



F2 — Automaten und formale Sprachen

Berndt Farwer

Fachbereich Informatik

AB „Theoretische Grundlagen der Informatik“ (TGI)

Universität Hamburg

farwer@informatik.uni-hamburg.de



Ankündigung



Bibliothek Aktuelles

Woche der Elektronischen Medien vom 10. bis 14. Juni 2002

Informationskompetenz

Wie finde ich die für meine Arbeit wichtigen Informationen?

Internet – und was gibt es noch?

Wie werte ich meine Rechercheergebnisse aus?

Bei den Antworten auf diese und ähnliche Fragen möchten wir Sie gern unterstützen und damit Ihre Informationskompetenz stärken, so dass Sie in der Lage sind, aus dem vielfältigen und unübersichtlichen Angebot die Informationen herauszufiltern und zu verwerten, die für Ihr Vorhaben relevant sind.

Wir wollen die verfügbaren elektronischen Ressourcen vorstellen und dazu animieren, die oftmals sehr teuer eingekauften Angebote für die eigene Lern- und auch Lehrsituation zu nutzen.



Ankündigung

Vorträge Montag, 10.6.2002

14.00 Uhr, Vogt-Kölln-Str. 30, Raum D125

Dr. Stefan Gradmann (RRZ / VCB)

Thema: **Die Projekte GAP und FIGARO:**

Innovative Techniken des Webpublishing

Dienstag, 11.6.2002

14.00 Uhr, Vogt-Kölln-Str. 30, Raum D125

Thorsten Ahlers (SUB Hamburg)

Thema: **Elektronische Zeitschriften : Der Weg zum Volltext**

Donnerstag, 13.6.2002

14.00 Uhr, Vogt-Kölln-Str. 30, Raum D125

Thomas Hapke (TUB Harburg)

Thema: **Effektive Suchstrategien in Datenbanken**

Praktische Unterstützung bei konkreten Fragen **Montag, 10.6. bis Freitag, 14.6.2002**
10.00 Uhr und 12.00 Uhr
Info-Raum in der Bibliothek



Ankündigung

Weiteres Angebot **Dienstag, 11.6.2002**
INSPEC-Workshop im Fachbereich Physik

10.00 bis 12.00 Uhr
Jungiusstr. 11, Eingang A, Hörsaal AP
Überblick und Einführung

13.30 bis 15.30 Uhr
Jungiusstr. 13, Raum 314 (CIP-Pool)
Praktische Übungen

Buchausstellung in der **Thema "Elektronische Medien und Kunst"**
Bibliothek Präsentiert wird eine Auswahl der in der
Informatik-Bibliothek vorhandenen Bücher zu
folgenden Gebieten:

Medientheorie Visualisierung, Information

Medientechnik Mediengestaltung, Desktop Publishing, Design

Wissenschaftstheorie, Kunstphilosophie, Bewusstsein, Metaphor
Philosophie

Elektronische Kunst Computerkunst, Prix Ars Electronica

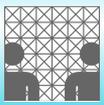
Instructional Design Gebrauchsgrafik, Grafi

Virtual Reality, Cyberspace, Thematische Kartographie,
Cyberspace als Karte Informationssysteme



Themen

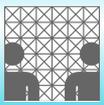
- Für die heutige Vorlesung geplant:
 - initiale Zusammenhangskomponente
 - Grenzen der regulären Sprachen
 - Pumping-Lemma
 - Abschlusseigenschaften
 - Entscheidungsprobleme für endliche Automaten



init. Zusammenhangskomponente

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

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



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.

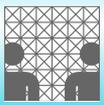


init. Zusammenhangskomponente

- 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 $A := (Z, \Sigma, \delta, z_0, Z_{\text{end}})$ 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 $M_{k+1} = M_k$.
- Das Verfahren endet, wenn das erste mal $M_k = M_{k+1}$ ist, spätestens bei $k = |Z| - 1$.

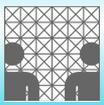


init. Zusammenhangskomponente

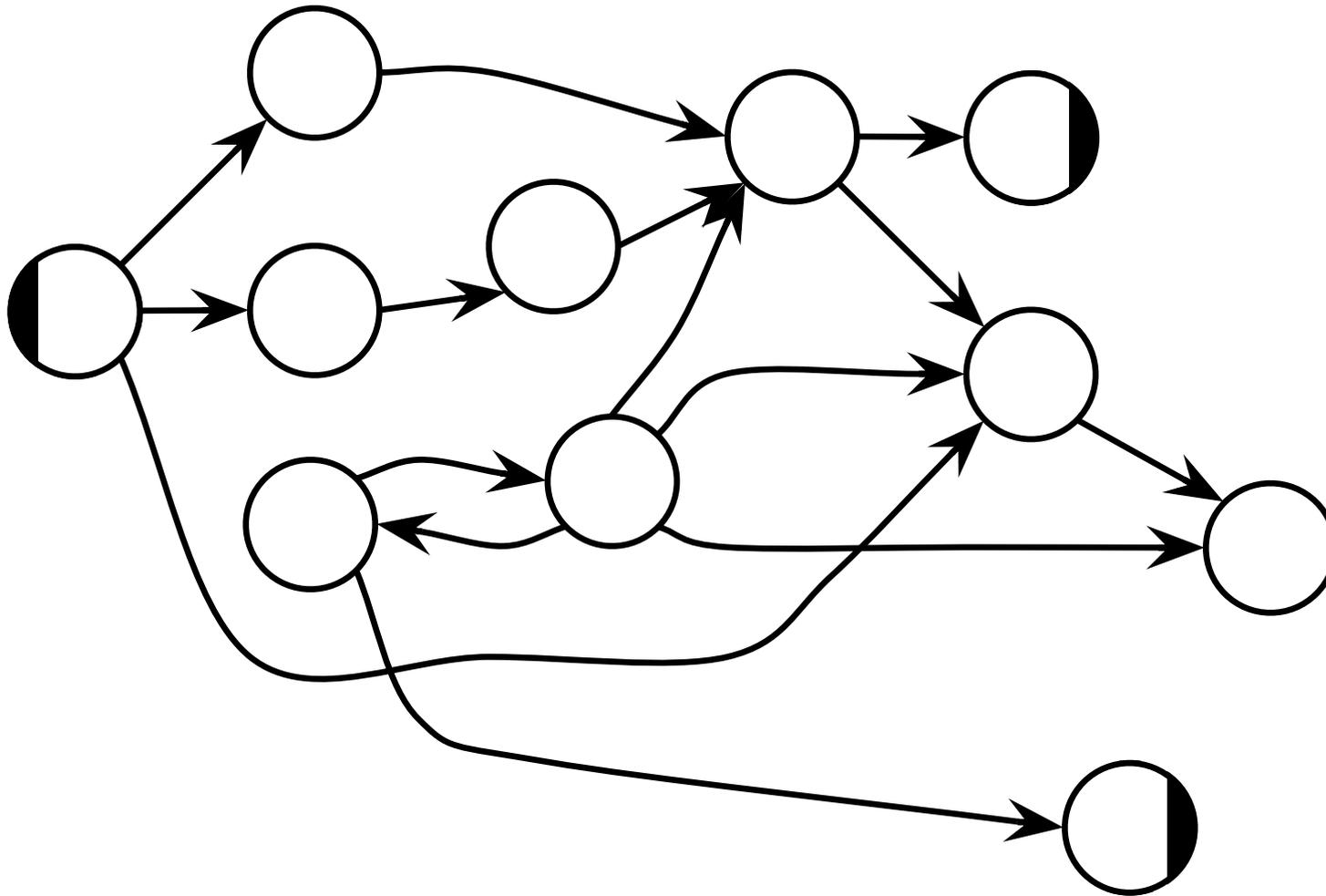
- Sei $A := (Z, \Sigma, \delta, z_0, Z_{\text{end}})$ ein beliebiger DFA. Wir berechnen schrittweise die Mengen $M_i \subseteq Z$:

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

- 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 $M_{k+1} = M_k$.
- 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 .

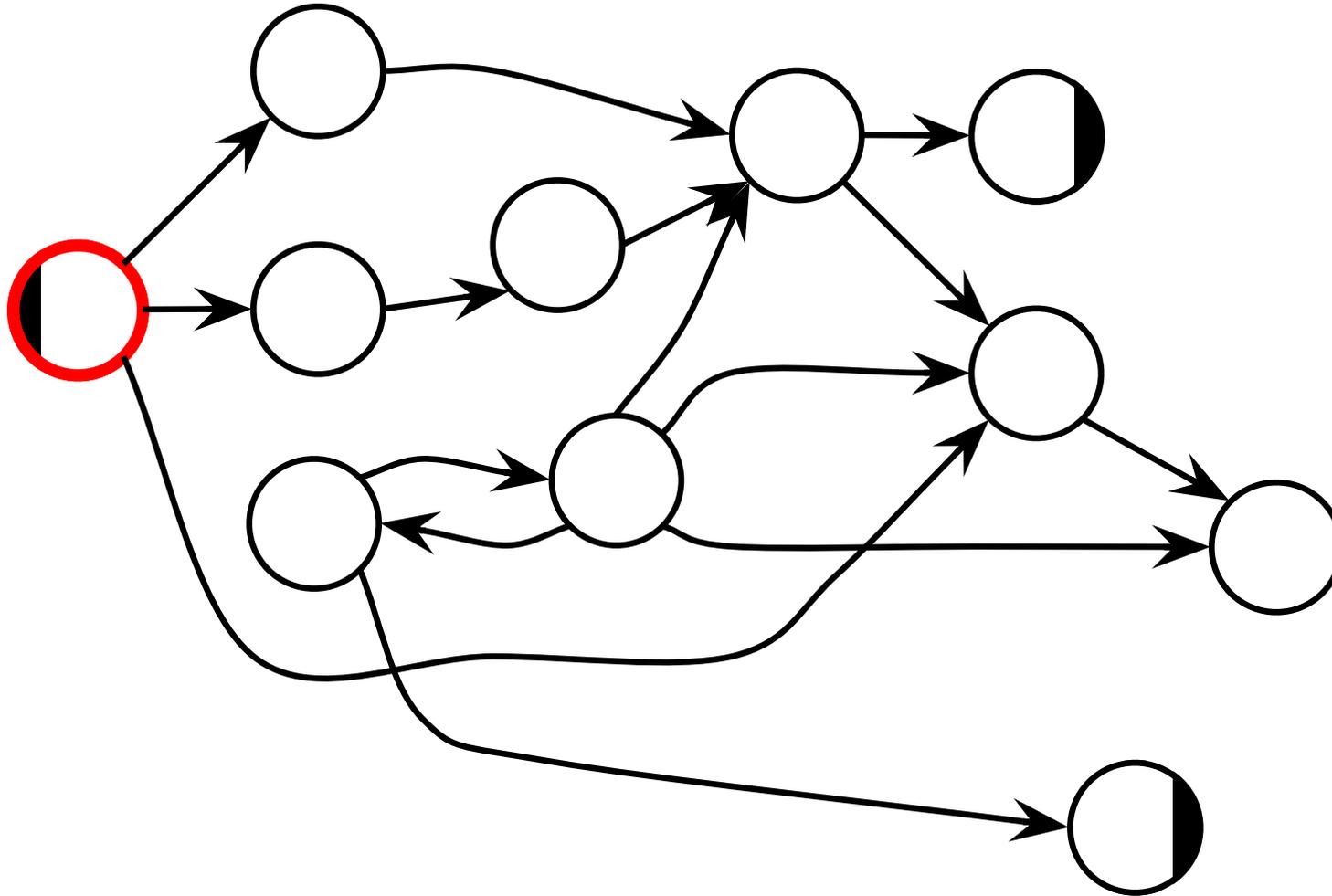


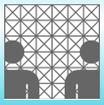
Anwendung des Algorithmus



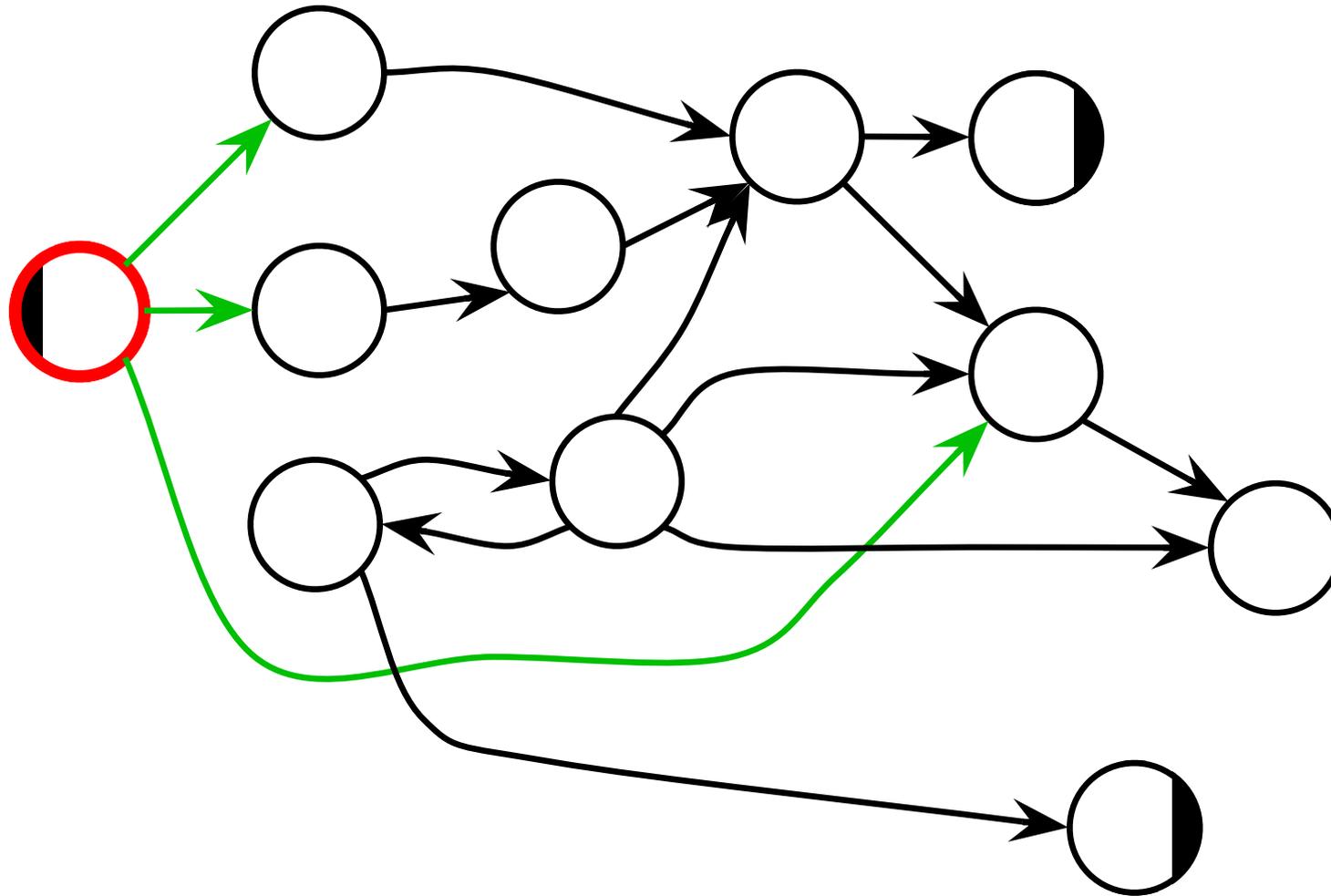


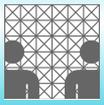
Anwendung des Algorithmus



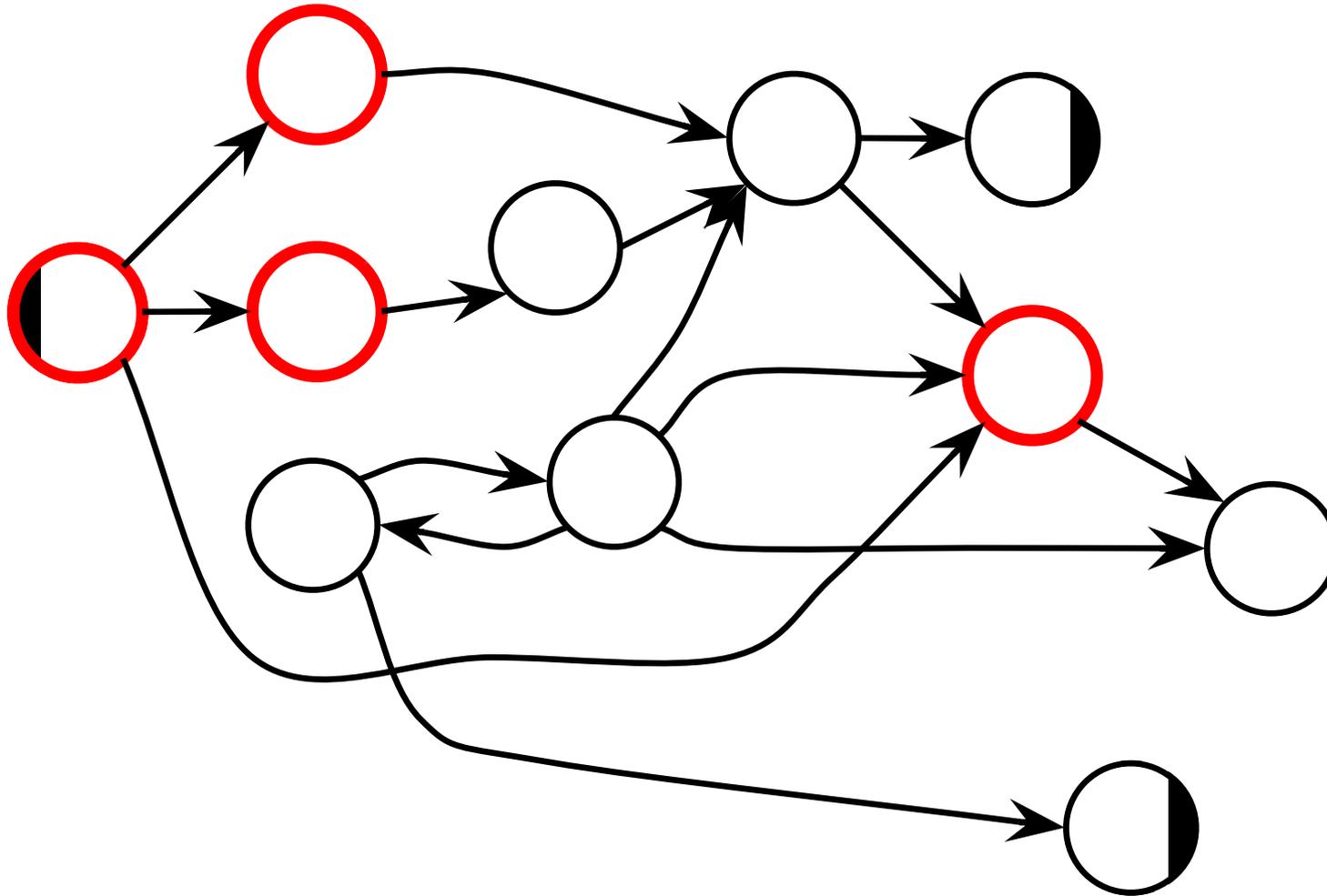


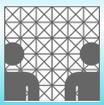
Anwendung des Algorithmus



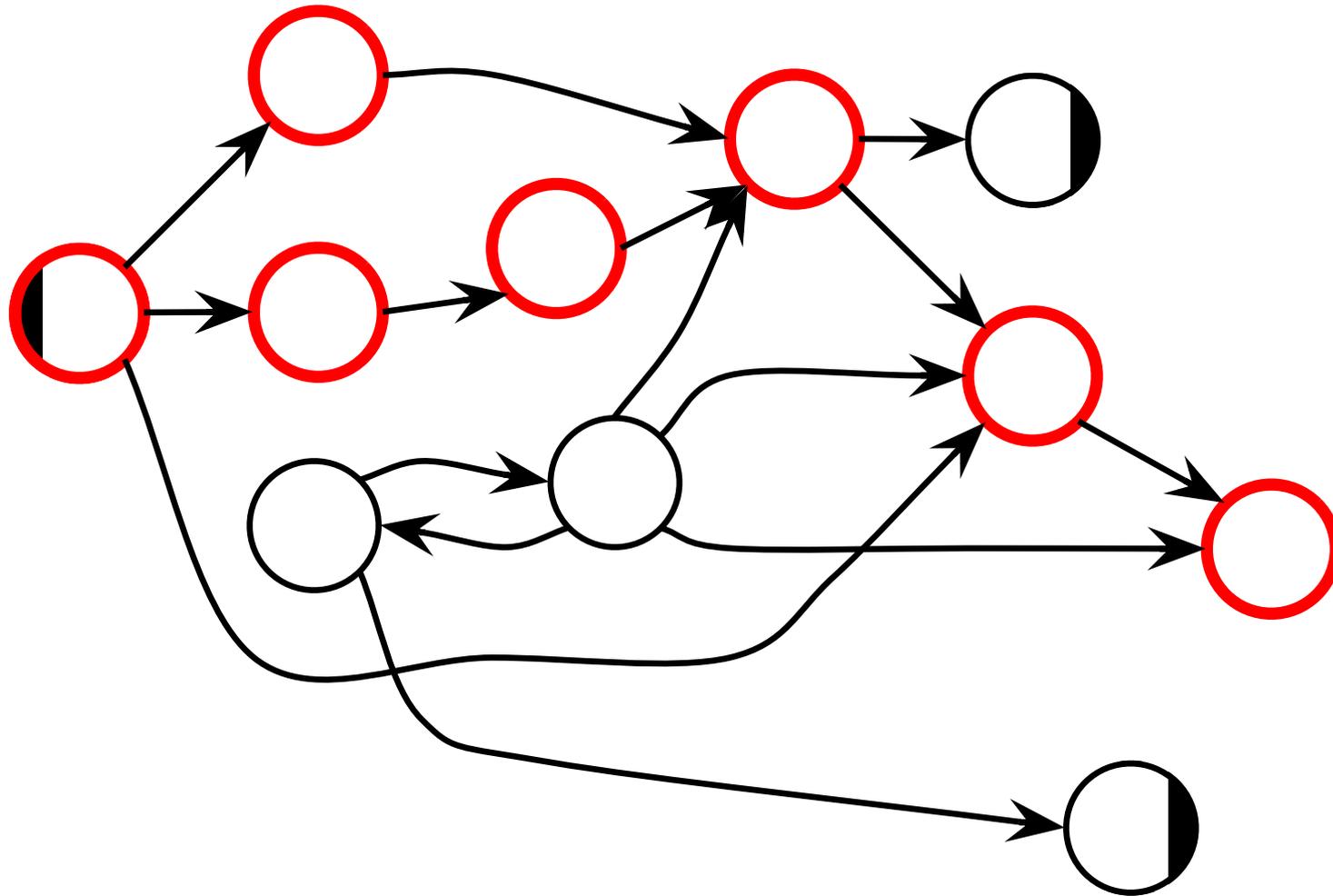


Anwendung des Algorithmus



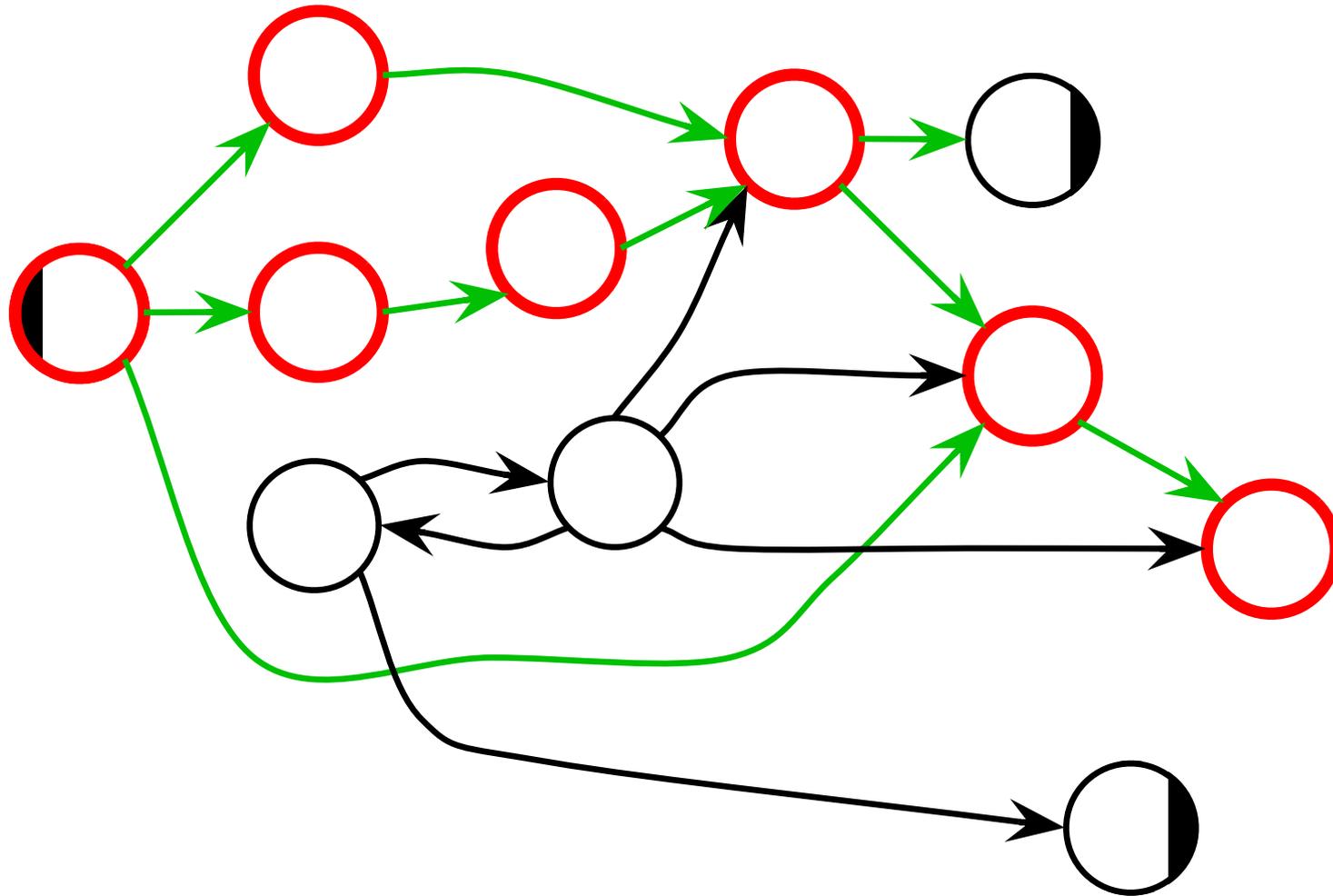


Anwendung des Algorithmus



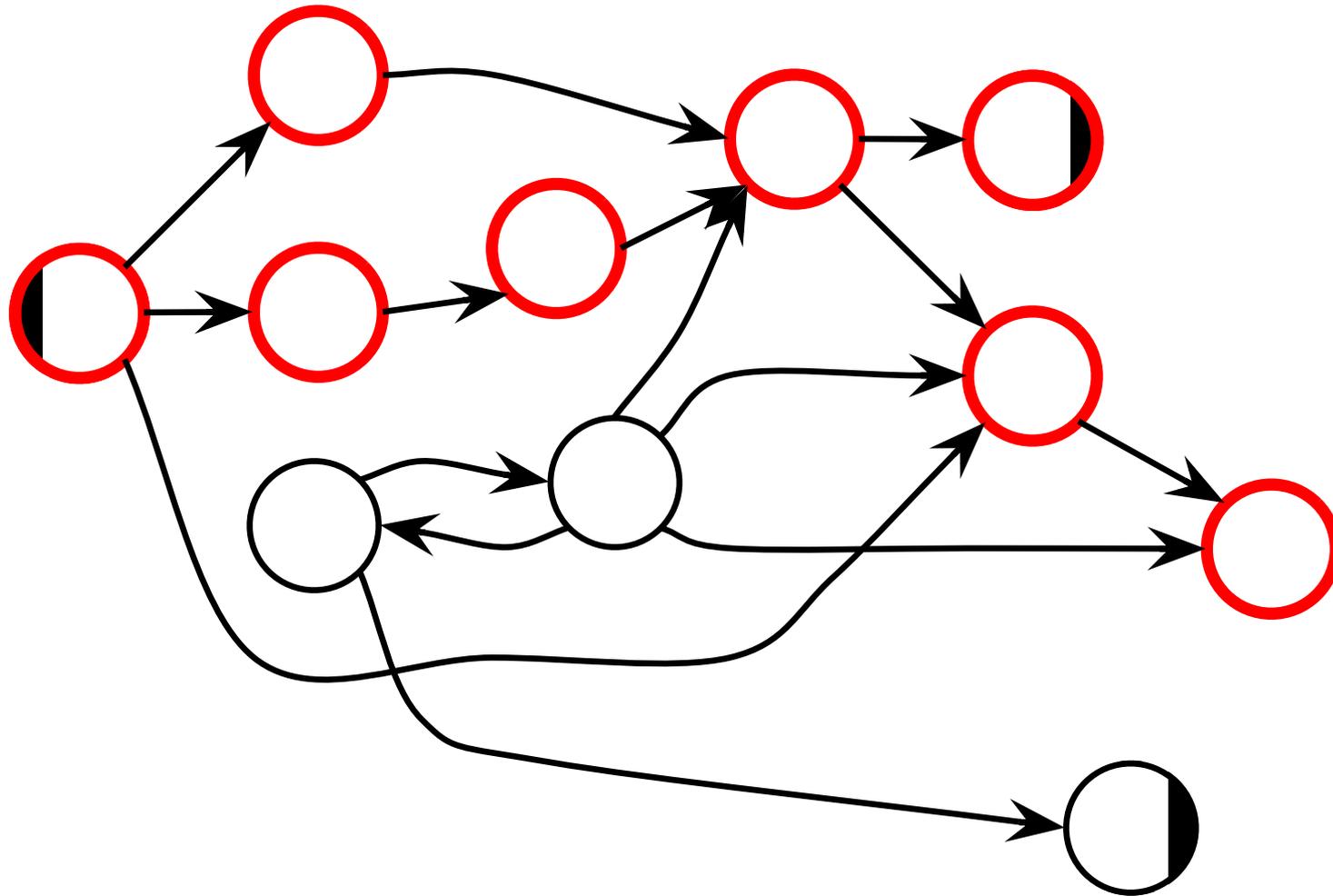


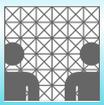
Anwendung des Algorithmus



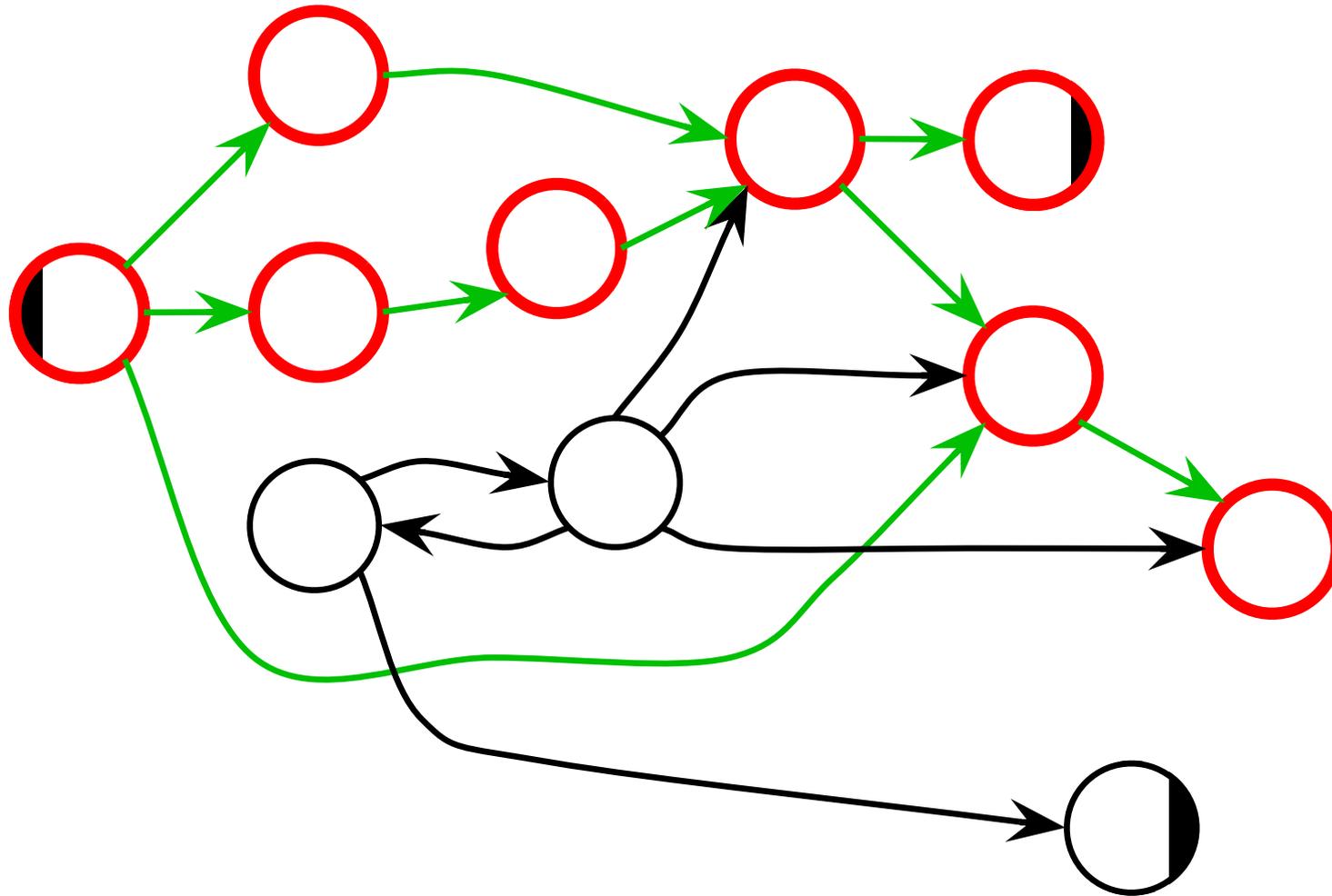


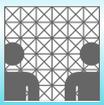
Anwendung des Algorithmus



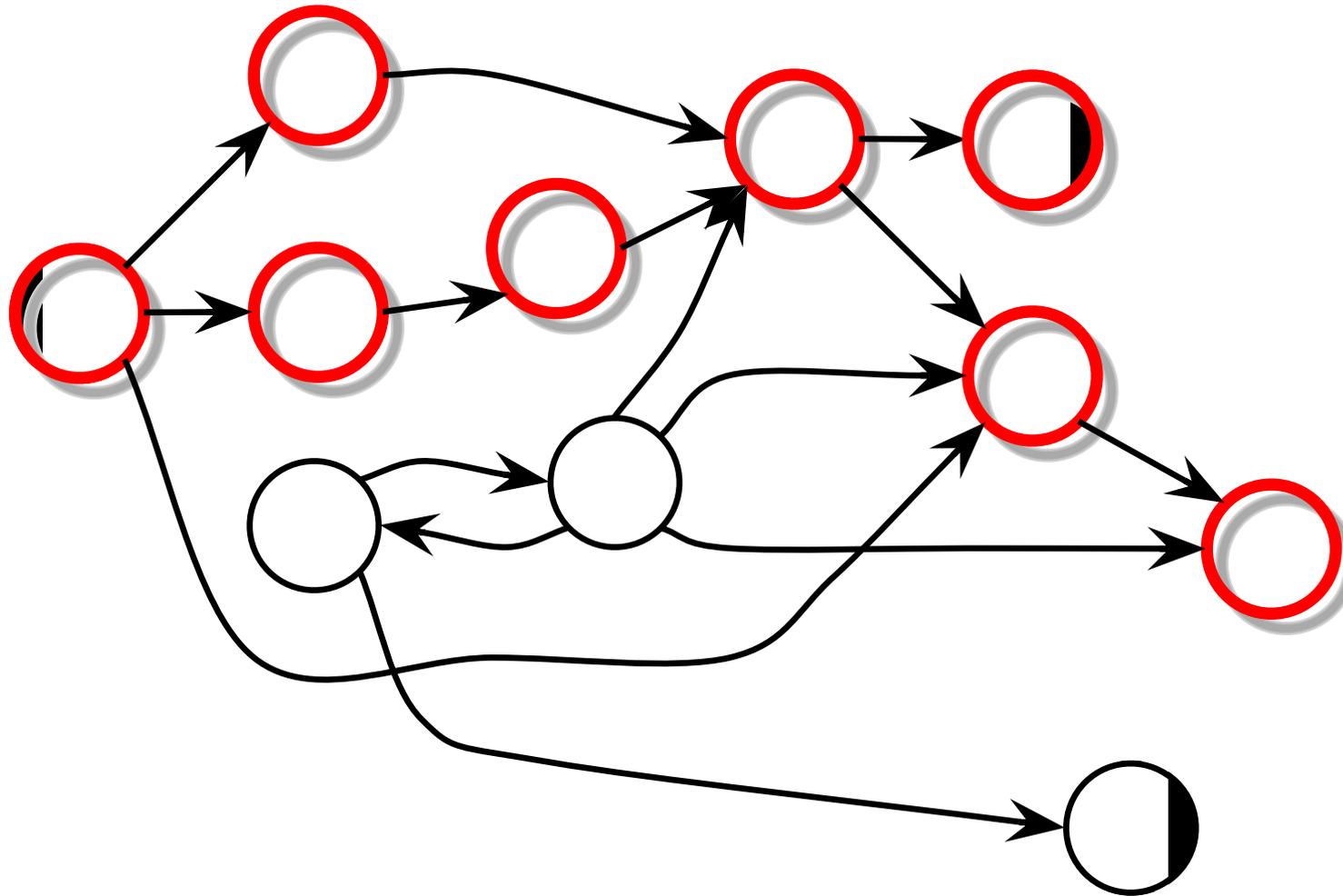


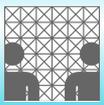
Anwendung des Algorithmus



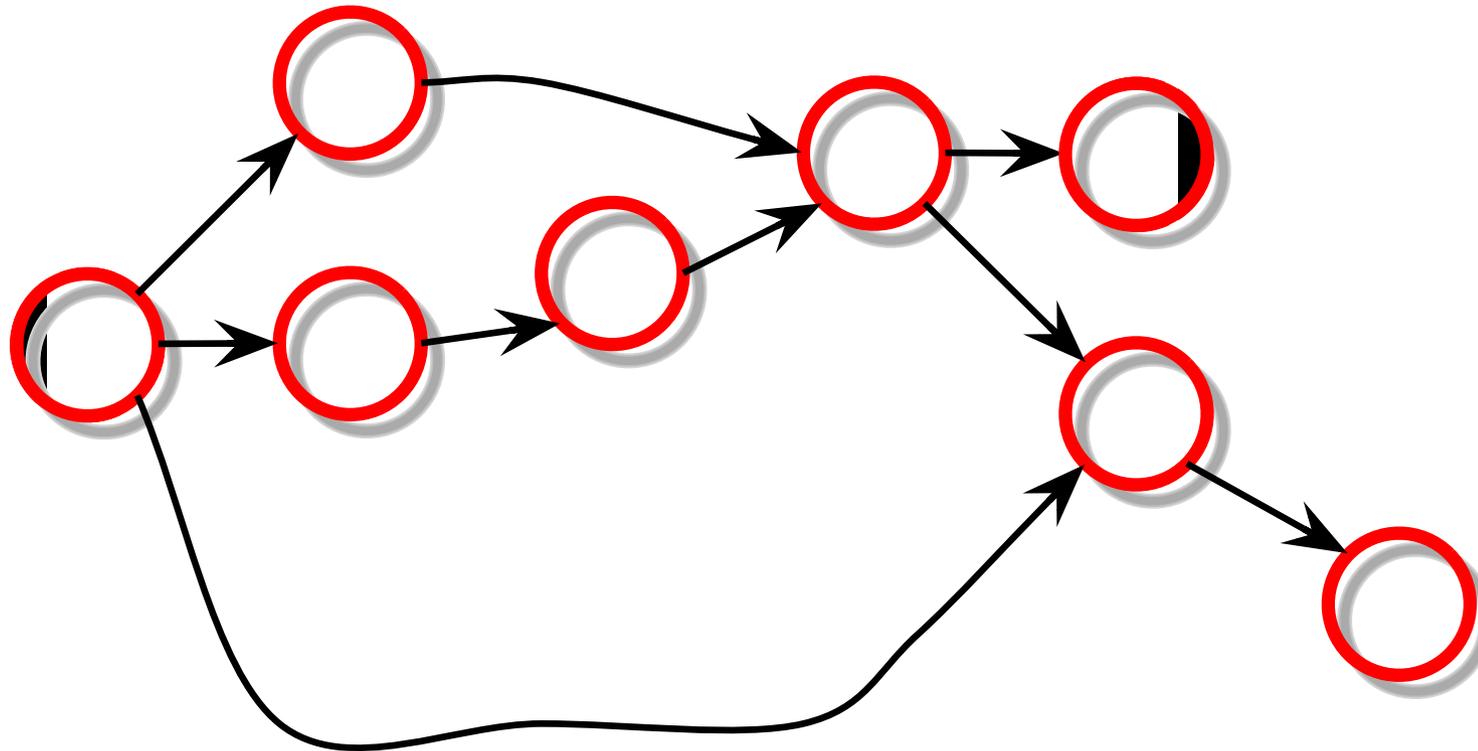


Anwendung des Algorithmus





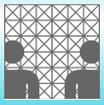
Anwendung des Algorithmus





Grenzen von *Reg*

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



Grenzen von *Reg*

- 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“.



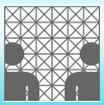
Grenzen von *Reg*

- 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.

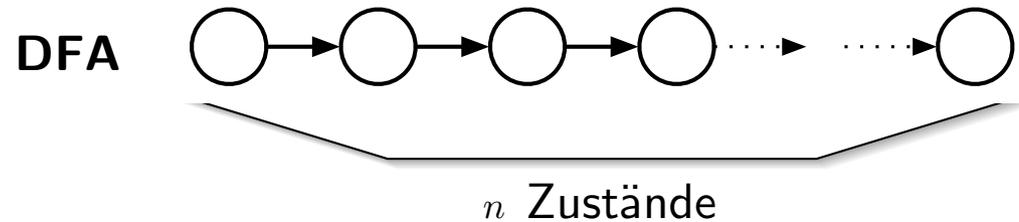


Grenzen von *Reg*

- 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!

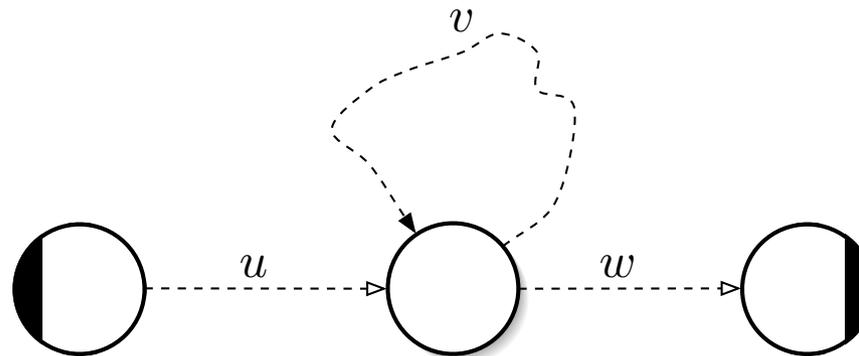


Motivation: Pumping-Lemma



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

Dazu muss eine Schleife durchlaufen werden!

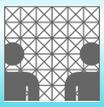


Dann wird aber auch jedes Wort der Form
 $uv^i w$ akzeptiert.



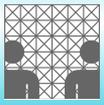
Pumping-Lemma für $\mathcal{R}eg$

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



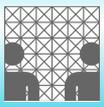
Pumping-Lemma für $\mathcal{R}eg$

- **Theorem:** Zu jeder regulären Menge $R \in \mathcal{R}eg$ gibt es eine Zahl $n \in \mathbb{N}$, 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 $n \in \mathbb{N}$, 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 $n \in \mathbb{N}$, 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) $\{u\}\{v\}^*\{w\} \subseteq R$



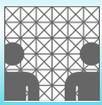
Beweis: Pumping-Lemma

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



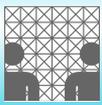
Beweis: Pumping-Lemma

- 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 $z = x_1x_2 \dots x_m \in R$, 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 $z = x_1x_2 \dots x_m \in R$, mit $m \geq n$ und $x_i \in \Sigma$, ein beliebiges Wort der Sprache.
- Die $m + 1$ Zustände $z_0, (z_0)^{x_1}, (z_0)^{x_1x_2}, \dots, (z_0)^{x_1x_2 \dots x_m}$ sind von z_0 aus erreichbar und werden auf diesem Erfolgspfad für die Akzeptierung von z besucht.



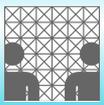
Beweis: Pumping-Lemma

- 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 $z = x_1x_2 \dots x_m \in R$, mit $m \geq n$ und $x_i \in \Sigma$, ein beliebiges Wort der Sprache.
- Die $m + 1$ Zustände $z_0, (z_0)^{x_1}, (z_0)^{x_1x_2}, \dots, (z_0)^{x_1x_2 \dots x_m}$ 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
 $(z_0)^{x_1 x_2 \dots x_i} = (z_0)^{x_1 x_2 \dots x_j}$.



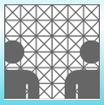
Beweis: Pumping-Lemma (2)

- 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
 $(z_0)^{x_1 x_2 \dots x_i} = (z_0)^{x_1 x_2 \dots x_j}$.
- 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.
- Dass auch (iii) gilt, sieht man leicht ein, denn mit
 $z' := (z_0)^u$ gilt doch
 $z' = (z')^\lambda = (z')^v = (z')^{vv} = \dots$. 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 $DUP := \{a^n b^n \mid n \in \mathbb{N}\}$ nicht regulär ist:



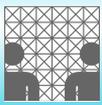
Beispiel: Pumping-Lemma

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



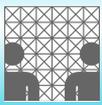
Beispiel: Pumping-Lemma

- 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 $|z| > k$.



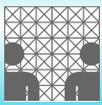
Beispiel: Pumping-Lemma

- Wir wollen mit Hilfe des uvw-Theorems zeigen, dass die Sprache $DUP := \{a^n b^n \mid n \in \mathbb{N}\}$ 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 $1 \leq m \leq k$.



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 $DUP \notin \mathcal{Reg}$ mit einer weiteren Methode nachgewiesen.



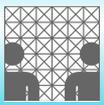
Grenzen von $\mathcal{R}eg$ (Forts.)

- Eine oftmals sehr einfache Methode, von Mengen zu zeigen, dass sie nicht regulär sind, ist die Verwendung der Abschlusseigenschaften von $\mathcal{R}eg$.



Grenzen von Reg (Forts.)

- Eine oftmals sehr einfache Methode, von Mengen zu zeigen, dass sie nicht regulär sind, ist die Verwendung der Abschlusseigenschaften von Reg .
 - Zunächst nimmt man an, die fragliche Menge sei regulär.



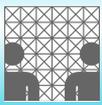
Grenzen von *Reg* (Forts.)

- Eine oftmals sehr einfache Methode, von Mengen zu zeigen, dass sie nicht regulär sind, ist die Verwendung der Abschlusseigenschaften von *Reg*.
 - Zunächst nimmt man an, die fragliche Menge sei regulär.
 - Dann verwendet man Transformationen mit Abschluss-Operatoren, um solche Sprachen zu erhalten, von denen man schon weiß, oder für die man vielleicht leichter zeigen kann, dass diese nicht regulär sind.



Grenzen von *Reg* (Forts.)

- Eine oftmals sehr einfache Methode, von Mengen zu zeigen, dass sie nicht regulär sind, ist die Verwendung der Abschlusseigenschaften von *Reg*.
 - Zunächst nimmt man an, die fragliche Menge sei regulär.
 - Dann verwendet man Transformationen mit Abschluss-Operatoren, um solche Sprachen zu erhalten, von denen man schon weiß, oder für die man vielleicht leichter zeigen kann, dass diese nicht regulär sind.
 - Somit ist ein Widerspruch erzielt und die ursprüngliche Menge kann nicht regulär gewesen sein.



Grenzen von *Reg* (Forts.)

- Eine oftmals sehr einfache Methode, von Mengen zu zeigen, dass sie nicht regulär sind, ist die Verwendung der Abschlusseigenschaften von *Reg*.
 - Zunächst nimmt man an, die fragliche Menge sei regulär.
 - Dann verwendet man Transformationen mit Abschluss-Operatoren, um solche Sprachen zu erhalten, von denen man schon weiß, oder für die man vielleicht leichter zeigen kann, dass diese nicht regulär sind.
 - Somit ist ein Widerspruch erzielt und die ursprüngliche Menge kann nicht regulär gewesen sein.
- Varianten dieses Vorgehens ...



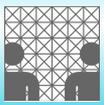
Beispielanwendung: DUP

- Wir zeigen, dass die Menge $C := \{w \in \{a, b\}^* \mid |w|_a = |w|_b\}$ nicht regulär ist:



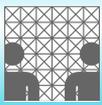
Beispielanwendung: DUP

- Wir zeigen, dass die Menge $C := \{w \in \{a, b\}^* \mid |w|_a = |w|_b\}$ nicht regulär ist:
 - Setze $D := C \cap \{a\}^*\{b\}^*$. Mit $C \in \mathcal{REG}$ wäre auch $D \in \mathcal{REG}$, es ist aber $D = DUP = \{a^n b^n \mid n \in \mathbb{N}\}$ bekanntlich nicht regulär.



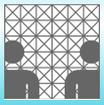
Beispielanwendung: DUP

- Wir zeigen, dass die Menge $C := \{w \in \{a, b\}^* \mid |w|_a = |w|_b\}$ nicht regulär ist:
 - Setze $D := C \cap \{a\}^* \{b\}^*$. Mit $C \in \mathcal{REG}$ wäre auch $D \in \mathcal{REG}$, es ist aber $D = DUP = \{a^n b^n \mid n \in \mathbb{N}\}$ bekanntlich nicht regulär.
- Das uvw -Theorem ist nicht direkt benutzbar, um zu beweisen, dass die Sprache $L := \{c^k a^l b^m \mid k, l, m \in \mathbb{N} : k = 0 \text{ oder } l = m\}$ nicht regulär ist.



Beispielanwendung: DUP

- Wir zeigen, dass die Menge $C := \{w \in \{a, b\}^* \mid |w|_a = |w|_b\}$ nicht regulär ist:
 - Setze $D := C \cap \{a\}^* \{b\}^*$. Mit $C \in \mathcal{REG}$ wäre auch $D \in \mathcal{REG}$, es ist aber $D = DUP = \{a^n b^n \mid n \in \mathbb{N}\}$ bekanntlich nicht regulär.
- Das uvw -Theorem ist nicht direkt benutzbar, um zu beweisen, dass die Sprache $L := \{c^k a^l b^m \mid k, l, m \in \mathbb{N} : k = 0 \text{ oder } l = m\}$ nicht regulär ist.
 - Wir definieren nun die Menge $E := L \cap \{c\} \{a\}^* \{b\}^*$, die regulär wäre, wenn L dies ist.



Entscheidbarkeit

- Ein **Entscheidungsproblem** ist eine Fragestellung, die mit **JA** oder **NEIN** beantwortet werden muss.



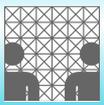
Entscheidbarkeit

- Ein **Entscheidungsproblem** ist eine Fragestellung, die mit **JA** oder **NEIN** beantwortet werden muss.
- Wir nennen ein Entscheidungsproblem **entscheidbar**, gdw. es einen deterministischen Algorithmus gibt, der die Fragestellung in endlich vielen elementaren Schritten beantwortet.



Entscheidbarkeit

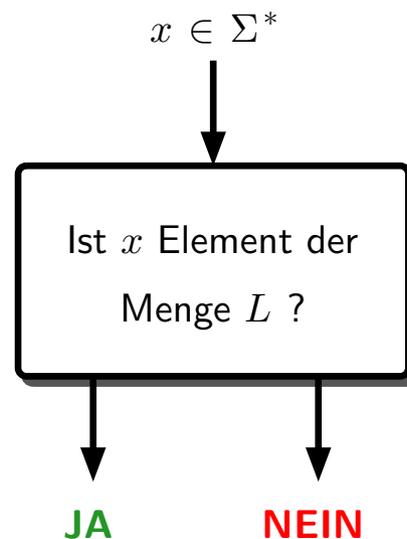
- Ein **Entscheidungsproblem** ist eine Fragestellung, die mit **JA** oder **NEIN** beantwortet werden muss.
- Wir nennen ein Entscheidungsproblem **entscheidbar**, gdw. es einen deterministischen Algorithmus gibt, der die Fragestellung in endlich vielen elementaren Schritten beantwortet.
- D.h. sowohl eine **positive** als auch eine **negative** Antwort muss in endlicher Zeit erreicht werden!

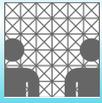


Entscheidbarkeit

- Ein **Entscheidungsproblem** ist eine Fragestellung, die mit **JA** oder **NEIN** beantwortet werden muss.
- Wir nennen ein Entscheidungsproblem **entscheidbar**, gdw. es einen deterministischen Algorithmus gibt, der die Fragestellung in endlich vielen elementaren Schritten beantwortet.
- D.h. sowohl eine **positive** als auch eine **negative** Antwort muss in endlicher Zeit erreicht werden!

Entscheidbarkeit/Beweisbarkeit





spezielles Wortproblem

- ... was sind typische Entscheidungsprobleme?



spezielles Wortproblem

- ... was sind typische Entscheidungsprobleme?
- **Definition:** (Wortproblem für reguläre Menge L)
Eingabe: Ein Wort $w \in \Sigma^*$
Frage: Gilt $w \in L$?



spezielles Wortproblem

- ... was sind typische Entscheidungsprobleme?
- **Definition:** (Wortproblem für reguläre Menge L)
Eingabe: Ein Wort $w \in \Sigma^*$
Frage: Gilt $w \in L$?
- ... ist entscheidbar.



spezielles Wortproblem

- ... was sind typische Entscheidungsprobleme?
- **Definition:** (Wortproblem für reguläre Menge L)
Eingabe: Ein Wort $w \in \Sigma^*$
Frage: Gilt $w \in L$?
- ... ist entscheidbar.
- Algorithmus: der DFA für die Sprache L .



spezielles Wortproblem

- ... was sind typische Entscheidungsprobleme?
- **Definition:** (Wortproblem für reguläre Menge L)
Eingabe: Ein Wort $w \in \Sigma^*$
Frage: Gilt $w \in L$?
- ... ist entscheidbar.
- Algorithmus: der DFA für die Sprache L .
- Komplexität: Länge des zu prüfenden Wortes w .



allgemeines Wortproblem

- **Definition:** (Allg. Wortproblem für λ -freie NFA's)

Eingabe: Ein NFA $A = (Z, \Sigma, K, z_0, Z_{\text{end}})$ und ein Wort $w \in \Sigma^*$

Frage: Gilt $w \in L(A)$?



allgemeines Wortproblem

- **Definition:** (Allg. Wortproblem für λ -freie NFA's)
 - Eingabe:** Ein NFA $A = (Z, \Sigma, K, z_0, Z_{\text{end}})$ und ein Wort $w \in \Sigma^*$
 - Frage:** Gilt $w \in L(A)$?
- ... ist natürlich auch entscheidbar.



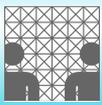
allgemeines Wortproblem

- **Definition:** (Allg. Wortproblem für λ -freie NFA's)
Eingabe: Ein NFA $A = (Z, \Sigma, K, z_0, Z_{\text{end}})$ und ein Wort $w \in \Sigma^*$
Frage: Gilt $w \in L(A)$?
- ... ist natürlich auch entscheidbar.
- 1. Möglichkeit für einen Algorithmus: Konstruiere aus NFA einen DFA \Rightarrow spezielles Wortproblem.



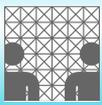
allgemeines Wortproblem

- **Definition:** (Allg. Wortproblem für λ -freie NFA's)
Eingabe: Ein NFA $A = (Z, \Sigma, K, z_0, Z_{\text{end}})$ und ein Wort $w \in \Sigma^*$
Frage: Gilt $w \in L(A)$?
- ... ist natürlich auch entscheidbar.
- 1. Möglichkeit für einen Algorithmus: Konstruiere aus NFA einen DFA \Rightarrow spezielles Wortproblem.
- Komplexität: exponentiell
(Potenzautomatenkonstr.)



allgemeines Wortproblem

- **Definition:** (Allg. Wortproblem für λ -freie NFA's)
 - Eingabe:** Ein NFA $A = (Z, \Sigma, K, z_0, Z_{\text{end}})$ und ein Wort $w \in \Sigma^*$
 - Frage:** Gilt $w \in L(A)$?
- ... ist natürlich auch entscheidbar.
- 1. Möglichkeit für einen Algorithmus: Konstruiere aus NFA einen DFA \Rightarrow spezielles Wortproblem.
- Komplexität: exponentiell (Potenzautomatenkonstr.)
- 2. Möglichkeit: Konstruktion des Potenzautomaten nur für die möglichen Pfade in dem λ -freien NFA entlang des Wortes $w := w_1w_2w_3 \dots w_n$.



allgemeines Wortproblem

- **Definition:** (Allg. Wortproblem für λ -freie NFA's)
 - Eingabe:** Ein NFA $A = (Z, \Sigma, K, z_0, Z_{\text{end}})$ und ein Wort $w \in \Sigma^*$
 - Frage:** Gilt $w \in L(A)$?
- ... ist natürlich auch entscheidbar.
- 1. Möglichkeit für einen Algorithmus: Konstruiere aus NFA einen DFA \Rightarrow spezielles Wortproblem.
- Komplexität: exponentiell (Potenzautomatenkonstr.)
- 2. Möglichkeit: Konstruktion des Potenzautomaten nur für die möglichen Pfade in dem λ -freien NFA entlang des Wortes $w := w_1w_2w_3 \dots w_n$.
- Komplexität: polynomiell, nämlich $n |Z|^2$



Leerheitsproblem

- **Definition:** (Leerheitsproblem für endliche Automaten)

Eingabe: Ein DFA oder ein NFA A

Frage: Ist $L(A) = \emptyset$?



Leerheitsproblem

- **Definition:** (Leerheitsproblem für endliche Automaten)
Eingabe: Ein DFA oder ein NFA A
Frage: Ist $L(A) = \emptyset$?
- ... ist entscheidbar.



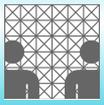
Leerheitsproblem

- **Definition:** (Leerheitsproblem für endliche Automaten)

Eingabe: Ein DFA oder ein NFA A

Frage: Ist $L(A) = \emptyset$?

- ... ist entscheidbar.
- Algorithmus: ähnlich dem Algorithmus zum Auffinden der initialen Zusammenhangskomponente.



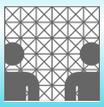
Leerheitsproblem

- **Definition:** (Leerheitsproblem für endliche Automaten)
Eingabe: Ein DFA oder ein NFA A
Frage: Ist $L(A) = \emptyset$?
- ... ist entscheidbar.
- Algorithmus: ähnlich dem Algorithmus zum Auffinden der initialen Zusammenhangskomponente.
- Komplexität: linear in der Größe des Automaten.



Leerheitsproblem

- **Definition:** (Leerheitsproblem für endliche Automaten)
 - Eingabe:** Ein DFA oder ein NFA A
 - Frage:** Ist $L(A) = \emptyset$?
- ... ist entscheidbar.
- Algorithmus: ähnlich dem Algorithmus zum Auffinden der initialen Zusammenhangskomponente.
- Komplexität: linear in der Größe des Automaten.
- Zur Größe des Automaten gehört: Anzahl der Zustände, Anzahl der Kanten



Äquivalenzproblem

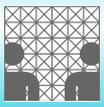
- **Definition:** (Äquivalenzproblem für DFAs)

Eingabe: Zwei DFAs

$$A = (Z_A, \Sigma_A, \delta_A, z_{A,0}, Z_{A,\text{end}}) \text{ und}$$

$$B = (Z_B, \Sigma_B, \delta_B, z_{B,0}, Z_{B,\text{end}})$$

Frage: Gilt $L(A) = L(B)$?



Äquivalenzproblem

- **Definition:** (Äquivalenzproblem für DFAs)

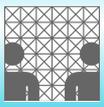
Eingabe: Zwei DFAs

$$A = (Z_A, \Sigma_A, \delta_A, z_{A,0}, Z_{A,\text{end}}) \text{ und}$$

$$B = (Z_B, \Sigma_B, \delta_B, z_{B,0}, Z_{B,\text{end}})$$

Frage: Gilt $L(A) = L(B)$?

- ... ist entscheidbar.



Äquivalenzproblem

- **Definition:** (Äquivalenzproblem für DFAs)

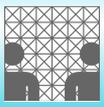
Eingabe: Zwei DFAs

$$A = (Z_A, \Sigma_A, \delta_A, z_{A,0}, Z_{A,\text{end}}) \text{ und}$$

$$B = (Z_B, \Sigma_B, \delta_B, z_{B,0}, Z_{B,\text{end}})$$

Frage: Gilt $L(A) = L(B)$?

- ... ist entscheidbar.
- Algorithmus: $L(A) = L(B)$ gilt gdw. sowohl $L(A) \cap \overline{L(B)} = \emptyset$ als auch $L(B) \cap \overline{L(A)} = \emptyset$ ist.



Äquivalenzproblem

- **Definition:** (Äquivalenzproblem für DFAs)

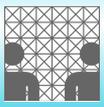
Eingabe: Zwei DFAs

$$A = (Z_A, \Sigma_A, \delta_A, z_{A,0}, Z_{A,\text{end}}) \text{ und}$$

$$B = (Z_B, \Sigma_B, \delta_B, z_{B,0}, Z_{B,\text{end}})$$

Frage: Gilt $L(A) = L(B)$?

- ... ist entscheidbar.
- Algorithmus: $L(A) = L(B)$ gilt gdw. sowohl $L(A) \cap \overline{L(B)} = \emptyset$ als auch $L(B) \cap \overline{L(A)} = \emptyset$ ist.
 - vervollständige die DFAs



Äquivalenzproblem

- **Definition:** (Äquivalenzproblem für DFAs)

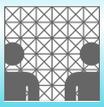
Eingabe: Zwei DFAs

$$A = (Z_A, \Sigma_A, \delta_A, z_{A,0}, Z_{A,\text{end}}) \text{ und}$$

$$B = (Z_B, \Sigma_B, \delta_B, z_{B,0}, Z_{B,\text{end}})$$

Frage: Gilt $L(A) = L(B)$?

- ... ist entscheidbar.
- Algorithmus: $L(A) = L(B)$ gilt gdw. sowohl $L(A) \cap \overline{L(B)} = \emptyset$ als auch $L(B) \cap \overline{L(A)} = \emptyset$ ist.
 - vervollständige die DFAs
 - konstruiere DFAs für das Komplement



Äquivalenzproblem

- **Definition:** (Äquivalenzproblem für DFAs)

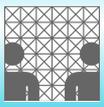
Eingabe: Zwei DFAs

$$A = (Z_A, \Sigma_A, \delta_A, z_{A,0}, Z_{A,\text{end}}) \text{ und}$$

$$B = (Z_B, \Sigma_B, \delta_B, z_{B,0}, Z_{B,\text{end}})$$

Frage: Gilt $L(A) = L(B)$?

- ... ist entscheidbar.
- Algorithmus: $L(A) = L(B)$ gilt gdw. sowohl $L(A) \cap \overline{L(B)} = \emptyset$ als auch $L(B) \cap \overline{L(A)} = \emptyset$ ist.
 - vervollständige die DFAs
 - konstruiere DFAs für das Komplement
 - konstruiere Produktautomaten



Äquivalenzproblem

- **Definition:** (Äquivalenzproblem für DFAs)

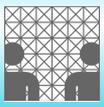
Eingabe: Zwei DFAs

$$A = (Z_A, \Sigma_A, \delta_A, z_{A,0}, Z_{A,\text{end}}) \text{ und}$$

$$B = (Z_B, \Sigma_B, \delta_B, z_{B,0}, Z_{B,\text{end}})$$

Frage: Gilt $L(A) = L(B)$?

- ... ist entscheidbar.
- Algorithmus: $L(A) = L(B)$ gilt gdw. sowohl $L(A) \cap \overline{L(B)} = \emptyset$ als auch $L(B) \cap \overline{L(A)} = \emptyset$ ist.
 - vervollständige die DFAs
 - konstruiere DFAs für das Komplement
 - konstruiere Produktautomaten
 - \Rightarrow Leerheitsproblem



Äquivalenzproblem

- **Definition:** (Äquivalenzproblem für DFAs)

Eingabe: Zwei DFAs

$$A = (Z_A, \Sigma_A, \delta_A, z_{A,0}, Z_{A,\text{end}}) \text{ und}$$

$$B = (Z_B, \Sigma_B, \delta_B, z_{B,0}, Z_{B,\text{end}})$$

Frage: Gilt $L(A) = L(B)$?

- ... ist entscheidbar.
- Algorithmus: $L(A) = L(B)$ gilt gdw. sowohl $L(A) \cap \overline{L(B)} = \emptyset$ als auch $L(B) \cap \overline{L(A)} = \emptyset$ ist.
 - vervollständige die DFAs
 - konstruiere DFAs für das Komplement
 - konstruiere Produktautomaten
 - \Rightarrow Leerheitsproblem
- Komplexität: $|\Sigma_A \cap \Sigma_B| \cdot |Z_A| \cdot |Z_B|$ Schritte



Universalitätsproblem

- **Definition:** (Universalitätsproblem für DFAs)

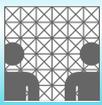
Eingabe: Ein DFA $A = (Z, \Sigma, \delta, z_0, Z_{\text{end}})$

Frage: Ist $L(A) = \Sigma^*$?



Universalitätsproblem

- **Definition:** (Universalitätsproblem für DFAs)
Eingabe: Ein DFA $A = (Z, \Sigma, \delta, z_0, Z_{\text{end}})$
Frage: Ist $L(A) = \Sigma^*$?
- ... ist entscheidbar.



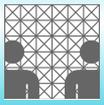
Universalitätsproblem

- **Definition:** (Universalitätsproblem für DFAs)
Eingabe: Ein DFA $A = (Z, \Sigma, \delta, z_0, Z_{\text{end}})$
Frage: Ist $L(A) = \Sigma^*$?
- ... ist entscheidbar.
- Algorithmus: Ist jeder Zustand Endzustand und verlassen jeden Zustand $|\Sigma|$ Kanten?



Universalitätsproblem

- **Definition:** (Universalitätsproblem für DFAs)
Eingabe: Ein DFA $A = (Z, \Sigma, \delta, z_0, Z_{\text{end}})$
Frage: Ist $L(A) = \Sigma^*$?
- ... ist entscheidbar.
- Algorithmus: Ist jeder Zustand Endzustand und verlassen jeden Zustand $|\Sigma|$ Kanten?
- Komplexität: linearer Aufwand



Ausblick

- Für die Definition von Programmiersprachen reichen reguläre Mengen nicht aus, deshalb suchen wir nach einer größeren Sprachfamilie ...



Ausblick

- Für die Definition von Programmiersprachen reichen reguläre Mengen nicht aus, deshalb suchen wir nach einer größeren Sprachfamilie ...
- Grammatiken



Ausblick

- Für die Definition von Programmiersprachen reichen reguläre Mengen nicht aus, deshalb suchen wir nach einer größeren Sprachfamilie ...
- Grammatiken
- kontextfreie Sprachen



Ausblick

- Für die Definition von Programmiersprachen reichen reguläre Mengen nicht aus, deshalb suchen wir nach einer größeren Sprachfamilie ...
- Grammatiken
- kontextfreie Sprachen
- Abschlusseigenschaften