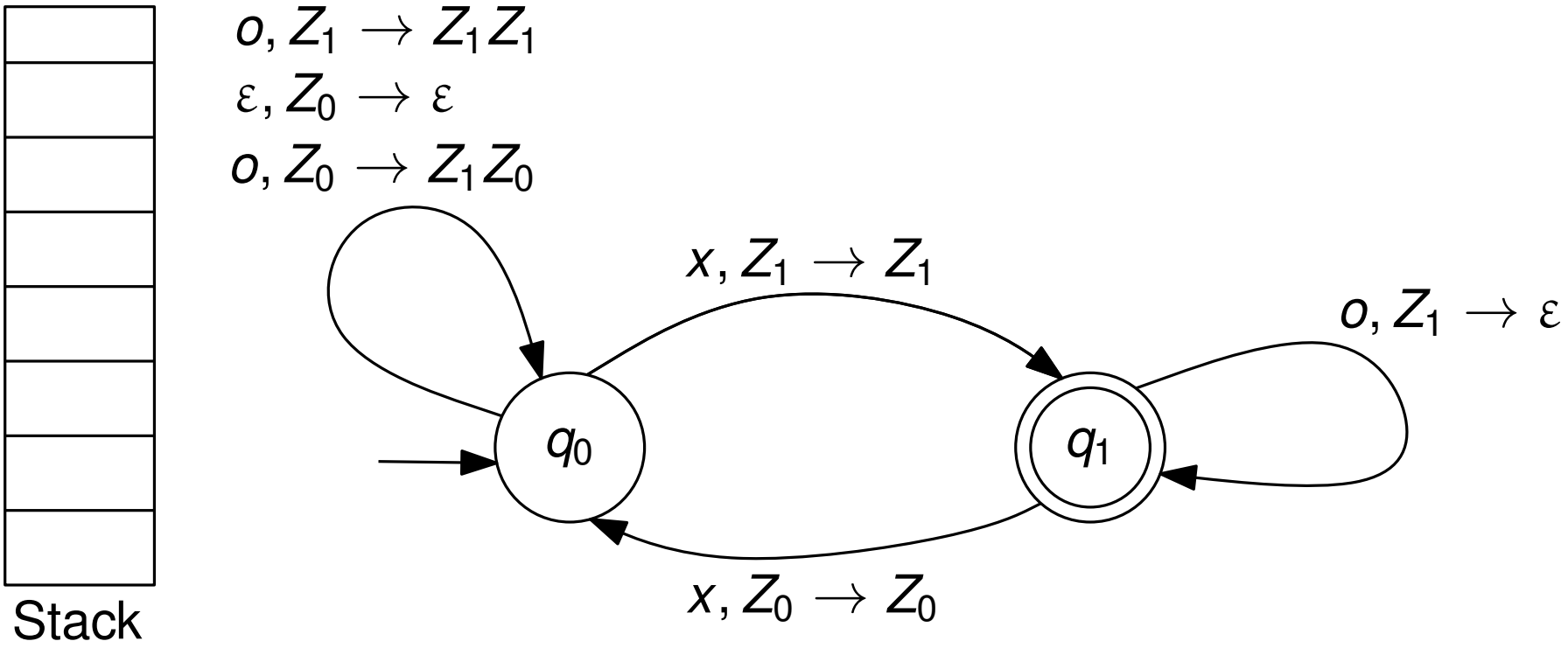
$o, Z_0 \to Z_1 Z_0$





Institut für Theoretische Informatik
Lehrstuhl Algorithmik

Determinismus



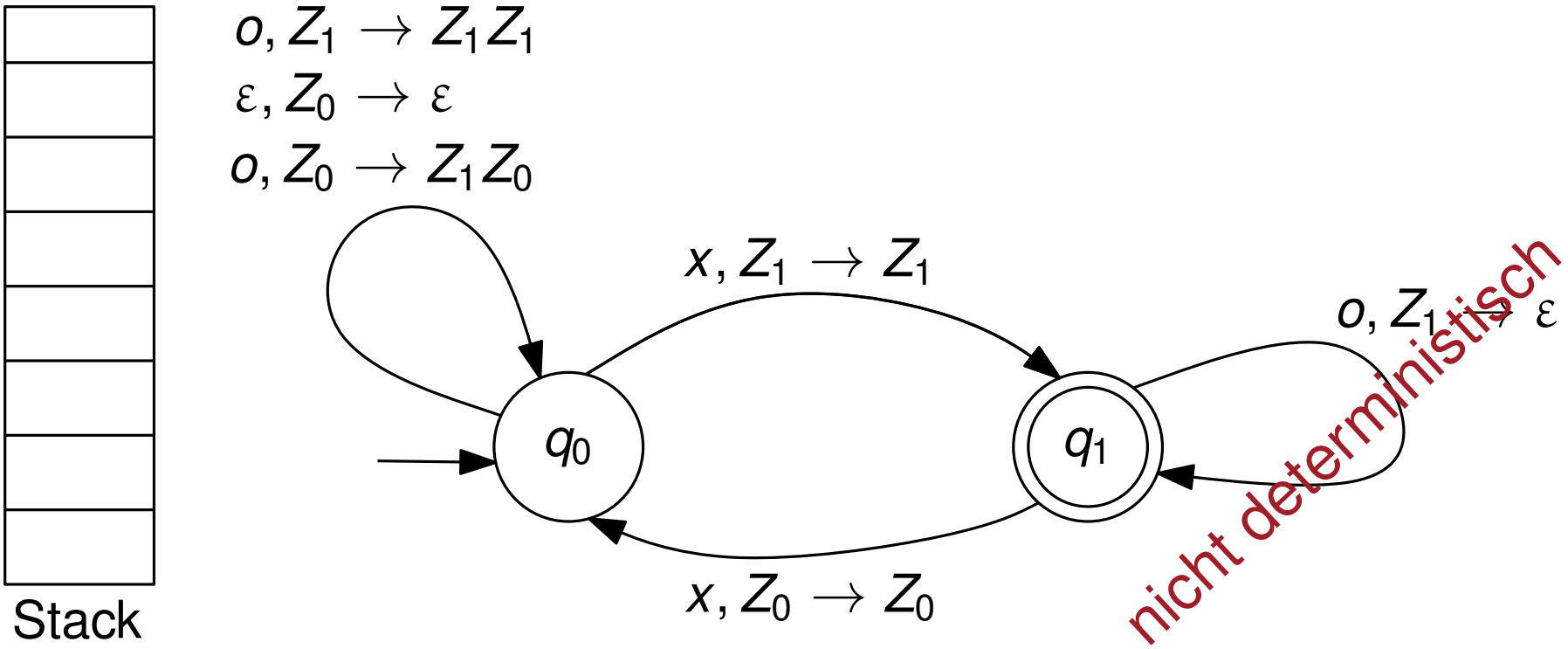
Ein Kellerautomat ist *deterministisch*, falls

$$|\delta(q, a, Z)| = 1 \quad \text{und} \quad \delta(q, \varepsilon, Z) = \emptyset$$

für alle $q \in Q, a \in \Sigma, Z \in \Gamma$ gilt.

Institut für Theoretische Informatik
Lehrstuhl Algorithmik

Determinismus



Ein Kellerautomat ist *deterministisch*, falls

$$\left|\delta(q, a, Z)\right| = 1 \quad \text{und} \quad \delta(q, \varepsilon, Z) = \emptyset$$

für alle $q \in Q, a \in \Sigma, Z \in \Gamma$ gilt.

Institut für Theoretische Informatik
Lehrstuhl Algorithmik

Akzeptanz eines Wortes

Sei $w \in \Sigma^\star$ das Eingabewort.

Akzeptanz durch leeren Stack

Es gibt zulässige Folge von Konfigurationen aus der Anfangskonfiguration (q_0, w, Z_0) in eine Konfiguration $(q, \varepsilon, \varepsilon)$, $q \in Q$,.

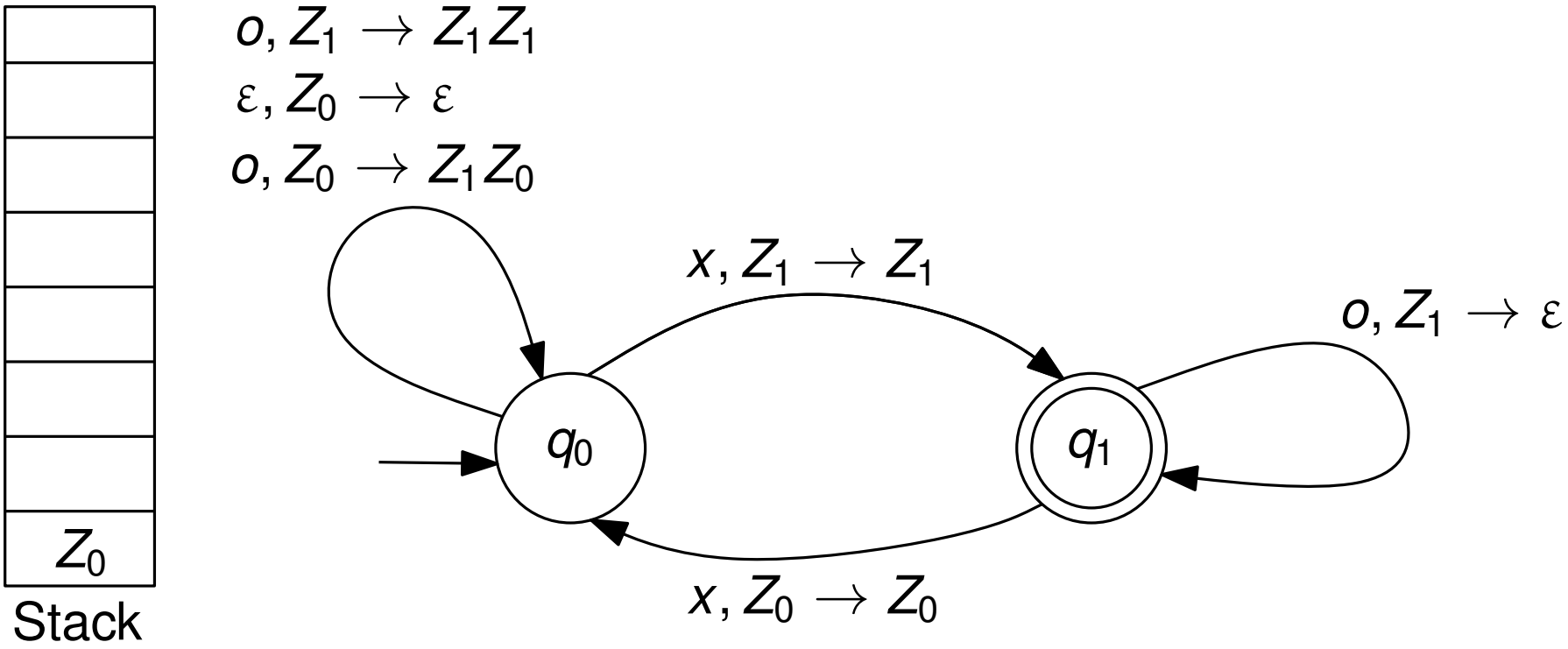
Akzeptanz durch Endzustände

Es gibt zulässige Folge von Konfigurationen aus der Anfangskonfiguration (q_0, w, Z_0) in eine Konfiguration (q, ε, γ) mit $q \in F$ und $\gamma \in \Gamma^*$ gibt.

Konfiguration (q, w, α)**:**

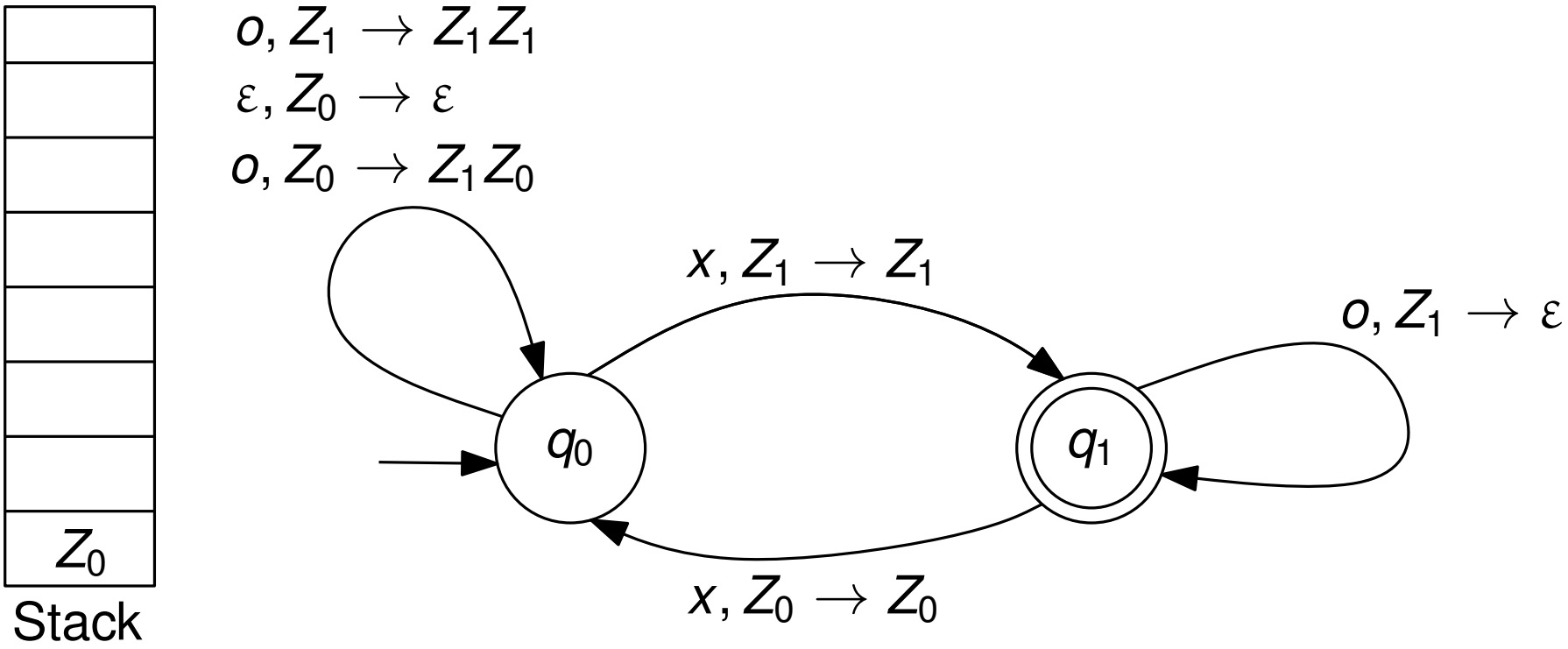
- q= Zustand,
- w= der Teil der Eingabe, der noch nicht gelesen wurde,
- α= Stackinhalt.

Akzeptanz eines Wortes



Institut für Theoretische Informatik
Lehrstuhl Algorithmik

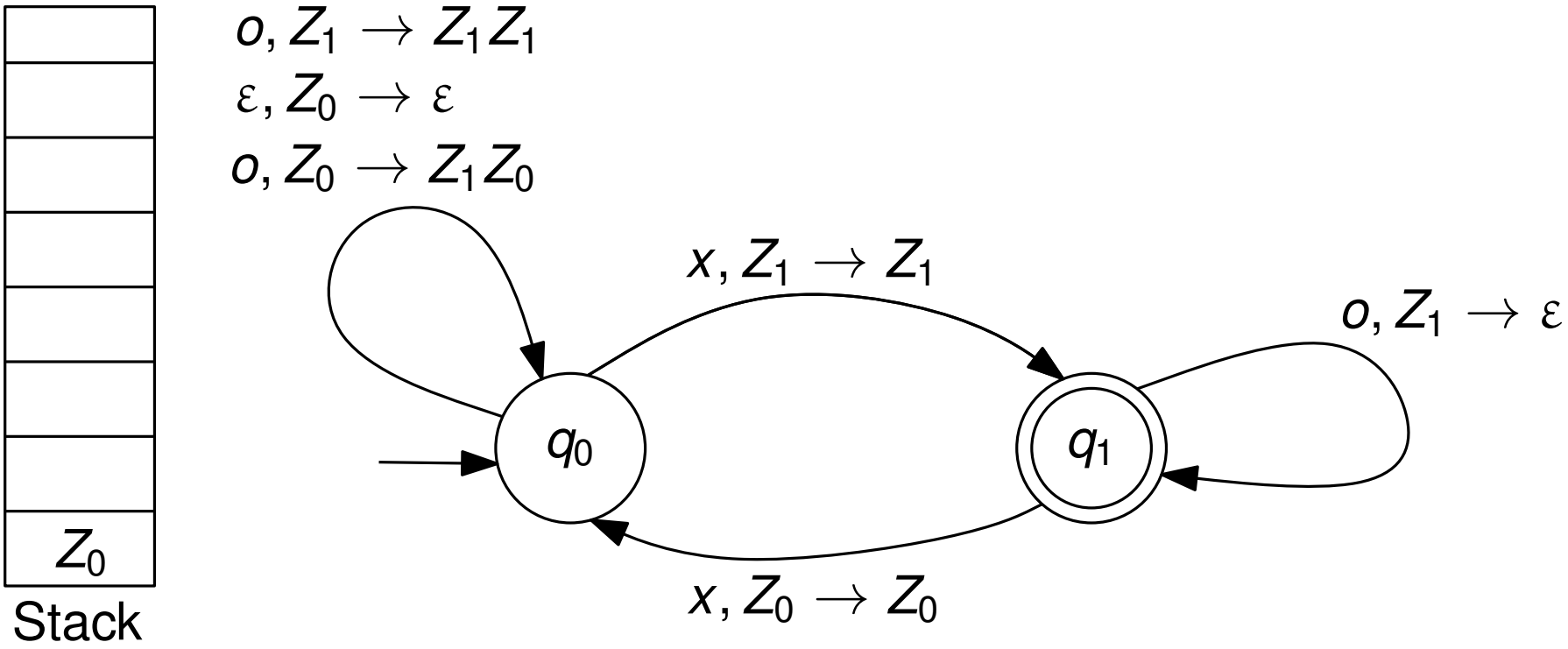
Akzeptanz eines Wortes



Gesucht: Sprache L_F, die $\mathcal{A}$ mit Endzuständen akzeptiert.

Institut für Theoretische Informatik
Lehrstuhl Algorithmik

Akzeptanz eines Wortes

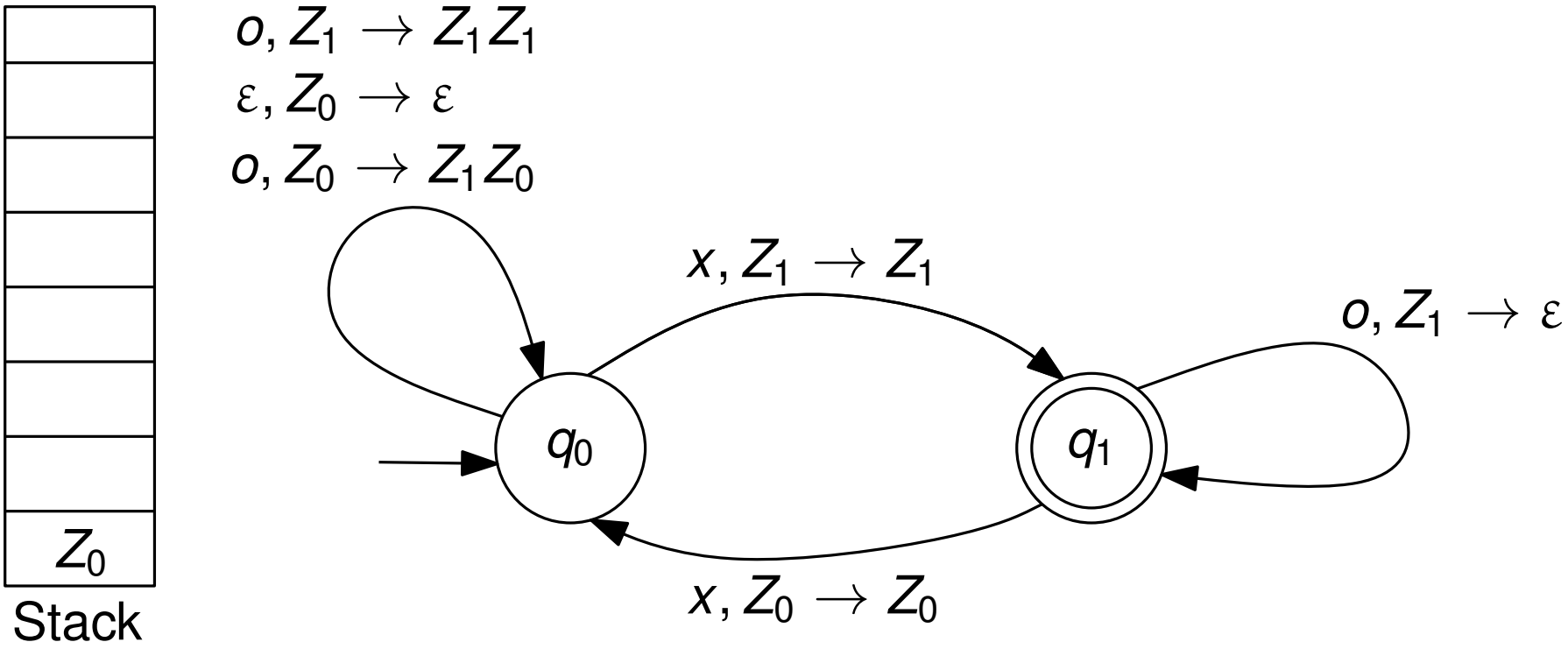


Gesucht: Sprache L_F, die $\mathcal{A}$ mit Endzuständen akzeptiert.

$$L_F = \left\{ o^{i_1} x o^{i_1} x o^{i_2} x o^{i_2} x \ldots o^{i_k} x o^{i_k} x o^{i} x o^{j} \mid i_1, i_2, \ldots, i_k, i \geq 1, i \geq j \right\}$$

Institut für Theoretische Informatik
Lehrstuhl Algorithmik

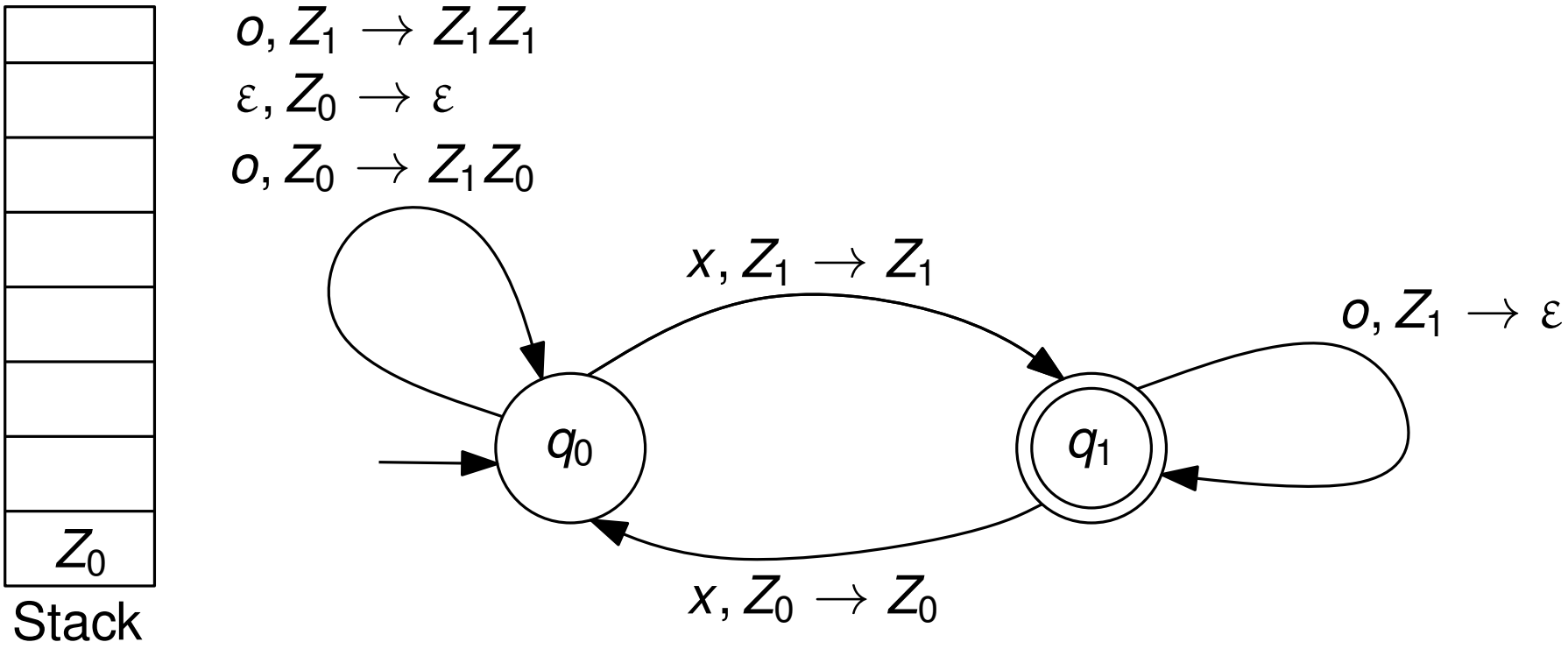
Akzeptanz eines Wortes



Gesucht: Sprache L_ε, die $\mathcal{A}$ durch leeren Stack akzeptiert.

Institut für Theoretische Informatik
Lehrstuhl Algorithmik

Akzeptanz eines Wortes



Gesucht: Sprache L_ε, die $\mathcal{A}$ durch leeren Stack akzeptiert.

Sei $L = \{o^i x o^i x \mid i > 0\}$, dann ist $L_\varepsilon = L^\star$

Institut für Theoretische Informatik
Lehrstuhl Algorithmik

Grammatik von L_ε

Sei $L = \{o^i x o^i x \mid i > 0\}$, dann ist $L_\varepsilon = L^\star$

Die kontextfreie Grammatik $G = (\{o, x\}, \{S, B\}, S, R)$ für L_ε mit

$$R = \{S \to \varepsilon \mid SS \mid oBox,$$
$$B \to oBo \mid x\}$$

Institut für Theoretische Informatik
Lehrstuhl Algorithmik

Greibach-Normalform

Eine kontextfreie Grammatik ist in *Greibach-Normalform*, wenn alle Ableitungsregeln von der Form

$$A \to a\alpha \text{ mit } A \in V, a \in \Sigma \text{ und } \alpha \in V^*$$

sind.

Für jede kontextfreie Grammatik G, für die $L(G)$ das leere Wort nicht enthält, kann eine (äquivalente) kontextfreie Grammatik G' mit $L(G) = L(G')$ in Greibach-Normalform konstruiert werden.

Greibach-Normalform

Ersetzung (i). Eine Regel

$$A \to \alpha_1 B \alpha_2$$

wobei

$$B \to \beta_1, \quad B \to \beta_2, \quad \ldots, \quad B \to \beta_r$$

alle Regeln sind, deren linke Seite B ist, kann durch die Regeln

$$A \to \alpha_1 \beta_1 \alpha_2$$
$$A \to \alpha_1 \beta_2 \alpha_2$$
$$\ldots$$
$$A \to \alpha_1 \beta_r \alpha_2$$

ersetzt werden.

Institut für Theoretische Informatik
Lehrstuhl Algorithmik

Greibach-Normalform

Ersetzung (ii). Seien

$$\begin{aligned} A &\to A\alpha_1, \ldots, A \to A\alpha_r \\ A &\to \beta_1, \ldots, A \to \beta_s \end{aligned}$$

alle Regeln, deren linke Seite A ist, wobei β_i nicht mit A beginnen. Dann können die Regeln

$$A \to A\alpha_1, \ldots, A \to A\alpha_r$$

durch die Regeln

$$\begin{aligned} A &\to \beta_1 B, \ldots, A \to \beta_s B \\ B &\to \alpha_1, \ldots, B \to \alpha_r, \\ B &\to \alpha_1 B, \ldots, B \to \alpha_r B \end{aligned}$$

ersetzt werden. Dabei sei B eine neu eingeführte Variable.

Verfahren

Annahme: G befindet sich in Chomsky-Normalform

1. Schritt: Falls $A_i \to A_j\alpha$ Regel ist, so gilt $j > i$.

2. Schritt: Bringe Regeln mit linker Seite in V in Form

$$A_k \to a\alpha, a \in \Sigma, \alpha \in (V')^*$$

3. Schritt: Ersetze Regeln $B_i \to A_j\alpha$, $\alpha \in (\Sigma \cup V')^*$ mit Ersetzung (i).

Institut für Theoretische Informatik
Lehrstuhl Algorithmik

Konstruktion von Greibach-Normalform

$G = (\Sigma, V, S, R)$ mit $\Sigma = \{a, b, c\}$ und $V = \{S, B, C\}$

$$
\begin{aligned}
S &\rightarrow BC \mid a \\
B &\rightarrow SC \mid c \\
C &\rightarrow BS \mid b
\end{aligned}
$$

Institut für Theoretische Informatik
Lehrstuhl Algorithmik

Konstruktion von Greibach-Normalform

$G = (\Sigma, V, S, R)$ mit $\Sigma = \{a, b, c\}$ und $V = \{S, B, C\}$

$$
\begin{aligned}
S &\rightarrow BC \mid a \\
B &\rightarrow SC \mid c \\
C &\rightarrow BS \mid b
\end{aligned}
$$

0. Schritt: Nichtterminale umbenennen.

$$
\begin{aligned}
A_1 &\rightarrow A_2 A_3 \mid a \\
A_2 &\rightarrow A_1 A_3 \mid c \\
A_3 &\rightarrow A_2 A_1 \mid b
\end{aligned}
$$

Institut für Theoretische Informatik
Lehrstuhl Algorithmik

Konstruktion von Greibach-Normalform

1. Schritt: Falls $A_i \to A_j \alpha$ Regel ist, so gilt $j > i$.

$$
\begin{aligned}
A_1 &\to A_2 A_3 \mid a \\
A_2 &\to A_1 A_3 \mid c \\
A_3 &\to A_2 A_1 \mid b
\end{aligned}
$$

Institut für Theoretische Informatik
Lehrstuhl Algorithmik

Konstruktion von Greibach-Normalform

1. Schritt: Falls $A_i \to A_j \alpha$ Regel ist, so gilt $j > i$.

$$
\begin{aligned}
A_1 &\to A_2 A_3 \mid a \\
A_2 &\to A_1 A_3 \mid c \\
A_3 &\to A_2 A_1 \mid b
\end{aligned}
$$

Ersetzung (i) auf A_2 →

$$
\begin{aligned}
A_1 &\to A_2 A_3 \mid a \\
A_2 &\to A_2 A_3 A_3 \mid a A_3 \mid c \\
A_3 &\to A_2 A_1 \mid b
\end{aligned}
$$

Institut für Theoretische Informatik
Lehrstuhl Algorithmik

Konstruktion von Greibach-Normalform

1. Schritt: Falls $A_i \to A_j\alpha$ Regel ist, so gilt $j > i$.

$$
\begin{aligned}
A_1 &\to A_2A_3 \mid a \\
A_2 &\to A_1A_3 \mid c \\
A_3 &\to A_2A_1 \mid b
\end{aligned}
$$

Ersetzung (i) auf A_2

$$
\begin{aligned}
A_1 &\to A_2A_3 \mid a \\
A_2 &\to A_2A_3A_3 \mid aA_3 \mid c \\
A_3 &\to A_2A_1 \mid b
\end{aligned}
$$

Ersetzung (ii) auf A_2

$$
\begin{aligned}
A_1 &\to A_2A_3 \mid a \\
A_2 &\to aA_3 \mid c \mid aA_3B_1 \mid cB_1 \\
A_3 &\to A_2A_1 \mid b \\
B_1 &\to A_3A_3 \mid A_3A_3B_1
\end{aligned}
$$

Institut für Theoretische Informatik
Lehrstuhl Algorithmik

Konstruktion von Greibach-Normalform

1. Schritt: Falls $A_i \to A_j\alpha$ Regel ist, so gilt $j > i$.

$$
\begin{array}{lcl}
A_1 & \to & A_2A_3 \mid a \\
A_2 & \to & A_1A_3 \mid c \\
A_3 & \to & A_2A_1 \mid b
\end{array}
$$

Ersetzung (i) auf A_2

$$
\begin{array}{lcl}
A_1 & \to & A_2A_3 \mid a \\
A_2 & \to & A_2A_3A_3 \mid aA_3 \mid c \\
A_3 & \to & A_2A_1 \mid b
\end{array}
$$

Ersetzung (ii) auf A_2

$$
\begin{array}{lcl}
A_1 & \to & A_2A_3 \mid a \\
A_2 & \to & aA_3 \mid c \mid aA_3B_1 \mid cB_1 \\
A_3 & \to & A_2A_1 \mid b \\
B_1 & \to & A_3A_3 \mid A_3A_3B_1
\end{array}
$$

Ersetzung (ii) auf A_3

$$
\begin{array}{lcl}
A_1 & \to & A_2A_3 \mid a \\
A_2 & \to & aA_3 \mid c \mid aA_3B_1 \mid cB_1 \\
A_3 & \to & aA_3A_1 \mid cA_1 \mid aA_3B_1A_1 \mid cB_1A_1 \mid b \\
B_1 & \to & A_3A_3 \mid A_3A_3B_1
\end{array}
$$

Institut für Theoretische Informatik
Lehrstuhl Algorithmik

Konstruktion von Greibach-Normalform

2. Schritt: Bringe Regeln mit linker Seite in V in Form

$$A_k \to a\alpha, a \in \Sigma, \alpha \in (V')^\star$$

$$
\begin{array}{lcl}
A_1 & \to & A_2A_3 \mid a \\
A_2 & \to & aA_3 \mid c \mid aA_3B_1 \mid cB_1 \\
A_3 & \to & aA_3A_1 \mid cA_1 \mid aA_3B_1A_1 \mid cB_1A_1 \mid b \\
B_1 & \to & A_3A_3 \mid A_3A_3B_1
\end{array}
$$

Institut für Theoretische Informatik
Lehrstuhl Algorithmik

Konstruktion von Greibach-Normalform

2. Schritt: Bringe Regeln mit linker Seite in V in Form

$$A_k \to a\alpha, a \in \Sigma, \alpha \in (V')^\star$$

$$\begin{aligned}
A_1 &\to A_2A_3 \mid a \\
A_2 &\to aA_3 \mid c \mid aA_3B_1 \mid cB_1 \\
A_3 &\to aA_3A_1 \mid cA_1 \mid aA_3B_1A_1 \mid cB_1A_1 \mid b \\
B_1 &\to A_3A_3 \mid A_3A_3B_1
\end{aligned}$$

Ersetzung (i) auf A_2 →

$$\begin{aligned}
A_1 &\to aA_3A_3 \mid cA_3 \mid aA_3B_1A_3 \mid cB_1A_3 \mid a \\
A_2 &\to aA_3 \mid c \mid aA_3B_1 \mid cB_1 \\
A_3 &\to aA_3A_1 \mid cA_1 \mid aA_3B_1A_1 \mid cB_1A_1 \mid b \\
B_1 &\to A_3A_3 \mid A_3A_3B_1
\end{aligned}$$

Institut für Theoretische Informatik
Lehrstuhl Algorithmik

Konstruktion von Greibach-Normalform

3. Schritt: Ersetze Regeln $B_i \to A_j\alpha$, $\alpha \in (\Sigma \cup V')^\star$ mit Ersetzung (i).

$$
\begin{aligned}
A_1 &\to aA_3A_3 \mid cA_3 \mid aA_3B_1A_3 \mid cB_1A_3 \mid a \\
A_2 &\to aA_3 \mid c \mid aA_3B_1 \mid cB_1 \\
A_3 &\to aA_3A_1 \mid cA_1 \mid aA_3B_1A_1 \mid cB_1A_1 \mid b \\
B_1 &\to A_3A_3 \mid A_3A_3B_1
\end{aligned}
$$

Ersetzung (i) auf A_3 →

Konstruktion von Greibach-Normalform

3. Schritt: Ersetze Regeln $B_i \to A_j\alpha$, $\alpha \in (\Sigma \cup V')^\star$ mit Ersetzung (i).

$$
\begin{aligned}
A_1 &\to aA_3A_3 \mid cA_3 \mid aA_3B_1A_3 \mid cB_1A_3 \mid a \\
A_2 &\to aA_3 \mid c \mid aA_3B_1 \mid cB_1 \\
A_3 &\to aA_3A_1 \mid cA_1 \mid aA_3B_1A_1 \mid cB_1A_1 \mid b \\
B_1 &\to A_3A_3 \mid A_3A_3B_1
\end{aligned}
$$

Ersetzung (i) auf A_3

$$
\begin{aligned}
A_1 &\to aA_3A_3 \mid cA_3 \mid aA_3B_1A_3 \mid cB_1A_3 \mid a \\
A_2 &\to aA_3 \mid c \mid aA_3B_1 \mid cB_1 \\
A_3 &\to aA_3A_1 \mid cA_1 \mid aA_3B_1A_1 \mid cB_1A_1 \mid b \\
B_1 &\to aA_3A_1A_3 \mid cA_1A_3 \mid aA_3B_1A_1A_3 \mid cB_1A_1A_3 \mid \\
&\quad\ bA_3 \mid aA_3A_1A_3B_1 \mid cA_1A_3B_1 \mid aA_3B_1A_1A_3B_1 \mid cB_1A_1A_3B_1 \mid bA_3B_1
\end{aligned}
$$

Institut für Theoretische Informatik
Lehrstuhl Algorithmik

Greibach-Normalform und Kellerautomaten

Sei $G = (\Sigma, V, S, R)$ eine Grammatik in Greibach-Normalform.

Kellerautomat $\mathcal{A} = (Q, \Sigma, \Gamma, \delta, q_0, Z_0)$

$$
\begin{aligned}
Q &:= \{q_0\} \\
\Gamma &:= V \\
Z_0 &:= S \\
\delta(q_0, a, A) &:= \{(q_0, \alpha) \mid (A \to a\alpha) \in R\}
\end{aligned}
$$

Institut für Theoretische Informatik
Lehrstuhl Algorithmik

Greibach-Normalform und Kellerautomaten

$$\delta(q_0, a, A) := \{(q_0, \alpha) | (A \to a\alpha) \in R\}$$

$$\begin{aligned}
A_1 &\to aA_3A_3 \mid cA_3 \mid aA_3B_1A_3 \mid cB_1A_3 \mid a \\
A_2 &\to aA_3 \mid c \mid aA_3B_1 \mid cB_1 \\
A_3 &\to aA_3A_1 \mid cA_1 \mid aA_3B_1A_1 \mid cB_1A_1 \mid b \\
B_1 &\to aA_3A_1A_3 \mid cA_1A_3 \mid aA_3B_1A_1A_3 \mid cB_1A_1A_3 \mid \\
&\quad bA_3 \mid aA_3A_1A_3B_1 \mid cA_1A_3B_1 \mid aA_3B_1A_1A_3B_1 \mid cB_1A_1A_3B_1 \mid bA_3B_1
\end{aligned}$$

Für A_1

$$\delta(q_0, a, A_1) = \{(q_0, A_3A_3), (q_0, A_3B_1A_3), (q_0, \varepsilon)\}$$

$$\delta(q_0, b, A_1) = \{\}$$

$$\delta(q_0, c, A_1) = \{(q_0, A_3), (q_0, B_1A_3)\}$$

Analog für A_1, A_2, A_3, B_1.

Umgekehrte Polnische Notation

UPN: Schreibweise von arithmetischen Ausdrücken

$$\left(\frac{5}{7-(1+1)} \times 3\right) - (2+(1+1))$$

wird geschrieben als

$$5\,7\,1\,1\,+\,-\,/\,3\,\times\,2\,1\,1\,+\,+\,-$$

Institut für Theoretische Informatik
Lehrstuhl Algorithmik

Umgekehrte Polnische Notation

UPN: Schreibweise von arithmetischen Ausdrücken

$$\left(\frac{5}{7-(1+1)} \times 3\right) - (2+(1+1))$$

wird geschrieben als

$$5\,7\,1\,1\,+\,-\,/\,3\,\times\,2\,1\,1\,+\,+\,-$$

Institut für Theoretische Informatik
Lehrstuhl Algorithmik

Umgekehrte Polnische Notation

UPN: Schreibweise von arithmetischen Ausdrücken

$$\left(\frac{5}{7-(1+1)} \times 3\right) - (2+(1+1))$$

wird geschrieben als

$$5\ 7\ 1\ 1\ +\ -\ /\ 3\ \times\ 2\ 1\ 1\ +\ +\ -$$

Institut für Theoretische Informatik
Lehrstuhl Algorithmik

Umgekehrte Polnische Notation

UPN: Schreibweise von arithmetischen Ausdrücken

$$\left(\frac{5}{7-(1+1)} \times 3\right) - (2+(1+1))$$

wird geschrieben als

$$5\,7\,1\,1\,+\,-\,/\,3\,\times\,2\,1\,1\,+\,+\,-$$

Gesucht: kontextfr. Gr., die arithmetische Ausdrücke in UPN erzeugt.

Ausdruck	**Zahl**	**Verknüpfung**
$A \to Z$	$Z \to 0 \mid 1 \mid \ldots \mid 9$	$V \to + \mid - \mid \times \mid /$
$A \to AAV$		

Institut für Theoretische Informatik
Lehrstuhl Algorithmik

Umgekehrte Polnische Notation

Kontextfreie Grammatik, die arithmetische Ausdrücke in UPN erzeugt:

Ausdruck	**Zahl**	**Verknüpfung**
$A \to Z$	$Z \to 0 \mid 1 \mid \ldots \mid 9$	$V \to + \mid - \mid \times \mid /$
$A \to AAV$		

Gesucht: äquivalente Grammatik in Greibach-Normalform.

Umgekehrte Polnische Notation

Kontextfreie Grammatik, die arithmetische Ausdrücke in UPN erzeugt:

Ausdruck	**Zahl**	**Verknüpfung**
$A \to Z$	$Z \to 0 \mid 1 \mid \ldots \mid 9$	$V \to + \mid - \mid \times \mid /$
$A \to AAV$		

Gesucht: äquivalente Grammatik in Greibach-Normalform.

Verknüpfung

$$V \to + \mid - \mid \times \mid /$$

Institut für Theoretische Informatik
Lehrstuhl Algorithmik

Umgekehrte Polnische Notation

Kontextfreie Grammatik, die arithmetische Ausdrücke in UPN erzeugt:

Ausdruck	**Zahl**	**Verknüpfung**
$A \to Z$	$Z \to 0 \mid 1 \mid \ldots \mid 9$	$V \to + \mid - \mid \times \mid /$
$A \to AAV$		

Gesucht: äquivalente Grammatik in Greibach-Normalform.

Verknüpfung

$$V \to + \mid - \mid \times \mid /$$

Zahl

$$Z \to 0 \mid 1 \mid \ldots \mid 9$$

Institut für Theoretische Informatik
Lehrstuhl Algorithmik

Umgekehrte Polnische Notation

Kontextfreie Grammatik, die arithmetische Ausdrücke in UPN erzeugt:

Ausdruck

$$A \to Z$$
$$A \to AAV$$

Zahl

$$Z \to 0 \mid 1 \mid \ldots \mid 9$$

Verknüpfung

$$V \to + \mid - \mid \times \mid /$$

Gesucht: äquivalente Grammatik in Greibach-Normalform.

Ausdruck

$$A \to 0 \mid 1 \mid \ldots \mid 9$$
$$A \to 0R \mid 1R \mid \ldots \mid 9R$$

Verknüpfung

$$V \to + \mid - \mid \times \mid /$$

Zahl

$$Z \to 0 \mid 1 \mid \ldots \mid 9$$

Institut für Theoretische Informatik
Lehrstuhl Algorithmik

Umgekehrte Polnische Notation

Kontextfreie Grammatik, die arithmetische Ausdrücke in UPN erzeugt:

Ausdruck

$$A \to Z$$
$$A \to AAV$$

Zahl

$$Z \to 0 \mid 1 \mid \ldots \mid 9$$

Verknüpfung

$$V \to + \mid - \mid \times \mid /$$

Gesucht: äquivalente Grammatik in Greibach-Normalform.

Ausdruck

$$A \to 0 \mid 1 \mid \ldots \mid 9$$
$$A \to 0R \mid 1R \mid \ldots \mid 9R$$

Rest „$R \to AV \mid AVAV \mid \ldots$“

$$R \to AV$$
$$R \to AVR$$

Verknüpfung

$$V \to + \mid - \mid \times \mid /$$

Zahl

$$Z \to 0 \mid 1 \mid \ldots \mid 9$$

Institut für Theoretische Informatik
Lehrstuhl Algorithmik

Umgekehrte Polnische Notation

Kontextfreie Grammatik, die arithmetische Ausdrücke in UPN erzeugt:

Ausdruck

$$A \to Z$$
$$A \to AAV$$

Zahl

$$Z \to 0 \mid 1 \mid \ldots \mid 9$$

Verknüpfung

$$V \to + \mid - \mid \times \mid /$$

Gesucht: äquivalente Grammatik in Greibach-Normalform.

Ausdruck

$$A \to 0 \mid 1 \mid \ldots \mid 9$$
$$A \to 0R \mid 1R \mid \ldots \mid 9R$$

Rest: „$R \to AV \mid AVAV \mid \ldots$“

$$R \to 0V \mid 1V \mid \ldots$$
$$R \to 0RV \mid 1RV \mid \ldots$$
$$R \to 0VR \mid 1VR \mid \ldots$$
$$R \to 0RVR \mid 1RVR \mid \ldots$$

Verknüpfung

$$V \to + \mid - \mid \times \mid /$$

Zahl

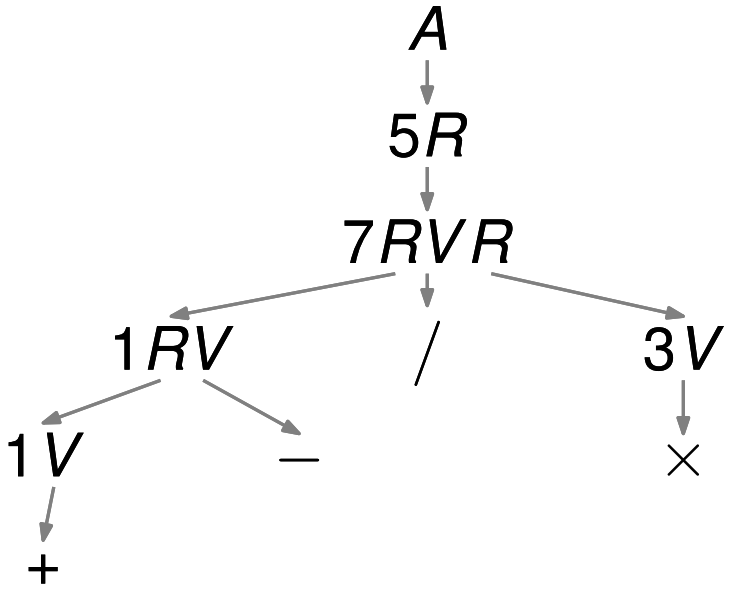
$$Z \to 0 \mid 1 \mid \ldots \mid 9$$

Institut für Theoretische Informatik
Lehrstuhl Algorithmik

Umgekehrte Polnische Notation

Ausdruck	**Rest** „$R \to AV \mid AVAV \mid \ldots$“	**Verknüpfung**
$A \to 0 \mid 1 \mid \ldots \mid 9$	$R \to 0V \mid 1V \mid \ldots$	$V \to + \mid - \mid \times \mid /$
$A \to 0R \mid 1R \mid \ldots \mid 9R$	$R \to 0RV \mid 1RV \mid \ldots$	
	$R \to 0VR \mid 1VR \mid \ldots$	
	$R \to 0RVR \mid 1RVR \mid \ldots$	

Gesucht: Ableitung von $5\,7\,1\,1\;+\,-\,/\,3\,\times$



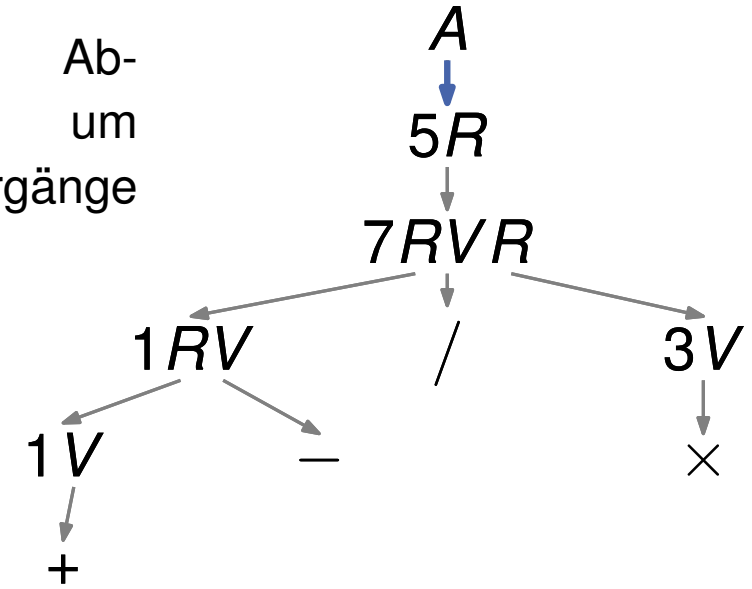
Institut für Theoretische Informatik
Lehrstuhl Algorithmik

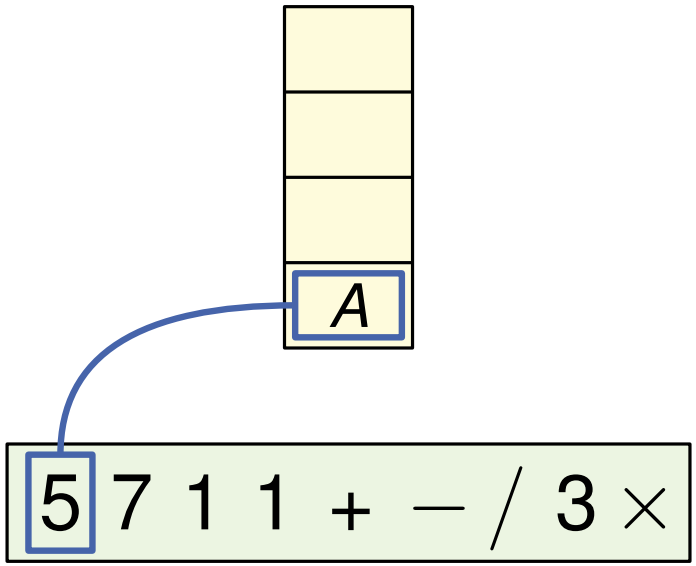
Umgekehrte Polnische Notation

Ausdruck	**Rest** „$R \to AV \mid AVAV \mid \ldots$“	**Verknüpfung**
$A \to 0 \mid 1 \mid \ldots \mid 9$	$R \to 0V \mid 1V \mid \ldots$	$V \to + \mid - \mid \times \mid /$
$A \to 0R \mid 1R \mid \ldots \mid 9R$	$R \to 0RV \mid 1RV \mid \ldots$	
	$R \to 0VR \mid 1VR \mid \ldots$	
	$R \to 0RVR \mid 1RVR \mid \ldots$	

Gesucht: PDA-Abarbeitung von $5\,7\,1\,1\,+\,-\,/\,3\,\times$

Benutze den Ableitungsbaum, um günstige Übergänge zu finden.





Institut für Theoretische Informatik
Lehrstuhl Algorithmik

Umgekehrte Polnische Notation

Ausdruck

$A \to 0 \mid 1 \mid \ldots \mid 9$

$A \to 0R \mid 1R \mid \ldots \mid 9R$

Rest „$R \to AV \mid AVAV \mid \ldots$“

$R \to 0V \mid 1V \mid \ldots$

$R \to 0RV \mid 1RV \mid \ldots$

$R \to 0VR \mid 1VR \mid \ldots$

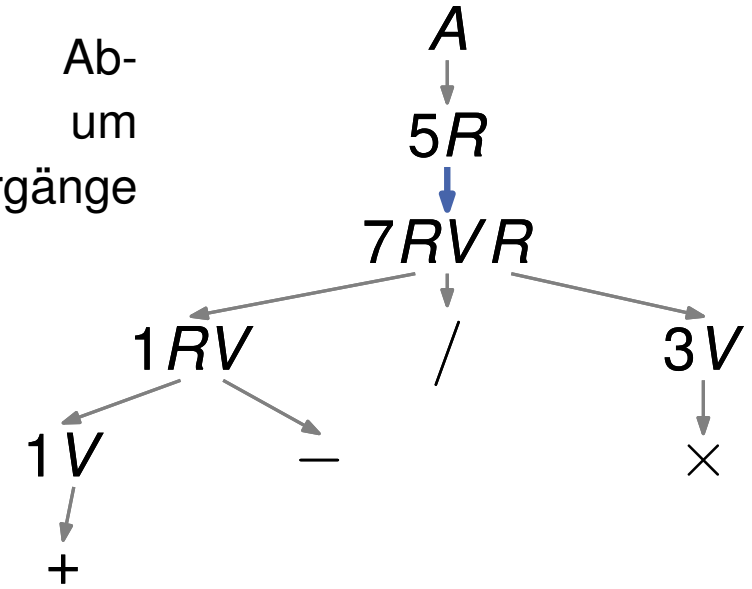
$R \to 0RVR \mid 1RVR \mid \ldots$

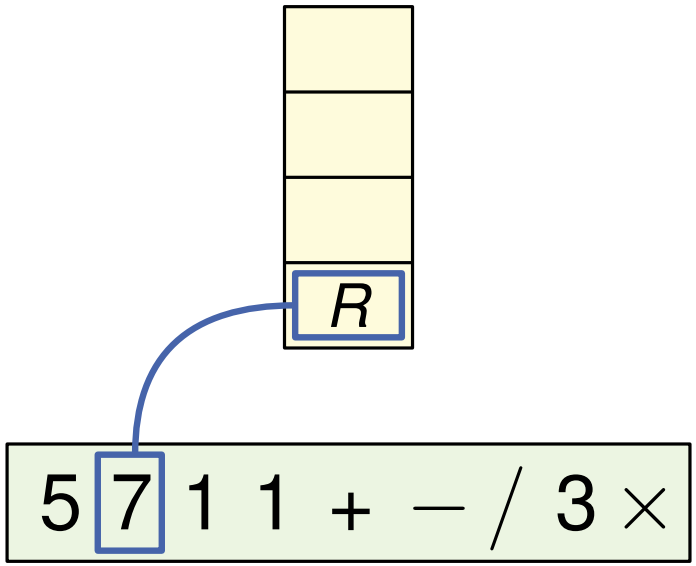
Verknüpfung

$V \to + \mid - \mid \times \mid /$

Gesucht: PDA-Abarbeitung von 5 7 1 1 + − / 3 ×

Benutze den Ableitungsbaum, um günstige Übergänge zu finden.





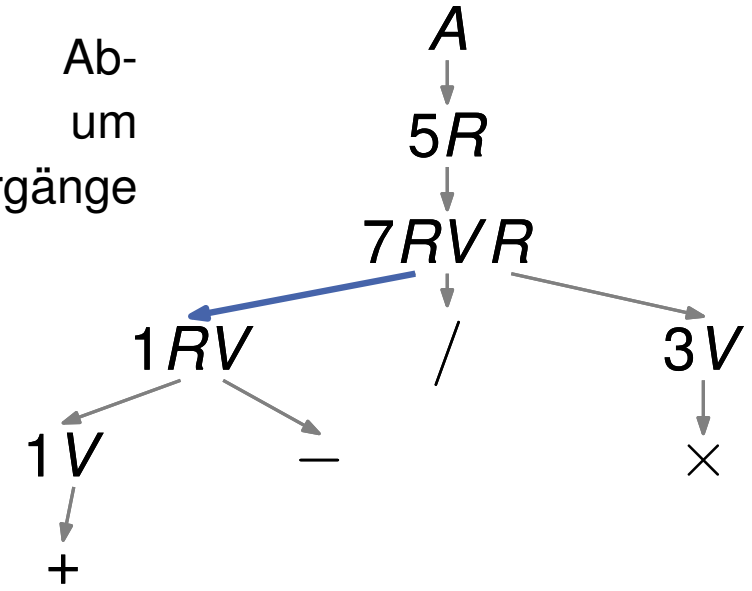
Institut für Theoretische Informatik
Lehrstuhl Algorithmik

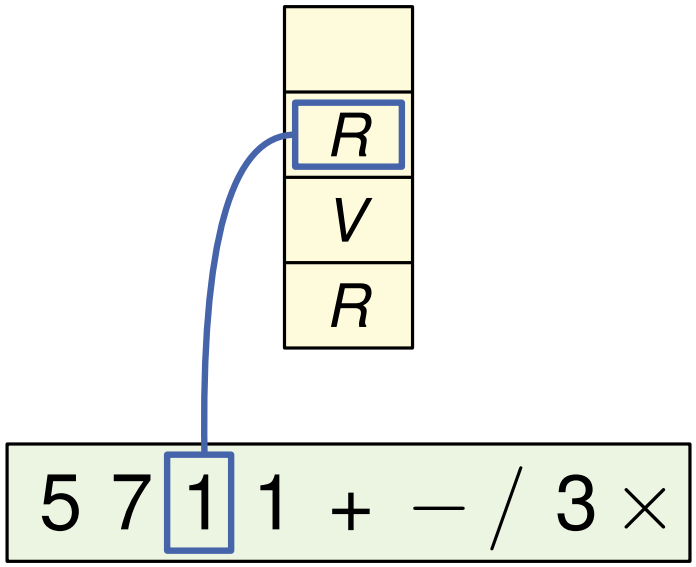
Umgekehrte Polnische Notation

Ausdruck	**Rest** „$R \to AV \mid AVAV \mid \ldots$“	**Verknüpfung**
$A \to 0 \mid 1 \mid \ldots \mid 9$	$R \to 0V \mid 1V \mid \ldots$	$V \to + \mid - \mid \times \mid /$
$A \to 0R \mid 1R \mid \ldots \mid 9R$	$R \to 0RV \mid 1RV \mid \ldots$	
	$R \to 0VR \mid 1VR \mid \ldots$	
	$R \to 0RVR \mid 1RVR \mid \ldots$	

Gesucht: PDA-Abarbeitung von 5 7 1 1 + − / 3 ×

Benutze den Ableitungsbaum, um günstige Übergänge zu finden.





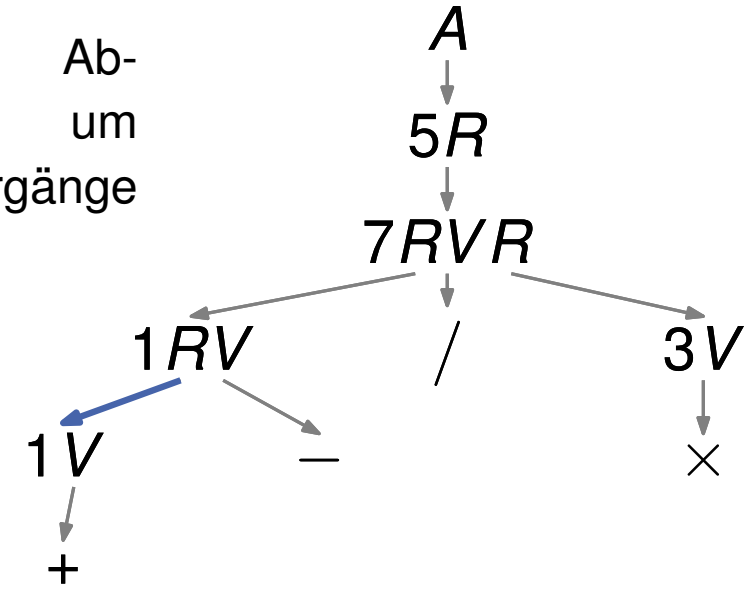
Institut für Theoretische Informatik
Lehrstuhl Algorithmik

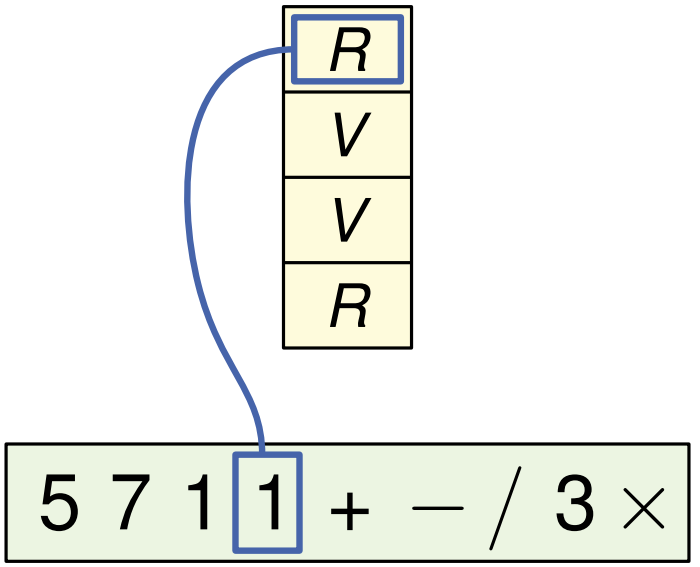
Umgekehrte Polnische Notation

Ausdruck	**Rest** „$R \to AV \mid AVAV \mid \ldots$“	**Verknüpfung**
$A \to 0 \mid 1 \mid \ldots \mid 9$	$R \to 0V \mid 1V \mid \ldots$	$V \to + \mid - \mid \times \mid /$
$A \to 0R \mid 1R \mid \ldots \mid 9R$	$R \to 0RV \mid 1RV \mid \ldots$	
	$R \to 0VR \mid 1VR \mid \ldots$	
	$R \to 0RVR \mid 1RVR \mid \ldots$	

Gesucht: PDA-Abarbeitung von 5 7 1 1 + − / 3 ×

Benutze den Ableitungsbaum, um günstige Übergänge zu finden.





Institut für Theoretische Informatik
Lehrstuhl Algorithmik

Umgekehrte Polnische Notation

Ausdruck

$A \to 0 \mid 1 \mid \ldots \mid 9$

$A \to 0R \mid 1R \mid \ldots \mid 9R$

Rest „$R \to AV \mid AVAV \mid \ldots$“

$R \to 0V \mid 1V \mid \ldots$

$R \to 0RV \mid 1RV \mid \ldots$

$R \to 0VR \mid 1VR \mid \ldots$

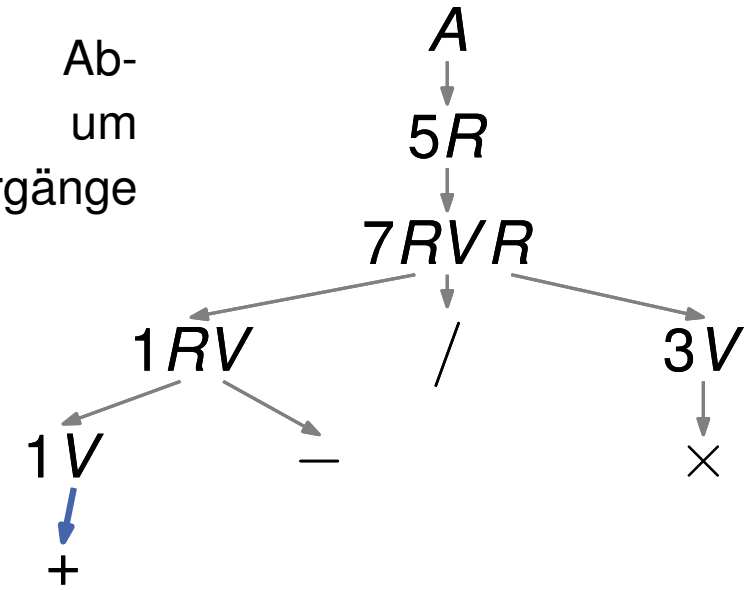
$R \to 0RVR \mid 1RVR \mid \ldots$

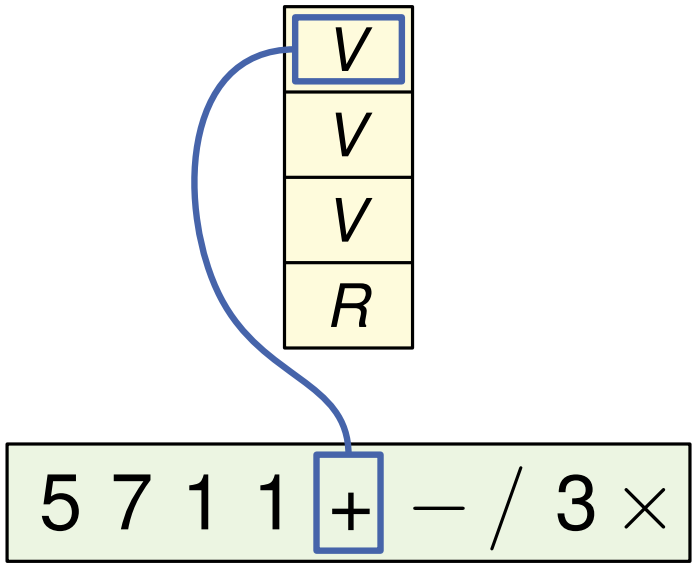
Verknüpfung

$V \to + \mid - \mid \times \mid /$

Gesucht: PDA-Abarbeitung von 5 7 1 1 + − / 3 ×

Benutze den Ableitungsbaum, um günstige Übergänge zu finden.





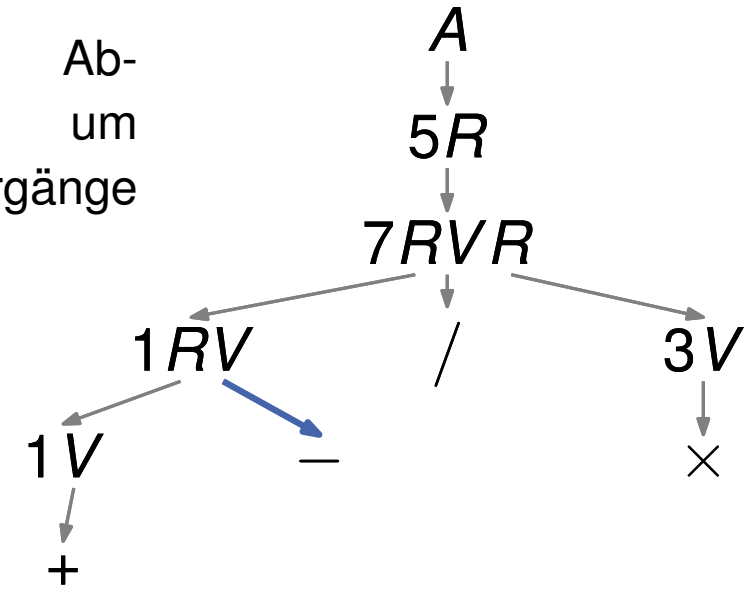
Institut für Theoretische Informatik
Lehrstuhl Algorithmik

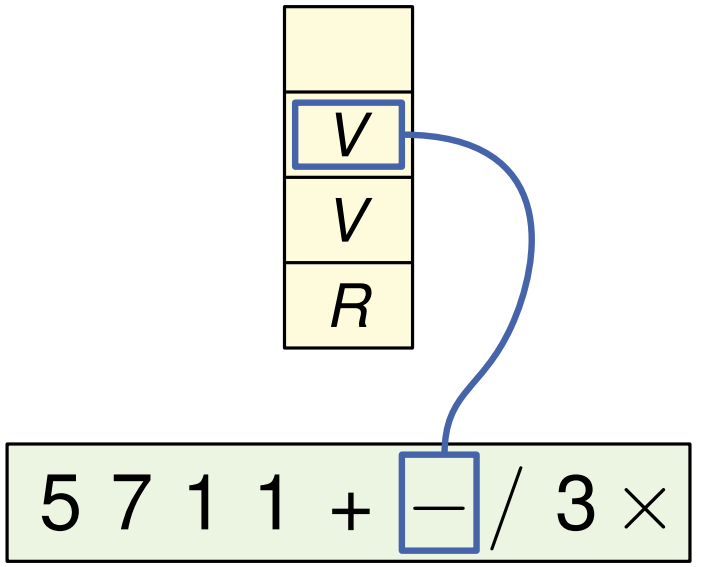
Umgekehrte Polnische Notation

Ausdruck	**Rest** „$R \to AV \mid AVAV \mid \ldots$“	**Verknüpfung**
$A \to 0 \mid 1 \mid \ldots \mid 9$	$R \to 0V \mid 1V \mid \ldots$	$V \to + \mid - \mid \times \mid /$
$A \to 0R \mid 1R \mid \ldots \mid 9R$	$R \to 0RV \mid 1RV \mid \ldots$	
	$R \to 0VR \mid 1VR \mid \ldots$	
	$R \to 0RVR \mid 1RVR \mid \ldots$	

Gesucht: PDA-Abarbeitung von $5\,7\,1\,1\,+\,-\,/\,3\,\times$

Benutze den Ableitungsbaum, um günstige Übergänge zu finden.





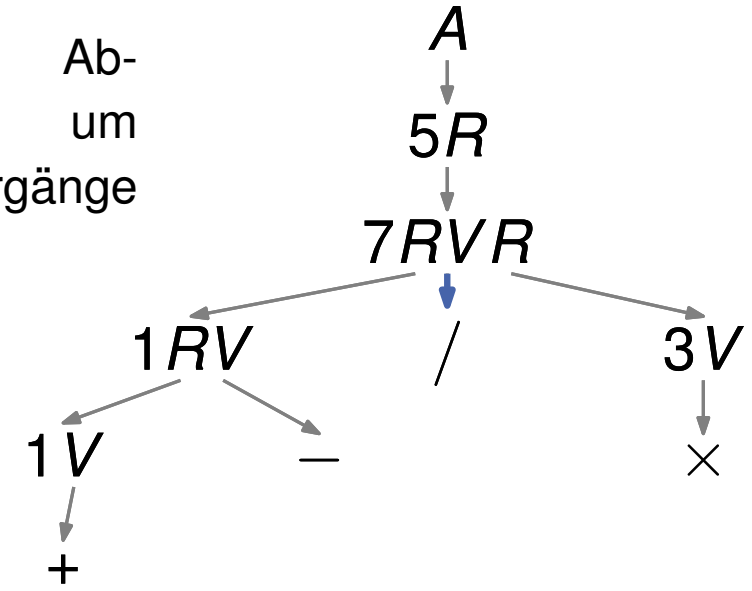
Institut für Theoretische Informatik
Lehrstuhl Algorithmik

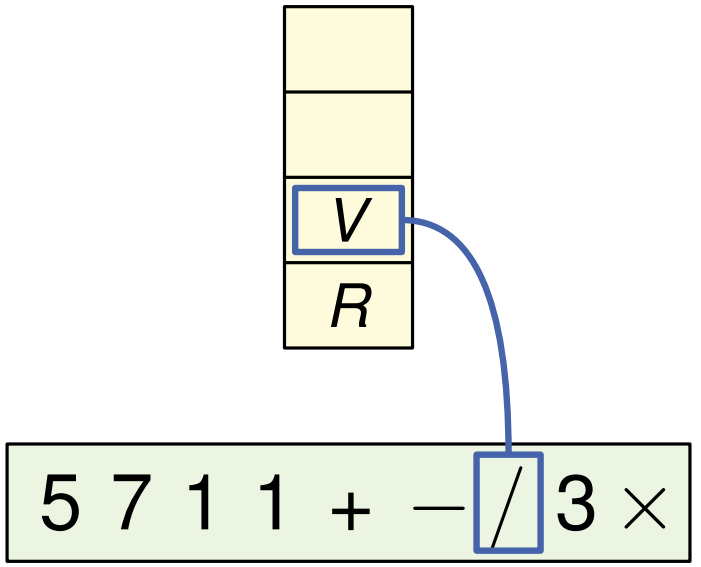
Umgekehrte Polnische Notation

Ausdruck	**Rest** „$R \to AV \mid AVAV \mid \ldots$“	**Verknüpfung**
$A \to 0 \mid 1 \mid \ldots \mid 9$	$R \to 0V \mid 1V \mid \ldots$	$V \to + \mid - \mid \times \mid /$
$A \to 0R \mid 1R \mid \ldots \mid 9R$	$R \to 0RV \mid 1RV \mid \ldots$	
	$R \to 0VR \mid 1VR \mid \ldots$	
	$R \to 0RVR \mid 1RVR \mid \ldots$	

Gesucht: PDA-Abarbeitung von $5\,7\,1\,1\,+\,-\,/\,3\,\times$

Benutze den Ableitungsbaum, um günstige Übergänge zu finden.





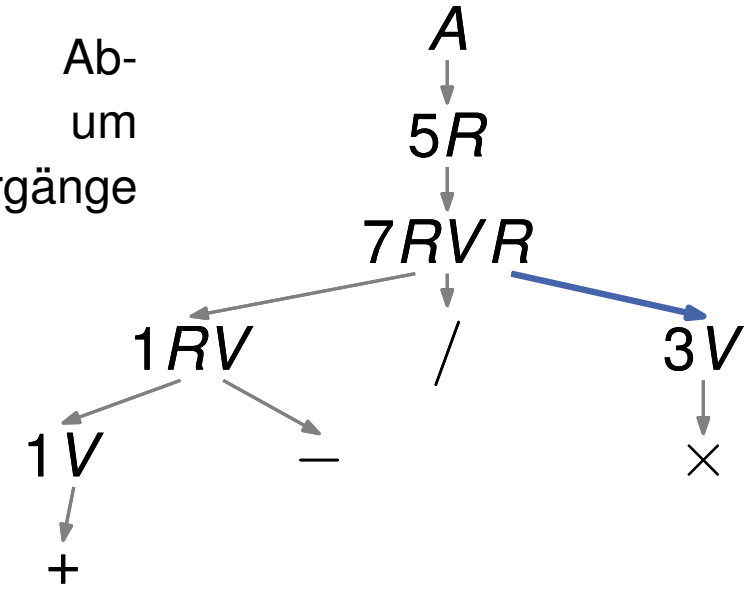
Institut für Theoretische Informatik
Lehrstuhl Algorithmik

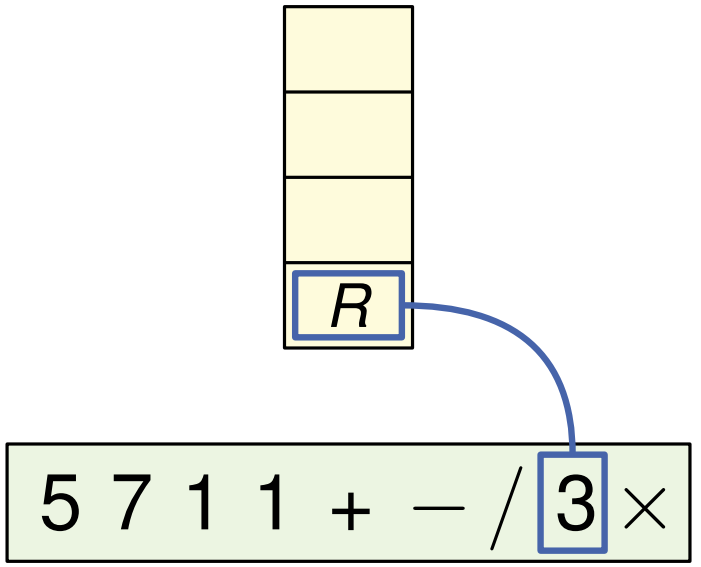
Umgekehrte Polnische Notation

Ausdruck	**Rest** „$R \to AV \mid AVAV \mid \ldots$“	**Verknüpfung**
$A \to 0 \mid 1 \mid \ldots \mid 9$	$R \to 0V \mid 1V \mid \ldots$	$V \to + \mid - \mid \times \mid /$
$A \to 0R \mid 1R \mid \ldots \mid 9R$	$R \to 0RV \mid 1RV \mid \ldots$	
	$R \to 0VR \mid 1VR \mid \ldots$	
	$R \to 0RVR \mid 1RVR \mid \ldots$	

Gesucht: PDA-Abarbeitung von $5\,7\,1\,1\,+\,-\,/\,3\,\times$

Benutze den Ableitungsbaum, um günstige Übergänge zu finden.





Institut für Theoretische Informatik
Lehrstuhl Algorithmik

Umgekehrte Polnische Notation

Ausdruck

$A \to 0 \mid 1 \mid \ldots \mid 9$

$A \to 0R \mid 1R \mid \ldots \mid 9R$

Rest „$R \to AV \mid AVAV \mid \ldots$“

$R \to 0V \mid 1V \mid \ldots$

$R \to 0RV \mid 1RV \mid \ldots$

$R \to 0VR \mid 1VR \mid \ldots$

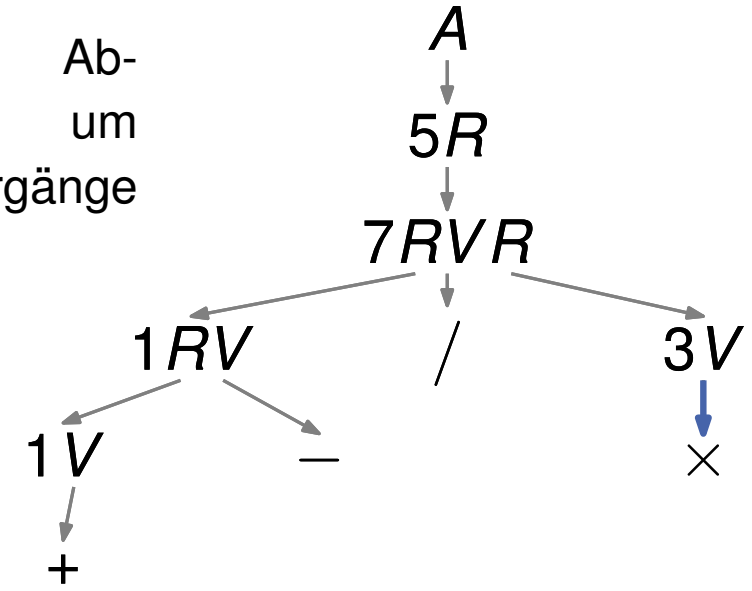
$R \to 0RVR \mid 1RVR \mid \ldots$

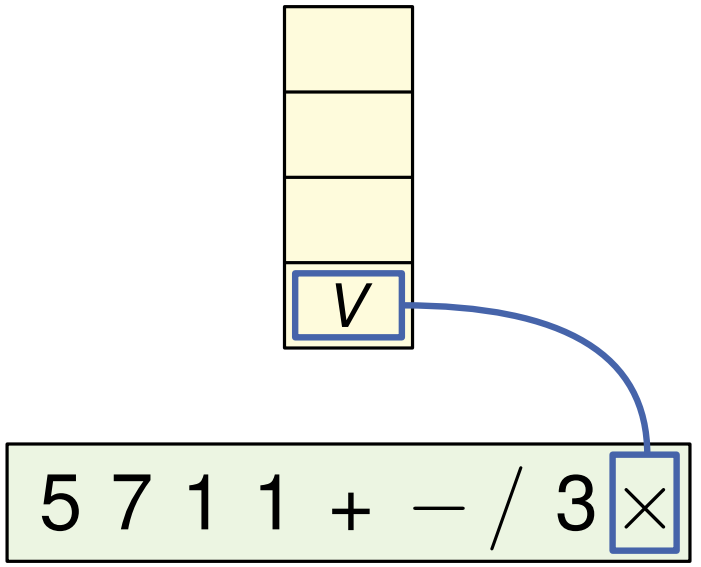
Verknüpfung

$V \to + \mid - \mid \times \mid /$

Gesucht: PDA-Abarbeitung von $5\,7\,1\,1\,+\,-\,/\,3\,\times$

Benutze den Ableitungsbaum, um günstige Übergänge zu finden.





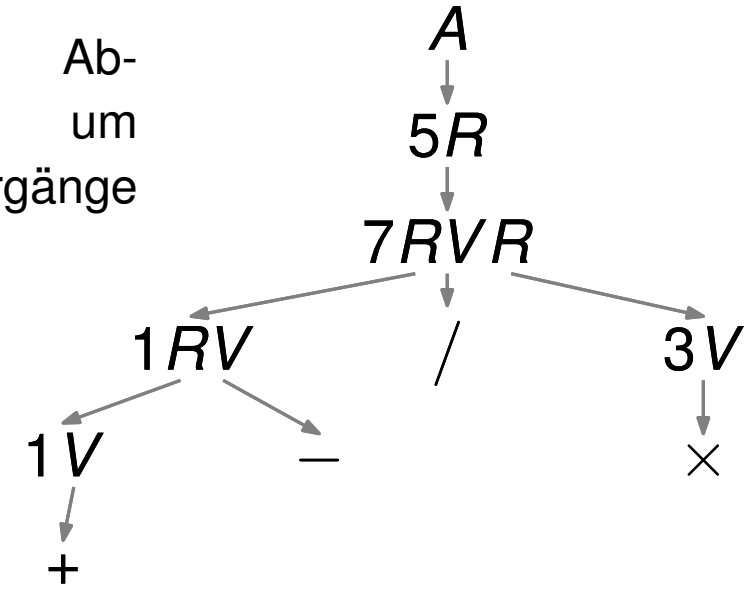
Institut für Theoretische Informatik
Lehrstuhl Algorithmik

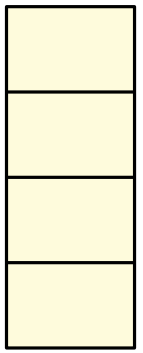
Umgekehrte Polnische Notation

Ausdruck	**Rest** „$R \to AV \mid AVAV \mid \ldots$“	**Verknüpfung**
$A \to 0 \mid 1 \mid \ldots \mid 9$	$R \to 0V \mid 1V \mid \ldots$	$V \to + \mid - \mid \times \mid /$
$A \to 0R \mid 1R \mid \ldots \mid 9R$	$R \to 0RV \mid 1RV \mid \ldots$	
	$R \to 0VR \mid 1VR \mid \ldots$	
	$R \to 0RVR \mid 1RVR \mid \ldots$	

Gesucht: PDA-Abarbeitung von $5\,7\,1\,1\,+\,-\,/\,3\,\times$

Benutze den Ableitungsbaum, um günstige Übergänge zu finden.





Akzeptiert durch leeren Stack

$5\,7\,1\,1\,+\,-\,/\,3\,\times$ □

Institut für Theoretische Informatik
Lehrstuhl Algorithmik