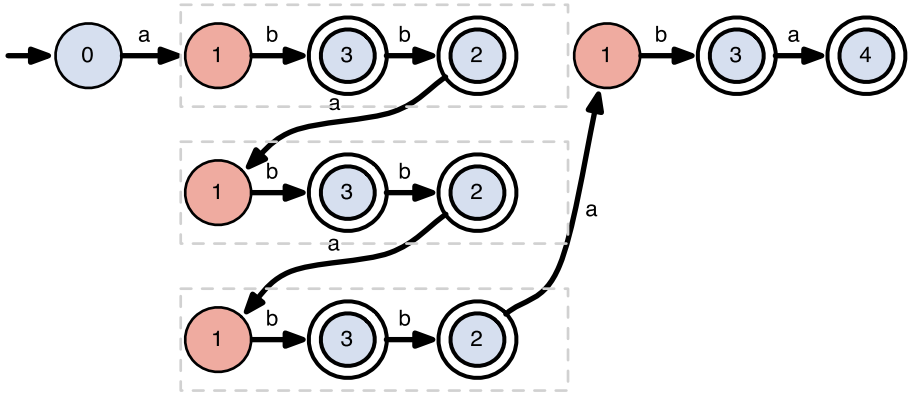
Berechenbarkeitstheorie

17. Vorlesung



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

Letztes Mal:

Letztes Mal:

Es gibt eine Quine-Turingmaschine S:
S gibt bei jeder Eingabe $\langle S \rangle$ aus.

Letztes Mal:

Es gibt eine Quine-Turingmaschine S:
S gibt bei jeder Eingabe $\langle S \rangle$ aus.

Rekursionstheorem

Sei T eine TM , welche $t\colon \Sigma^* \times \Sigma^* \to \Sigma^*$ berechnet. Dann existiert eine durch die TM R beschriebene berechenbare Funktion $r\colon \Sigma^* \to \Sigma^*$, so dass für alle Eingaben $w \in \Sigma^*$ gilt

$$r(w) = t(\langle R \rangle, w).$$

Letztes Mal:

Es gibt eine Quine-Turingmaschine S:
S gibt bei jeder Eingabe $\langle S \rangle$ aus.

Rekursionstheorem

Sei T eine TM , welche $t\colon \Sigma^* \times \Sigma^* \to \Sigma^*$ berechnet. Dann existiert eine durch die TM R beschriebene berechenbare Funktion $r\colon \Sigma^* \to \Sigma^*$, so dass für alle Eingaben $w \in \Sigma^*$ gilt

$$r(w) = t(\langle R \rangle, w).$$

Interpretation und Anwendung

Letztes Mal:

Es gibt eine Quine-Turingmaschine S:
S gibt bei jeder Eingabe $\langle S \rangle$ aus.

Rekursionstheorem

Sei T eine TM , welche $t\colon \Sigma^* \times \Sigma^* \to \Sigma^*$ berechnet. Dann existiert eine durch die TM R beschriebene berechenbare Funktion $r\colon \Sigma^* \to \Sigma^*$, so dass für alle Eingaben $w \in \Sigma^*$ gilt

$$r(w) = t(\langle R \rangle, w).$$

Interpretation und Anwendung

- wir können für eine TM T annehmen, dass $\langle T \rangle$ Teil der Eingabe war und für uns verfügbar ist

Letztes Mal:

Es gibt eine Quine-Turingmaschine S:
S gibt bei jeder Eingabe $\langle S \rangle$ aus.

Rekursionstheorem

Sei T eine TM , welche $t\colon \Sigma^* \times \Sigma^* \to \Sigma^*$ berechnet. Dann existiert eine durch die TM R beschriebene berechenbare Funktion $r\colon \Sigma^* \to \Sigma^*$, so dass für alle Eingaben $w \in \Sigma^*$ gilt

$$r(w) = t(\langle R \rangle, w).$$

Interpretation und Anwendung

- wir können für eine TM T annehmen, dass $\langle T \rangle$ Teil der Eingabe war und für uns verfügbar ist
- dann gibt es eine TM R, die genauso arbeitet, ohne dass $\langle R \rangle$ als Eingabe übergeben wurde

Letztes Mal:

Es gibt eine Quine-Turingmaschine S:
S gibt bei jeder Eingabe $\langle S \rangle$ aus.

Rekursionstheorem

Sei T eine TM , welche $t\colon \Sigma^* \times \Sigma^* \to \Sigma^*$ berechnet. Dann existiert eine durch die TM R beschriebene berechenbare Funktion $r\colon \Sigma^* \to \Sigma^*$, so dass für alle Eingaben $w \in \Sigma^*$ gilt

$$r(w) = t(\langle R \rangle, w).$$

Interpretation und Anwendung

- wir können für eine TM T annehmen, dass $\langle T \rangle$ Teil der Eingabe war und für uns verfügbar ist
- dann gibt es eine TM R, die genauso arbeitet, ohne dass $\langle R \rangle$ als Eingabe übergeben wurde
- wir können immer annehmen, dass eine TM über ein Unterprogramm `getYourOwnCode` verfügt

Beweis des Rekursionstheorems

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

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

Beweis des Rekursionstheorems

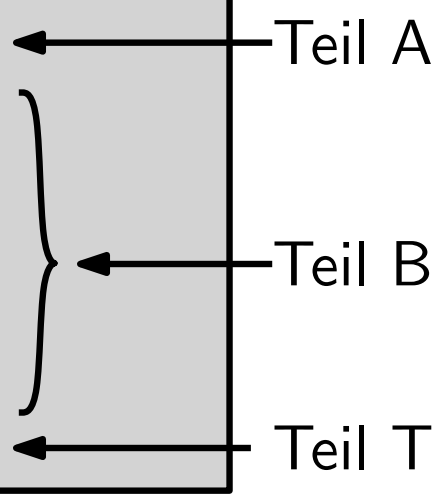
(1) print(w) TM-Programm, welches $\#w$ hinter die Eingabe druckt

(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



Beweis des Rekursionstheorems

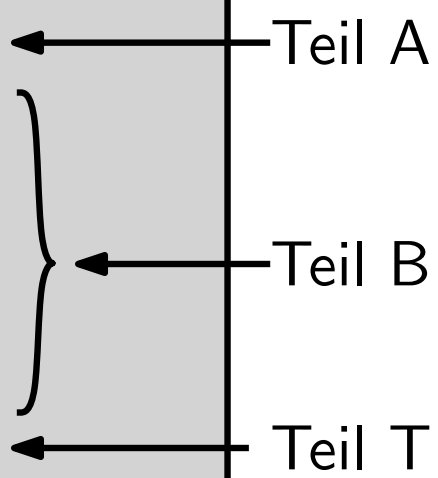
(1) print(w) TM-Programm, welches $\#w$ hinter die Eingabe druckt

(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



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

Alternativer Beweis zu $A_{\mathsf{TM}} \notin \mathbb{E}$

- angenommen es gibt einen Entscheider H für A_{TM}

Alternativer Beweis zu $A_{\mathsf{TM}} \notin \mathbb{E}$

- angenommen es gibt einen Entscheider H für A_{TM}
- wir konstruieren auf H basierend, die Anti-Simulationsmaschine B

Alternativer Beweis zu $A_{\mathsf{TM}} \notin \mathbb{E}$

- angenommen es gibt einen Entscheider H für A_{TM}
- wir konstruieren auf H basierend, die Anti-Simulationsmaschine B

$B(z)$
1. Sichere z
2. Bestimme $\langle B \rangle$
3. Simuliere $H(\langle B, z \rangle)$
4. Akzeptiere, wenn Simulation verwirft, ansonsten verwerfe

Alternativer Beweis zu $A_{\mathsf{TM}} \notin \mathbb{E}$

- angenommen es gibt einen Entscheider H für A_{TM}
- wir konstruieren auf H basierend, die Anti-Simulationsmaschine B

$B(z)$
1. Sichere z
2. Bestimme $\langle B \rangle$
3. Simuliere $H(\langle B, z \rangle)$
4. Akzeptiere, wenn Simulation verwirft, ansonsten verwerfe

- $B(z)$ akzeptiert $\iff$ $H(\langle B, z \rangle)$ verwirft $\iff$ $B(z)$ verwirfft

Alternativer Beweis zu $A_{\mathsf{TM}} \notin \mathbb{E}$

- angenommen es gibt einen Entscheider H für A_{TM}
- wir konstruieren auf H basierend, die Anti-Simulationsmaschine B

$B(z)$
1. Sichere z
2. Bestimme $\langle B \rangle$
3. Simuliere $H(\langle B, z \rangle)$
4. Akzeptiere, wenn Simulation verwirft, ansonsten verwerfe

- $B(z)$ akzeptiert $\iff$ $H(\langle B, z \rangle)$ verwirft $\iff$ $B(z)$ verwirfft
- Widerspruch zur Annahme $\rightarrow$ Entscheider H für A_{TM} kann nicht existieren

Alternativer Beweis zu $A_{\mathsf{TM}} \notin \mathbb{E}$

- angenommen es gibt einen Entscheider H für A_{TM}
- wir konstruieren auf H basierend, die Anti-Simulationsmaschine B

$B(z)$
1. Sichere z
2. Bestimme $\langle B \rangle$
3. Simuliere $H(\langle B, z \rangle)$
4. Akzeptiere, wenn Simulation verwirft, ansonsten verwerfe

- $B(z)$ akzeptiert $\iff$ $H(\langle B, z \rangle)$ verwirft $\iff$ $B(z)$ verwirfft
- Widerspruch zur Annahme $\rightarrow$ Entscheider H für A_{TM} kann nicht existieren
- A_{TM} ist nicht entscheidbar

Def.: Eine TM M heißt **minimal**, gdw. es keine äquivalente TM M' gibt mit $|\langle M' \rangle| < |\langle M \rangle|$

Satz 26

$$\text{MIN}_{\text{TM}} = \{\langle M \rangle \mid M \text{ ist minimal } \} \notin \mathbb{A}$$

Def.: Eine TM M heißt **minimal**, gdw. es keine äquivalente TM M' gibt mit $|\langle M' \rangle| < |\langle M \rangle|$

Satz 26

$$\text{MIN}_{\text{TM}} = \{\langle M \rangle \mid M \text{ ist minimal } \} \notin \mathbb{A}$$

Beweis

Def.: Eine TM M heißt **minimal**, gdw. es keine äquivalente TM M' gibt mit $|\langle M' \rangle| < |\langle M \rangle|$

Satz 26

$$\text{MIN}_{\text{TM}} = \{\langle M \rangle \mid M \text{ ist minimal } \} \notin \mathbb{A}$$

Beweis

- Annahme: Es gibt Aufzähler E für MIN_{TM}

Def.: Eine TM M heißt **minimal**, gdw. es keine äquivalente TM M' gibt mit $|\langle M' \rangle| < |\langle M \rangle|$

Satz 26

$$\text{MIN}_{\text{TM}} = \{\langle M \rangle \mid M \text{ ist minimal } \} \notin \mathbb{A}$$

Beweis

- Annahme: Es gibt Aufzähler E für MIN_{TM}
- Konstruiere TM C wie folgt

$C(z)$

1. Bestimme $\langle C \rangle$
2. Bestimme erste TM D in der Aufzählung von E mit $|\langle D \rangle| > |\langle C \rangle|$
3. Simuliere $D(z)$

Def.: Eine TM M heißt **minimal**, gdw. es keine äquivalente TM M' gibt mit $|\langle M' \rangle| < |\langle M \rangle|$

Satz 26

$$\text{MIN}_{\text{TM}} = \{\langle M \rangle \mid M \text{ ist minimal } \} \notin \mathbb{A}$$

Beweis

- Annahme: Es gibt Aufzähler E für MIN_{TM}
- Konstruiere TM C wie folgt

$C(z)$
1. Bestimme $\langle C \rangle$
2. Bestimme erste TM D in der Aufzählung von E mit $|\langle D \rangle| > |\langle C \rangle|$
3. Simuliere $D(z)$

- es gibt beliebig lange minimale TM $\rightarrow$ Suche nach D erfolgreich

Def.: Eine TM M heißt **minimal**, gdw. es keine äquivalente TM M' gibt mit $|\langle M' \rangle| < |\langle M \rangle|$

Satz 26

$$\text{MIN}_{\text{TM}} = \{\langle M \rangle \mid M \text{ ist minimal } \} \notin \mathbb{A}$$

Beweis

- Annahme: Es gibt Aufzähler E für MIN_{TM}
- Konstruiere TM C wie folgt

$C(z)$
1. Bestimme $\langle C \rangle$
2. Bestimme erste TM D in der Aufzählung von E mit $|\langle D \rangle| > |\langle C \rangle|$
3. Simuliere $D(z)$

- es gibt beliebig lange minimale TM $\rightarrow$ Suche nach D erfolgreich
- C arbeitet wie D

Def.: Eine TM M heißt **minimal**, gdw. es keine äquivalente TM M' gibt mit $|\langle M' \rangle| < |\langle M \rangle|$

Satz 26

$$\text{MIN}_{\text{TM}} = \{\langle M \rangle \mid M \text{ ist minimal } \} \notin \mathbb{A}$$

Beweis

- Annahme: Es gibt Aufzähler E für MIN_{TM}
- Konstruiere TM C wie folgt

$C(z)$
1. Bestimme $\langle C \rangle$
2. Bestimme erste TM D in der Aufzählung von E mit $|\langle D \rangle| > |\langle C \rangle|$
3. Simuliere $D(z)$

- es gibt beliebig lange minimale TM $\rightarrow$ Suche nach D erfolgreich
- C arbeitet wie D
- C hat aber kürzere Kodierung als minmale TM D

Def.: Eine TM M heißt **minimal**, gdw. es keine äquivalente TM M' gibt mit $|\langle M' \rangle| < |\langle M \rangle|$

Satz 26

$$\text{MIN}_{\text{TM}} = \{\langle M \rangle \mid M \text{ ist minimal } \} \notin \mathbb{A}$$

Beweis

- Annahme: Es gibt Aufzähler E für MIN_{TM}
- Konstruiere TM C wie folgt

$C(z)$
1. Bestimme $\langle C \rangle$
2. Bestimme erste TM D in der Aufzählung von E mit $|\langle D \rangle| > |\langle C \rangle|$
3. Simuliere $D(z)$

- es gibt beliebig lange minimale TM $\rightarrow$ Suche nach D erfolgreich
- C arbeitet wie D
- C hat aber kürzere Kodierung als minmale TM D
- Widerspruch zur Annahme $\rightarrow$ E kann nicht existieren

□

Das Postsche Korrespondenz-Problem (PKP)

Das Postsche Korrespondenz-Problem (PKP)

PKP Definition

Eingabe: Folge von ℓ Wörterpaaren (x_i, y_i).

Frage: Existiert eine nicht-leere Folge $(i_1, i_2, \ldots, i_k)$ mit $x_{i_1} x_{i_2} \cdots x_{i_k} = y_{i_1} y_{i_2} \cdots y_{i_k}$?

Das Postsche Korrespondenz-Problem (PKP)

PKP Definition

Eingabe: Folge von ℓ Wörterpaaren (x_i, y_i).

Frage: Existiert eine nicht-leere Folge $(i_1, i_2, \ldots, i_k)$ mit $x_{i_1} x_{i_2} \cdots x_{i_k} = y_{i_1} y_{i_2} \cdots y_{i_k}$?

- die Wörterpaare (x_i, y_i) heißen auch Dominotypen

Das Postsche Korrespondenz-Problem (PKP)

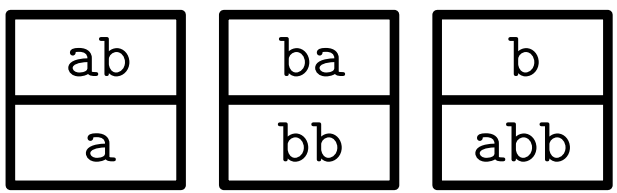
PKP Definition

Eingabe: Folge von ℓ Wörterpaaren (x_i, y_i).

Frage: Existiert eine nicht-leere Folge $(i_1, i_2, \ldots, i_k)$ mit $x_{i_1} x_{i_2} \cdots x_{i_k} = y_{i_1} y_{i_2} \cdots y_{i_k}$?

- die Wörterpaare (x_i, y_i) heißen auch Dominotypen

Darstellung der Eingabe:
$((\mathtt{ab}, \mathtt{a}), (\mathtt{ba}, \mathtt{bb}), (\mathtt{b}, \mathtt{abb}))$



Das Postsche Korrespondenz-Problem (PKP)

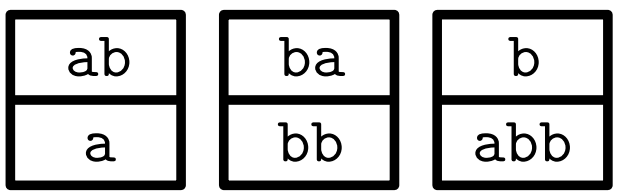
PKP Definition

Eingabe: Folge von ℓ Wörterpaaren (x_i, y_i).

Frage: Existiert eine nicht-leere Folge $(i_1, i_2, \ldots, i_k)$ mit $x_{i_1} x_{i_2} \cdots x_{i_k} = y_{i_1} y_{i_2} \cdots y_{i_k}$?

- die Wörterpaare (x_i, y_i) heißen auch Dominotypen

  Darstellung der Eingabe:
  $((\mathtt{ab}, \mathtt{a}), (\mathtt{ba}, \mathtt{bb}), (\mathtt{b}, \mathtt{abb}))$



- Lösung zur Frage heißt PKP-Folge. Das gebildete Wort Lösungswort.

Das Postsche Korrespondenz-Problem (PKP)

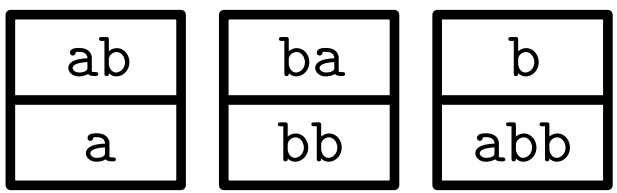
PKP Definition

Eingabe: Folge von ℓ Wörterpaaren (x_i, y_i).

Frage: Existiert eine nicht-leere Folge $(i_1, i_2, \ldots, i_k)$ mit $x_{i_1} x_{i_2} \cdots x_{i_k} = y_{i_1} y_{i_2} \cdots y_{i_k}$?

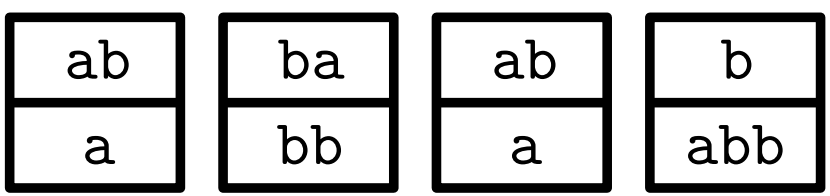
- die Wörterpaare (x_i, y_i) heißen auch Dominotypen

  Darstellung der Eingabe:
  $((\mathtt{ab}, \mathtt{a}), (\mathtt{ba}, \mathtt{bb}), (\mathtt{b}, \mathtt{abb}))$



- Lösung zur Frage heißt PKP-Folge. Das gebildete Wort Lösungswort.

  Im Beispiel:
  (Wiederholung möglich)



Das Postsche Korrespondenz-Problem (PKP)

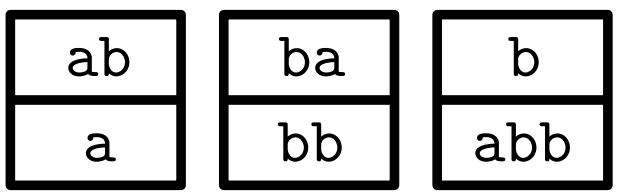
PKP Definition

Eingabe: Folge von ℓ Wörterpaaren (x_i, y_i).

Frage: Existiert eine nicht-leere Folge $(i_1, i_2, \ldots, i_k)$ mit $x_{i_1} x_{i_2} \cdots x_{i_k} = y_{i_1} y_{i_2} \cdots y_{i_k}$?

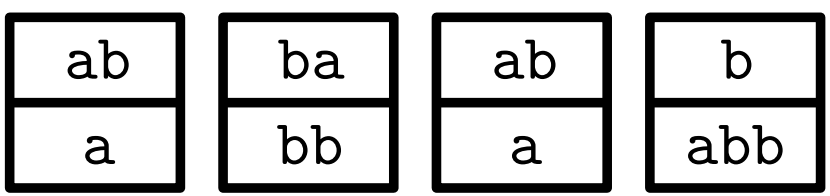
- die Wörterpaare (x_i, y_i) heißen auch Dominotypen

  Darstellung der Eingabe:
  $((\mathtt{ab}, \mathtt{a}), (\mathtt{ba}, \mathtt{bb}), (\mathtt{b}, \mathtt{abb}))$

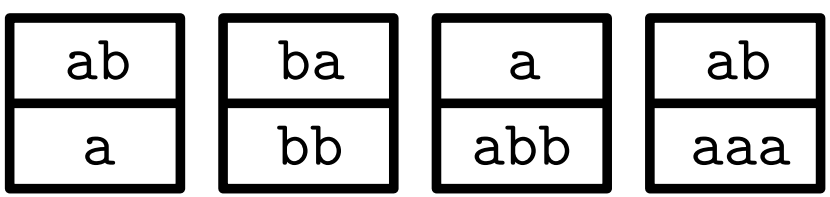


- Lösung zur Frage heißt PKP-Folge. Das gebildete Wort Lösungswort.

  Im Beispiel:
  (Wiederholung möglich)



- PKP Instanz ohne Lösung:



Das Postsche Korrespondenz-Problem (PKP)

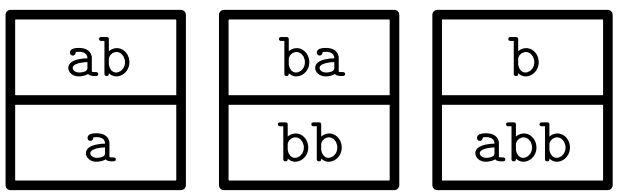
PKP Definition

Eingabe: Folge von ℓ Wörterpaaren (x_i, y_i).

Frage: Existiert eine nicht-leere Folge $(i_1, i_2, \ldots, i_k)$ mit $x_{i_1} x_{i_2} \cdots x_{i_k} = y_{i_1} y_{i_2} \cdots y_{i_k}$?

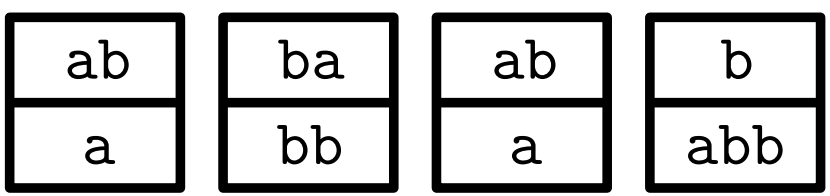
- die Wörterpaare (x_i, y_i) heißen auch Dominotypen

  Darstellung der Eingabe:
  $((\mathtt{ab}, \mathtt{a}), (\mathtt{ba}, \mathtt{bb}), (\mathtt{b}, \mathtt{abb}))$

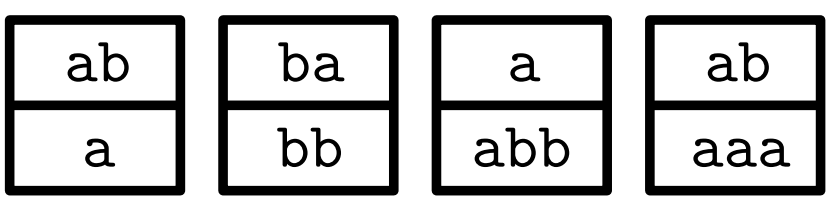


- Lösung zur Frage heißt PKP-Folge. Das gebildete Wort Lösungswort.

  Im Beispiel:
  (Wiederholung möglich)



- PKP Instanz ohne Lösung:
  (Kein Abschluss möglich)



- **Modikiziertes PKP (MPKP):** Wie PKP nur, dass man fordert, dass mit dem ersten Dominotyp begonnen werden muss!

$$\text{PKP} = \{\langle \mathcal{S} \rangle \mid \exists \text{ PKP-Folge für } \mathcal{S}\}$$
$$\text{MPKP} = \{\langle \mathcal{S} \rangle \mid \exists \text{ PKP-Folge für } \mathcal{S} \text{ mit } i_1 = 1 \text{ beginnend}\}$$

- **Modikiziertes PKP (MPKP):** Wie PKP nur, dass man fordẽrt, dass mit dem ersten Dominotyp begonnen werden muss!

$$\text{PKP} = \{\langle \mathcal{S} \rangle \mid \exists \text{ PKP-Folge für } \mathcal{S}\}$$
$$\text{MPKP} = \{\langle \mathcal{S} \rangle \mid \exists \text{ PKP-Folge für } \mathcal{S} \text{ mit } i_1 = 1 \text{ beginnend}\}$$

Satz 27

$$\text{MPKP} \leq_m \text{PKP}$$

- **Modikiziertes PKP (MPKP):** Wie PKP nur, dass man fordert, dass mit dem ersten Dominotyp begonnen werden muss!

$$\text{PKP} = \{\langle \mathcal{S} \rangle \mid \exists \text{ PKP-Folge für } \mathcal{S}\}$$
$$\text{MPKP} = \{\langle \mathcal{S} \rangle \mid \exists \text{ PKP-Folge für } \mathcal{S} \text{ mit } i_1 = 1 \text{ beginnend}\}$$

Satz 27

$$\text{MPKP} \leq_m \text{PKP}$$

Beweis

- **Idee :** modifiziere MPKP Instanz so, dass es nur PKP-Folgen mit i_1 beginnend geben kann

- **Modikiziertes PKP (MPKP):** Wie PKP nur, dass man fordert, dass mit dem ersten Dominotyp begonnen werden muss!

$$\text{PKP} = \{\langle \mathcal{S} \rangle \mid \exists \text{ PKP-Folge für } \mathcal{S}\}$$
$$\text{MPKP} = \{\langle \mathcal{S} \rangle \mid \exists \text{ PKP-Folge für } \mathcal{S} \text{ mit } i_1 = 1 \text{ beginnend}\}$$

Satz 27

$$\text{MPKP} \leq_m \text{PKP}$$

Beweis

- **Idee :** modifiziere MPKP Instanz so, dass es nur PKP-Folgen mit i_1 beginnend geben kann
- **Plan :** Wenn (x_1, y_1) das einzige Paar mit gleichem Anfangszeichen, dann muss jede PKP-Folge mit i_1 beginnen

- **Modikiziertes PKP (MPKP):** Wie PKP nur, dass man fordert, dass mit dem ersten Dominotyp begonnen werden muss!

$$\text{PKP} = \{\langle \mathcal{S} \rangle \mid \exists \text{ PKP-Folge für } \mathcal{S}\}$$
$$\text{MPKP} = \{\langle \mathcal{S} \rangle \mid \exists \text{ PKP-Folge für } \mathcal{S} \text{ mit } i_1 = 1 \text{ beginnend}\}$$

Satz 27

$$\text{MPKP} \leq_m \text{PKP}$$

Beweis

- **Idee :** modifiziere MPKP Instanz so, dass es nur PKP-Folgen mit i_1 beginnend geben kann
- **Plan :** Wenn (x_1, y_1) das einzige Paar mit gleichem Anfangszeichen, dann muss jede PKP-Folge mit i_1 beginnen
- Ersteze (x_1, y_1) durch $(\#x_1, \#y_1)$ ($\#$ neues Zeichen)

- **Modikiziertes PKP (MPKP):** Wie PKP nur, dass man fordert, dass mit dem ersten Dominotyp begonnen werden muss!

$$\text{PKP} = \{\langle \mathcal{S} \rangle \mid \exists \text{ PKP-Folge für } \mathcal{S}\}$$
$$\text{MPKP} = \{\langle \mathcal{S} \rangle \mid \exists \text{ PKP-Folge für } \mathcal{S} \text{ mit } i_1 = 1 \text{ beginnend}\}$$

Satz 27

$$\text{MPKP} \leq_m \text{PKP}$$

Beweis

- **Idee :** modifiziere MPKP Instanz so, dass es nur PKP-Folgen mit i_1 beginnend geben kann
- **Plan :** Wenn (x_1, y_1) das einzige Paar mit gleichem Anfangszeichen, dann muss jede PKP-Folge mit i_1 beginnen
- Ersteze (x_1, y_1) durch $(\#x_1, \#y_1)$ ($\#$ neues Zeichen)

1. **Problem:** i_1 kann in PKP-Folge mehrfach auftauchen. Nach Modifikation ist die PKP-Folge nicht mehr möglich.

- **Modikiziertes PKP (MPKP):** Wie PKP nur, dass man fordert, dass mit dem ersten Dominotyp begonnen werden muss!

$$\text{PKP} = \{\langle \mathcal{S} \rangle \mid \exists \text{ PKP-Folge für } \mathcal{S}\}$$
$$\text{MPKP} = \{\langle \mathcal{S} \rangle \mid \exists \text{ PKP-Folge für } \mathcal{S} \text{ mit } i_1 = 1 \text{ beginnend}\}$$

Satz 27

$$\text{MPKP} \leq_m \text{PKP}$$

Beweis

- **Idee :** modifiziere MPKP Instanz so, dass es nur PKP-Folgen mit i_1 beginnend geben kann
- **Plan :** Wenn (x_1, y_1) das einzige Paar mit gleichem Anfangszeichen, dann muss jede PKP-Folge mit i_1 beginnen
- Ersteze (x_1, y_1) durch $(\#x_1, \#y_1)$ ($\#$ neues Zeichen)

1. **Problem:** i_1 kann in PKP-Folge mehrfach auftauchen. Nach Modifikation ist die PKP-Folge nicht mehr möglich.
2. **Problem:** Auch andere Paare können gleiches Anfangszeichen haben.

Verbesserte Idee

Verbesserte Idee

(1.) Modifiziere jedes Paar (x_i, y_i) zu (x'_i, y'_i) mit

- $x_i = a_1 a_2 a_3 \cdots a_n \longrightarrow x'_i = a_1 \# a_2 \# \cdots \# a_n \#$ $\quad (a_i \in \Sigma)$
- $y_i = b_1 b_2 b_3 \cdots b_m \longrightarrow y'_i = \# b_1 \# b_2 \# \cdots \# b_m$ $\quad (b_i \in \Sigma)$

Verbesserte Idee

(1.) Modifiziere jedes Paar (x_i, y_i) zu (x_i', y_i') mit

- $x_i = a_1 a_2 a_3 \cdots a_n \longrightarrow x_i' = a_1 \# a_2 \# \cdots \# a_n \#$ $\quad (a_i \in \Sigma)$
- $y_i = b_1 b_2 b_3 \cdots b_m \longrightarrow y_i' = \# b_1 \# b_2 \# \cdots \# b_m$ $\quad (b_i \in \Sigma)$

(2.) Füge als $(\ell+1)$tes Paar $(x_{\ell+1}, y_{\ell+1})$ hinzu mit

- $x_1 = a_1 a_2 a_3 \cdots a_n \longrightarrow x_{\ell+1} = \# a_1 \# a_2 \# \cdots \# a_n \#$ $\quad (a_i \in \Sigma)$
- $y_1 = b_1 b_2 b_3 \cdots b_m \longrightarrow y_{\ell+1} = \# b_1 \# b_2 \# \cdots \# b_m$ $\quad (b_i \in \Sigma)$

Verbesserte Idee

(1.) Modifiziere jedes Paar (x_i, y_i) zu (x'_i, y'_i) mit

- $x_i = a_1 a_2 a_3 \cdots a_n \longrightarrow x'_i = a_1 \# a_2 \# \cdots \# a_n \#$ $\quad (a_i \in \Sigma)$
- $y_i = b_1 b_2 b_3 \cdots b_m \longrightarrow y'_i = \# b_1 \# b_2 \# \cdots \# b_m$ $\quad (b_i \in \Sigma)$

(2.) Füge als $(\ell + 1)$tes Paar $(x_{\ell+1}, y_{\ell+1})$ hinzu mit

- $x_1 = a_1 a_2 a_3 \cdots a_n \longrightarrow x_{\ell+1} = \# a_1 \# a_2 \# \cdots \# a_n \#$ $\quad (a_i \in \Sigma)$
- $y_1 = b_1 b_2 b_3 \cdots b_m \longrightarrow y_{\ell+1} = \# b_1 \# b_2 \# \cdots \# b_m$ $\quad (b_i \in \Sigma)$

(3.) Füge als $(\ell + 2)$tes Paar $(\$, \#\$)$ hinzu ($\$ \notin \Sigma$)

Verbesserte Idee

(1.) Modifiziere jedes Paar (x_i, y_i) zu (x'_i, y'_i) mit

- $x_i = a_1a_2a_3 \cdots a_n \longrightarrow x'_i = a_1\#a_2\# \cdots \#a_n\# \quad (a_i \in \Sigma)$
- $y_i = b_1b_2b_3 \cdots b_m \longrightarrow y'_i = \#b_1\#b_2\# \cdots \#b_m \quad (b_i \in \Sigma)$

(2.) Füge als $(\ell+1)$tes Paar $(x_{\ell+1}, y_{\ell+1})$ hinzu mit

- $x_1 = a_1a_2a_3 \cdots a_n \longrightarrow x_{\ell+1} = \#a_1\#a_2\# \cdots \#a_n\# \quad (a_i \in \Sigma)$
- $y_1 = b_1b_2b_3 \cdots b_m \longrightarrow y_{\ell+1} = \#b_1\#b_2\# \cdots \#b_m \quad (b_i \in \Sigma)$

(3.) Füge als $(\ell+2)$tes Paar $(\$, \#\$)$ hinzu ($\$ \notin \Sigma$)

Nur das $(\ell+1)$te Paar beginnt mit gleichen Zeichen (es gibt aber auch eine Variante von (x_1, y_1) ohne gleiche 1. Zeichen)

Verbesserte Idee

(1.) Modifiziere jedes Paar (x_i, y_i) zu (x'_i, y'_i) mit

- $x_i = a_1a_2a_3\cdots a_n \longrightarrow x'_i = a_1\#a_2\#\cdots\#a_n\# \quad (a_i \in \Sigma)$
- $y_i = b_1b_2b_3\cdots b_m \longrightarrow y'_i = \#b_1\#b_2\#\cdots\#b_m \quad (b_i \in \Sigma)$

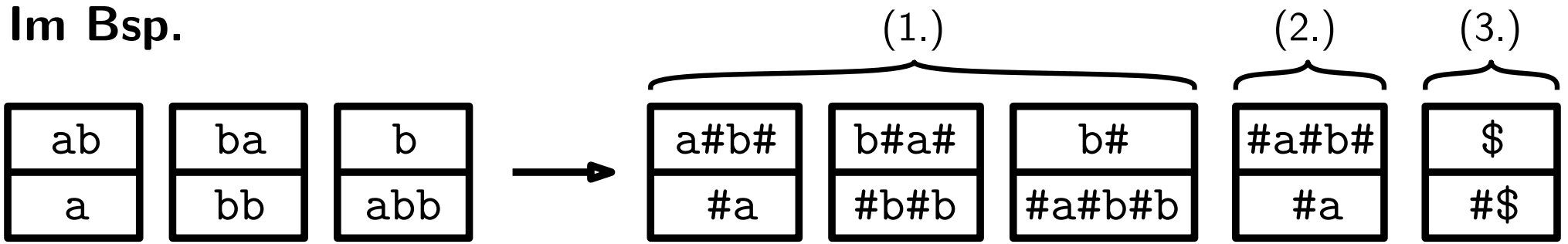
(2.) Füge als $(\ell+1)$tes Paar $(x_{\ell+1}, y_{\ell+1})$ hinzu mit

- $x_1 = a_1a_2a_3\cdots a_n \longrightarrow x_{\ell+1} = \#a_1\#a_2\#\cdots\#a_n\# \quad (a_i \in \Sigma)$
- $y_1 = b_1b_2b_3\cdots b_m \longrightarrow y_{\ell+1} = \#b_1\#b_2\#\cdots\#b_m \quad (b_i \in \Sigma)$

(3.) Füge als $(\ell+2)$tes Paar $(\$, \#\$)$ hinzu ($\$ \notin \Sigma$)

Nur das $(\ell+1)$te Paar beginnt mit gleichen Zeichen (es gibt aber auch eine Variante von (x_1, y_1) ohne gleiche 1. Zeichen)

Im Bsp.



Behauptung: Modifikation durch (1.) (2.) (3.) von $\mathcal{S}$ nach $f(\mathcal{S})$ ist Reduktion von MPKP auf PKP

Behauptung: Modifikation durch (1.) (2.) (3.) von $\mathcal{S}$ nach $f(\mathcal{S})$ ist Reduktion von MPKP auf PKP

1. $\mathcal{S} \in$ **MPKP** $\Rightarrow f(\mathcal{S}) \in$ **PKP**

Behauptung: Modifikation durch (1.) (2.) (3.) von $\mathcal{S}$ nach $f(\mathcal{S})$ ist Reduktion von MPKP auf PKP

1. $\mathcal{S} \in$ **MPKP** $\Rightarrow f(\mathcal{S}) \in$ **PKP**

- Sei $(1, i_2, i_3, \ldots, i_k)$ (M)PKP-Folge für $\mathcal{S}$

Behauptung: Modifikation durch (1.) (2.) (3.) von $\mathcal{S}$ nach $f(\mathcal{S})$ ist Reduktion von MPKP auf PKP

1. $\mathcal{S} \in$ **MPKP** $\Rightarrow f(\mathcal{S}) \in$ **PKP**

- Sei $(1, i_2, i_3, \ldots, i_k)$ (M)PKP-Folge für $\mathcal{S}$
- $\rightarrow (\ell + 1, i_2, i_3, \ldots, i_k, \ell + 2)$ ist PKP-Folge für $f(\mathcal{S})$

Behauptung: Modifikation durch (1.) (2.) (3.) von $\mathcal{S}$ nach $f(\mathcal{S})$ ist Reduktion von MPKP auf PKP

1. $\mathcal{S} \in$ **MPKP** $\Rightarrow f(\mathcal{S}) \in$ **PKP**

- Sei $(1, i_2, i_3, \ldots, i_k)$ (M)PKP-Folge für $\mathcal{S}$
- $\rightarrow (\ell + 1, i_2, i_3, \ldots, i_k, \ell + 2)$ ist PKP-Folge für $f(\mathcal{S})$

Im Bsp.

ab	ba	ab	b
a	bb	a	abb

(1,2,1,3) (M)PKP-Folge für $\mathcal{S}$

#a#b#	b#a#	a#b#	b#	$
#a	#b#b	#a	#a#b#b	#$

(4,2,1,3,5) PKP-Folge für $\mathcal{S}$

Behauptung: Modifikation durch (1.) (2.) (3.) von $\mathcal{S}$ nach $f(\mathcal{S})$ ist Reduktion von MPKP auf PKP

1. $\mathcal{S} \in$ **MPKP** $\Rightarrow f(\mathcal{S}) \in$ **PKP**

- Sei $(1, i_2, i_3, \ldots, i_k)$ (M)PKP-Folge für $\mathcal{S}$
- $\rightarrow (\ell + 1, i_2, i_3, \ldots, i_k, \ell + 2)$ ist PKP-Folge für $f(\mathcal{S})$

Im Bsp.

ab	ba	ab	b
a	bb	a	abb

(1,2,1,3) (M)PKP-Folge für $\mathcal{S}$

#a#b#	b#a#	a#b#	b#	$
#a	#b#b	#a	#a#b#b	#$

(4,2,1,3,5) PKP-Folge für $\mathcal{S}$

- Nach Konstruktion ist Lösungswort von $\mathcal{S}$, mit # durchsetzt und mit $ abgeschlossen Lösungswort für $f(\mathcal{S})$.

2. $f(\mathcal{S}) \in$ **PKP** $\Rightarrow \mathcal{S} \in$ **MPKP**

2. $f(\mathcal{S}) \in$ **PKP** $\Rightarrow \mathcal{S} \in$ **MPKP**

Sei $(i_1, i_2, i_3, \ldots, i_k)$ eine kürzeste PKP-Folge für $f(\mathcal{S})$

$f(\mathcal{S})$ Bsp.

#a#b#	a#b#	b#a#	b#	$
#a	#a	#b#b	#a#b#b	#$

2. $f(\mathcal{S}) \in$ **PKP** $\Rightarrow \mathcal{S} \in$ **MPKP**

Sei $(i_1, i_2, i_3, \ldots, i_k)$ eine kürzeste PKP-Folge für $f(\mathcal{S})$

$f(\mathcal{S})$ Bsp.

#a#b#	a#b#	b#a#	b#	$
#a	#a	#b#b	#a#b#b	#$

1. Nur das erste Domino fängt mit den gleichen Zeichen an, also ist $i_1 = \ell + 1$

2. $f(\mathcal{S}) \in$ **PKP** $\Rightarrow \mathcal{S} \in$ **MPKP**

Sei $(i_1, i_2, i_3, \ldots, i_k)$ eine kürzeste PKP-Folge für $f(\mathcal{S})$

$f(\mathcal{S})$ Bsp.

#a#b#	a#b#	b#a#	b#	$
#a	#a	#b#b	#a#b#b	#$

1. Nur das erste Domino fängt mit den gleichen Zeichen an, also ist $i_1 = \ell + 1$
2. Nur das letzte Domino hört mit den gleichen Zeichen auf, also ist $i_k = \ell + 2$

2. $f(\mathcal{S}) \in$ **PKP** $\Rightarrow \mathcal{S} \in$ **MPKP**

Sei $(i_1, i_2, i_3, \ldots, i_k)$ eine kürzeste PKP-Folge für $f(\mathcal{S})$

$f(\mathcal{S})$ Bsp.

#a#b#	a#b#	b#a#	b#	$
#a	#a	#b#b	#a#b#b	#$

1. Nur das erste Domino fängt mit den gleichen Zeichen an, also ist $i_1 = \ell + 1$
2. Nur das letzte Domino hört mit den gleichen Zeichen auf, also ist $i_k = \ell + 2$
3. Der $(\ell + 2)$te Dominotyp tritt nur einmal auf, denn sonst gäbe es eine kürzere PKP-Folge.

2. $f(\mathcal{S}) \in$ **PKP** $\Rightarrow \mathcal{S} \in$ **MPKP**

Sei $(i_1, i_2, i_3, \ldots, i_k)$ eine kürzeste PKP-Folge für $f(\mathcal{S})$

$f(\mathcal{S})$ Bsp.

#a#b#	a#b#	b#a#	b#	$
#a	#a	#b#b	#a#b#b	#$

1. Nur das erste Domino fängt mit den gleichen Zeichen an, also ist $i_1 = \ell + 1$
2. Nur das letzte Domino hört mit den gleichen Zeichen auf, also ist $i_k = \ell + 2$
3. Der $(\ell + 2)$te Dominotyp tritt nur einmal auf, denn sonst gäbe es eine kürzere PKP-Folge.
4. Der $(\ell + 1)$te Dominotyp tritt nur 1x auf, denn sonst gäbe es bei einmaliger Benutzung von Typ $\ell + 2$ zu viele $\#$ in der oberen Reihe.

2. $f(\mathcal{S}) \in$ **PKP** $\Rightarrow \mathcal{S} \in$ **MPKP**

Sei $(i_1, i_2, i_3, \ldots, i_k)$ eine kürzeste PKP-Folge für $f(\mathcal{S})$

$f(\mathcal{S})$ Bsp.

#a#b#	a#b#	b#a#	b#	$
#a	#a	#b#b	#a#b#b	#$

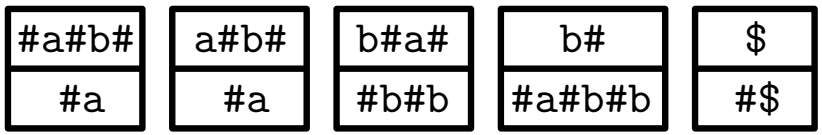
1. Nur das erste Domino fängt mit den gleichen Zeichen an, also ist $i_1 = \ell + 1$
2. Nur das letzte Domino hört mit den gleichen Zeichen auf, also ist $i_k = \ell + 2$
3. Der $(\ell + 2)$te Dominotyp tritt nur einmal auf, denn sonst gäbe es eine kürzere PKP-Folge.
4. Der $(\ell + 1)$te Dominotyp tritt nur 1x auf, denn sonst gäbe es bei einmaliger Benutzung von Typ $\ell + 2$ zu viele # in der oberen Reihe.

Lösungswort beginnt mit #, endet auf $ und jedes zweite Zeichen ist #

2. $f(\mathcal{S}) \in$ **PKP** $\Rightarrow \mathcal{S} \in$ **MPKP**

Sei $(i_1, i_2, i_3, \ldots, i_k)$ eine kürzeste PKP-Folge für $f(\mathcal{S})$

$f(\mathcal{S})$ Bsp.



1. Nur das erste Domino fängt mit den gleichen Zeichen an, also ist $i_1 = \ell + 1$
2. Nur das letzte Domino hört mit den gleichen Zeichen auf, also ist $i_k = \ell + 2$
3. Der $(\ell + 2)$te Dominotyp tritt nur einmal auf, denn sonst gäbe es eine kürzere PKP-Folge.
4. Der $(\ell + 1)$te Dominotyp tritt nur 1x auf, denn sonst gäbe es bei einmaliger Benutzung von Typ $\ell + 2$ zu viele # in der oberen Reihe.

Lösungswort beginnt mit #, endet auf $ und jedes zweite Zeichen ist #

$\rightarrow (1, i_2, i_3, \ldots, i_{k-1})$ ist (M)PKP-Folge für $\mathcal{S}$

2. $f(\mathcal{S}) \in$ PKP $\Rightarrow \mathcal{S} \in$ MPKP

Sei $(i_1, i_2, i_3, \ldots, i_k)$ eine kürzeste PKP-Folge für $f(\mathcal{S})$

$f(\mathcal{S})$ Bsp.

#a#b#	a#b#	b#a#	b#	$
#a	#a	#b#b	#a#b#b	#$

1. Nur das erste Domino fängt mit den gleichen Zeichen an, also ist $i_1 = \ell + 1$
2. Nur das letzte Domino hört mit den gleichen Zeichen auf, also ist $i_k = \ell + 2$
3. Der $(\ell + 2)$te Dominotyp tritt nur einmal auf, denn sonst gäbe es eine kürzere PKP-Folge.
4. Der $(\ell + 1)$te Dominotyp tritt nur 1x auf, denn sonst gäbe es bei einmaliger Benutzung von Typ $\ell + 2$ zu viele # in der oberen Reihe.

Lösungswort beginnt mit #, endet auf $ und jedes zweite Zeichen ist #

$\rightarrow (1, i_2, i_3, \ldots, i_{k-1})$ ist (M)PKP-Folge für $\mathcal{S}$

$\rightarrow$ **f ist Reduktion für MPKP $\leq_m$ PKP**

Satz 28

PKP ist **nicht entscheidbar**

Satz 28

PKP ist **nicht entscheidbar**

Beweis

- **Wir zeigen:** $A_{\mathsf{TM}} \leq_m$ MPKP

Satz 28

PKP ist **nicht entscheidbar**

Beweis

- **Wir zeigen:** $A_{\mathsf{TM}} \leq_m$ MPKP
- **Idee:** Lösungswort für MPKP Instanz ist akz. Berechnungspfad für $M(w)$

Satz 28

PKP ist **nicht entscheidbar**

Beweis

- **Wir zeigen:** $A_{\mathsf{TM}} \leq_m$ MPKP
- **Idee:** Lösungswort für MPKP Instanz ist akz. Berechnungspfad für $M(w)$

Kodierung Berechnungspfad:

Startkonf.#Folgekonf.1#Folgekonf.2#$\cdots$#akz. Konf.

Satz 28

PKP ist **nicht entscheidbar**

Beweis

- **Wir zeigen:** $A_{\mathsf{TM}} \leq_m$ MPKP
- **Idee:** Lösungswort für MPKP Instanz ist akz. Berechnungspfad für $M(w)$

Kodierung Berechnungspfad:

Startkonf.#Folgekonf.1#Folgekonf.2#$\cdots$#akz. Konf.

Folgekonfiguration ist fast überall identisch

Satz 28

PKP ist **nicht entscheidbar**

Beweis

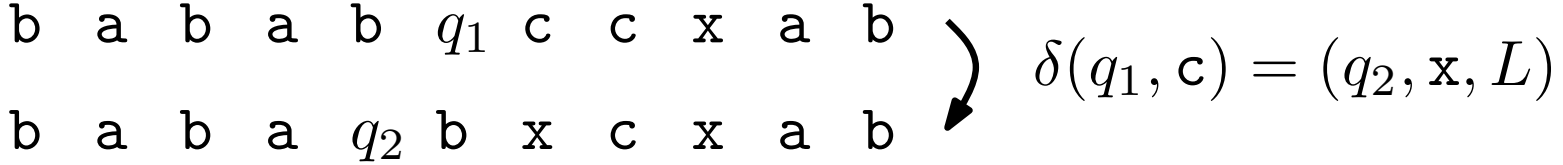
- **Wir zeigen:** $A_{\mathsf{TM}} \leq_m$ MPKP
- **Idee:** Lösungswort für MPKP Instanz ist akz. Berechnungspfad für $M(w)$

Kodierung Berechnungspfad:

Startkonf.#Folgekonf.1#Folgekonf.2#$\cdots$#akz. Konf.

Folgekonfiguration ist fast überall identisch

Bsp.



Satz 28

PKP ist **nicht entscheidbar**

Beweis

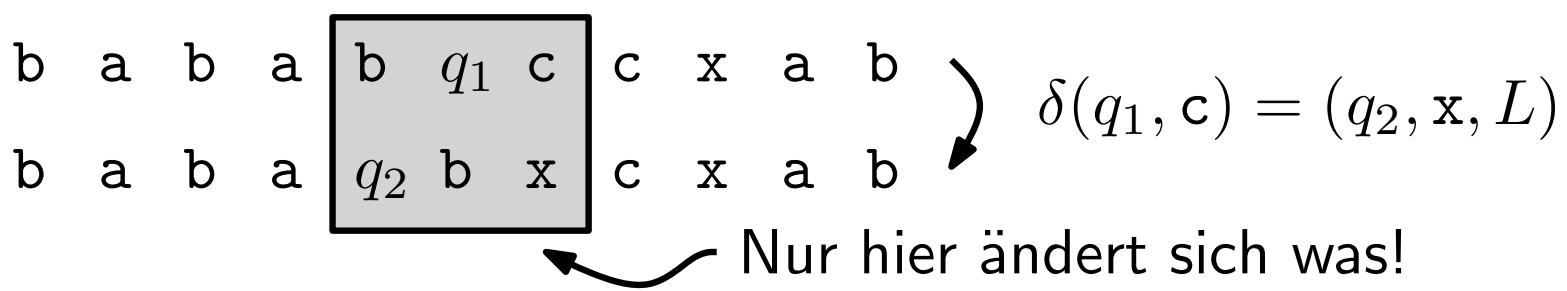
- **Wir zeigen:** $A_{\mathsf{TM}} \leq_m$ MPKP
- **Idee:** Lösungswort für MPKP Instanz ist akz. Berechnungspfad für $M(w)$

Kodierung Berechnungspfad:

Startkonf.#Folgekonf.1#Folgekonf.2#$\cdots$#akz. Konf.

Folgekonfiguration ist fast überall identisch

Bsp.



Satz 28

PKP ist **nicht entscheidbar**

Beweis

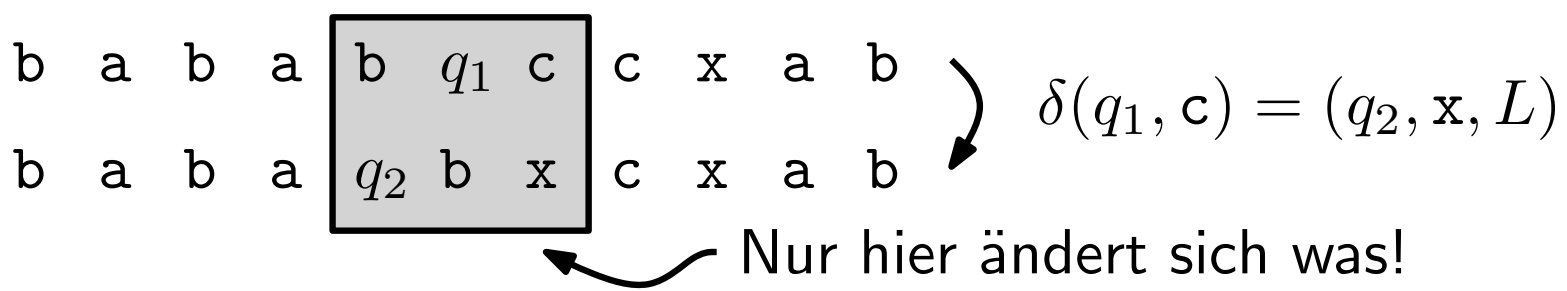
- **Wir zeigen:** $A_{\mathsf{TM}} \leq_m$ MPKP
- **Idee:** Lösungswort für MPKP Instanz ist akz. Berechnungspfad für $M(w)$

Kodierung Berechnungspfad:

Startkonf.#Folgekonf.1#Folgekonf.2#$\cdots$#akz. Konf.

Folgekonfiguration ist fast überall identisch

Bsp.



- Dominos "kopieren" Konfiguration, erlauben dabei Übergänge

* **Startdomino**

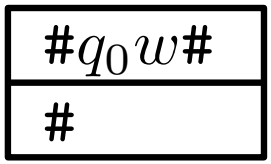
$\#q_0w\#$
#

* **Startdomino**

$\#q_0w\#$
#

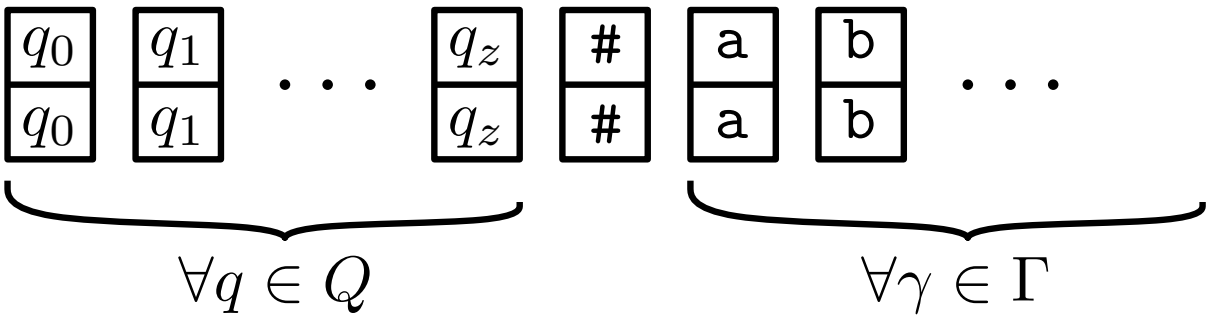
obere Wort "läuft vor"

* **Startdomino**

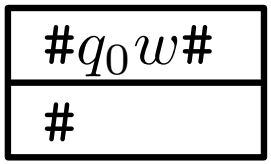


obere Wort "läuft vor"

* **Kopierdominos**

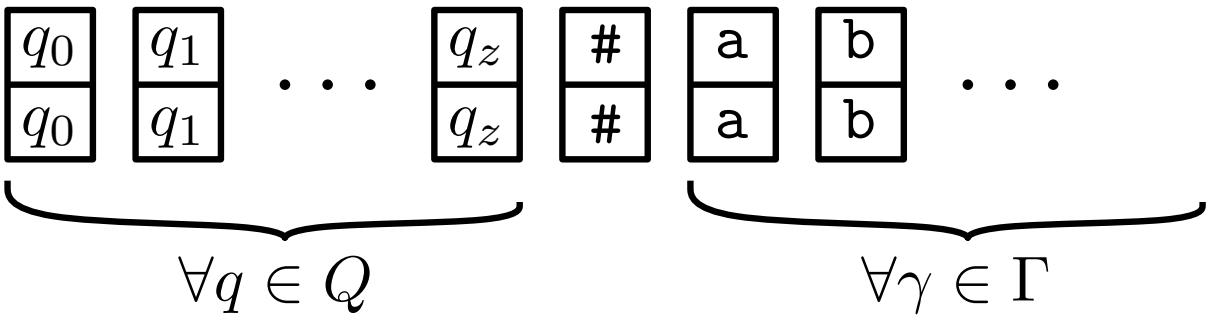


* **Startdomino**

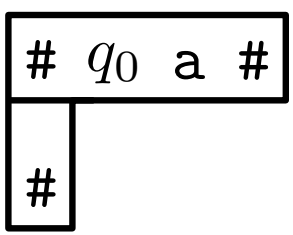


obere Wort "läuft vor"

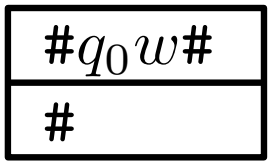
* **Kopierdominos**



Folgendes ist jetzt möglich

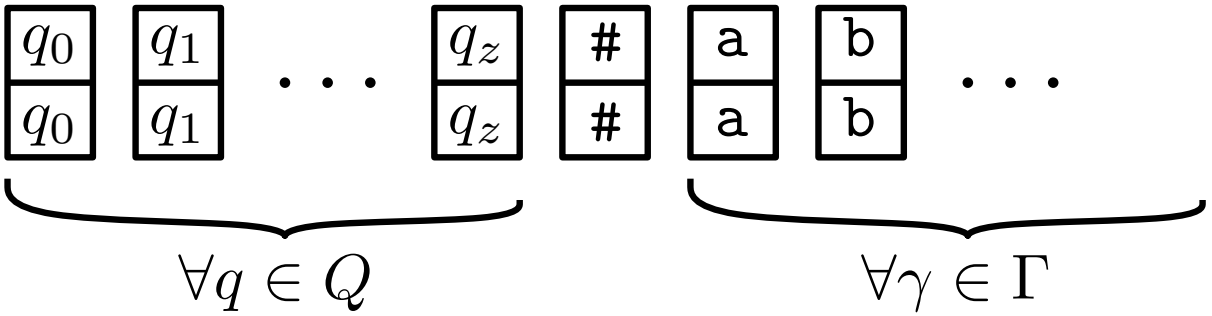


* **Startdomino**

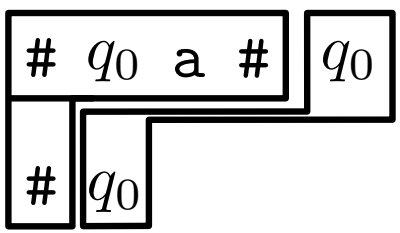


obere Wort "läuft vor"

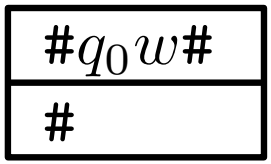
* **Kopierdominos**



Folgendes ist jetzt möglich

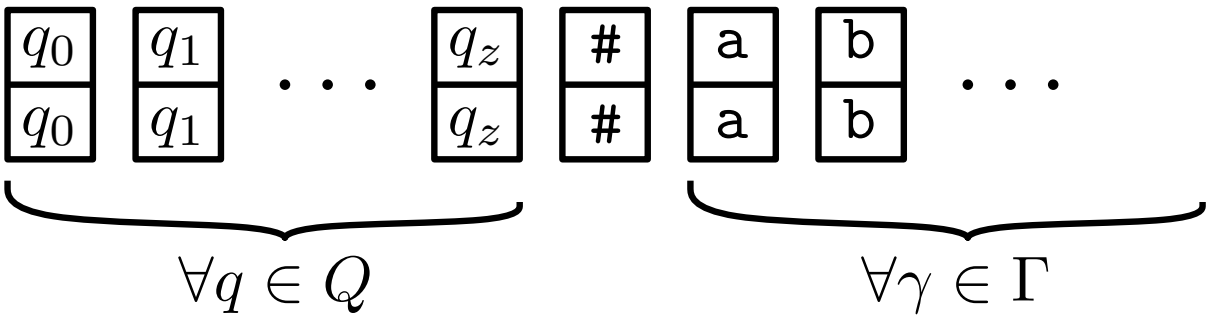


* **Startdomino**

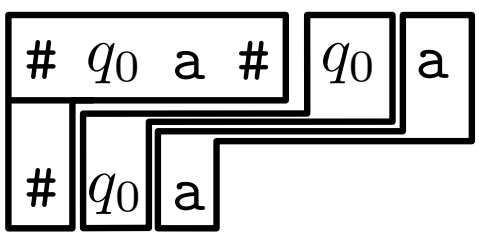


obere Wort "läuft vor"

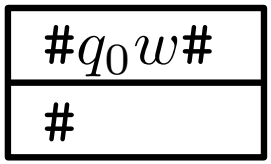
* **Kopierdominos**



Folgendes ist jetzt möglich

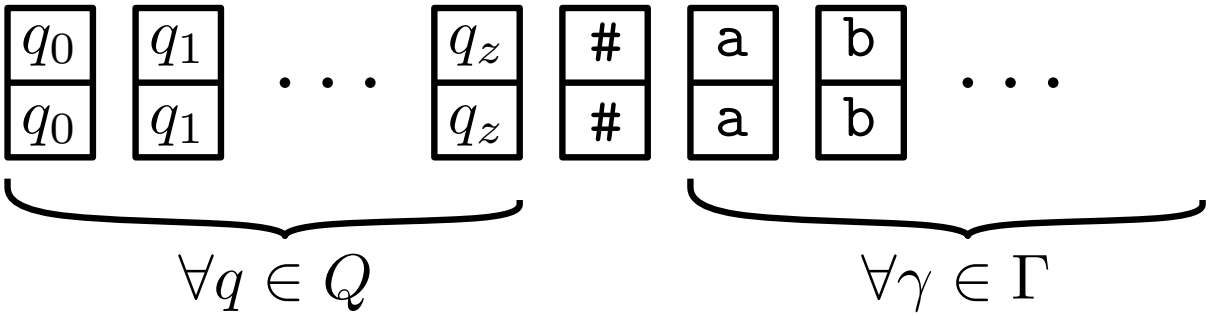


* **Startdomino**

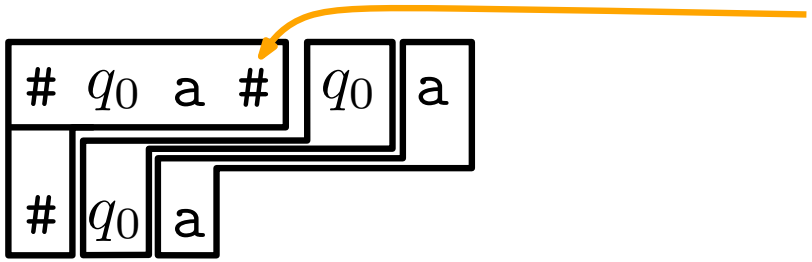


obere Wort "läuft vor"

* **Kopierdominos**

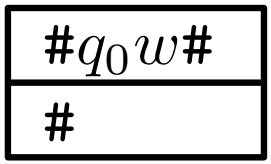


Folgendes ist jetzt möglich



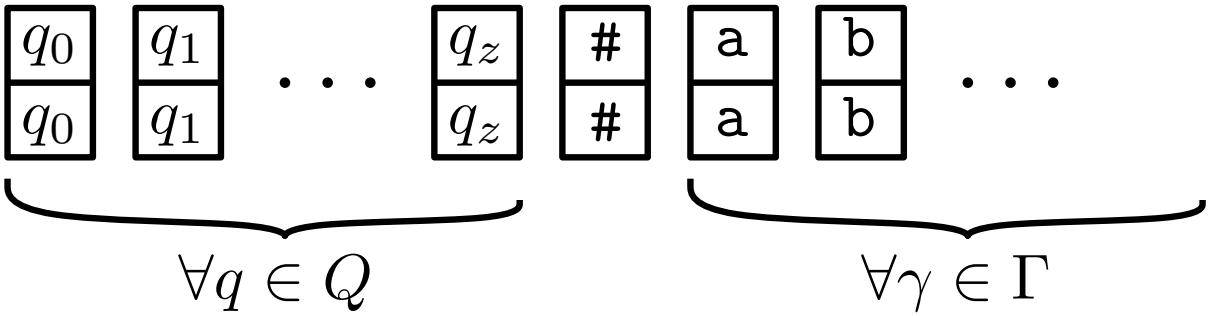
Zeichen erzwingt unsere Auswahl

* **Startdomino**

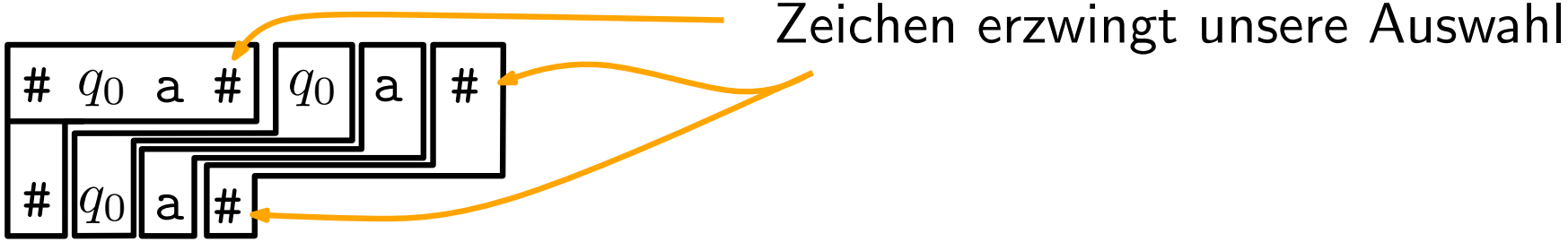


obere Wort "läuft vor"

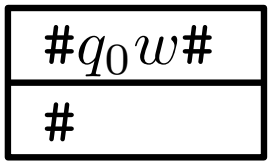
* **Kopierdominos**



Folgendes ist jetzt möglich

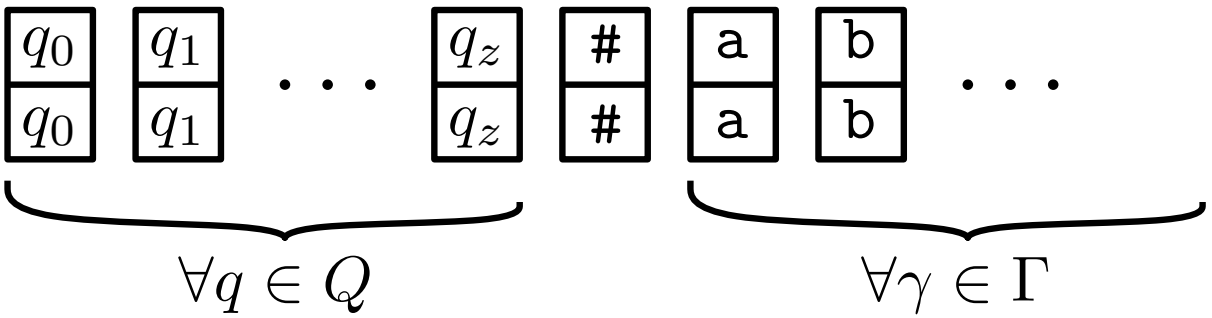


* **Startdomino**

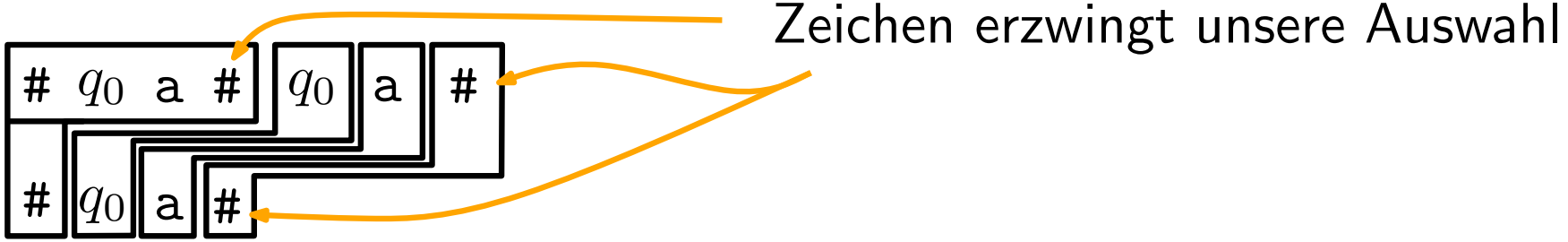


obere Wort "läuft vor"

* **Kopierdominos**



Folgendes ist jetzt möglich



Im Lösungswort würde eine wiederholte Kopie der Startkonfiguration stehen müssen. PKP-Folge kann aber nicht abgeschlossen werden.

* Übergangsdominos

c	p
q	a

für alle Befehle $\delta(q, \mathtt{a}) = (p, \mathtt{c}, R)$

p	b	c
b	q	a

für alle Befehle $\delta(q, \mathtt{a}) = (p, \mathtt{c}, L), \forall b \in \Gamma$

* Übergangsdominos

c	p
q	a

für alle Befehle $\delta(q, \mathtt{a}) = (p, \mathtt{c}, R)$

p	b	c
b	q	a

für alle Befehle $\delta(q, \mathtt{a}) = (p, \mathtt{c}, L), \forall b \in \Gamma$

Jetzt ist es möglich, entweder eine Konfiguration zu wiederholen, oder die Folgekonfiguration zu "schreiben".

* Übergangsdominos

c	p
q	a

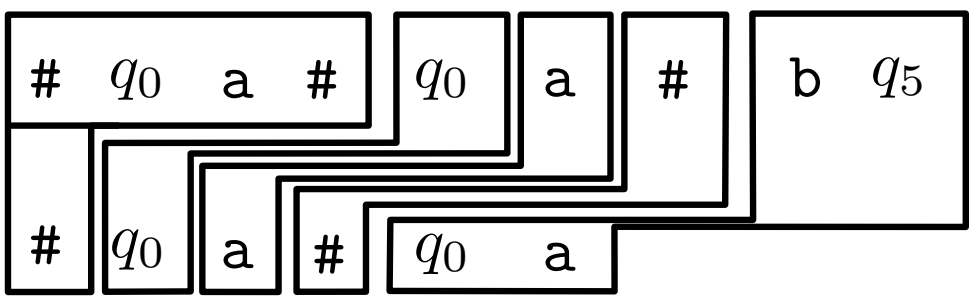
für alle Befehle $\delta(q, \mathtt{a}) = (p, \mathtt{c}, R)$

p	b	c
b	q	a

für alle Befehle $\delta(q, \mathtt{a}) = (p, \mathtt{c}, L), \forall b \in \Gamma$

Jetzt ist es möglich, entweder eine Konfiguration zu wiederholen, oder die Folgekonfiguration zu "schreiben".

Bsp.



für
$\delta(q_0, \mathtt{a}) = (q_5, \mathtt{b}, R)$

* Übergangsdominos

c	p
q	a

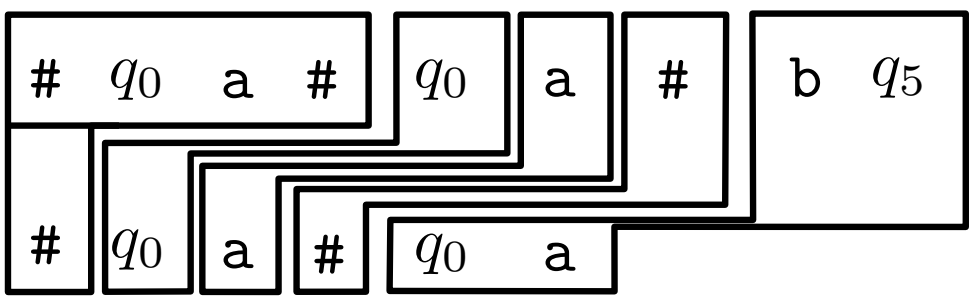
für alle Befehle $\delta(q, \mathtt{a}) = (p, \mathtt{c}, R)$

p	b	c
b	q	a

für alle Befehle $\delta(q, \mathtt{a}) = (p, \mathtt{c}, L), \forall b \in \Gamma$

Jetzt ist es möglich, entweder eine Konfiguration zu wiederholen, oder die Folgekonfiguration zu "schreiben".

Bsp.



für
$\delta(q_0, \mathtt{a}) = (q_5, \mathtt{b}, R)$

* Übergangsdominos für den Rand

Kopiere Konfiguration und füge vorne/hinten Blank an

* Übergangsdominos

c	p
q	a

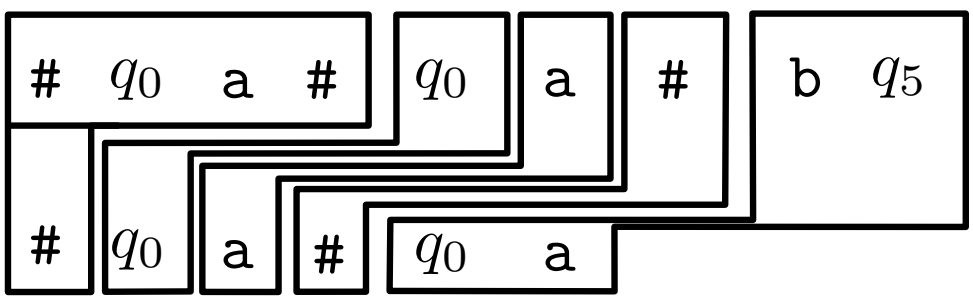
für alle Befehle $\delta(q, \mathtt{a}) = (p, \mathtt{c}, R)$

p	b	c
b	q	a

für alle Befehle $\delta(q, \mathtt{a}) = (p, \mathtt{c}, L), \forall b \in \Gamma$

Jetzt ist es möglich, entweder eine Konfiguration zu wiederholen, oder die Folgekonfiguration zu "schreiben".

Bsp.



für
$\delta(q_0, \mathtt{a}) = (q_5, \mathtt{b}, R)$

* Übergangsdominos für den Rand

Kopiere Konfiguration und füge vorne/hinten Blank an

q □ #
q #

□ q
q

für alle $q \in Q$

Jetzt ist zwar jeder Berechnungspfad darstellbar, ich kann die PKP-Folge aber immer noch nicht *beenden*.

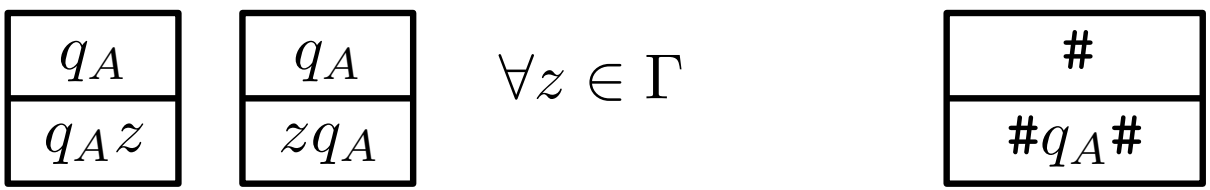
Jetzt ist zwar jeder Berechnungspfad darstellbar, ich kann die PKP-Folge aber immer noch nicht *beenden*.

Beenden soll nur möglich sein, wenn in meiner aktuell kopierten Konfiguration q_A enthalten ist.

Jetzt ist zwar jeder Berechnungspfad darstellbar, ich kann die PKP-Folge aber immer noch nicht *beenden*.

Beenden soll nur möglich sein, wenn in meiner aktuell kopierten Konfiguration q_A enthalten ist.

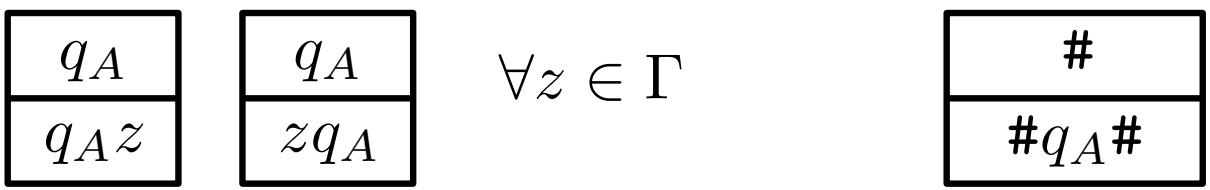
* **Pumpdominos**



Jetzt ist zwar jeder Berechnungspfad darstellbar, ich kann die PKP-Folge aber immer noch nicht *beenden*.

Beenden soll nur möglich sein, wenn in meiner aktuell kopierten Konfiguration q_A enthalten ist.

* **Pumpdominos**

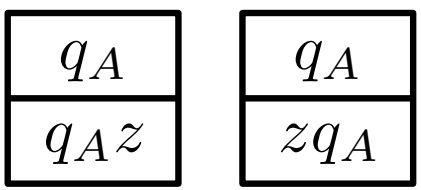


Entfernt Zeichen, vor oder nach q_A

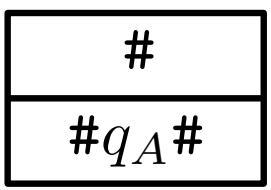
Jetzt ist zwar jeder Berechnungspfad darstellbar, ich kann die PKP-Folge aber immer noch nicht *beenden*.

Beenden soll nur möglich sein, wenn in meiner aktuell kopierten Konfiguration q_A enthalten ist.

* **Pumpdominos**



$\forall z \in \Gamma$



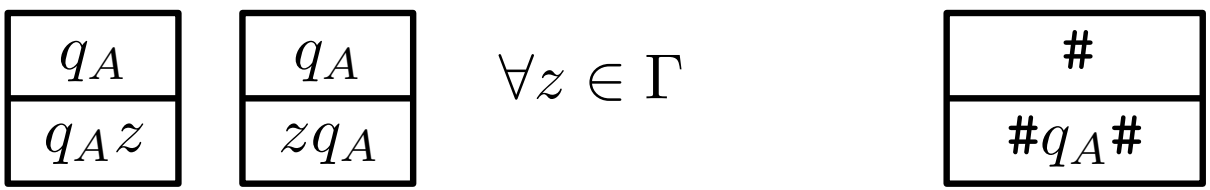
Entfernt Zeichen, vor oder nach q_A

Entfernt q_A

Jetzt ist zwar jeder Berechnungspfad darstellbar, ich kann die PKP-Folge aber immer noch nicht *beenden*.

Beenden soll nur möglich sein, wenn in meiner aktuell kopierten Konfiguration q_A enthalten ist.

* **Pumpdominos**



Entfernt Zeichen, vor oder nach q_A

Entfernt q_A

- **Reduktion f:** Zu einer Turingmaschine $M(w)$ konstruiere die besprochenen Dominotypen $\mathcal{S} = f(\langle M, w \rangle)$

Jetzt ist zwar jeder Berechnungspfad darstellbar, ich kann die PKP-Folge aber immer noch nicht *beenden*.

Beenden soll nur möglich sein, wenn in meiner aktuell kopierten Konfiguration q_A enthalten ist.

* **Pumpdominos**

q_A
$q_A z$

q_A
$z q_A$

$\forall z \in \Gamma$

#
$\#q_A\#$

Entfernt Zeichen, vor oder nach q_A

Entfernt q_A

- **Reduktion f:** Zu einer Turingmaschine $M(w)$ konstruiere die besprochenen Dominotypen $\mathcal{S} = f(\langle M, w \rangle)$
- **Noch zu zeigen:** M akzeptiert w genau dann, wenn es eine PKP-Folge für $\mathcal{S}$ gibt

Nach Konstruktion kann man zu jedem akzept. Lauf eine PKP-Folge wie folgt ermitteln

Nach Konstruktion kann man zu jedem akzept. Lauf eine PKP-Folge wie folgt ermitteln

- Lösungswort besteht zuerst aus Lauf (evtl. mit eingefügten Blanks)

Nach Konstruktion kann man zu jedem akzept. Lauf eine PKP-Folge wie folgt ermitteln

- Lösungswort besteht zuerst aus Lauf (evtl. mit eingefügten Blanks)
- danach kommt eine Sequenz von Teilwörtern einer akzeptierenden Konfiguration, bei denen jeweils ein Zeichen entfernt wird
- zuletzt steht $q_A\#$

Nach Konstruktion kann man zu jedem akzept. Lauf eine PKP-Folge wie folgt ermitteln

- Lösungswort besteht zuerst aus Lauf (evtl. mit eingefügten Blanks)
- danach kommt eine Sequenz von Teilwörtern einer akzeptierenden Konfiguration, bei denen jeweils ein Zeichen entfernt wird
- zuletzt steht q_A#

$\rightarrow$ Wenn $M(w)$ akzeptierend, dann $\mathcal{S} \in$ MPKP

Nach Konstruktion kann man zu jedem akzept. Lauf eine PKP-Folge wie folgt ermitteln

- Lösungswort besteht zuerst aus Lauf (evtl. mit eingefügten Blanks)
- danach kommt eine Sequenz von Teilwörtern einer akzeptierenden Konfiguration, bei denen jeweils ein Zeichen entfernt wird
- zuletzt steht q_A#

$\rightarrow$ Wenn $M(w)$ akzeptierend, dann $\mathcal{S} \in$ MPKP

- Jedes Lösungswort beginnt mit der Startkonfiguration

Nach Konstruktion kann man zu jedem akzept. Lauf eine PKP-Folge wie folgt ermitteln

- Lösungswort besteht zuerst aus Lauf (evtl. mit eingefügten Blanks)
- danach kommt eine Sequenz von Teilwörtern einer akzeptierenden Konfiguration, bei denen jeweils ein Zeichen entfernt wird
- zuletzt steht q_A#

$\rightarrow$ Wenn $M(w)$ akzeptierend, dann $\mathcal{S} \in$ MPKP

- Jedes Lösungswort beginnt mit der Startkonfiguration
- nach jedem # steht entweder die vorige Konfiguration, die Nachfolgekonfiguration, oder die vorige Konfiguration mit $\square$ erweitert

Nach Konstruktion kann man zu jedem akzept. Lauf eine PKP-Folge wie folgt ermitteln

- Lösungswort besteht zuerst aus Lauf (evtl. mit eingefügten Blanks)
- danach kommt eine Sequenz von Teilwörtern einer akzeptierenden Konfiguration, bei denen jeweils ein Zeichen entfernt wird
- zuletzt steht q_A#

$\rightarrow$ Wenn $M(w)$ akzeptierend, dann $\mathcal{S} \in$ MPKP

- Jedes Lösungswort beginnt mit der Startkonfiguration
- nach jedem # steht entweder die vorige Konfiguration, die Nachfolgekonfiguration, oder die vorige Konfiguration mit $\square$ erweitert

$\rightarrow$ Bei $M(w)$ nicht akz. erreicht man nie q_A, man kann kein Domino benutzen mit $|x_i| < |y_i| \rightarrow \mathcal{S} \notin$ MPKP

Nach Konstruktion kann man zu jedem akzept. Lauf eine PKP-Folge wie folgt ermitteln

- Lösungswort besteht zuerst aus Lauf (evtl. mit eingefügten Blanks)
- danach kommt eine Sequenz von Teilwörtern einer akzeptierenden Konfiguration, bei denen jeweils ein Zeichen entfernt wird
- zuletzt steht $q_A\#$

$\rightarrow$ Wenn $M(w)$ akzeptierend, dann $\mathcal{S} \in$ MPKP

- Jedes Lösungswort beginnt mit der Startkonfiguration
- nach jedem $\#$ steht entweder die vorige Konfiguration, die Nachfolgekonfiguration, oder die vorige Konfiguration mit $\square$ erweitert

$\rightarrow$ Bei $M(w)$ nicht akz. erreicht man nie q_A, man kann kein Domino benutzen mit $|x_i| < |y_i| \rightarrow \mathcal{S} \notin$ MPKP

Es gilt also $A_{\text{TM}} \leq_m$ MPKP $\leq_m$ PKP

Nach Konstruktion kann man zu jedem akzept. Lauf eine PKP-Folge wie folgt ermitteln

- Lösungswort besteht zuerst aus Lauf (evtl. mit eingefügten Blanks)
- danach kommt eine Sequenz von Teilwörtern einer akzeptierenden Konfiguration, bei denen jeweils ein Zeichen entfernt wird
- zuletzt steht $q_A\#$

$\rightarrow$ Wenn $M(w)$ akzeptierend, dann $\mathcal{S} \in$ MPKP

- Jedes Lösungswort beginnt mit der Startkonfiguration
- nach jedem $\#$ steht entweder die vorige Konfiguration, die Nachfolgekonfiguration, oder die vorige Konfiguration mit $\square$ erweitert

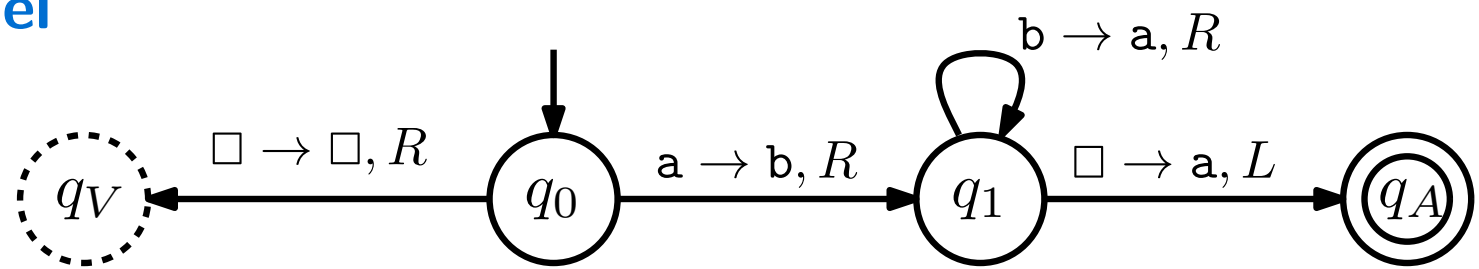
$\rightarrow$ Bei $M(w)$ nicht akz. erreicht man nie q_A, man kann kein Domino benutzen mit $|x_i| < |y_i| \rightarrow \mathcal{S} \notin$ MPKP

Es gilt also $A_{\mathsf{TM}} \leq_m$ MPKP $\leq_m$ PKP

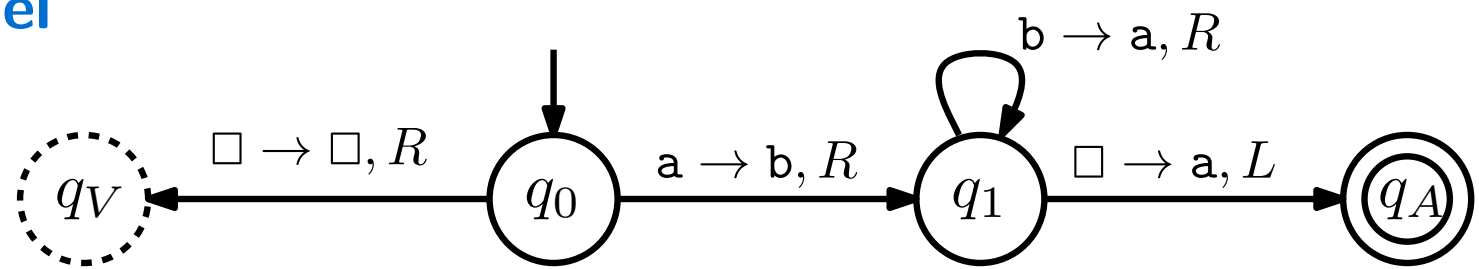
$\rightarrow$ also: $A_{\mathsf{TM}} \leq_m$ PKP und deshalb ist PKP **nicht entscheidbar**

$\square$

Beispiel


$\mathtt{b} \to \mathtt{a}, R$
$\square \to \square, R$
$\mathtt{a} \to \mathtt{b}, R$
$\square \to \mathtt{a}, L$
q_V
q_0
q_1
q_A

Beispiel

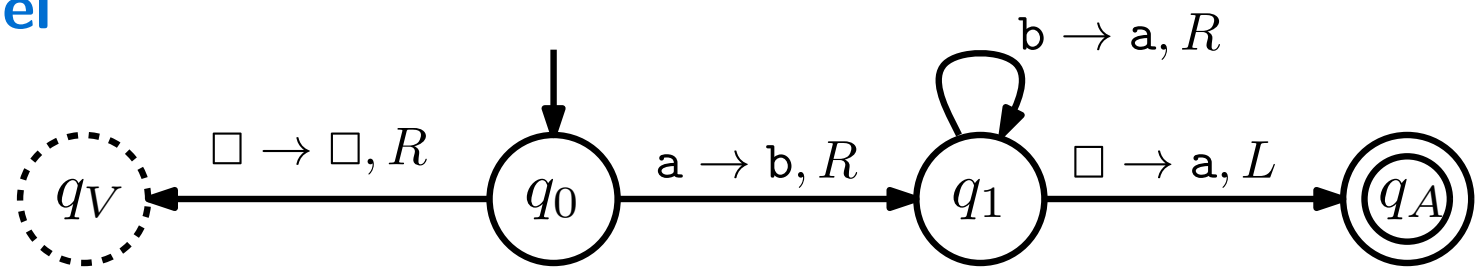


Dominotypen (für Eingabe a)

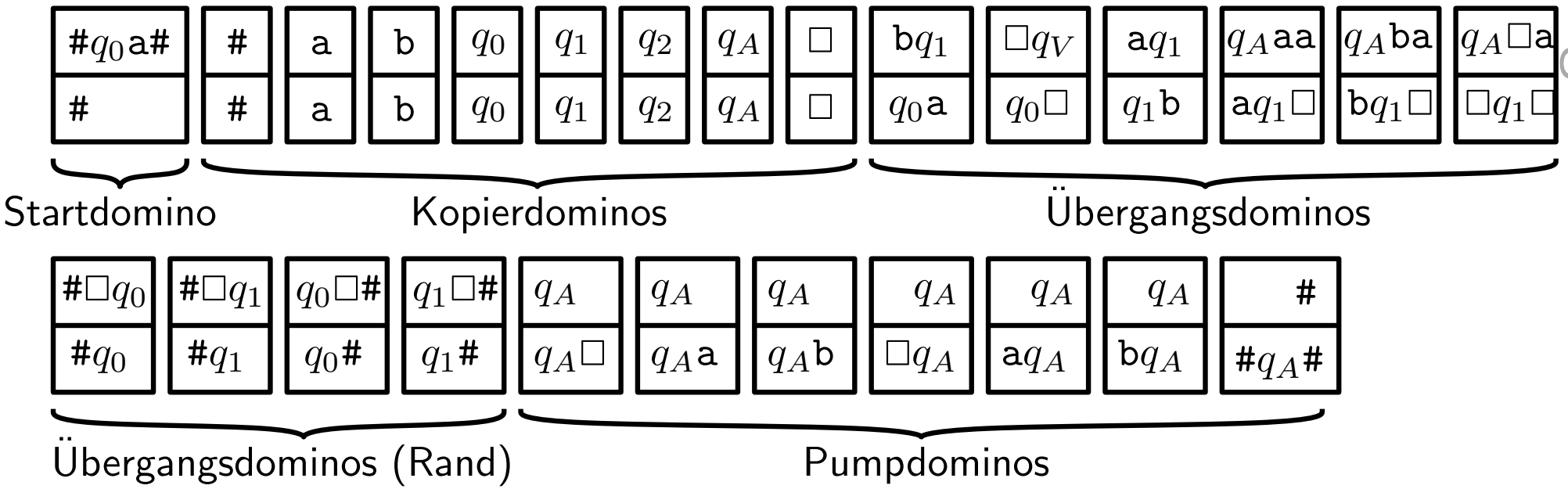
$\#q_0\mathtt{a}\#$	$\#$	$\mathtt{a}$	$\mathtt{b}$	q_0	q_1	q_2	q_A	$\square$	$\mathtt{b}q_1$	$\square q_V$	$\mathtt{a}q_1$	$q_A\mathtt{aa}$	$q_A\mathtt{ba}$	$q_A\square\mathtt{a}$
$\#$	$\#$	$\mathtt{a}$	$\mathtt{b}$	q_0	q_1	q_2	q_A	$\square$	$q_0\mathtt{a}$	$q_0\square$	$q_1\mathtt{b}$	$\mathtt{a}q_1\square$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

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

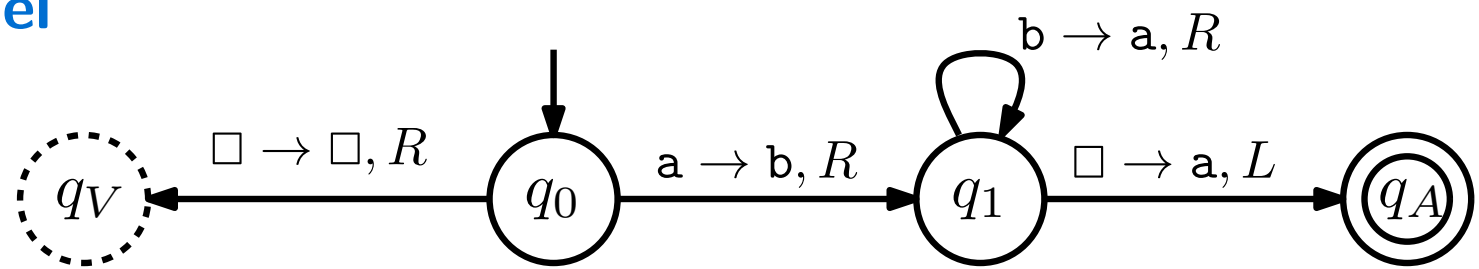
Beispiel



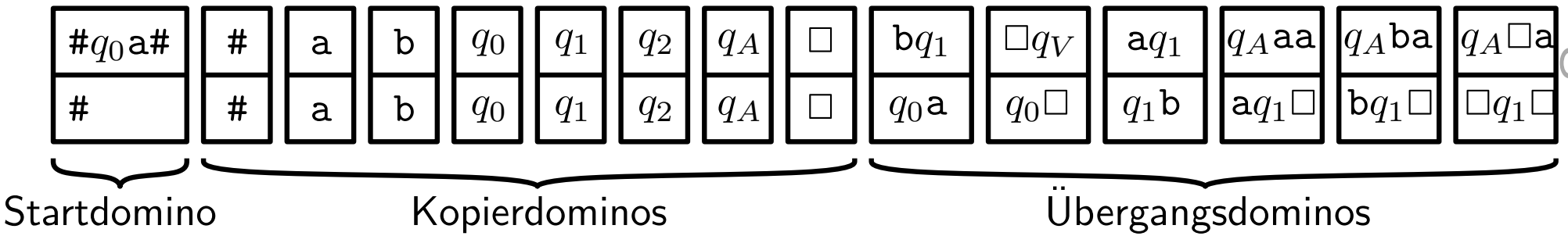
Dominotypen (für Eingabe a)

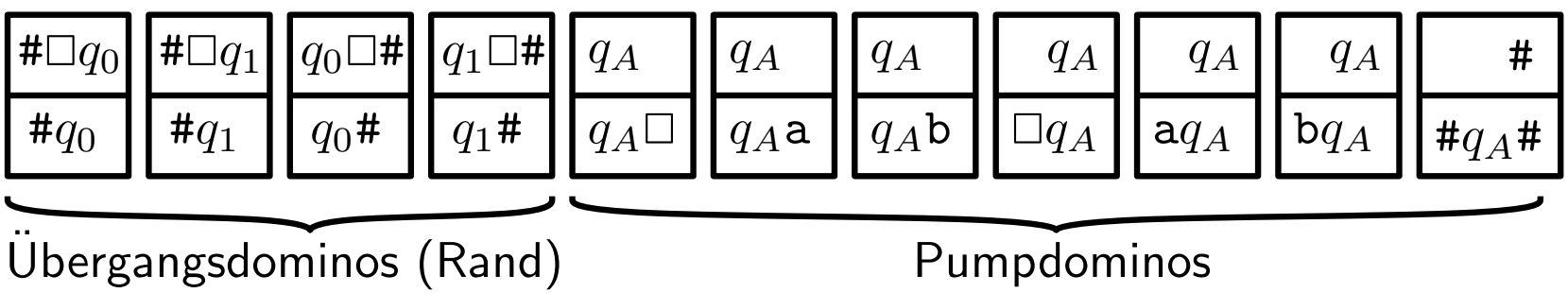


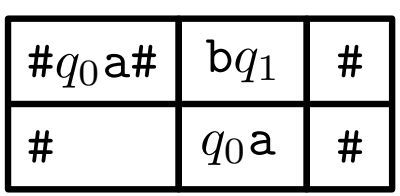
Beispiel



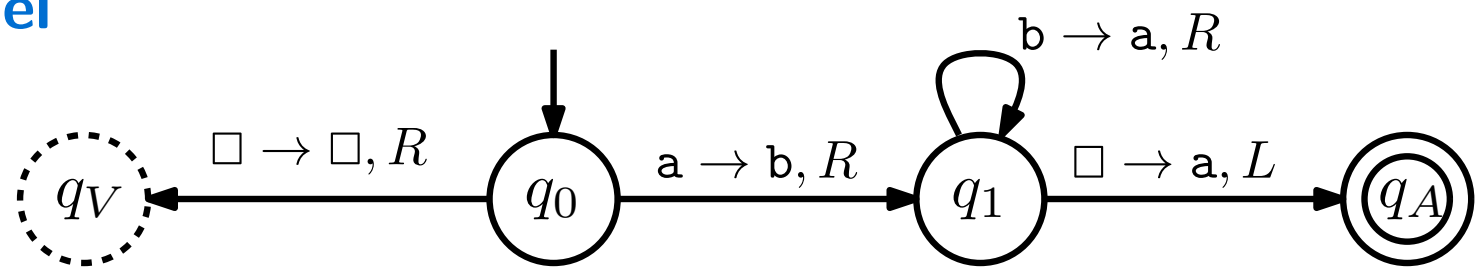
Dominotypen (für Eingabe a)







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	$\square$	$q_0\mathtt{a}$	$q_0\square$	$q_1\mathtt{b}$	$\mathtt{a}q_1\square$	$\mathtt{b}q_1\square$	$\square q_1\square$

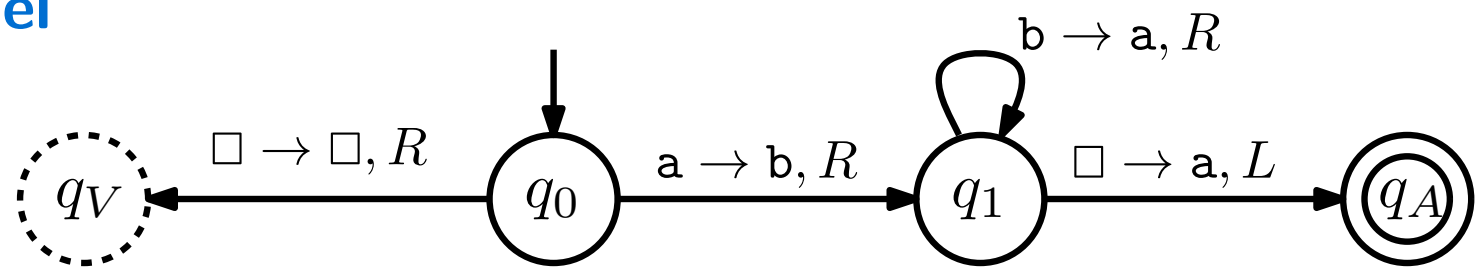
Startdomino — Kopierdominos — Übergangsdominos

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

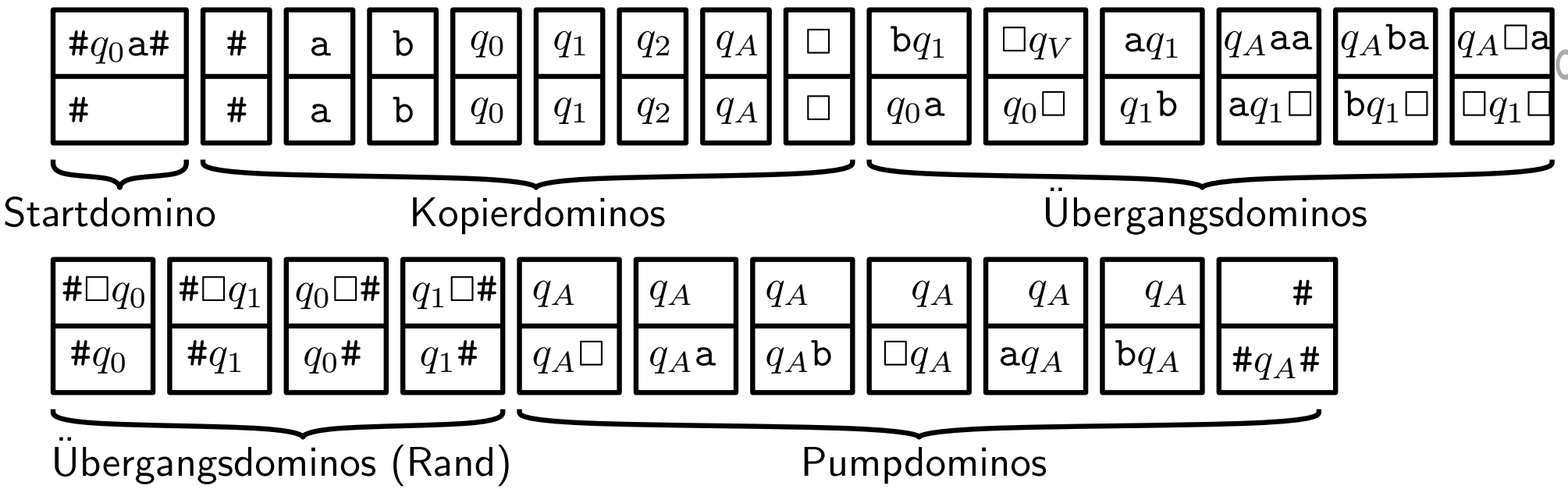
Übergangsdominos (Rand) — Pumpdominos

$\#q_0\mathtt{a}\#$	$\mathtt{b}q_1$	$\#$	b	$q_1\square\#$
$\#$	$q_0\mathtt{a}$	$\#$	b	$q_1\#$

Beispiel

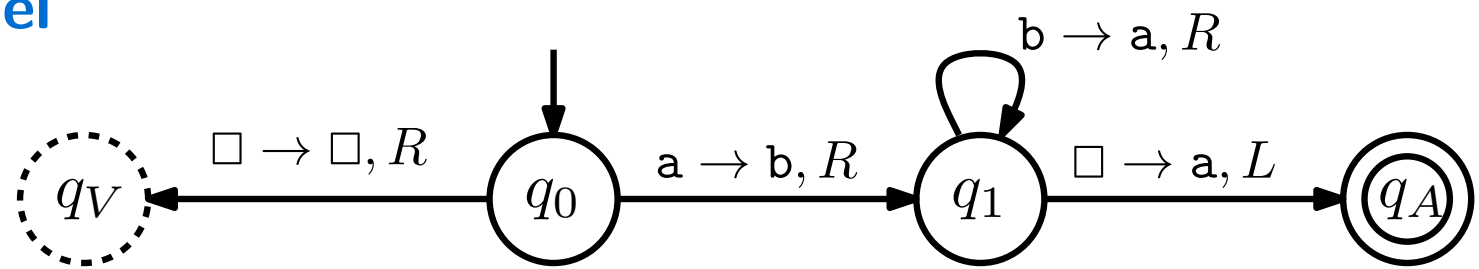


Dominotypen (für Eingabe a)



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

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	$\square$	$q_0\mathtt{a}$	$q_0\square$	$q_1\mathtt{b}$	$\mathtt{a}q_1\square$	$\mathtt{b}q_1\square$	$\square q_1\square$

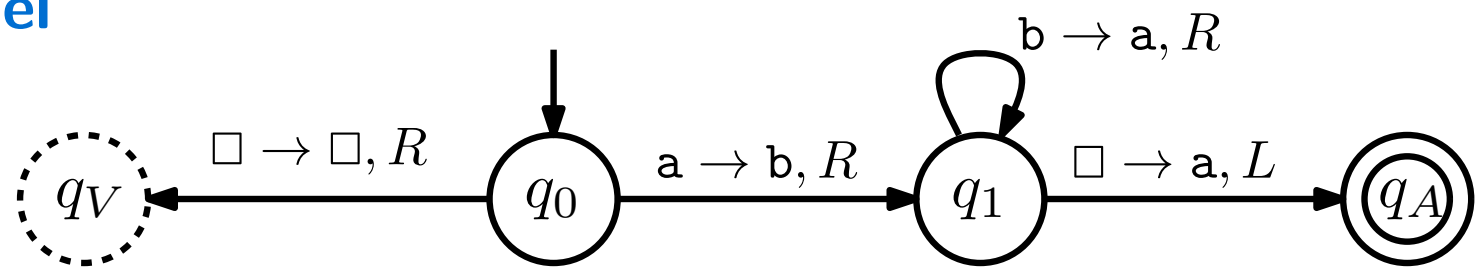
Startdomino | Kopierdominos | Übergangsdominos

$\#\square q_0$	$\#\square q_1$	$q_0\square\#$	$q_1\square\#$	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	$\#$	$q_A\mathtt{a}$	$\#\ q_A\#$

Beispiel



Dominotypen (für Eingabe a)

$\#q_0$a#	#	a	b	q_0	q_1	q_2	q_A	□	bq_1	□q_V	aq_1	$q_A$aa	$q_A$ba	q_A□a
#	#	a	b	q_0	q_1	q_2	q_A	□	q_0a	q_0□	q_1b	aq_1□	bq_1□	□q_1□

Startdomino | Kopierdominos | Übergangsdominos

#□q_0	#□q_1	q_0□#	q_1□#	q_A	q_A	q_A	q_A	q_A	q_A	#
#q_0	#q_1	q_0#	q_1#	q_A□	q_Aa	q_Ab	□q_A	aq_A	bq_A	#q_A#

Übergangsdominos (Rand) | Pumpdominos

#q_0a#	bq_1	#	b	q_1□#	$q_A$ba	#	q_A	a	#	q_A	#
#	q_0a	#	b	q_1#	bq_1□	#	q_Ab	a	#	q_Aa	# q_A#

Lösungswort: #q_0a#bq_1#bq_1□#$q_A$ba#q_Aa#q_A#