# F2 — Automaten und formale Sprachen

Matthias Jantzen

(nach und mit Folienvorlagen von Berndt Farwer)

Fachbereich Informatik

AB „Theoretische Grundlagen der Informatik“ (TGI)

Universität Hamburg

*jantzen@informatik.uni-hamburg.de*

# Themen

- Für die heutige Vorlesung geplant:
  - Homomorphismen
  - Substitutionen
  - initiale Zusammenhangskomponente
  - Grenzen der regulären Sprachen
    - Pumping-Lemma
    - Abschlusseigenschaften

# Homomorphismen

- Das Wechseln des Alphabets und der Austausch einzelner Symbole macht aus einer regulären Menge wiederum eine reguläre Menge.

# Homomorphismen

- Das Wechseln des Alphabets und der Austausch einzelner Symbole macht aus einer regulären Menge wiederum eine reguläre Menge.
- **Definition:** Eine Funktion $h$, für die $h(x \cdot y) = h(x) \circ h(y)$ gilt, wird strukturerhaltend oder **Homomorphismus** genannt.

# Homomorphismen

- Das Wechseln des Alphabets und der Austausch einzelner Symbole macht aus einer regulären Menge wiederum eine reguläre Menge.
- **Definition:** Eine Funktion $h$, für die $h(x \cdot y) = h(x) \circ h(y)$ gilt, wird strukturerhaltend oder **Homomorphismus** genannt.
- **Beispiel:** Sei $\Sigma = \{a, b\}$, $\Gamma = \{b, c\}$ und $h : \Sigma^* \longrightarrow \Gamma^*$ mit

$$h(a) := c, \quad h(b) := bb.$$

# Homomorphismen

- Das Wechseln des Alphabets und der Austausch einzelner Symbole macht aus einer regulären Menge wiederum eine reguläre Menge.
- **Definition:** Eine Funktion $h$, für die $h(x \cdot y) = h(x) \circ h(y)$ gilt, wird strukturerhaltend oder **Homomorphismus** genannt.
- **Beispiel:** Sei $\Sigma = \{a, b\}$, $\Gamma = \{b, c\}$ und $h : \Sigma^* \longrightarrow \Gamma^*$ mit

$$h(a) := c, \quad h(b) := bb.$$

  - Dann gilt: $h(abab) = h(aba)h(b) = h(aba)bb = h(a)h(b)h(a)bb = cbbcbb$.

# Homomorphismen

- Das Wechseln des Alphabets und der Austausch einzelner Symbole macht aus einer regulären Menge wiederum eine reguläre Menge.
- **Definition:** Eine Funktion $h$, für die $h(x \cdot y) = h(x) \circ h(y)$ gilt, wird strukturerhaltend oder **Homomorphismus** genannt.
- **Beispiel:** Sei $\Sigma = \{a, b\}$, $\Gamma = \{b, c\}$ und $h : \Sigma^* \longrightarrow \Gamma^*$ mit

$$h(a) := c, \quad h(b) := bb.$$

  - Dann gilt: $h(abab) = h(aba)h(b) = h(aba)bb = h(a)h(b)h(a)bb = cbbcbb$.
  - $h$ ist ein Homomorphismus, bei dem sowohl $\circ$ als auch $\cdot$ die Konkatenation von Wörtern ist. $(h(\lambda) = \lambda)$

# Abschlussop.: Homomorphismus

- **Theorem:** Sei $L \in \mathcal{R}at(\Sigma)$ und $h : \Sigma^* \longrightarrow \Gamma^*$ ein Homomorphismus. Dann ist $h(L) := \{h(v) \mid v \in L\} \in \mathcal{R}at(\Gamma)$

# Abschlussop.: Homomorphismus

- **Theorem:** Sei $L \in \mathcal{R}at(\Sigma)$ und $h : \Sigma^* \longrightarrow \Gamma^*$ ein Homomorphismus. Dann ist $h(L) := \{h(v) \mid v \in L\} \in \mathcal{R}at(\Gamma)$
- **Beweis:** Man ersetze die Symbole $a \in \Sigma$ in dem rationalen Ausdruck für $L$ jeweils durch $h(a)$. Es resultiert ein rationaler Ausdruck für $h(L)$.

# Abschlussop.: Homomorphismus

- **Theorem:** Sei $L \in \mathcal{R}at(\Sigma)$ und $h : \Sigma^* \longrightarrow \Gamma^*$ ein Homomorphismus. Dann ist $h(L) := \{h(v) \mid v \in L\} \in \mathcal{R}at(\Gamma)$
- **Beweis:** Man ersetze die Symbole $a \in \Sigma$ in dem rationalen Ausdruck für $L$ jeweils durch $h(a)$. Es resultiert ein rationaler Ausdruck für $h(L)$.
- Eine ähnliche Konstruktion ist auch mit NFAs möglich.

# Abschlussop.: Homomorphismus

- **Theorem:** Sei $L \in \mathcal{R}at(\Sigma)$ und $h : \Sigma^* \longrightarrow \Gamma^*$ ein Homomorphismus. Dann ist $h(L) := \{h(v) \mid v \in L\} \in \mathcal{R}at(\Gamma)$
- **Beweis:** Man ersetze die Symbole $a \in \Sigma$ in dem rationalen Ausdruck für $L$ jeweils durch $h(a)$. Es resultiert ein rationaler Ausdruck für $h(L)$.
- Eine ähnliche Konstruktion ist auch mit NFAs möglich.
- Diese Idee wird nun verallgemeinert, indem für einzelne Symbole ganze Sprachen **substituiert** werden.

# Substitution

- **Definition** Eine **Substitution** ist ein Homomorphismus $s : \Sigma^* \longrightarrow 2^{\Gamma^*}$, wobei $\Sigma$ und $\Gamma$ Alphabete sind und $s$ folgende Eigenschaften besitzt:

# Substitution

- **Definition** Eine **Substitution** ist ein Homomorphismus $s : \Sigma^* \longrightarrow 2^{\Gamma^*}$, wobei $\Sigma$ und $\Gamma$ Alphabete sind und $s$ folgende Eigenschaften besitzt:

  1. Für jedes $a \in \Sigma$ ist $s(a) \subseteq \Gamma^*$ definiert.

# Substitution

- **Definition** Eine **Substitution** ist ein Homomorphismus $s : \Sigma^* \longrightarrow 2^{\Gamma^*}$, wobei $\Sigma$ und $\Gamma$ Alphabete sind und $s$ folgende Eigenschaften besitzt:
  1. Für jedes $a \in \Sigma$ ist $s(a) \subseteq \Gamma^*$ definiert.
  2. $s(\lambda) := \{\lambda\}$.

# Substitution

- **Definition** Eine **Substitution** ist ein Homomorphismus $s : \Sigma^* \longrightarrow 2^{\Gamma^*}$, wobei $\Sigma$ und $\Gamma$ Alphabete sind und $s$ folgende Eigenschaften besitzt:

  1. Für jedes $a \in \Sigma$ ist $s(a) \subseteq \Gamma^*$ definiert.
  2. $s(\lambda) := \{\lambda\}$.
  3. $\forall u, v \in \Sigma^* : s(u \cdot v) = s(u) \cdot s(v)$

# Substitution

- **Definition** Eine **Substitution** ist ein Homomorphismus $s : \Sigma^* \longrightarrow 2^{\Gamma^*}$, wobei $\Sigma$ und $\Gamma$ Alphabete sind und $s$ folgende Eigenschaften besitzt:

    1. Für jedes $a \in \Sigma$ ist $s(a) \subseteq \Gamma^*$ definiert.
    2. $s(\lambda) := \{\lambda\}$.
    3. $\forall u, v \in \Sigma^* : s(u \cdot v) = s(u) \cdot s(v)$

- Ist $s(a)$ regulär (bzw. endlich) für jedes $a \in \Sigma$, dann heißt $s$ **reguläre** bzw. **endliche** Substitution.

# Substitution

- **Definition** Eine **Substitution** ist ein Homomorphismus $s : \Sigma^* \longrightarrow 2^{\Gamma^*}$, wobei $\Sigma$ und $\Gamma$ Alphabete sind und $s$ folgende Eigenschaften besitzt:
  1. Für jedes $a \in \Sigma$ ist $s(a) \subseteq \Gamma^*$ definiert.
  2. $s(\lambda) := \{\lambda\}$.
  3. $\forall u, v \in \Sigma^* : s(u \cdot v) = s(u) \cdot s(v)$
- Ist $s(a)$ regulär (bzw. endlich) für jedes $a \in \Sigma$, dann heißt $s$ **reguläre** bzw. **endliche** Substitution.
- kanonische Erweiterung:

$$s(L) := \bigcup_{w \in L} s(w)$$

# Beispiel: Substitution

- Seien

$$s : \{0,1\}^* \longrightarrow 2^{\{a,b\}^*} \text{ mit } 0 \mapsto \{a\}, \quad 1 \mapsto \{b\}^*$$

$$s' : \{0,1\}^* \longrightarrow 2^{\{a,b\}^*} \text{ mit } 0 \mapsto (ab)^+a, \quad 1 \mapsto \emptyset$$

# Beispiel: Substitution

- Seien

$$s : \{0,1\}^* \longrightarrow 2^{\{a,b\}^*} \text{ mit } 0 \mapsto \{a\}, \quad 1 \mapsto \{b\}^*$$

$$s' : \{0,1\}^* \longrightarrow 2^{\{a,b\}^*} \text{ mit } 0 \mapsto (ab)^+a, \quad 1 \mapsto \emptyset$$

- $s(01) = s(0)s(1) = ab^* = \{a, ab, abb, abbb, \ldots\}$

# Beispiel: Substitution

- Seien

$$s : \{0,1\}^* \longrightarrow 2^{\{a,b\}^*} \text{ mit } 0 \mapsto \{a\}, \quad 1 \mapsto \{b\}^*$$

$$s' : \{0,1\}^* \longrightarrow 2^{\{a,b\}^*} \text{ mit } 0 \mapsto (ab)^+a, \quad 1 \mapsto \emptyset$$

- $s(01) = s(0)s(1) = ab^* = \{a, ab, abb, abbb, \ldots\}$
- $s'(01) = s'(0)s'(1) = (ab)^+a \cdot \emptyset = \emptyset$

# Beispiel: Substitution

- Seien

$$s : \{0,1\}^* \longrightarrow 2^{\{a,b\}^*} \text{ mit } 0 \mapsto \{a\}, \quad 1 \mapsto \{b\}^*$$

$$s' : \{0,1\}^* \longrightarrow 2^{\{a,b\}^*} \text{ mit } 0 \mapsto (ab)^+a, \quad 1 \mapsto \emptyset$$

- $s(01) = s(0)s(1) = ab^* = \{a, ab, abb, abbb, \ldots\}$
- $s'(01) = s'(0)s'(1) = (ab)^+a \cdot \emptyset = \emptyset$
- $s(0^*(0+1) + 1^*) = a^*(a + b^*) + (b^*)^* =$
  $a^+ + a^*b^* + b^* = a^*b^*$

# Beispiel: Substitution

- Seien

$$s : \{0,1\}^* \longrightarrow 2^{\{a,b\}^*} \text{ mit } 0 \mapsto \{a\}, \quad 1 \mapsto \{b\}^*$$

$$s' : \{0,1\}^* \longrightarrow 2^{\{a,b\}^*} \text{ mit } 0 \mapsto (ab)^+ a, \quad 1 \mapsto \emptyset$$

- $s(01) = s(0)s(1) = ab^* = \{a, ab, abb, abbb, \ldots\}$
- $s'(01) = s'(0)s'(1) = (ab)^+ a \cdot \emptyset = \emptyset$
- $s(0^*(0+1)+1^*) = a^*(a+b^*) + (b^*)^* =$
  $a^+ + a^*b^* + b^* = a^*b^*$
- $s'(0^*(0+1)+1^*) = ((ab)^+a)^*((ab)^+a + \emptyset) + \emptyset^* =$
  $((ab)^+a)^+ + \emptyset^* = ((ab)^+a)^*$

# Abschlussop.: reg. Substitution

- **Theorem:** Die Familie der regulären Mengen ist gegenüber regulären Substitutionen abgeschlossen.

# Abschlussop.: reg. Substitution

- **Theorem:** Die Familie der regulären Mengen ist gegenüber regulären Substitutionen abgeschlossen.
- **Beweis:** Jede reguläre Menge $R$ wird durch einen rationalen Ausdruck dargestellt.

# Abschlussop.: reg. Substitution

- **Theorem:** Die Familie der regulären Mengen ist gegenüber regulären Substitutionen abgeschlossen.
- **Beweis:** Jede reguläre Menge $R$ wird durch einen rationalen Ausdruck dargestellt.
  - Ersetzt man nun jedes Symbol $a$ in diesem Ausdruck durch den rationalen Ausdruck der $s(a)$ beschreibt, so ergibt sich ein rationaler Ausdruck für $s(R)$.

# Abschlussop.: reg. Substitution

- **Theorem:** Die Familie der regulären Mengen ist gegenüber regulären Substitutionen abgeschlossen.
- **Beweis:** Jede reguläre Menge $R$ wird durch einen rationalen Ausdruck dargestellt.
  - Ersetzt man nun jedes Symbol $a$ in diesem Ausdruck durch den rationalen Ausdruck der $s(a)$ beschreibt, so ergibt sich ein rationaler Ausdruck für $s(R)$.
  - ... aus dem vorigen **Beispiel:** $s(0^*(0+1)+1^*) = a^*(a+b^*)+(b^*)^*$ ist rationaler Ausdruck.

# Verkürzen von Wörtern

- Substitutionen ersetzen einzelne Symbole durch Wortmengen, also in der Regel sogar durch unendlich viele Wörter, unter denen auch das leere Wort $\lambda$ vorkommen darf.

# Verkürzen von Wörtern

- Substitutionen ersetzen einzelne Symbole durch Wortmengen, also in der Regel sogar durch unendlich viele Wörter, unter denen auch das leere Wort $\lambda$ vorkommen darf.
- Die Umkehrung, nämlich längere (Teil-)Wörter in einer Zeichenkette auf einzelne Symbole zu verkürzen, scheint zunächst nicht so leicht möglich.

# Verkürzen von Wörtern

- Substitutionen ersetzen einzelne Symbole durch Wortmengen, also in der Regel sogar durch unendlich viele Wörter, unter denen auch das leere Wort $\lambda$ vorkommen darf.
- Die Umkehrung, nämlich längere (Teil-)Wörter in einer Zeichenkette auf einzelne Symbole zu verkürzen, scheint zunächst nicht so leicht möglich.
- Mathematisch wird dies durch die Umkehrung eines Homomorphismus, also der Anwendung sogenannter **inverser Homomorphismen** geleistet.

# inverse Homomorphismen

- **Definition:** Seien $\Sigma$ und $\Gamma$ endliche Alphabete sowie $h : \Sigma^* \longrightarrow \Gamma^*$ ein Homomorphismus. Die zu $h$ inverse Funktion heißt **inverser Homomorphismus** und ist gegegen durch $h^{-1} : \Gamma^* \longrightarrow 2^{\Sigma^*}$.

# inverse Homomorphismen

- **Definition:** Seien $\Sigma$ und $\Gamma$ endliche Alphabete sowie $h : \Sigma^* \longrightarrow \Gamma^*$ ein Homomorphismus. Die zu $h$ inverse Funktion heißt **inverser Homomorphismus** und ist gegegen durch $h^{-1} : \Gamma^* \longrightarrow 2^{\Sigma^*}$.
- $h^{-1}$ definiert für jedes $w \in \Gamma^*$ die Menge aller möglichen Urbilder von $w$ durch:

$$h^{-1}(w) := \{v \in \Sigma^* \mid \exists w \in \Gamma^* : h(v) = w\}.$$

Diese Menge kann daher auch leer sein.

# inverse Homomorphismen

- **Definition:** Seien $\Sigma$ und $\Gamma$ endliche Alphabete sowie $h : \Sigma^* \longrightarrow \Gamma^*$ ein Homomorphismus. Die zu $h$ inverse Funktion heißt **inverser Homorphismus** und ist gegegen durch $h^{-1} : \Gamma^* \longrightarrow 2^{\Sigma^*}$.
- $h^{-1}$ definiert für jedes $w \in \Gamma^*$ die Menge aller möglichen Urbilder von $w$ durch:

$$h^{-1}(w) := \{v \in \Sigma^* \mid \exists w \in \Gamma^* : h(v) = w\}.$$

  Diese Menge kann daher auch leer sein.
- Erweiterung auf Sprachen:

$$h^{-1}(L) := \bigcup_{w \in L} h^{-1}(w)$$

# Beispiele: inv. Homomorphismen

- Wir definieren zwei Homomorphismen und betrachten die Anwendung auf
  (a) einzelne Wörter, (b) Sprachen.

# Beispiele: inv. Homomorphismen

- Wir definieren zwei Homomorphismen und betrachten die Anwendung auf
  (a) einzelne Wörter, (b) Sprachen.
  - (a) Sei $h : \{a, b, c\}^* \longrightarrow \{x, y\}^*$ definiert durch

    $$a \mapsto xyx, \quad b \mapsto xy, \quad c \mapsto yx.$$

    Dann ist $h^{-1}(xy) = \{b\}$, $h^{-1}(xx) = \emptyset$ und $h^{-1}(xyxyx) = \{ac, ba\}$.

# Beispiele: inv. Homomorphismen

- Wir definieren zwei Homomorphismen und betrachten die Anwendung auf
  (a) einzelne Wörter, (b) Sprachen.
  - (a) Sei $h : \{a, b, c\}^* \longrightarrow \{x, y\}^*$ definiert durch

    $$a \mapsto xyx, \quad b \mapsto xy, \quad c \mapsto yx.$$

    Dann ist $h^{-1}(xy) = \{b\}$, $h^{-1}(xx) = \emptyset$ und $h^{-1}(xyxyx) = \{ac, ba\}$.
  - (b) Sei $f : \{a, b, c\}^* \longrightarrow \{x, y\}^*$ definiert durch

    $$a \mapsto x, \quad b \mapsto y, \quad c \mapsto \lambda.$$

    Dann ist $h^{-1}(\{\lambda\}) = \{c\}^*$,
    $h^{-1}(\{y\}) = \{c\}^*\{b\}\{c\}^*$ und
    $h^{-1}(\{x, y\}^*) = \{a, b, c\}^*$.

# Abschlussoperator: inv. Hom.

- **Theorem:** Sei $L \in \mathcal{A}kz(\Gamma)$ und $h : \Sigma^* \longrightarrow \Gamma^*$ ein Homomorphismus, dann ist $h^{-1}(L) \in \mathcal{A}kz(\Sigma)$.

# Abschlussoperator: inv. Hom.

- **Theorem:** Sei $L \in \mathcal{A}kz(\Gamma)$ und $h : \Sigma^* \longrightarrow \Gamma^*$ ein Homomorphismus, dann ist $h^{-1}(L) \in \mathcal{A}kz(\Sigma)$.
- **Beweis:** Sei $L = L(A)$ für einen vDFA $A = (Z, \Gamma, \delta_1, z_0, Z_{\text{end}})$. Wir konstruieren einen DFA $B = (Z, \Sigma, \delta_2, z_0, Z_{\text{end}})$, der bei Eingabe von $x$ aus $\Sigma$ den vDFA $A$ auf der Eingabe $h(x)$ simuliert.

# Abschlussoperator: inv. Hom.

- **Theorem:** Sei $L \in \mathcal{A}kz(\Gamma)$ und $h : \Sigma^* \longrightarrow \Gamma^*$ ein Homomorphismus, dann ist $h^{-1}(L) \in \mathcal{A}kz(\Sigma)$.
- **Beweis:** Sei $L = L(A)$ für einen vDFA $A = (Z, \Gamma, \delta_1, z_0, Z_{\text{end}})$. Wir konstruieren einen DFA $B = (Z, \Sigma, \delta_2, z_0, Z_{\text{end}})$, der bei Eingabe von $x$ aus $\Sigma$ den vDFA $A$ auf der Eingabe $h(x)$ simuliert.
- Es wird $\delta_2$ definiert durch: $\forall (z, x) \in Z \times \Sigma : \delta_2(z, x) := (z)^{h(x)}$, womit in $B$ der mit $h(x)$ in $A$ erreichte Zustand $(z)^{h(x)}$ eingenommen wird.

# Abschlussoperator: inv. Hom.

- **Theorem:** Sei $L \in \mathcal{A}kz(\Gamma)$ und $h : \Sigma^* \longrightarrow \Gamma^*$ ein Homomorphismus, dann ist $h^{-1}(L) \in \mathcal{A}kz(\Sigma)$.
- **Beweis:** Sei $L = L(A)$ für einen vDFA $A = (Z, \Gamma, \delta_1, z_0, Z_{\text{end}})$. Wir konstruieren einen DFA $B = (Z, \Sigma, \delta_2, z_0, Z_{\text{end}})$, der bei Eingabe von $x$ aus $\Sigma$ den vDFA $A$ auf der Eingabe $h(x)$ simuliert.
- Es wird $\delta_2$ definiert durch: $\forall (z, x) \in Z \times \Sigma : \delta_2(z, x) := (z)^{h(x)}$, womit in $B$ der mit $h(x)$ in $A$ erreichte Zustand $(z)^{h(x)}$ eingenommen wird.
- Ist in $L(A)$ ein Wort $v \in (h(\Sigma))^*$, so wird jedes $u \in \Sigma^*$ mit $h(u) \in (h(\Sigma))^*$ von $B$ akzeptiert.

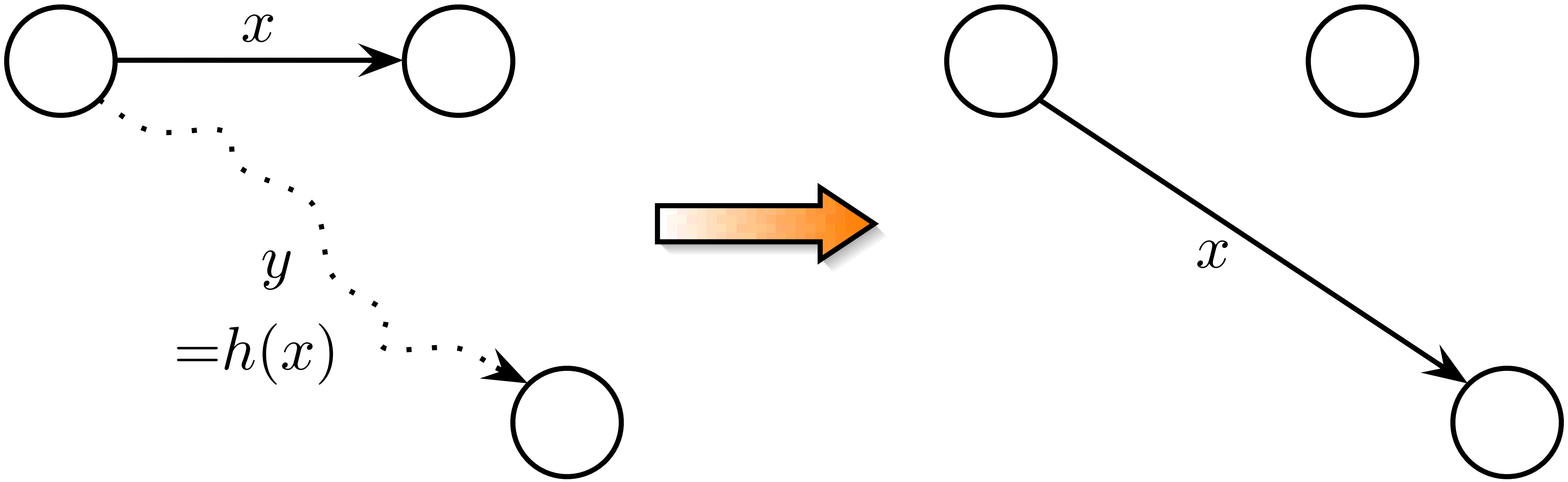
# Abschlussoperator: inv. Hom.

- **Theorem:** Sei $L \in \mathcal{A}kz(\Gamma)$ und $h : \Sigma^* \longrightarrow \Gamma^*$ ein Homomorphismus, dann ist $h^{-1}(L) \in \mathcal{A}kz(\Sigma)$.
- **Beweis:** Sei $L = L(A)$ für einen vDFA $A = (Z, \Gamma, \delta_1, z_0, Z_{\text{end}})$. Wir konstruieren einen DFA $B = (Z, \Sigma, \delta_2, z_0, Z_{\text{end}})$, der bei Eingabe von $x$ aus $\Sigma$ den vDFA $A$ auf der Eingabe $h(x)$ simuliert.
- Es wird $\delta_2$ definiert durch: $\forall (z, x) \in Z \times \Sigma : \delta_2(z, x) := (z)^{h(x)}$, womit in $B$ der mit $h(x)$ in $A$ erreichte Zustand $(z)^{h(x)}$ eingenommen wird.
- Ist in $L(A)$ ein Wort $v \in (h(\Sigma))^*$, so wird jedes $u \in \Sigma^*$ mit $h(u) \in (h(\Sigma))^*$ von $B$ akzeptiert.
- Die Umkehrung ist ebenso einfach.

# Illustration des Beweises

$$w \in L \quad \Leftrightarrow \quad h^{-1}(w) \in h^{-1}(L)$$

$$w \in (h(\Sigma)^*) \subseteq \Gamma^*$$




Die Konstruktion funktioniert, da ein Homomorphismus **strukturerhaltend** ist.

# init. Zusammenhangskomponente

- Sei $A := (Z, \Sigma, \delta, z_0, Z_{\text{end}})$ ein beliebiger DFA. Wir berechnen schrittweise die Mengen $M_i \subseteq Z$:

$$M_i := \begin{cases} M_{i-1} \cup \bigcup\limits_{z \in M_{i-1}, x \in \Sigma} \delta(z, x) & \text{für } i > 0 \\ \{z_0\} & \text{für } i = 0 \end{cases}$$

# init. Zusammenhangskomponente

- Sei $A := (Z, \Sigma, \delta, z_0, Z_{\textsf{end}})$ ein beliebiger DFA. Wir berechnen schrittweise die Mengen $M_i \subseteq Z$:

$$M_i := \begin{cases} M_{i-1} \cup \bigcup\limits_{z \in M_{i-1}, x \in \Sigma} \delta(z, x) & \text{für } i > 0 \\ \{z_0\} & \text{für } i = 0 \end{cases}$$

- Offensichtlich ist jeder Zustand $z \in M_i$, $i \in I\!N$ vom Startzustand aus erreichbar.

# init. Zusammenhangskomponente

- Sei $A := (Z, \Sigma, \delta, z_0, Z_{\text{end}})$ ein beliebiger DFA. Wir berechnen schrittweise die Mengen $M_i \subseteq Z$:

$$M_i := \begin{cases} M_{i-1} \cup \bigcup_{z \in M_{i-1}, x \in \Sigma} \delta(z, x) & \text{für } i > 0 \\ \{z_0\} & \text{für } i = 0 \end{cases}$$

- Offensichtlich ist jeder Zustand $z \in M_i$, $i \in \mathbb{N}$ vom Startzustand aus erreichbar.
- Wegen $M_{i-1} \subseteq M_i \subseteq Z$ und der Endlichkeit von $Z$ existiert ein Index $k \leq |Z|$, mit $M_{k+1} = M_k$.

# init. Zusammenhangskomponente

- Sei $A := (Z, \Sigma, \delta, z_0, Z_{\mathsf{end}})$ ein beliebiger DFA. Wir berechnen schrittweise die Mengen $M_i \subseteq Z$:

$$M_i := \begin{cases} M_{i-1} \cup \bigcup\limits_{z \in M_{i-1}, x \in \Sigma} \delta(z, x) & \text{für } i > 0 \\ \{z_0\} & \text{für } i = 0 \end{cases}$$

- Offensichtlich ist jeder Zustand $z \in M_i$, $i \in I\!N$ vom Startzustand aus erreichbar.
- Wegen $M_{i-1} \subseteq M_i \subseteq Z$ und der Endlichkeit von $Z$ existiert ein Index $k \leq |Z|$, mit $M_{k+1} = M_k$.
- Das Verfahren endet, wenn das erste mal $M_k = M_{k+1}$ ist, spätestens bei $k = |Z| - 1$.

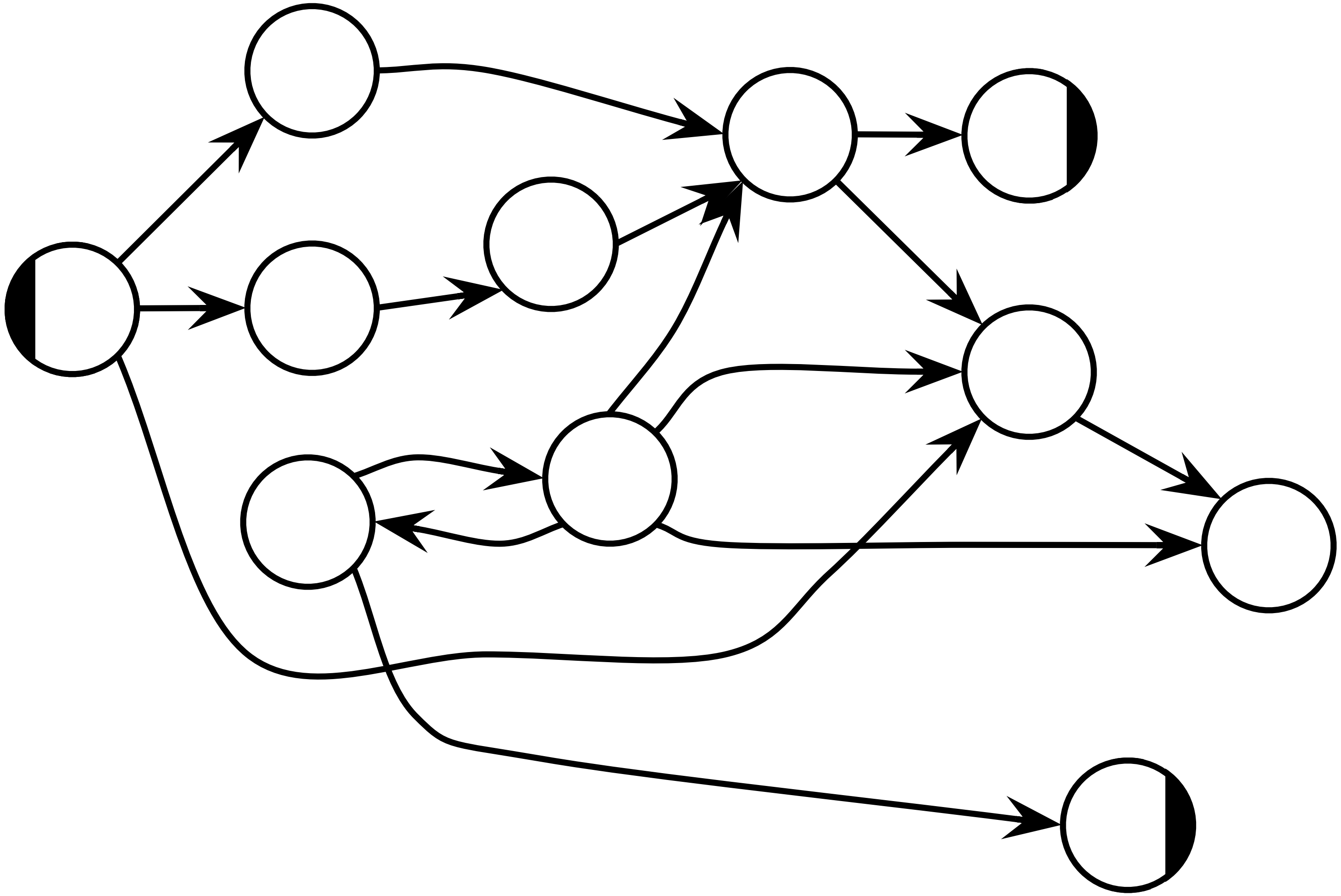
# init. Zusammenhangskomponente

- Sei $A := (Z, \Sigma, \delta, z_0, Z_{\text{end}})$ ein beliebiger DFA. Wir berechnen schrittweise die Mengen $M_i \subseteq Z$:

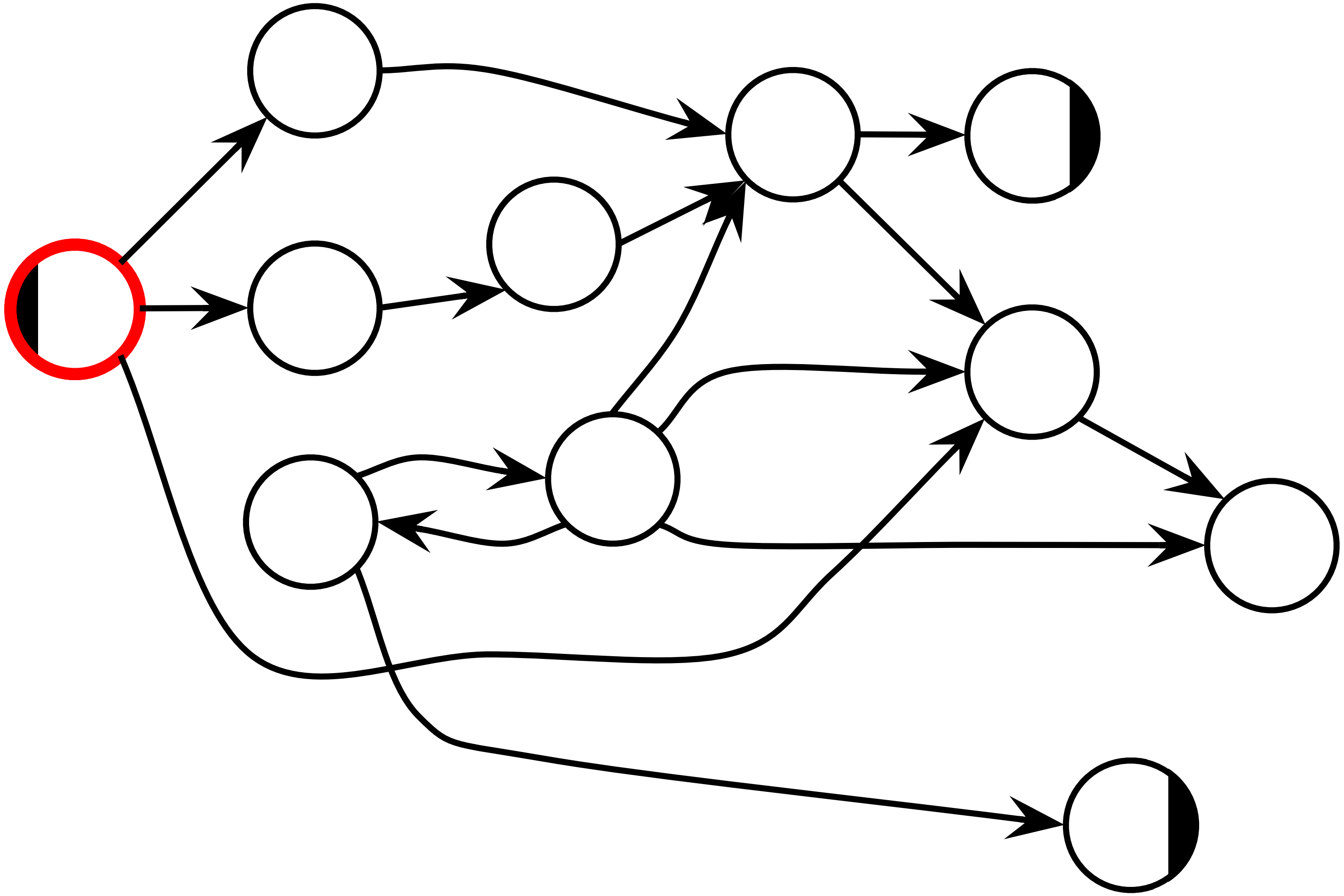
$$M_i := \begin{cases} M_{i-1} \cup \bigcup\limits_{z \in M_{i-1}, x \in \Sigma} \delta(z, x) & \text{für } i > 0 \\ \{z_0\} & \text{für } i = 0 \end{cases}$$

- Offensichtlich ist jeder Zustand $z \in M_i$, $i \in I\!N$ vom Startzustand aus erreichbar.
- Wegen $M_{i-1} \subseteq M_i \subseteq Z$ und der Endlichkeit von $Z$ existiert ein Index $k \leq |Z|$, mit $M_{k+1} = M_k$.
- Das Verfahren endet, wenn das erste mal $M_k = M_{k+1}$ ist, spätestens bei $k = |Z| - 1$.
- Die Menge $M_k$ enthält dann genau die von $z_0$ aus erreichbaren Zustände im DFA $A$.

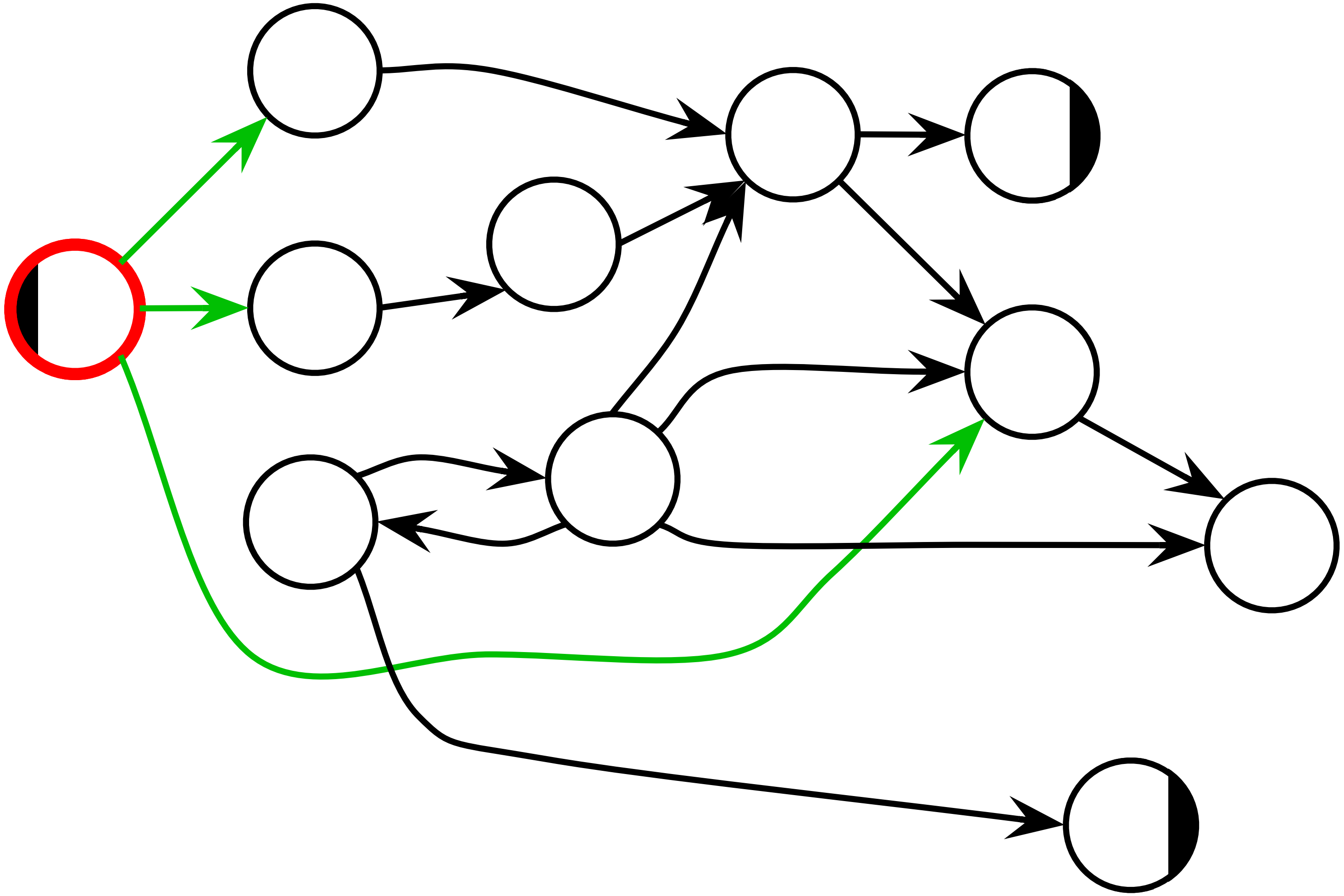
# Anwendung des Algorithmus



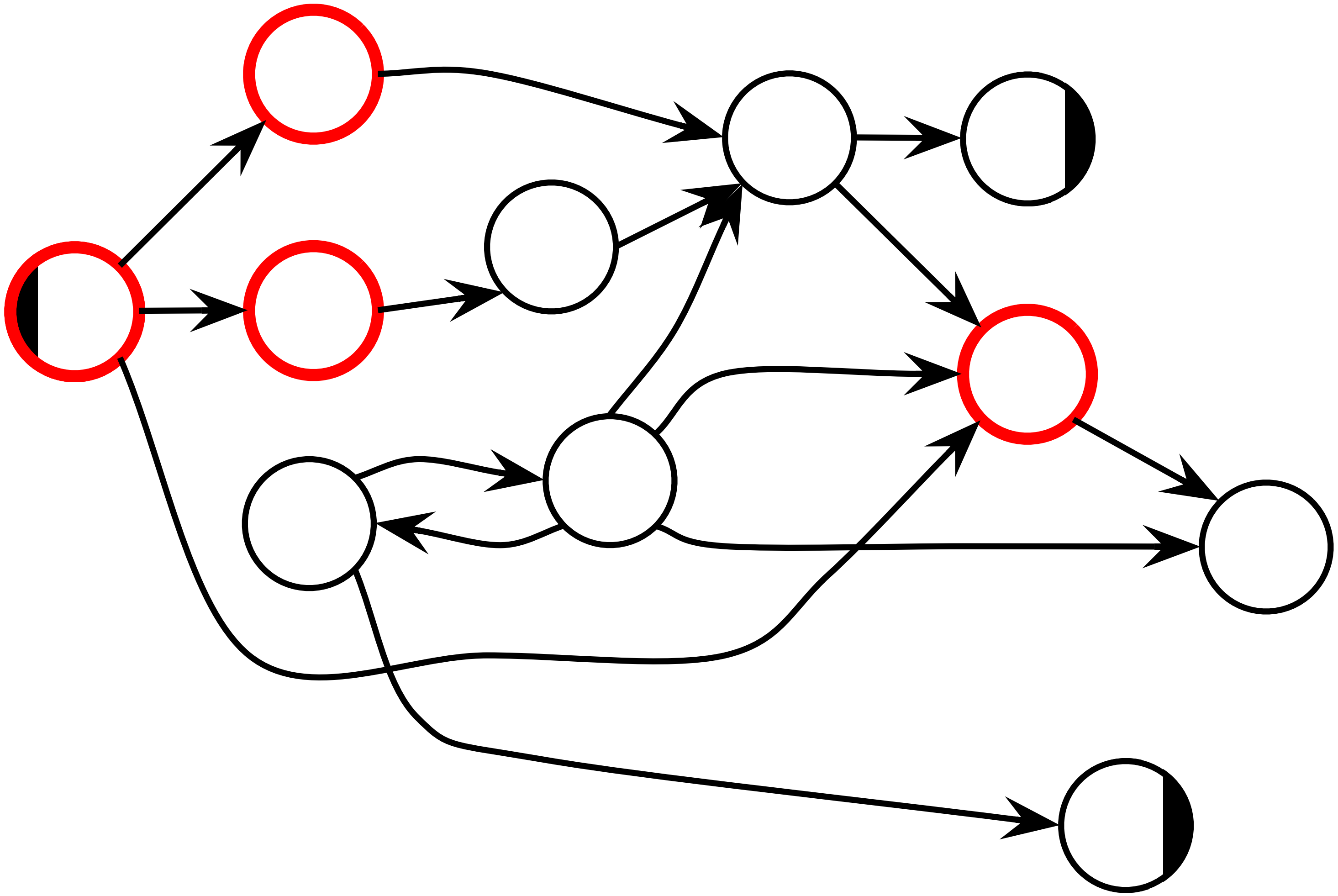
# Anwendung des Algorithmus



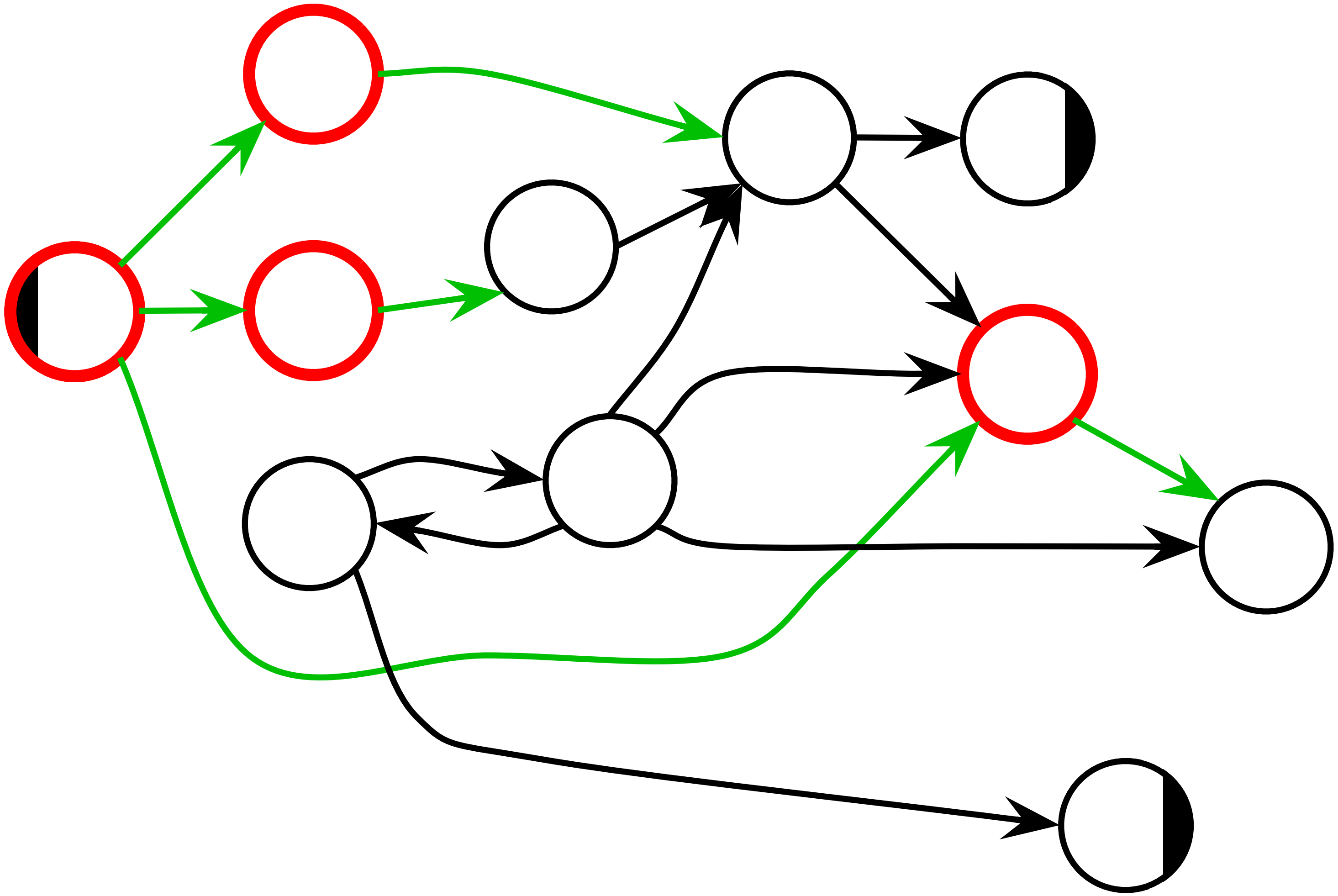
# Anwendung des Algorithmus



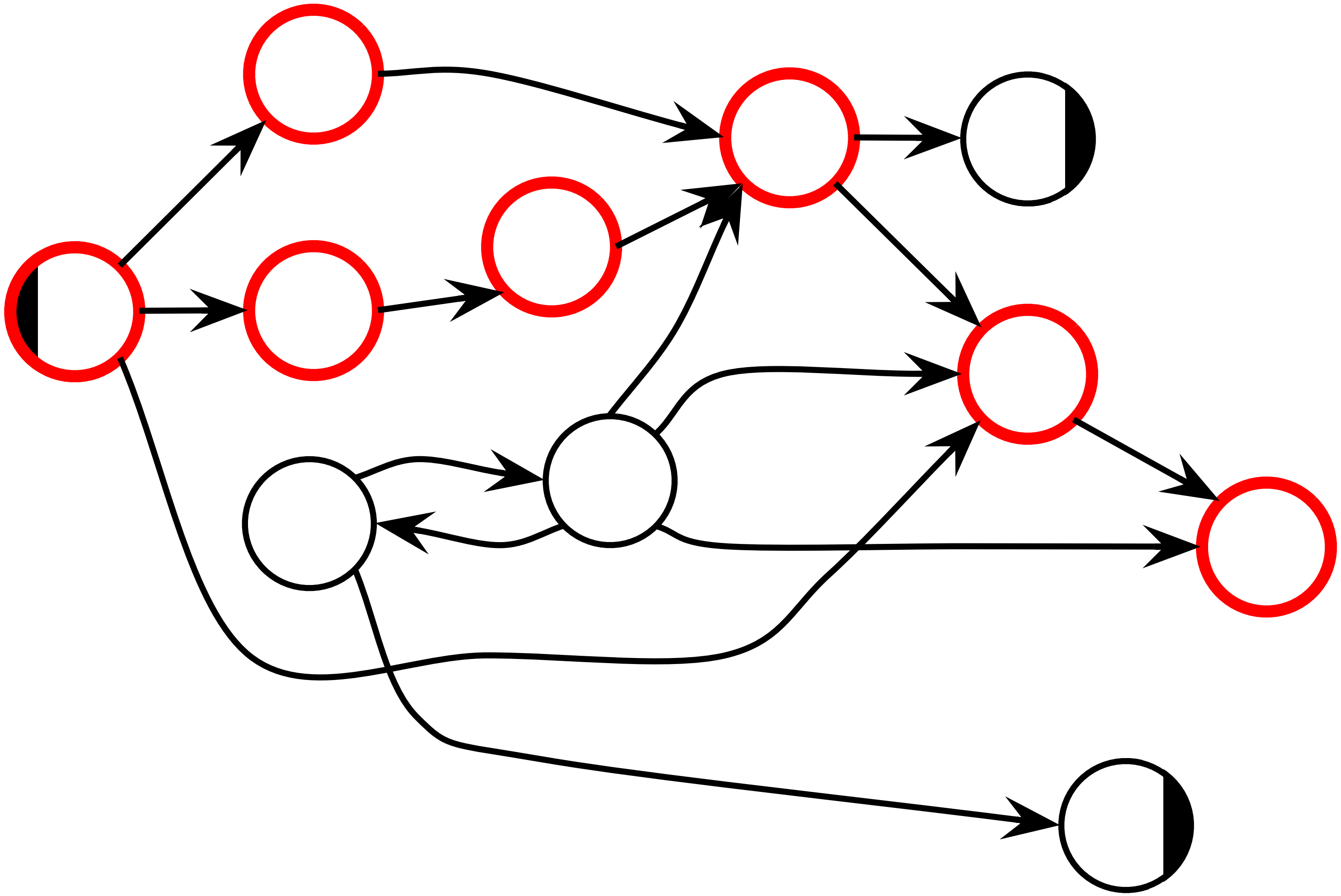
# Anwendung des Algorithmus



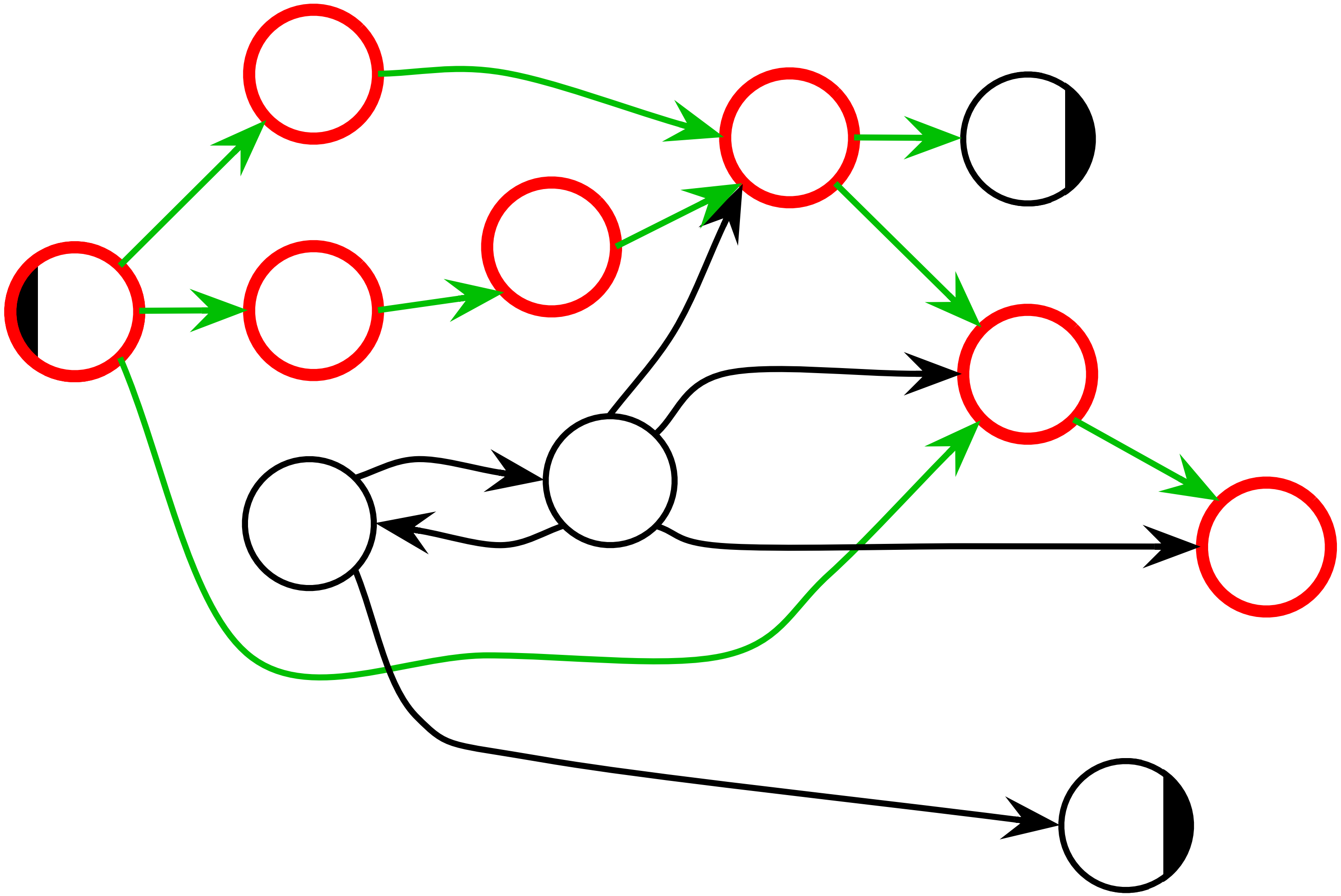
# Anwendung des Algorithmus



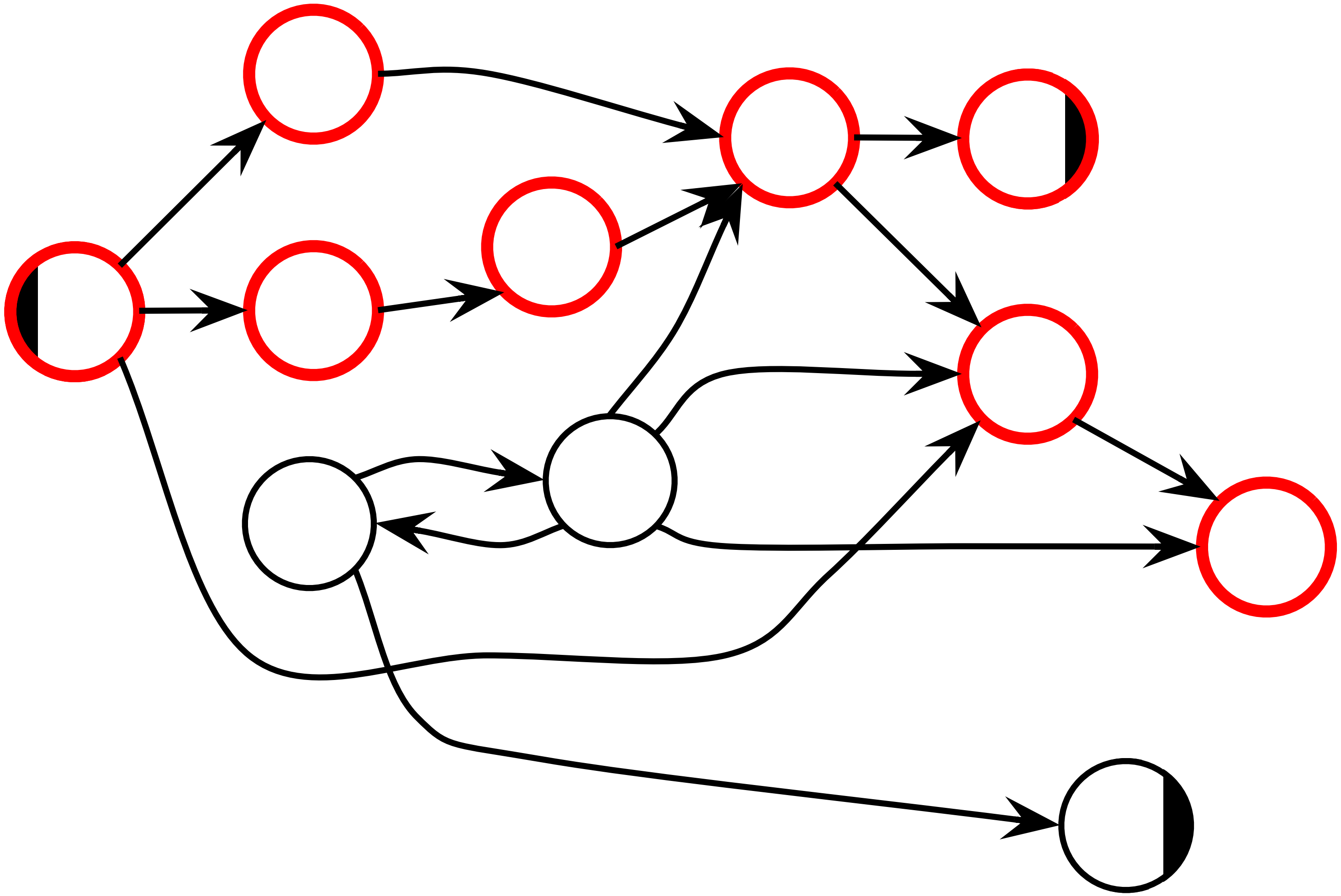
# Anwendung des Algorithmus



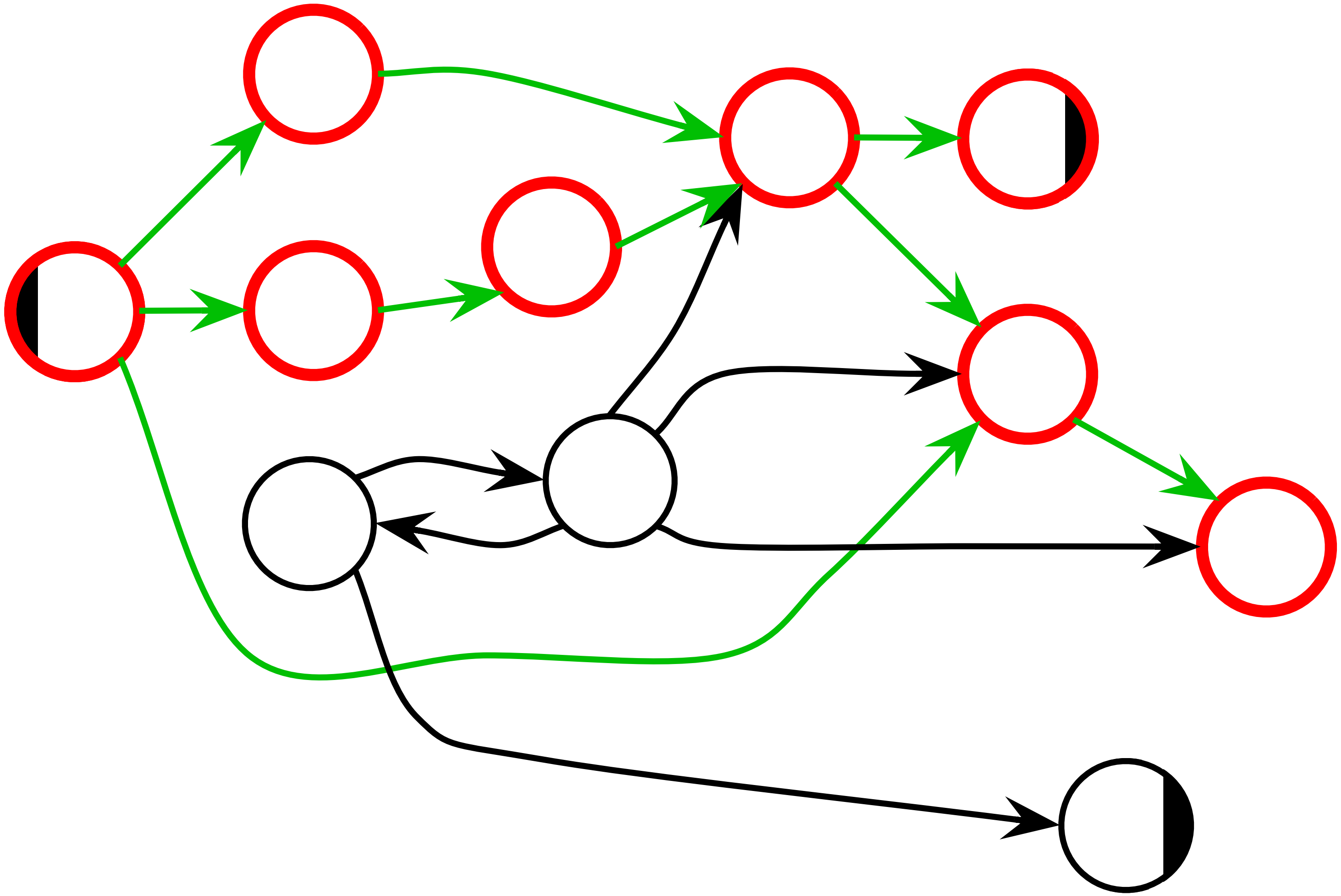
# Anwendung des Algorithmus



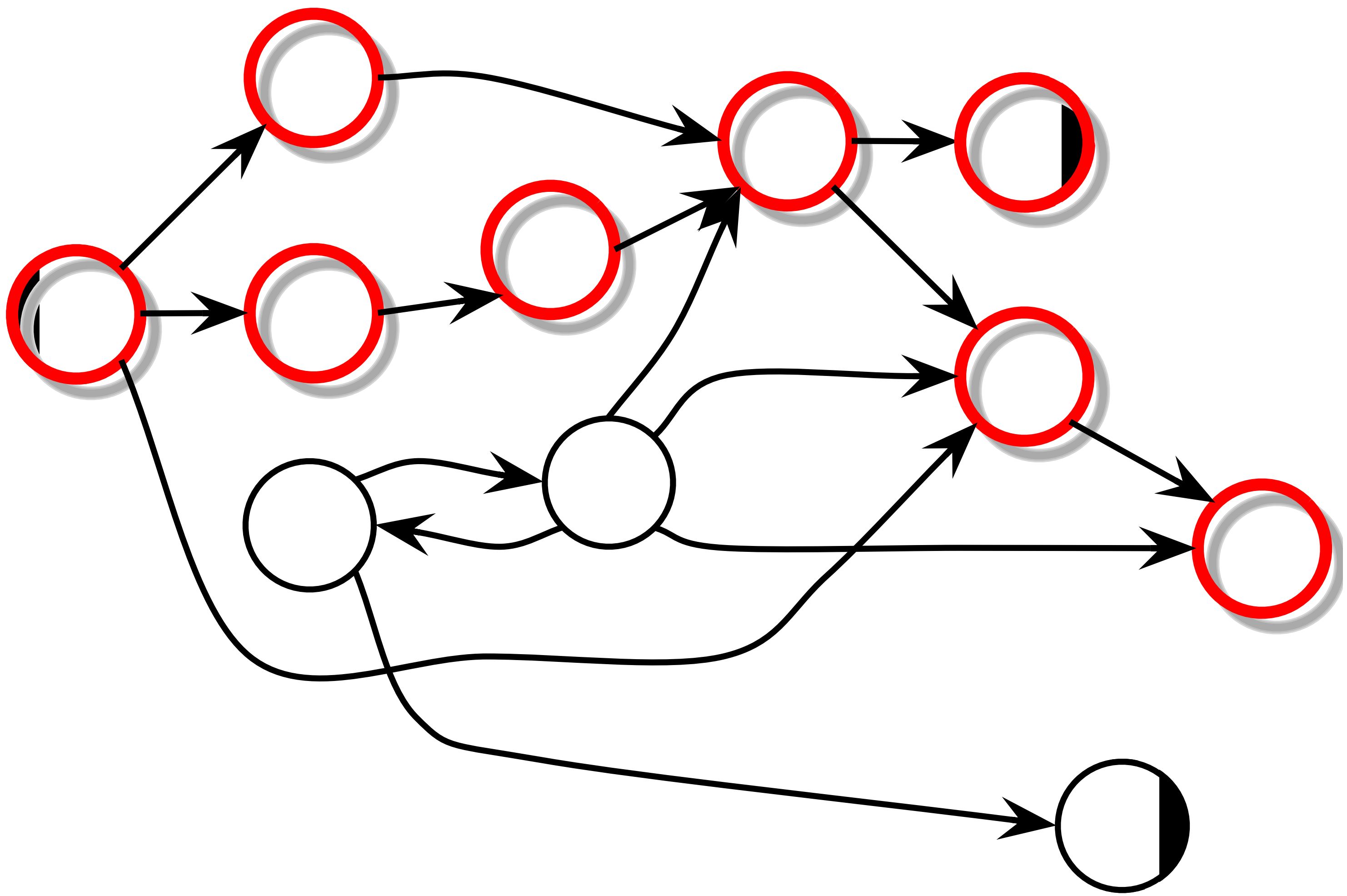
# Anwendung des Algorithmus



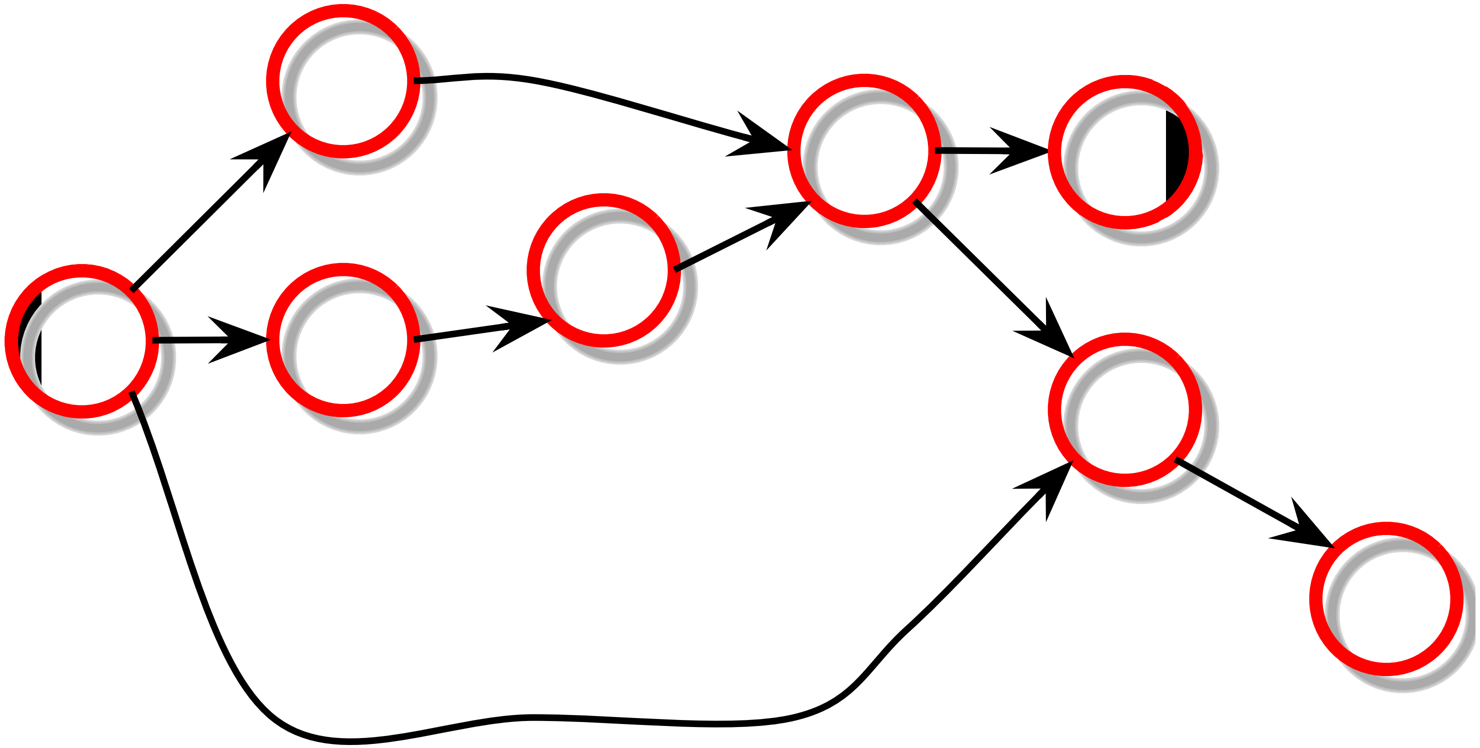
# Anwendung des Algorithmus



# Anwendung des Algorithmus



# Anwendung des Algorithmus



# Grenzen von $\mathcal{R}eg$

- Möchte man zeigen, dass eine Sprache $L$ nicht regulär ist, muss bewiesen werden, dass **kein** FA diese Sprache akzeptiert!

# Grenzen von $\mathcal{R}eg$

- Möchte man zeigen, dass eine Sprache $L$ nicht regulär ist, muss bewiesen werden, dass **kein** FA diese Sprache akzeptiert!
- Wichtiges Hilfsmittel ist das sogenannte Pumping-Lemma oder „uvw-Theorem“.

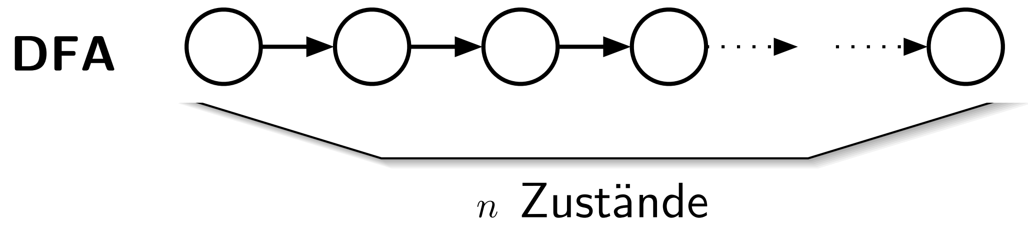
# Grenzen von $\mathcal{R}eg$

- Möchte man zeigen, dass eine Sprache $L$ nicht regulär ist, muss bewiesen werden, dass **kein** FA diese Sprache akzeptiert!
- Wichtiges Hilfsmittel ist das sogenannte Pumping-Lemma oder „uvw-Theorem".
- Es trifft eine Aussage über den Aufbau langer Wörter aus einer regulären Sprache.

# Grenzen von $\mathcal{R}eg$

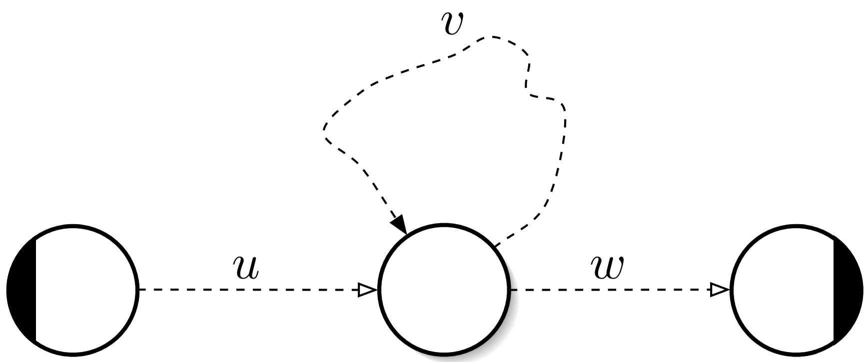
- Möchte man zeigen, dass eine Sprache $L$ nicht regulär ist, muss bewiesen werden, dass **kein** FA diese Sprache akzeptiert!
- Wichtiges Hilfsmittel ist das sogenannte Pumping-Lemma oder „uvw-Theorem“.
- Es trifft eine Aussage über den Aufbau langer Wörter aus einer regulären Sprache.
- Kann man zeigen, dass gemäß Pumping-Lemma Wörter in der Sprache enthalten sein müßten, sie es aber nicht sind, so wissen wir, dass die Sprache nicht regulär sein kann!

# Motivation: Pumping-Lemma




Ein akzeptiertes Wort mit Länge $\geq n$ muss auf seiner Erfolgsrechnung mindestens $n+1$ Zustände besuchen.

Dazu muss eine Schleife durchlaufen werden!




Dann wird aber auch jedes Wort der Form $uv^iw$ akzeptiert.

# Pumping-Lemma für $\mathcal{R}eg$

- **Theorem:** Zu jeder regulären Menge $R \in \mathcal{R}eg$ gibt es eine Zahl $n \in I\!N$, so dass jedes Wort $z \in R$ mit $|z| \geq n$ zerlegt werden kann in die Form $z = uvw$ mit:

# Pumping-Lemma für $\mathcal{R}eg$

- **Theorem:** Zu jeder regulären Menge $R \in \mathcal{R}eg$ gibt es eine Zahl $n \in I\!N$, so dass jedes Wort $z \in R$ mit $|z| \geq n$ zerlegt werden kann in die Form $z = uvw$ mit:

  (i) $|uv| \leq n$

# Pumping-Lemma für $\mathcal{R}eg$

- **Theorem:** Zu jeder regulären Menge $R \in \mathcal{R}eg$ gibt es eine Zahl $n \in I\!N$, so dass jedes Wort $z \in R$ mit $|z| \geq n$ zerlegt werden kann in die Form $z = uvw$ mit:

  (i) $|uv| \leq n$

  (ii) $|v| \geq 1$ und

# Pumping-Lemma für $\mathcal{R}eg$

- **Theorem:** Zu jeder regulären Menge $R \in \mathcal{R}eg$ gibt es eine Zahl $n \in I\!N$, so dass jedes Wort $z \in R$ mit $|z| \geq n$ zerlegt werden kann in die Form $z = uvw$ mit:
  (i) $|uv| \leq n$
  (ii) $|v| \geq 1$ und
  (iii) $\{u\}\{v\}^*\{w\} \subseteq R$

# Beweis: Pumping-Lemma

- Sei $A = (Z, \Sigma, K, z_0, Z_{\mathsf{end}})$ ein DFA der $R$ akzeptiert, d.h.: $L(A) = R$.

# Beweis: Pumping-Lemma

- Sei $A = (Z, \Sigma, K, z_0, Z_{\mathsf{end}})$ ein DFA der $R$ akzeptiert, d.h.: $L(A) = R$.
- Wähle $n := |Z|$. Sei nun $z = x_1 x_2 \ldots x_m \in R$, mit $m \geq n$ und $x_i \in \Sigma$, ein beliebiges Wort der Sprache.

# Beweis: Pumping-Lemma

- Sei $A = (Z, \Sigma, K, z_0, Z_{\text{end}})$ ein DFA der $R$ akzeptiert, d.h.: $L(A) = R$.
- Wähle $n := |Z|$. Sei nun $z = x_1 x_2 \ldots x_m \in R$, mit $m \geq n$ und $x_i \in \Sigma$, ein beliebiges Wort der Sprache.
- Die $m + 1$ Zustände $z_0, (z_0)^{x_1}, (z_0)^{x_1 x_2}, \ldots, (z_0)^{x_1 x_2 \ldots x_m}$ sind von $z_0$ aus erreichbar und werden auf diesem Erfolgspfad für die Akzeptierung von $z$ besucht.

# Beweis: Pumping-Lemma

- Sei $A = (Z, \Sigma, K, z_0, Z_{\text{end}})$ ein DFA der $R$ akzeptiert, d.h.: $L(A) = R$.
- Wähle $n := |Z|$. Sei nun $z = x_1 x_2 \dots x_m \in R$, mit $m \geq n$ und $x_i \in \Sigma$, ein beliebiges Wort der Sprache.
- Die $m + 1$ Zustände $z_0, (z_0)^{x_1}, (z_0)^{x_1 x_2}, \dots, (z_0)^{x_1 x_2 \dots x_m}$ sind von $z_0$ aus erreichbar und werden auf diesem Erfolgspfad für die Akzeptierung von $z$ besucht.
- Da maximal $n \leq m$ viele davon verschieden sein können, muss nach dem **Schubfachprinzip** spätestens mit dem $(n+1)$-ten, ein Zustand in dieser Folge zweimal vorgekommen sein.

# Beweis: Pumping-Lemma (2)

- Sei also für $0 \leq i < j \leq n$ gerade $(z_0)^{x_1 x_2 \ldots x_i} = (z_0)^{x_1 x_2 \ldots x_j}$.

# Beweis: Pumping-Lemma (2)

- Sei also für $0 \leq i < j \leq n$ gerade $(z_0)^{x_1 x_2 \dots x_i} = (z_0)^{x_1 x_2 \dots x_j}$.
- Dann finden wir mit $u := x_1 x_2 \dots x_i$, $v := x_{i+1} x_{i+2} \dots x_j$ und $w := x_{j+1} x_{j+2} \dots x_m$ gerade die verlangte Zerlegung von $z$, die (i) und (ii) erfüllt.

# Beweis: Pumping-Lemma (2)

- Sei also für $0 \leq i < j \leq n$ gerade $(z_0)^{x_1 x_2 \ldots x_i} = (z_0)^{x_1 x_2 \ldots x_j}$.
- Dann finden wir mit $u := x_1 x_2 \ldots x_i$, $v := x_{i+1} x_{i+2} \ldots x_j$ und $w := x_{j+1} x_{j+2} \ldots x_m$ gerade die verlangte Zerlegung von $z$, die (i) und (ii) erfüllt.
- Dass auch (iii) gilt, sieht man leicht ein, denn mit $z' := (z_0)^u$ gilt doch $z' = (z')^\lambda = (z')^v = (z')^{vv} = \ldots$. Also werden alle Wörter der Form $u \cdot v^i \cdot w$, für jedes $i \geq 0$, von $A$ ebenfalls akzeptiert.

# Beispiel: Pumping-Lemma

- Wir wollen mit Hilfe des uvw-Theorems zeigen, dass die Sprache $DUP := \{a^n b^n \mid n \in I\!N\}$ nicht regulär ist:

# Beispiel: Pumping-Lemma

- Wir wollen mit Hilfe des uvw-Theorems zeigen, dass die Sprache $DUP := \{a^n b^n \mid n \in I\!N\}$ nicht regulär ist:
- Wäre $DUP$ regulär, so gäbe es eine Zahl $k \in I\!N$, für die die Folgerung des $uvw$-Theorems zutrifft.

# Beispiel: Pumping-Lemma

- Wir wollen mit Hilfe des uvw-Theorems zeigen, dass die Sprache $DUP := \{a^n b^n \mid n \in \mathbb{N}\}$ nicht regulär ist:
- Wäre $DUP$ regulär, so gäbe es eine Zahl $k \in \mathbb{N}$, für die die Folgerung des $uvw$-Theorems zutrifft.
- $z := a^k b^k \in DUP$ ist ein Wort mit $|z| > k$.

# Beispiel: Pumping-Lemma

- Wir wollen mit Hilfe des uvw-Theorems zeigen, dass die Sprache $DUP := \{a^n b^n \mid n \in I\!N\}$ nicht regulär ist:
- Wäre $DUP$ regulär, so gäbe es eine Zahl $k \in I\!N$, für die die Folgerung des $uvw$-Theorems zutrifft.
- $z := a^k b^k \in DUP$ ist ein Wort mit $|z| > k$.
- Jede Zerlegung $z = uvw$ mit $|uv| \leq k$ bedeutet $v = a^m$ für ein $m \in I\!N$ mit $1 \leq m \leq k$.

# Beispiel: Pumping-Lemma

- Wir wollen mit Hilfe des uvw-Theorems zeigen, dass die Sprache $DUP := \{a^n b^n \mid n \in \mathbb{N}\}$ nicht regulär ist:
- Wäre $DUP$ regulär, so gäbe es eine Zahl $k \in \mathbb{N}$, für die die Folgerung des $uvw$-Theorems zutrifft.
- $z := a^k b^k \in DUP$ ist ein Wort mit $|z| > k$.
- Jede Zerlegung $z = uvw$ mit $|uv| \leq k$ bedeutet $v = a^m$ für ein $m \in \mathbb{N}$ mit $1 \leq m \leq k$.
- Damit müßte auch das Wort $a^{k-m} b^k$ ein Element der Sprache $DUP$ sein, was nicht stimmt.

# Beispiel: Pumping-Lemma

- Wir wollen mit Hilfe des uvw-Theorems zeigen, dass die Sprache $DUP := \{a^n b^n \mid n \in \mathbb{N}\}$ nicht regulär ist:
- Wäre $DUP$ regulär, so gäbe es eine Zahl $k \in \mathbb{N}$, für die die Folgerung des $uvw$-Theorems zutrifft.
- $z := a^k b^k \in DUP$ ist ein Wort mit $|z| > k$.
- Jede Zerlegung $z = uvw$ mit $|uv| \leq k$ bedeutet $v = a^m$ für ein $m \in \mathbb{N}$ mit $1 \leq m \leq k$.
- Damit müßte auch das Wort $a^{k-m}b^k$ ein Element der Sprache $DUP$ sein, was nicht stimmt.
- Damit ist $DUP \notin \mathcal{R}eg$ mit einer weiteren Methode nachgewiesen.

# Tutorial

Click here to show movie