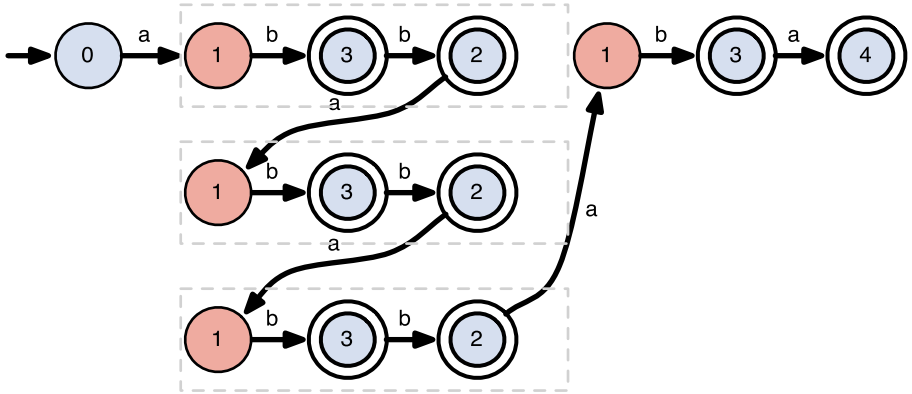
# Berechenbarkeitstheorie

25. Vorlesung

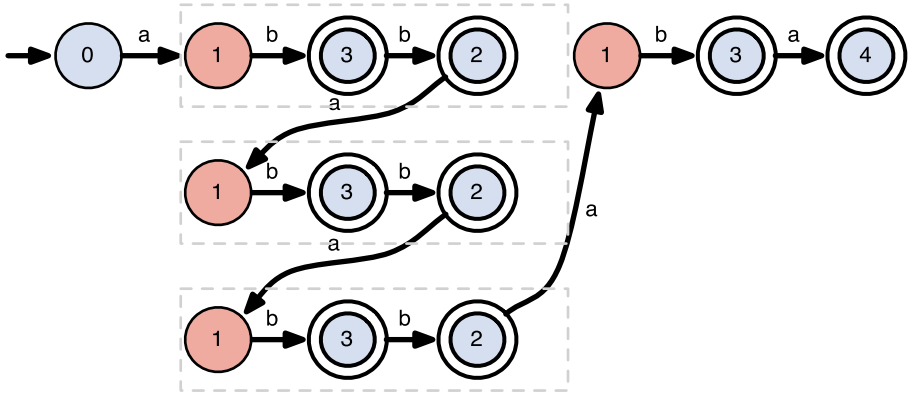



Dr. Franziska Jahnke
Institut für Mathematische Logik und Grundlagenforschung
WWU Münster

# Berechenbarkeitstheorie

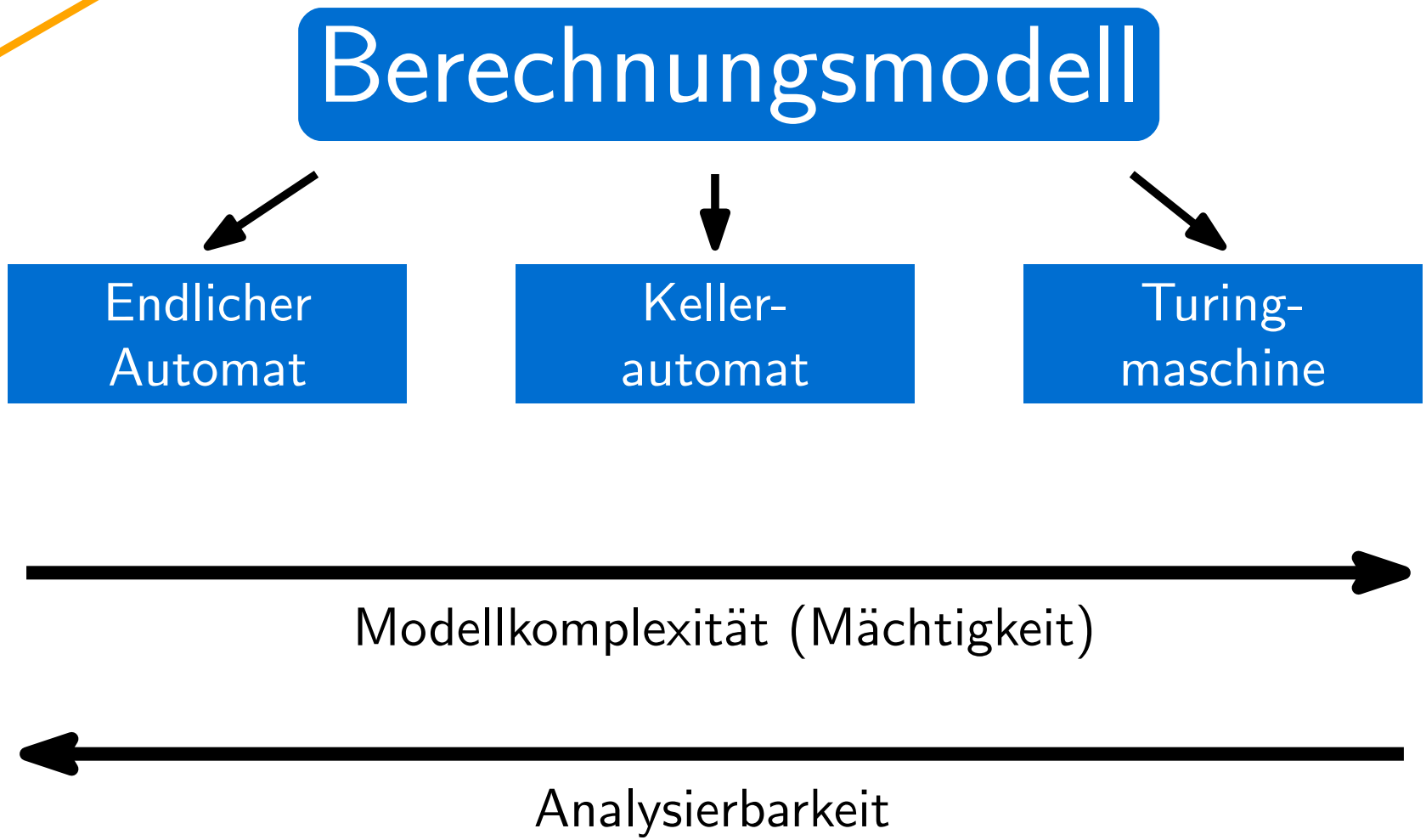
25. Vorlesung

Wrap-up




Dr. Franziska Jahnke
Institut für Mathematische Logik und Grundlagenforschung
WWU Münster

1. Vorlesung




**Was ist ein Computer?**

2. Vorlesung

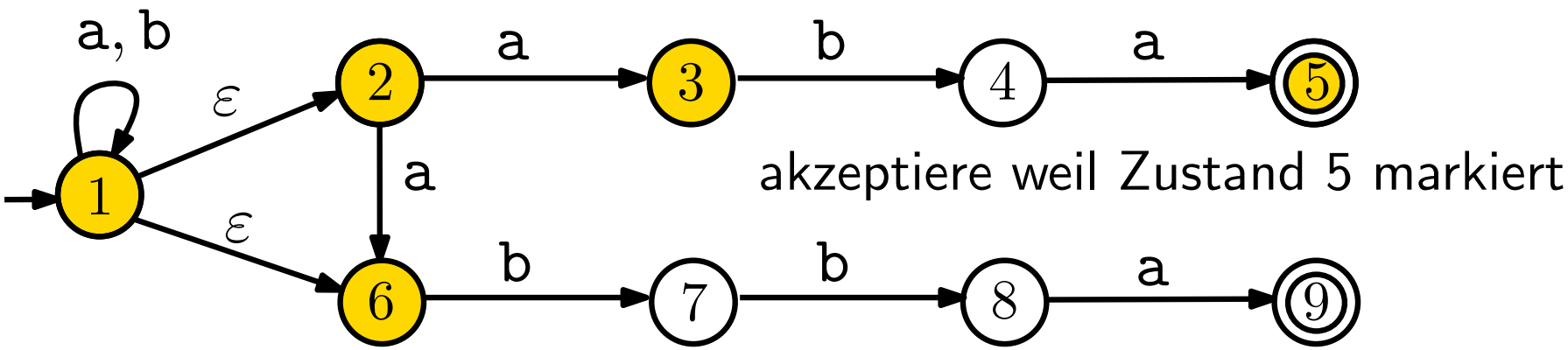
## Satz 2

$$\{L \mid L \text{ wird durch NEA erkannt}\} = \text{REG}$$

**Hinweis** Jeder DEA kann als NEA verstanden werden der weder Nichtdeterminismus noch $\varepsilon$-Übergänge nutzt.

**Frage** Kann ich einen NEA durch einen DEA simulieren?

**Berechnung $\delta^*(q_0, w)$ von Hand**
(In welchen Zuständen könnte ich aktuell sein?)




$w = \text{aba}$

**Idee** DEA Zustandsraum = Teilmengen von Zuständen

**NEA $\rightsquigarrow$ Potenzautomat**

# Vom DEA zum RA

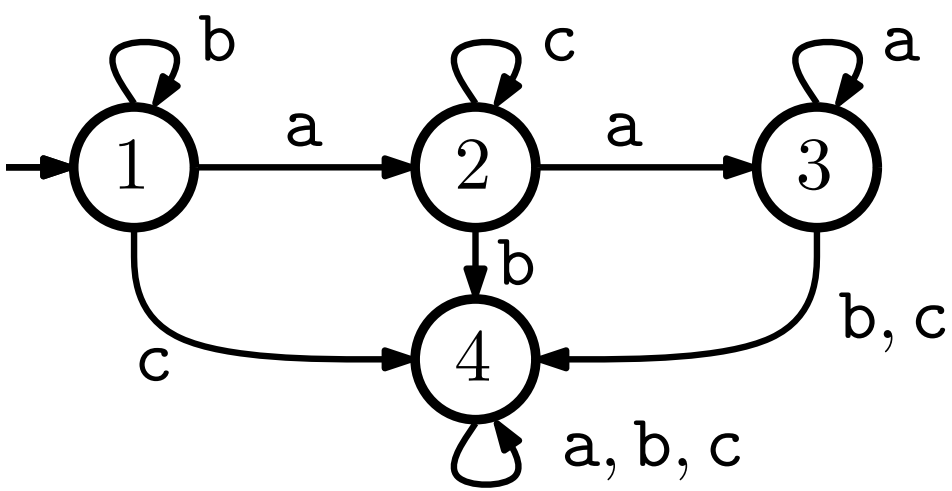


**Ansatz**

- Wenn $L \in$ REG, dann existiert ein DEA $M$ der $L$ er
- $M = (Q, \Sigma, \delta, 1, F)$ mit $Q = \{1, 2, \ldots, n\}$
- **Ziel:** Konstruiere RA $R$ mit $L(R) = L$

$R_{ij}^k :=$ RA für alle Wörter die von Zustand $i$ nach $j$ führen ohne einen Zustand $> k$ zu benutzen.

Anfangs- und Endzustand $i, j$ dürfen $> k$ sein!




**Bsp** $R_{22}^0 = \mathsf{c} + \varepsilon$

$R_{24}^2 = \mathsf{c}^*\mathsf{b}$

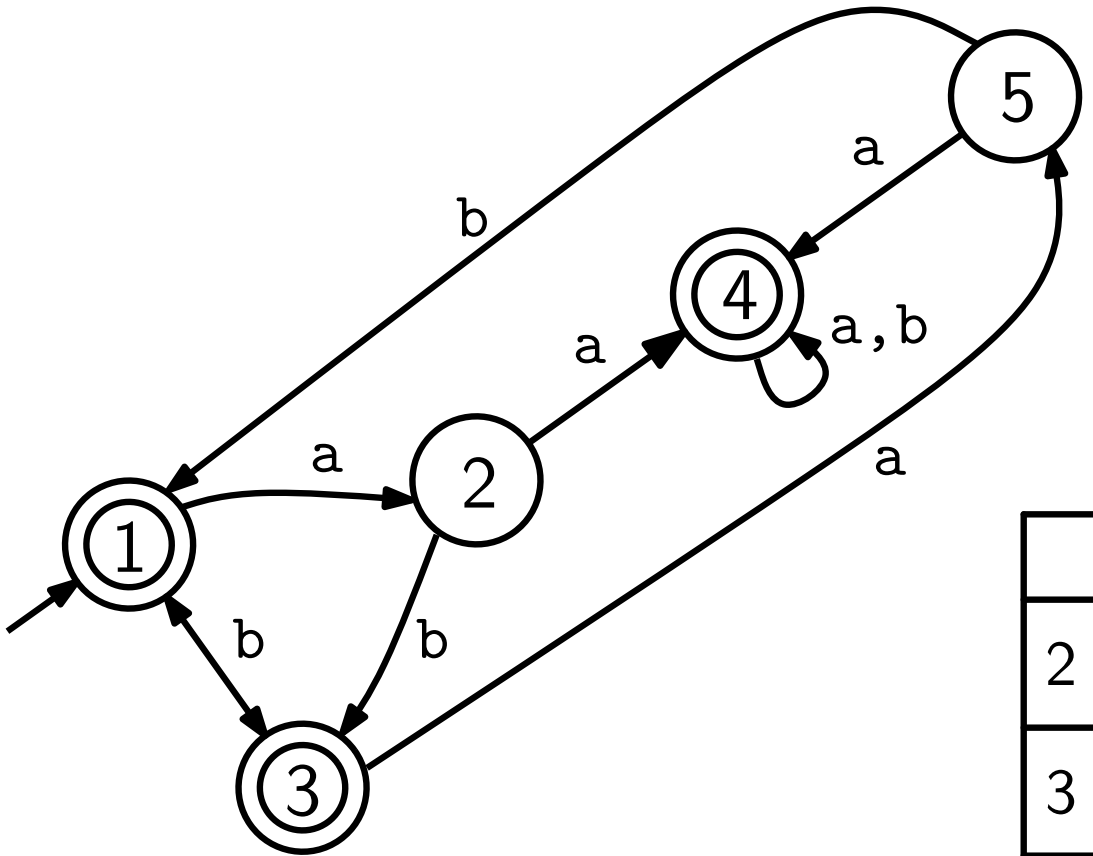
$R_{24}^3 = (\mathsf{c}^*\mathsf{b} + \mathsf{c}^*\mathsf{a}\mathsf{a}^*(\mathsf{b} + \mathsf{c}))$

**Satz von Kleene**

# Table-Filling Algorithmus im Beispiel



4. Vorlesung




$(1,4) \xrightarrow{a} (2,4)$

$(1,4) \xrightarrow{b} (3,4)$

| | 1 | 2 | 3 | 4 |
|---|---|---|---|---|
| 2 | 1 | | | |
| 3 | | 1 | | |
| 4 | 1 | 1 | | |
| 5 | 1 | | 1 | 1 |

**Äquivalente Zustände**

# Eigenschaften der MN Relation

$$(u \equiv_L v :\iff \forall z \in \Sigma^* : uz \in L \Leftrightarrow vz \in L)$$

Äquivalenzrelation

5. Vorlesung

25. Vorlesung

(1) $\equiv_L$ ist **symmetrisch**

(2) $\equiv_L$ ist **reflexiv**

(3) $\equiv_L$ ist **transitiv**

(4) $\forall a \in \Sigma : u \equiv_L v \implies ua \equiv_L va$

sonst trennt $a \circ$(Trennwort $uz, vz$) auch $u, v$

(4') $\forall z \in \Sigma^* : u \equiv_L v \implies uz \equiv_L vz$

folgt aus (4)

(5) $u \equiv_L v \implies u \in L \Leftrightarrow v \in L$

sonst trennbar mit $\varepsilon$

(1)+(2)+(3)+(4): **Rechtskongruenz**

(5): $\equiv_L$ **saturiert** $L$

**Myhill-Nerode Relation**

6. Vorlesung

## Satz 6 (Satz von Myhill–Nerode)

$$L \in \text{REG} \iff \equiv_L \text{ hat endlichen Index}$$

**Beweis** $(\Rightarrow)$

$L \in \text{REG} \Rightarrow \exists \text{DEA } M\colon L(M) = L$

$\Rightarrow \equiv_M$ saturiert $L$

$\Rightarrow \equiv_M$ verfeinert $\equiv_L$

$\Rightarrow$ Index $\equiv_L \leq$ Index $\equiv_M < \infty$

↓ Lemma 4

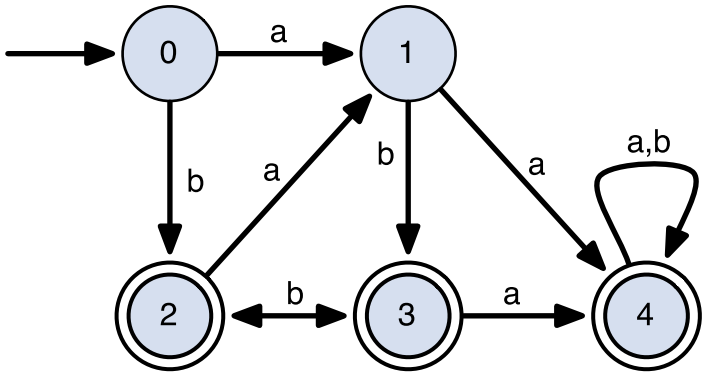
**Beweis** $(\Leftarrow)$

$\equiv_L$ hat endlichen Index $\Rightarrow M(\equiv_L)$ ist DEA der $L$ erkennt (nach Lemma 3)

$\Rightarrow L$ ist regulär

□

**Am Beispiel** $M$

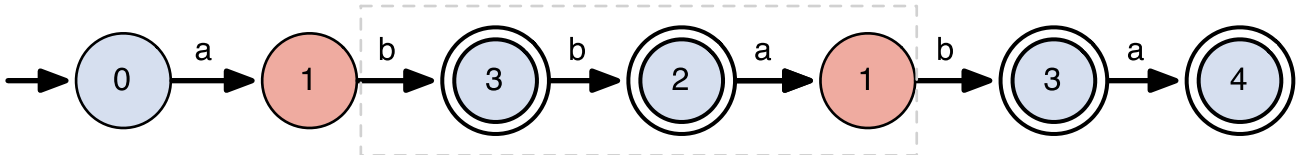



$|Q| = 5$

$k = 5$

7. Vorlesung

$w =$ abbaba hat folgenden Lauf



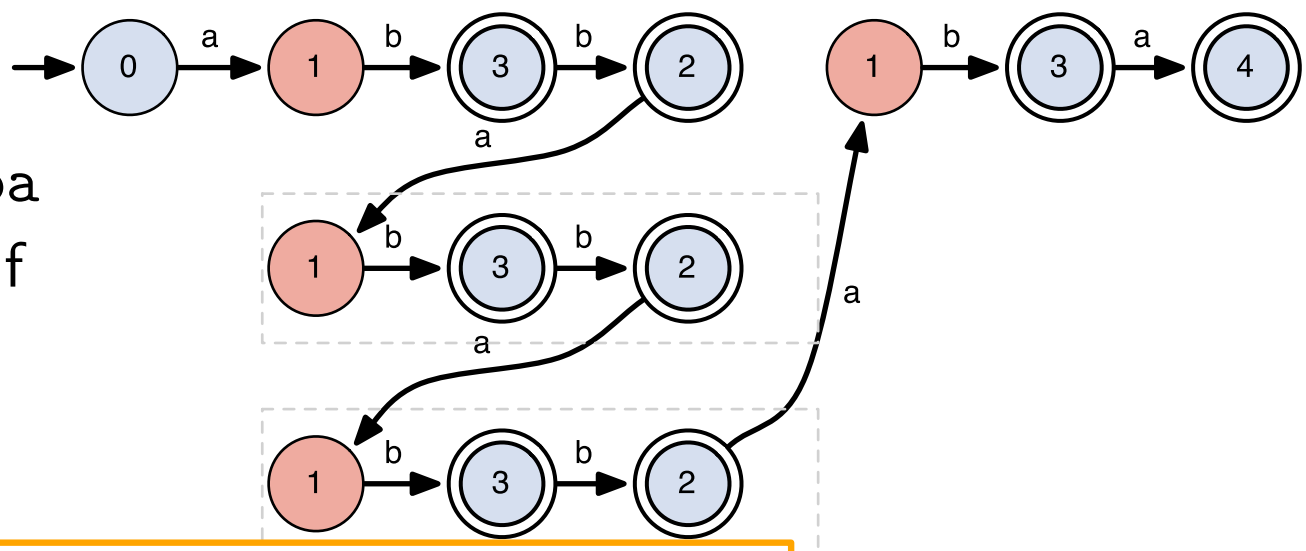

erster doppelter Zustand ist 1

$x =$ a, $y =$ bba, $z =$ ba

Wiederholung $i = 3$

$xyyyz =$ abbabbabbaba

hat akzeptierenden Lauf

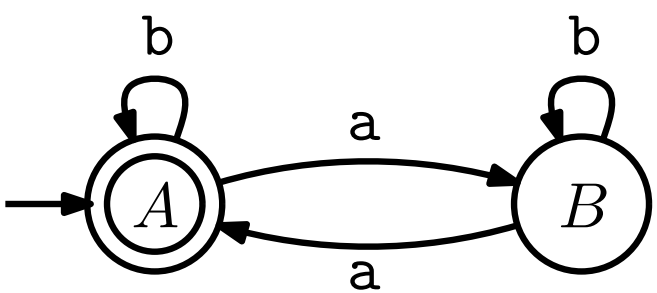



**Reguläres Pumpinglemma**

## Beweis $L \in \text{REG} \Rightarrow L \in \text{CFL}$

- sei $M = (Q, \Sigma, \delta, q_0, F)$ ein DEA, welcher $L$ erkennt
- Definiere Grammatik $G_M = (V, \Sigma, R, S)$ mit
  - $V = Q$,
  - $S = q_0$,
  - für alle $q \in Q$ und $a \in \Sigma$ mit $\delta(q, a) = p$ füge die Regel $q \to ap$ zu $R$ hinzu,
  - für alle $q \in F$ füge die Regel $q \to \varepsilon$ zu $R$ hinzu.

## Konstruktion am Beispiel




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

R: $A \to \mathtt{a}B \mid \mathtt{b}A \mid \varepsilon$

$B \to \mathtt{a}A \mid \mathtt{b}B$

**Bsp. 1**

$$
\begin{array}{lcl}
S & \to & AB \mid CA \\
A & \to & AA \mid CB \mid \texttt{a} \\
B & \to & AC \mid \texttt{b} \\
C & \to & \texttt{c}
\end{array}
\qquad w = \texttt{cbaac}
$$

| | 1 | 2 | 3 | 4 | 5 |
|---|---|---|---|---|---|
| 5 | | | | | $C$ |
| 4 | | | | $A$ | $B$ |
| 3 | | | $A$ | $A$ | |
| 2 | | $B$ | – | | |
| 1 | $C$ | $A$ | | | |

## CYK Algorithmus (für $G$ in CNF)

10. Vorlesung

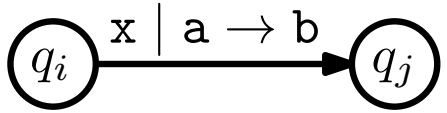
## Bsp. Kellerautomat für $\{\mathtt{a}^n\mathtt{b}^n \mid n \geq 0\}$

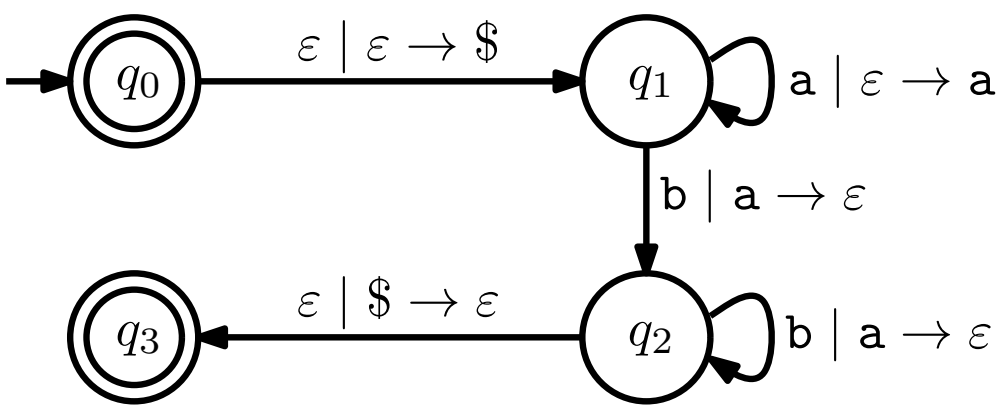
$Q = \{q_0, q_1, q_2, q_3\}$

$\Sigma = \{\mathtt{a}, \mathtt{b}\}$ $\quad \Gamma = \{\$, \mathtt{a}\}$ $\quad F = \{q_0, q_3\}$

| | | | |
|---|---|---|---|
| $\delta(q_0, \varepsilon, \varepsilon)$ | $=$ | $\{(q_1, \$)\}$ | Push \$ |
| $\delta(q_1, \mathtt{a}, \varepsilon)$ | $=$ | $\{(q_1, \mathtt{a})\}$ | Push a |
| $\delta(q_1, \mathtt{b}, \mathtt{a})$ | $=$ | $\{(q_2, \varepsilon)\}$ | Pop a |
| $\delta(q_2, \mathtt{b}, \mathtt{a})$ | $=$ | $\{(q_2, \varepsilon)\}$ | Pop a |
| $\delta(q_2, \varepsilon, \$)$ | $=$ | $\{(q_3, \varepsilon)\}$ | Pop \$ |

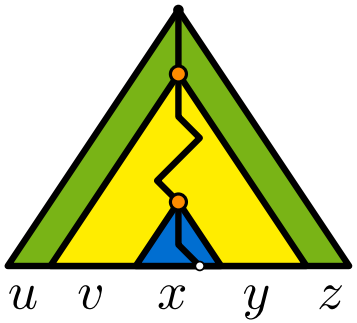
**Diagrammdarstellung** für $(q_j, \mathtt{b}) \in \delta(q_i, \mathtt{x}, \mathtt{a})$



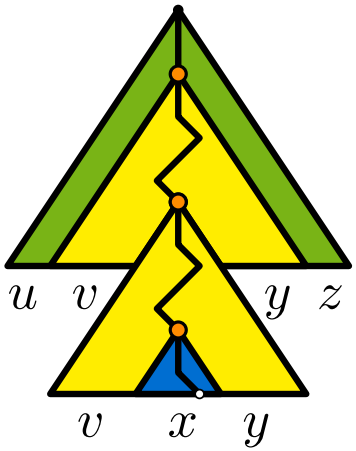




**Kellerautomat entspricht KF**

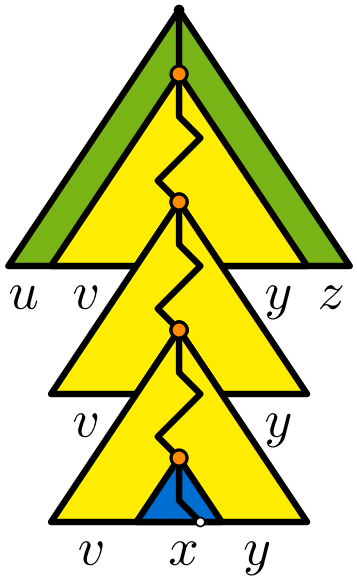
**(Schema)**



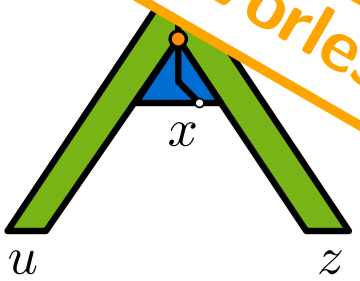

$w = uvxyz$




$uv^2xy^2z$



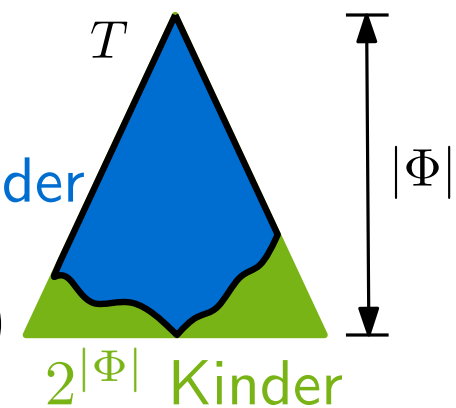

$uv^3xy^3z$




$uv^0xy^0z$

**(Details)**

- wähle $G$ in Chomsky Normalform
- wähle $k = 2^{|V|+1}$
- wähle Ableitungsbaum für $w$ und entferne Blätter $(T)$
- wähle als $\Phi$ einen längsten Wurzel–Blatt Pfad in $T$
- Kinder von $T = |w| \geq k = 2^{|V|+1}$, d.h. $|\Phi| \geq |V| + 1$



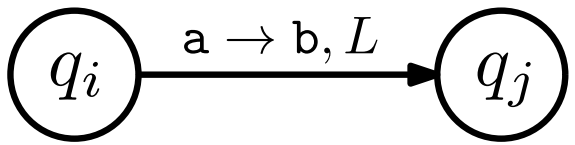

**Kontextfreies Pumpinglemma**

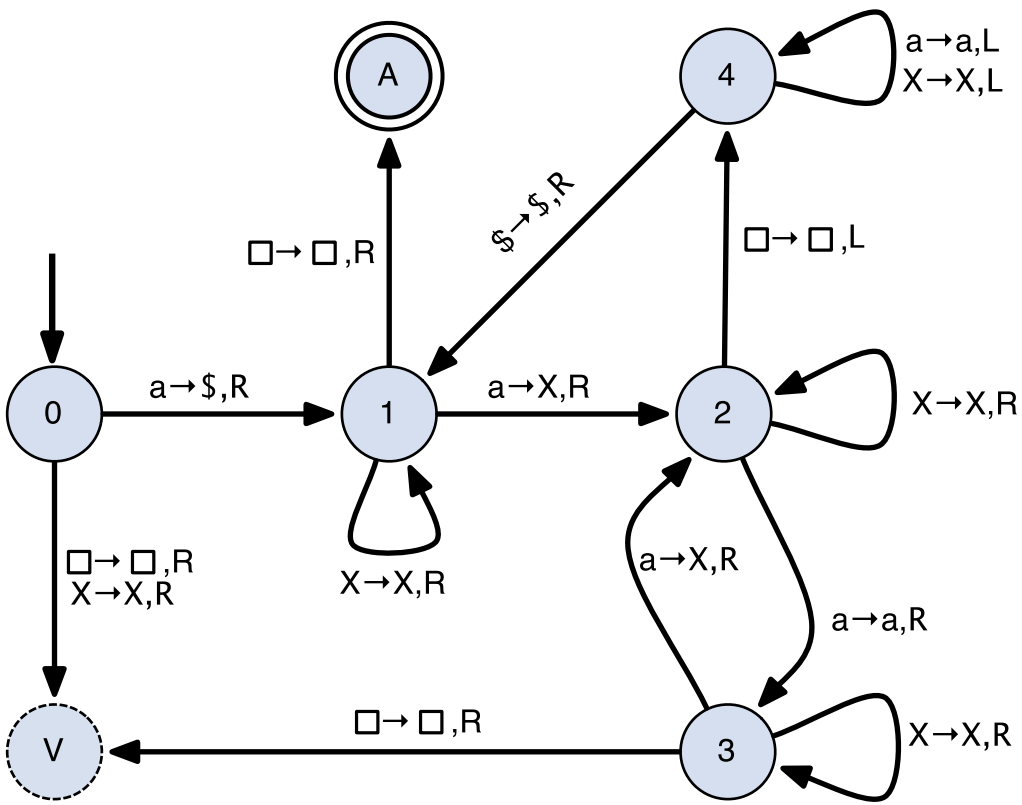
# Beispiel $L = \{\mathtt{a}^{2^n} \in \{\mathtt{a}\}^* \mid n \geq 0\}$



- **Idee:** $k$ ist 2er-Potenz ungleich 1 $\iff$ $k/2$ ist 2er-Potenz
- Kann ich die Eingabe wiederholt halbieren?
- $Q = \{q_0, q_1, q_2, q_3, q_4, q_A, q_V\}, \Gamma = \{\mathtt{a}, \mathtt{X}, \$, \square\}$
- Angabe von $\delta$ in Diagrammform

$\delta(q_i, \mathtt{a}) = (q_j, \mathtt{b}, L)$



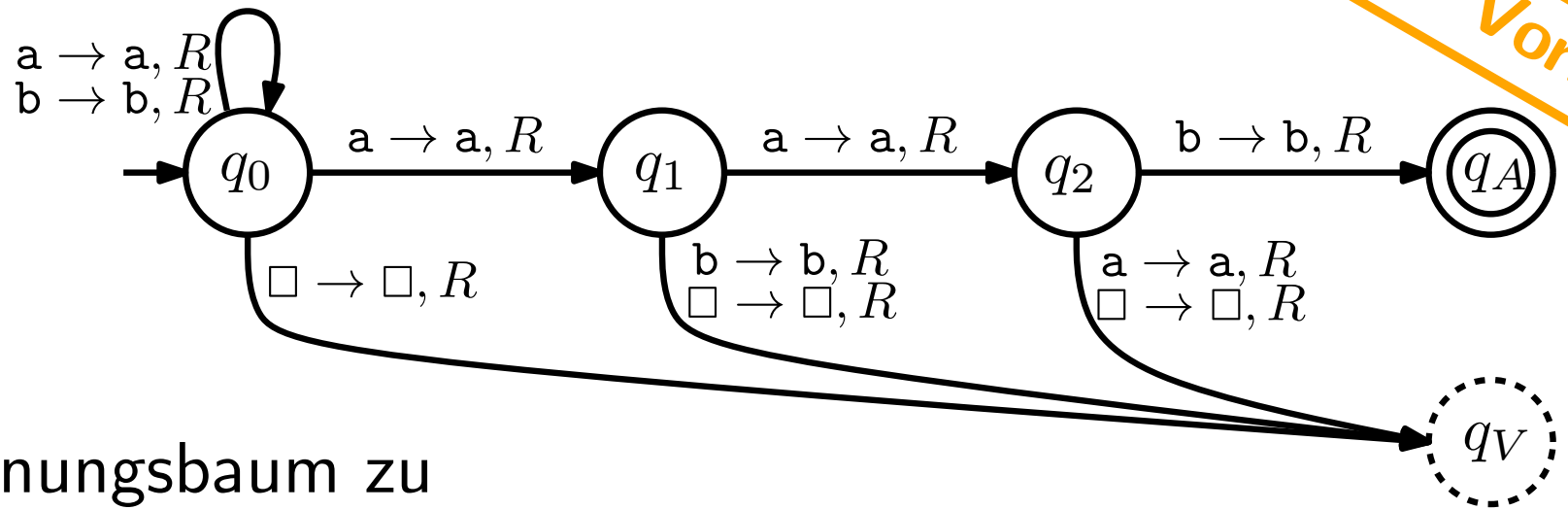




1. Markiere Anfang durch \$ $q_0$
2. war das das einzige Zeichen, dann akzeptiere $q_1$
3. lösche jedes 2. a auf dem Band $q_2, q_3$
   - 3.1. bei Misserfolg (ungerade Anzahl) verwerfe
   - 3.2. bei Erfolg (gerade Anzahl) gehe zum \$ nach links $q_4$
4. Goto 2.

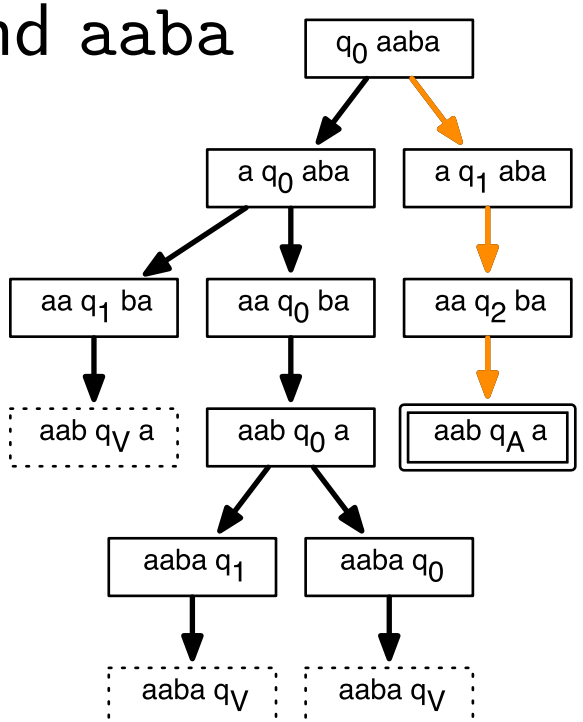
# Berechnungsbaum am Beispiel



TM $T$




Berechnungsbaum zu $T$ und aaba




**Definition**

Ein Wort $w$ wird von der NTM $T$ akzeptiert, gdw. im Berechnungsbaum zu $T$ und $w$ es eine akzeptierende Konfiguration gibt.

$\rightarrow$ aaba wird akzeptiert

- Zeichne alle $f_i$ in eine Tabelle ein

| | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | |
|---|---|---|---|---|---|---|---|---|---|
| $f_0$ | 0 | 1 | 0 | 0 | 0 | 1 | 1 | 1 | |
| $f_1$ | 0 | 0 | 1 | 1 | 0 | 0 | 1 | 1 | ... |
| $f_2$ | 0 | 0 | 0 | 1 | 0 | 0 | 1 | 1 | |
| $f_3$ | 1 | 1 | 0 | 1 | 0 | 0 | 0 | 1 | |
| $\vdots$ | | | | | $\vdots$ | | | | |
| $g$ | 1 | 1 | 1 | 0 | 0 | 1 | 0 | 1 | |

- Konstruiere für $(f_i)_{i \in \mathbf{N}}$ folgende Funktion

$$g(x) := \begin{cases} 1, \text{falls } f_x(x) = 0 \\ 0, \text{falls } f_x(x) = 1 \end{cases}$$

- $\forall i$: $g$ unterscheidet sich von $f_i$ an der Stelle $i$ $(f_i(i) \neq g(i))$
- $g$ fehlt in der Aufzählung der $f_i$, $g$ ist aber eine Funktion $\{0,1\}^* \to \mathbf{N}$
- es gibt keine Aufzählung der Funktionen $\{0,1\}^{\mathbf{N}}$, und somit ist diese Menge **überabzählbar**

□

**Existenz nicht-erkennbarer Sprachen**

**Satz 20**

Die Sprache $A_{\mathsf{TM}} = \{\langle M, w\rangle \mid M(w) \text{ akzeptiert}\}$ ist **nicht entscheidbar**.

14. Vorlesung

**Beweis**

Angenommen $A_{\mathsf{TM}} \in \mathbb{E}$, dann existiert für $A_{\mathsf{TM}}$ ein Entscheider $H$.

$$H(\langle M, w\rangle) = \begin{cases} \text{akzeptiert} & \text{M akzeptiert } w \\ \text{verwirft} & \text{M akzeptiert } w \text{ nicht} \end{cases}$$

Wir konstruieren die Diagonalisierungsmaschine $D(\langle M\rangle)$ wie folgt:

1. $D$ simuliert $H(\langle M, \langle M\rangle\rangle)$
2. $D$ gibt das entgegengesetzte Ergebnis der Simulation zurück

Also: $$D(\langle M\rangle) = \begin{cases} \text{akzeptiert} & \text{wenn } M(\langle M\rangle) \text{ verwirft} \\ \text{verwirft} & \text{wenn } M(\langle M\rangle) \text{ akzeptiert} \end{cases}$$

D.h.: $$D(\langle D\rangle) = \begin{cases} \text{akzeptiert} & \text{wenn } D(\langle D\rangle) \text{ verwirft} \\ \text{verwirft} & \text{wenn } D(\langle D\rangle) \text{ akzeptiert} \end{cases}$$

$\rightarrow$ Widerspruch, $D$, und somit $H$ kann nicht existieren

**Diagonalisierer/Antisimulation**

## Satz von Rice

Für eine nicht-triviale Eigenschaft $\mathcal{U} \subset \mathcal{R}$ gilt

$$L_{\mathcal{U}} := \{\langle M \rangle \mid f_M \in \mathcal{U}\} \notin \mathbb{E}.$$

**Beweis**

Wir zeigen: $\text{HALT} \leq_m L_{\mathcal{U}}$

$f : \langle M, w \rangle \to \langle M' \rangle$

Sei $f_0$ die überall undefinierte Funktion

**1. Fall:** $f_0 \notin \mathcal{U}$

wir wählen $f_u \in \mathcal{U}$ und TM $M_u$ die $f_u$ berechnet (existiert, da $\mathcal{U} \neq \emptyset$)

Die Reduktion arbeitet wie folgt: Zu einem gegebenen $\langle M, w \rangle$ konstruieren wir folgende Maschine $M'$

$M'(z)$
1. Simuliere $M(w)$
2. Simuliere $M_u(z)$

## Beweis des Rekursionstheorems

**(1)** print$(w)$ TM-Programm, welches $\#w$ hinter die Eingabe druckt

**(2)** seq$(\langle M_1, M_2, M_3\rangle) := \langle M'\rangle$
mit $M'(z)$ führt zuerst $M_1(z)$ aus und danach $M_2$ mit dem, was gerade auf dem Band steht, danach $M_3$

- $R$ enthält 3 Teile: A, B, T – Teil T stimmt mit TM $T$ überein

$R(y)$
1. Drucke $\#\langle B\rangle\#\langle T\rangle$ hinter Eingabe aufs Band
2. Sichere $y, \langle B\rangle$ und $\langle T\rangle$
3. Berechne print$(\#\langle B\rangle\#\langle T\rangle) = \langle A\rangle$
4. Berechne seq$(\langle A\rangle, \langle B\rangle, \langle T\rangle) = \langle R\rangle$
5. Drucke $\langle R, y\rangle$ aufs Band
6. führe $T(\langle R\rangle, y)$ aus

Teil A

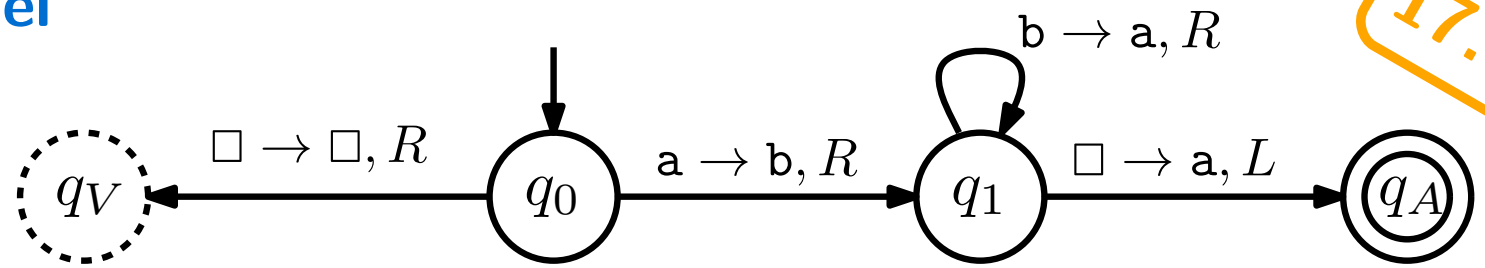
Teil B

Teil T

- $R =$ seq$(A, B, T)$
- $R(y)$ arbeitet wie $T(\langle R\rangle, y)$

16. Vorlesung

## Beispiel




## Dominotypen (für Eingabe a)

| $\#q_0\mathtt{a}\#$ | $\#$ | a | b | $q_0$ | $q_1$ | $q_2$ | $q_A$ | $\square$ | $\mathtt{b}q_1$ | $\square q_V$ | $\mathtt{a}q_1$ | $q_A\mathtt{aa}$ | $q_A\mathtt{ba}$ | $q_A\square\mathtt{a}$ |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| $\#$ | $\#$ | a | b | $q_0$ | $q_1$ | $q_2$ | $q_A$ | $\square$ | $q_0\mathtt{a}$ | $q_0\square$ | $q_1\mathtt{b}$ | $\mathtt{a}q_1\square$ | $\mathtt{b}q_1\square$ | $\square q_1\square$ |

Startdomino — Kopierdominos — Übergangsdominos

| $\#\square q_0$ | $\#\square q_1$ | $q_0\square\#$ | $q_1\square\#$ | $q_A$ | $q_A$ | $q_A$ | $q_A$ | $q_A$ | $q_A$ | $\#$ |
|---|---|---|---|---|---|---|---|---|---|---|
| $\#q_0$ | $\#q_1$ | $q_0\#$ | $q_1\#$ | $q_A\square$ | $q_A\mathtt{a}$ | $q_A\mathtt{b}$ | $\square q_A$ | $\mathtt{a}q_A$ | $\mathtt{b}q_A$ | $q_A\#$ |

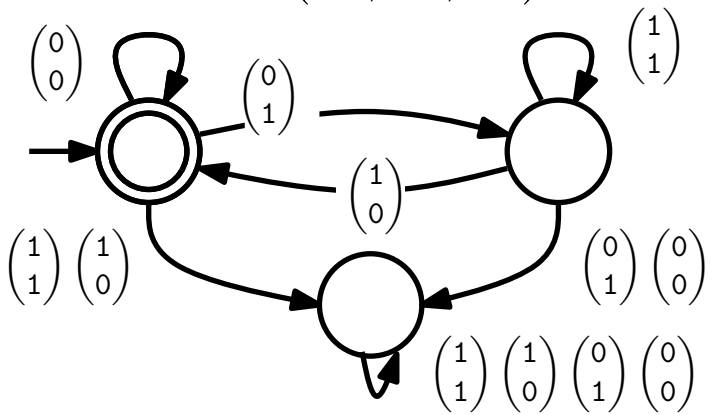
Übergangsdominos (Rand) — Pumpdominos

| $\#q_0\mathtt{a}\#$ | $\mathtt{b}q_1$ | $\#$ | b | $q_1\square\#$ | $q_A\mathtt{ba}$ | $\#$ | $q_A$ | a | $\#$ | $q_A$ | $\#$ | $\$$ |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| $\#$ | $q_0\mathtt{a}$ | $\#$ | b | $q_1\#$ | $\mathtt{b}q_1\square$ | $\#$ | $q_A\mathtt{b}$ | a | $\#$ | $q_A\mathtt{a}$ | $\#$ | $q_A\#\$$ |

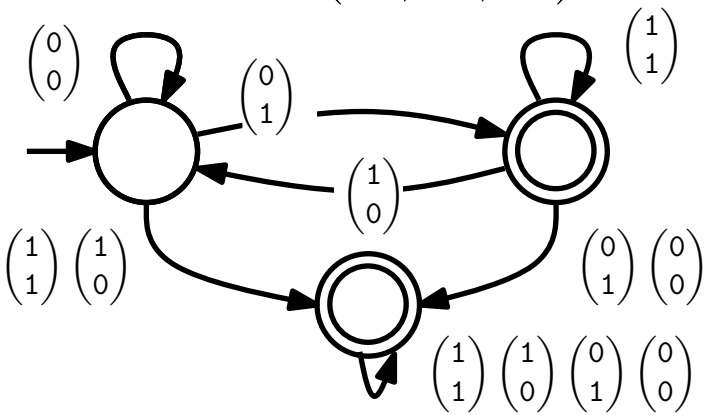
Lösungswort: $\#q_0\mathtt{a}\#\mathtt{b}q_1\#\mathtt{b}q_1\square\#q_A\mathtt{ba}\#q_A\mathtt{a}\#q_A\#\$$

**Bsp.** $\phi = \forall x_1 \exists x_2 \neg \text{PLUS}(x_2, x_2, x_1)$

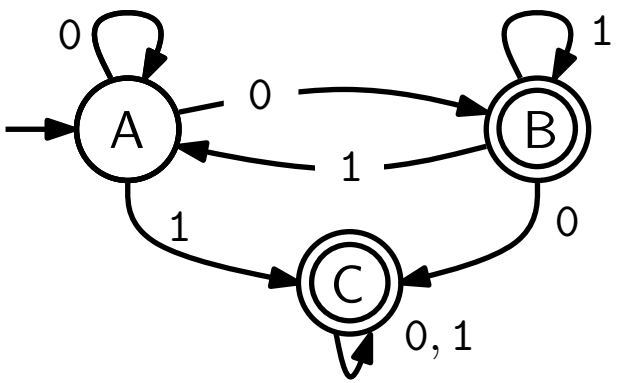
DEA zu $\text{PLUS}(x_2, x_2, x_1)$



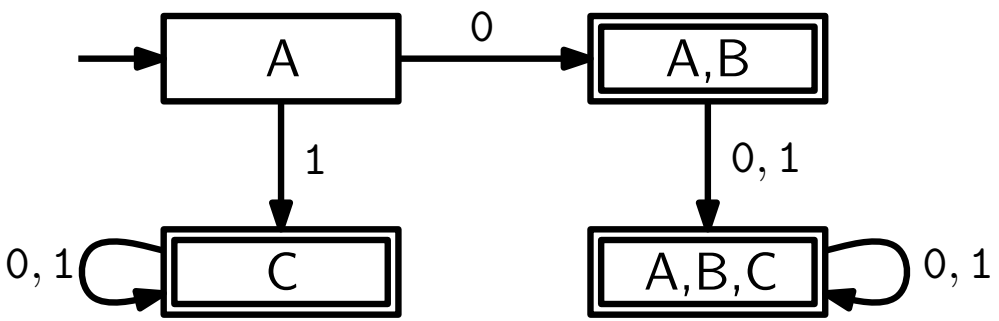

DEA zu $\neg\text{PLUS}(x_2, x_2, x_1)$




NEA zu $\exists x_2 \neg \text{PLUS}(x_2, x_2, x_1)$




DEA zu $\exists x_2 \neg \text{PLUS}(x_2, x_2, x_1)$




- $x_1$ ist mit $\forall$ quantifiziert, also testen wir ob alle mit $w \in \Sigma^+$ erreichbaren Zustände akzeptierend sind
- das stimmt, also ist $\phi$ wahr

18. Vorlesung

## Nachbetrachtung zum Satz 36

- Ist es nicht widersprüchlich, das wir beweisen konnten, dass e Aussage "wahr aber nicht beweisbar" ist?

$\rightarrow$ Beweis der Nichtbeweisbarkeit innerhalb der Erweiterung von $\mathrm{Th}(\mathbf{N}, +, \cdot)$ wurde innerhalb eines anderen Systems erbracht

## Formulierung als Unvollständigkeitssatz

- System $X$ heißt **konsistent**, gdw. $\neg\exists\phi[\phi \in X \wedge \neg\phi \in X]$
- System $X$ heißt **vollständig**, gdw. $\forall\phi[\phi \in X \vee \neg\phi \in X]$
- **vollständig und konsistent**:
  $\forall\phi[(\phi \in X \vee \neg\phi \in X) \wedge (\phi \notin X \vee \neg\phi \notin X)]$

widerspricht

**Gödels 1. Unvollständigkeitssatz**

Jedes axiomatische Beweissytem $X$, in welchem sich der Begriff des Beweis für Aussagen aus $\mathrm{Th}(\mathbf{N}, +, \cdot)$ formalisieren lässt, ist entweder konsistent oder vollständig.

- Satz 35: Es gibt wahre Aussagen in $X$, die nicht beweisbar sind. Unter Annahme $\forall\phi[\phi \in X \vee \phi \notin X]$ gilt $\exists\phi[\phi \notin X \wedge \neg\phi \notin X]$

**Unvollständigkeit**

# Die Klasse NP

- NP beinhaltet die Probleme, deren Lösung sich leicht verifizieren lässt
- **Def.:** Ein **Verifizierer für eine Sprache** $L$ ist eine TM für die gilt:
  $L = \{w \mid \exists z \in \Sigma^* : \langle w, z \rangle$ wird von $V$ akzeptiert$\}$

$z$ heißt Zeuge oder Zertifikat

Laufzeit von $V$ wird bzgl. $\langle w, z \rangle$ gemessen

- **Def.:** Ein Verifizierer $V$ für $L$ heißt **polynomiell**, gdw.
  (1) $\exists k$, sodass $L = \{w \mid \exists z \in \Sigma^* : |z| \leq |w|^k$ und $\langle w, z \rangle \in L(V)\}$
  (2) $V$ hat polynomielle Laufzeit

**Definition**

Die Komplexitätsklasse NP umfasst alle Sprachen, für die es einen polynomiellen Verifizierer gibt.

- Wenn $L$ aus P dann ist der polynomielle Entscheider für $L$ auch ein polynomieller Verifizierer für $L$ (er ignoriert den Zeugen)

**P und NP**

## Beweis

- $N$ ist polyzeit Halbband NTM für $L$ mit Laufzeit $t_N(n) \leq n^k - 3$
- Wir notieren den Lauf (Pfad im Berechnungsbaum) von $N(w)$ als **Tableau**

**Tableau:**

- $n^k \times n^k$ Tabelle
- Zellen der 1. und letzte Spalte enthalten #
- 1. Zeile enthält # Startkonfiguration + Blanks + #
- Zeile = Konfiguration des kodierten Laufes
- $i$te Zeile ist eine Folgekonfiguration der $(i-1)$en Zeile
- tritt eine akzeptierende Konfiguration auf, enthalten alle folgendes Zeilen eine Kopie dieser Zeile

**Bsp.**

| | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|
| # | $q_0$ | a | b | b | b | □ | □ | □ | □ | □ | # |
| # | b | $q_3$ | b | b | b | □ | □ | □ | □ | □ | # |

$(q_3, \mathtt{b}, R) \in \delta(q_0, \mathtt{a})$

| | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|
| # | a | a | b | b | b | $q_A$ | a | □ | □ | □ | # |
| # | a | a | b | b | b | $q_A$ | a | □ | □ | □ | # |

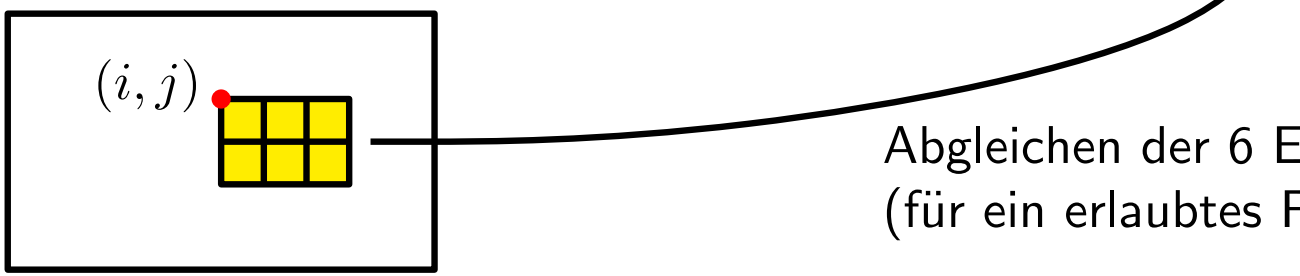
21. Vorlesung

**SAT ist NP-vollständig**

## ④ $\phi_{\text{trans}}$ sichert Konfigurationsübergänge zwischen Zeilen

$$\phi_{\text{trans}} = \bigwedge_{i,j} \bigvee_{a_1,\ldots,a_6 \text{ erl. Fenster}} (x_{i,j,a_1} \wedge x_{i+1,j,a_2} \cdots \wedge x_{i+2,j+1,a_6})$$

Für jedes Paar $(i, j)$ mit $i \leq n^k - 3$ und $i \leq n^k - 2$




Abgleichen der 6 Einträge des Fensters
(für ein erlaubtes Fenster)

22. Vorlesung

- wenn wir die Tabelle mit erlaubten Fenstern überdecken können, handelt es sich um ein korrekt ausgefülltes Tableau

---

- Also: Wenn $\phi$ erfüllbar ist, existiert ein korrekt ausgefülltes akz. Tableau, und somit auch ein akzeptierender Lauf für $N(w)$
- Weil $N$ polyzeit beschränkt, reicht es aus die Tableaubreite $n^k$ zu wählen

**SAT ist NP-vollständig**

**CLIQUE**
Eingabe: Graph $G$ und natürliche Zahl $k$
Frage: Hat $G$ eine $k$-Clique (d.h., $K_k$ als Teilgraph)?

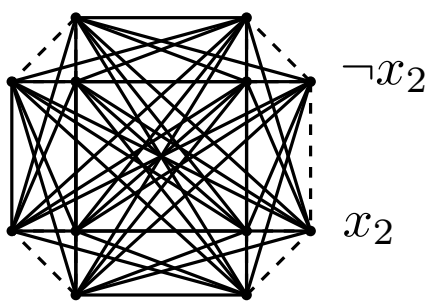
23. Vorlesung

**Satz 42**

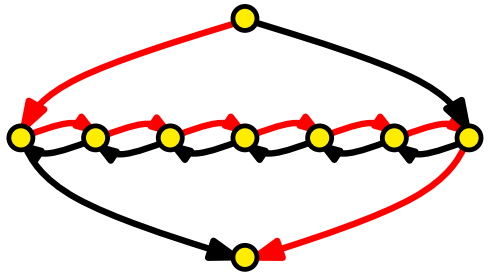
CLIQUE ist NP-vollständig.

**Beweis**

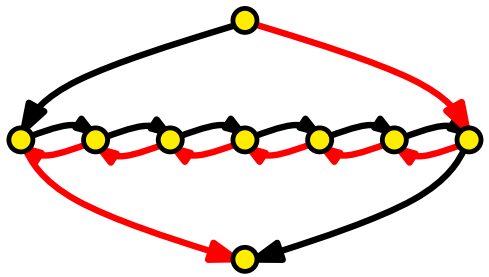
- CLIQUE$\in$ NP (Zeuge ist Knotenliste der Clique)
- wir zeigen NP-Schwerheit durch 3SAT$\leq_p$CLIQUE
- Reduktion wandelt Formel $\phi$ in 3-CNF in Graph $G$ und Zahl $k$ um
- $G$ ist wie folgt definiert:
  - ein Knoten pro Auftreten eines Literals in $\phi$
  - zwischen Knoten von Literalen einer Klausel gibt es keine Kanten
  - zwischen Knoten $x_i/\neg x_i$ gibt es keine Kanten
  - ansonsten sind alle Knotenpaare durch eine Kante verbunden
- $k$ ist die Anzahl der Klauseln in $\phi$



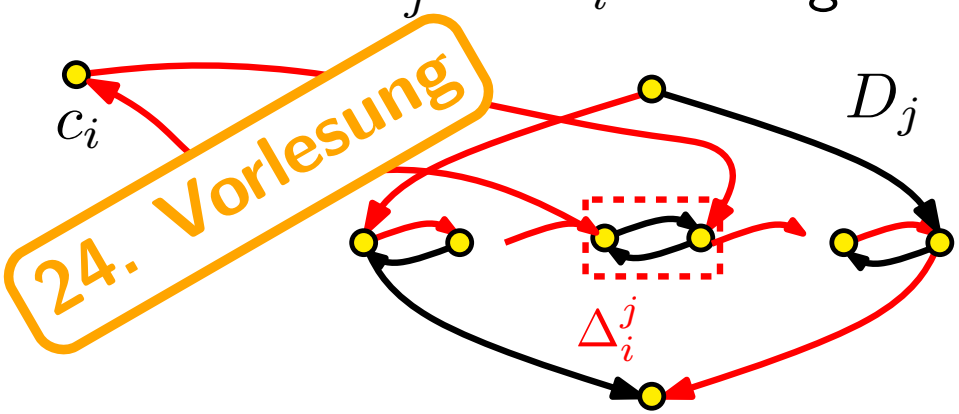
Ⓐ Wenn $x_i = 1$ durchlaufe $D_i$ wie folgt



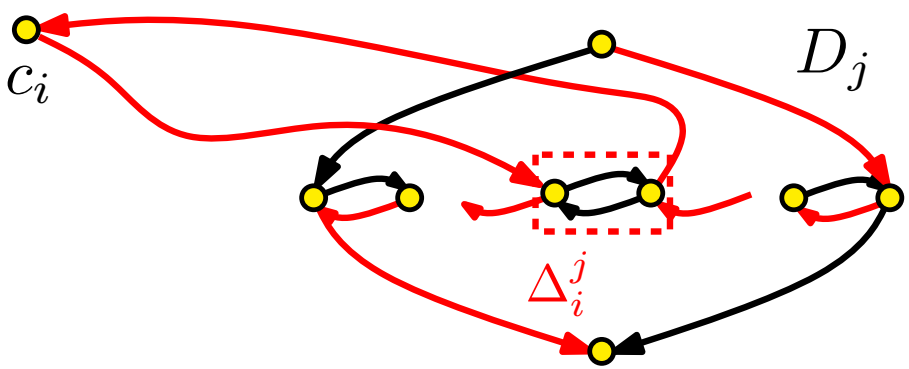
Ⓑ Wenn $x_i = 0$ durchlaufe $D_i$ wie folgt



- für jede Klauel $C_i$ wähle ein Literal $x_i/\neg x_i$, welches $C_i$ erfüllt
- verbinde $D_j$ mit $c_i$ wie folgt




24. Vorlesung




$C_i$ wird durch $x_j$ erfüllt

$C_i$ wird durch $\neg x_j$ erfüllt

- der so erzeugte Kantenzug ist ein (gerichteter) Hamiltonkreis

**TSP ist NP-vollständig**

# Rückblick: Zentrale Themen



- Wie definiert man einen Computer formal?

  ⇝ Automatentheorie und Turingmaschinen

- Gibt es Probleme, die ein Computer nicht lösen kann? Falls ja, können wir eins finden?

  ⇝ Berechenbarkeitstheorie

- Manche Probleme, die ein Computer lösen kann sind leicht, andere schwer. Wie können wir Probleme klassifizieren?

  ⇝ Komplexitätstheorie