Einfache Zusammenhänge

- Eine TM, die $t(n)$ Zeit (d.h. Schritte) zur Verfügung hat, kann nicht mehr als $t(n)$ Bandzellen besuchen.
  - Umgekehrt gilt dies nicht!
- Platz kann **wiederverwendet** werden, Zeit nicht!
- *Kann man trotzdem eine maximale Rechenzeit in Abhängigkeit vom verbrauchten Platz angeben?*
  - Ja! **Idee:** Sobald eine Konfiguration ein zweites Mal besucht wird, befinden wir uns in einer Endlosschleife!
  - Bleibt zu berechnen, **wieviele verschiedene Konfigurationen** bei einer gegebenen Platzbeschränkung möglich sind.

Anzahl der Konfigurationen

- Wovon hängt die Anzahl der möglichen unterschiedlichen Konfigurationen ab?
  - maximal verbrauchter Platz
    - **Kopfpositionen** auf den Arbeitsbändern
    - **Länge** möglicher Bandinschriften
  - Anzahl der Bandsymbole
    - **Permutationen** von Bandsymbolen
  - Länge der Eingabe
    - **Kopfpositionen** auf dem Eingabeband
  - Anzahl der Zustände in der endlichen Steuerung
    - aktueller **Zustand** einer Konfiguration

Randbedingungen für Schranken

- Wenn eine TM mit der Platzbeschränkung $s(n)$ arbeitet, dann ist $s(n) \geq 1$ für jedes $n \in \mathbb{N}$, denn die TM muss sich wenigstens ein Feld auf dem Arbeitsband ansehen!
  - TM **arbeitet mit Platzbeschränkung** $s(n)$, wenn sie $\max\{1, \lceil s(n) \rceil\}$-platzbeschränkt ist.
- Es ist ebenfalls nützlich anzunehmen, dass eine TM ihre Eingabe stets vollständig liest und ein weiteres Feld vorrückt, um deren Ende festzustellen.
  - TM **arbeitet mit Zeitbeschränkung** $t(n)$, wenn sie $\max\{n+1, t(n)\}$-zeitbeschränkt ist.

Beispiel $\{w\$w^{rev} \mid w \in \{a,b\}^*\}$

- $\{w\$w^{rev} \mid w \in \{a,b\}^*\}$ ist kontextfrei.
- Mit dem Verfahren von Cocke/Younger/Kasami kann sie somit in Polynomzeit analysiert werden.
- Wieviel Platz wird gebraucht? $s(n) = n$ genügt.
  - **Idee:** Akzeptieren nach dem **Kellerprinzip**, für Wörter, die in der Sprache enthalten sind, reicht dann sogar $\lceil \frac{n+1}{2} \rceil$.
- Es geht aber noch besser: $\log(n)$ Platzbedarf genügt!
  - Codieren der Anzahl bereits bearbeiteter Symbole von w bzw. w^{rev} in $w\$w^{rev}$ als Binärzahl: Ein Zähler gibt das nächste zu bearbeitende Zeichen an.

Bandkompression

Theorem:

Zu jeder $s(n)$-platzbeschränkten TM und jeder reellen Zahl $c \in \mathbb{R}$ mit $c > 0$ gibt es eine äquivalente TM die $c \cdot s(n)$-platzbeschränkt ist.

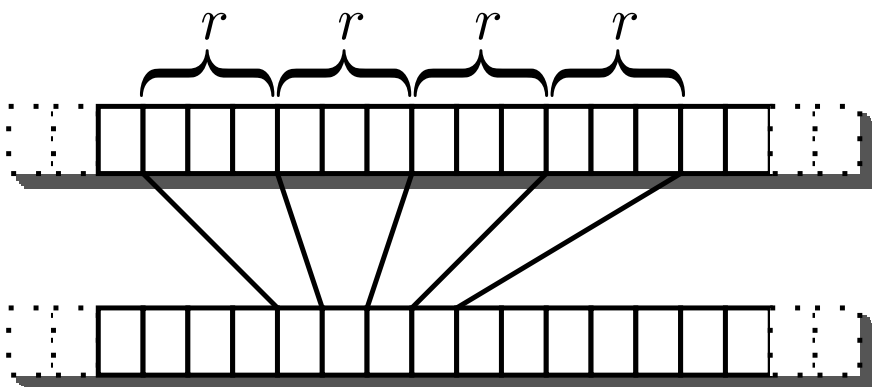
Beweis:

1. **Fall** Für $c \geq 1$ ist nichts zu beweisen.

2. **Fall** Für $c < 1$:

   - Konstruktion analog zum $|w|$-platzbeschränkten LBA B aus einem beliebigen LBA A für eine TM durchführen!
   - Wenn A $s(n)$-platzbeschränkt ist, so ist B dann nur noch $c \cdot s(n)$-platzbeschränkt.

Bandkompression (2)



- Zusammenfassen von jeweils r Symbolen
- Erweiterung des Bandalphabets
- r berechnet sich aus $r = \lceil \frac{s(n)}{c} \rceil$

Also: $xBand(f) = xBand(cf)$ für $x \in \{\mathcal{D}, \mathcal{N}\}$.

...zum Beispiel: $f(n) := 3n^2 - 70n + 5$ und $g(n) := n^2$

als Platzschranken nicht unterscheidbar!

linear speed-up

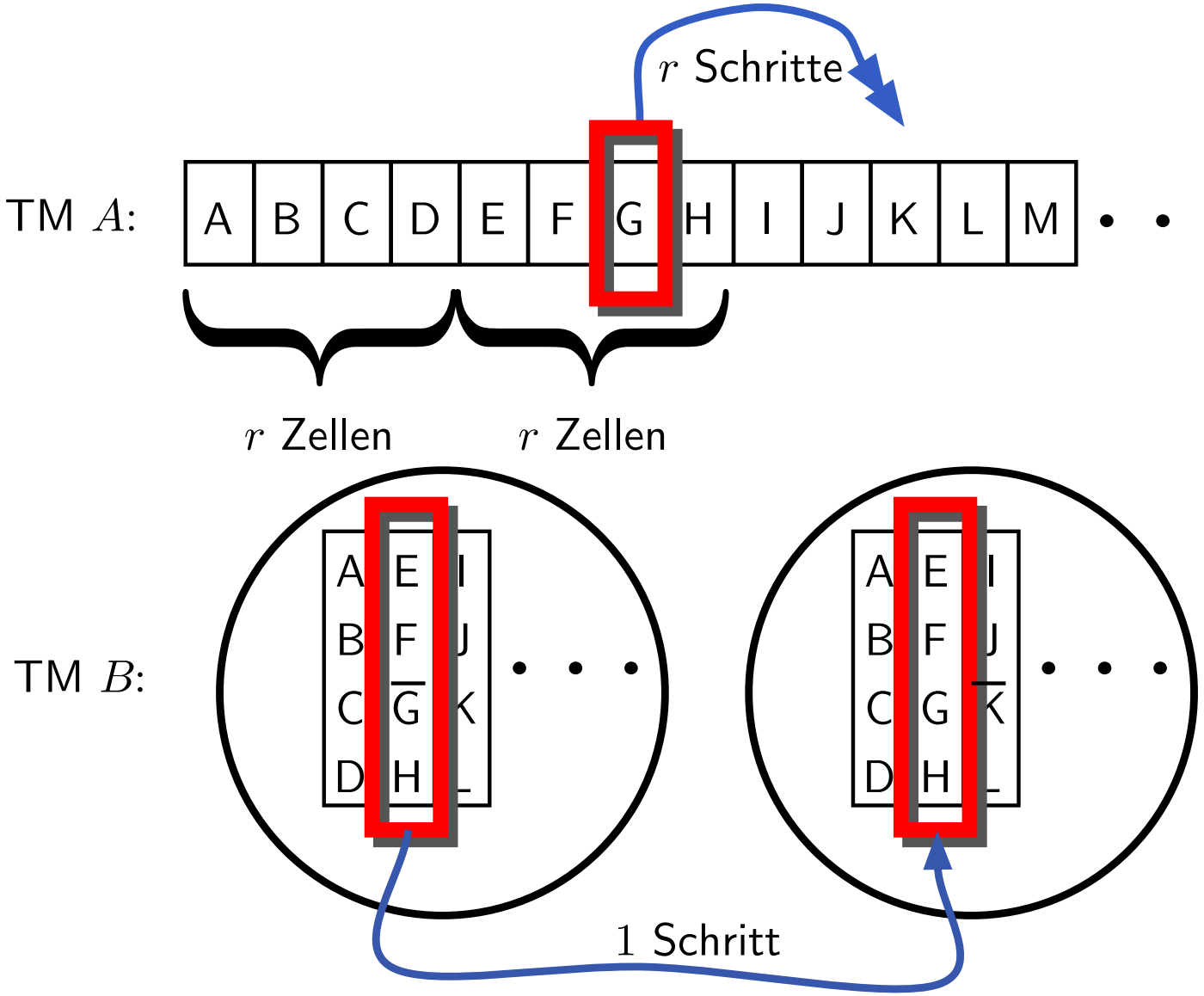
Theorem: (lineare Beschleunigung)

> Zu jeder $t(n)$-zeitbeschränkten TM mit $\inf_{n\to\infty}(\frac{t(n)}{n}) = \infty$ und jedem $c \in \mathbb{R}$ mit $c > 0$ gibt es eine äquivalente $c \cdot t(n)$-zeitbeschränkte TM.

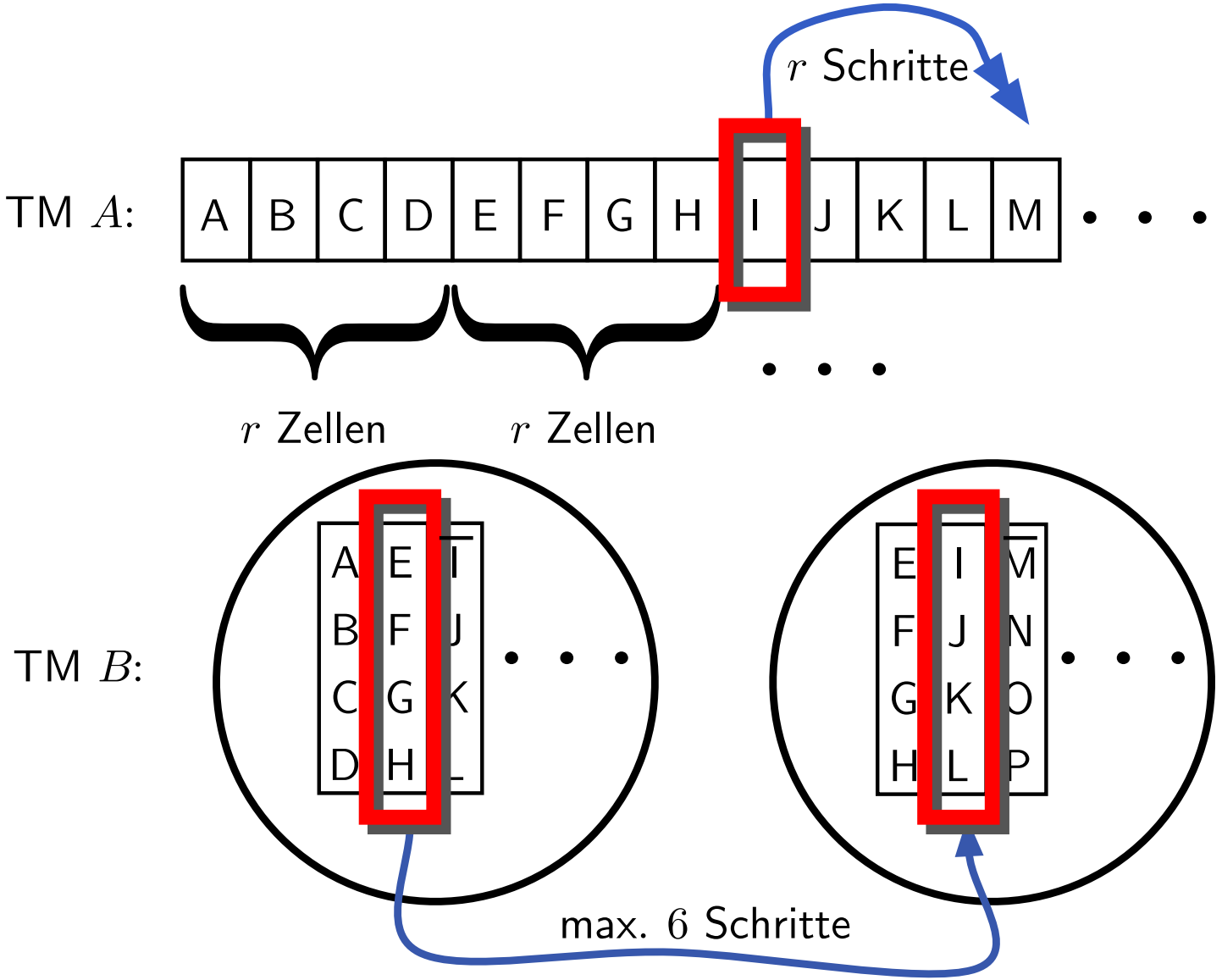
Beweisidee:

- Blockbildung (Länge r, wie bei der Bandkompression).
- Speichern von jeweils 3 Blöcken im Zustand der endlichen Kontrolle.
- Ein Schritt simuliert r Schritte der ursprünglichen TM.

Beweis: Speed-Up-Theorem



Beweis: Speed-Up-Theorem



Beweis: Speed-Up-Theorem (2)

- Verlassen des 3-Block-Bereiches kostet A mind. r Schritte, B nur 8 Schritte. $\Rightarrow$ Für $t(n)$ Schritte von A benötigt B max. $\lceil \frac{8t(n)}{r} \rceil$ Schritte.
- n Schritte, zum Lesen und Codieren der Eingabe.
- $\lceil \frac{n}{r} \rceil$ Schritte, Repositionieren des LSK von B.
- Wegen $\lceil x \rceil \leq x + 1$, sind max. $n + 1 + \frac{n}{r} + 1 + 8 \cdot (\frac{t(n)}{r})$ Schritte von B nötig.
- $\inf_{n \to \infty} (\frac{t(n)}{n}) = \infty \Rightarrow \forall d : \exists n_d : \forall n \geq n_d : (\frac{t(n)}{n}) \geq d$ (d.h. auch $n \leq (\frac{t(n)}{d})$).
- Für $n \geq 2$ (und somit $n + 2 \leq 2n$) und $n \geq n_d$: Schrittzahl von B max. $t(n) \left(\frac{2}{d} + \frac{1}{dr} + \frac{8}{r} \right)$.

Beweis: Speed-Up-Theorem (3)

- Wählen wir $r := \lceil \frac{16}{c} \rceil$ und $d := \lceil \frac{4}{c} \rceil + \frac{1}{8}$ dann gilt $r \cdot c \geq 16$ und $d \geq \frac{32+c}{8c}$.
- Einsetzen in die Formel für die max. Laufzeit von B:

$$\begin{aligned}
\frac{2}{d} + \frac{1}{dr} + \frac{8}{r} &= \frac{2}{d} + \frac{c}{drc} + \frac{8c}{rc} \\
&\leq \frac{2}{d} + \frac{c}{16d} + \frac{8c}{16} \\
&= \frac{32+c}{16d} + \frac{c}{2} \\
&= \frac{(32+c)c}{16cd} + \frac{c}{2} \\
&= \frac{(32+c)c}{8c \cdot 2d} + \frac{c}{2} \leq \frac{cd}{2d} + \frac{c}{2} \qquad = c
\end{aligned}$$

Asymptotische Analyse

- **Konstante Faktoren** und **additive Glieder** spielen bei Beschränkungsfunktionen keine entscheidende Rolle.
- Zur Klassifizierung von Funktionen verwendet man deshalb die bekannten Landau-Schreibweisen: die O-, Ω- und θ-Notation.
- *Wichtig:* Bei der asymptotischen Analyse gilt:

  THINK BIG

  das heißt, es kommt nur darauf an, wie sich die Funktionen für *große Argumente* verhalten.
- Eine **endliche Anzahl** von „Ausreißern“ ist immer erlaubt!

Zwei extreme Beispiele (1)

- Es gilt $\log(n) \in o(n^{0.0001})$.
- Das heißt, $\lim_{n\to\infty} \frac{\log(n)}{n^{0.0001}} = 0$
- kurz: $n^{0.0001}$ wächst viel schneller als $\log n$.
- Für $n = 10^{100}$:

$$\log 10^{100} = 100 > \left(10^{100}\right)^{0.0001} = 10^{0.01} = 1,023$$

- Für $n = 10^{10^{100}}$:

$$\log 10^{10^{100}} = 10^{100} < \left(10^{10^{100}}\right)^{0.0001} = 10^{10^{96}}$$

- Also: $\forall\, n > 10^{10^{100}} : \log(n) < n^{0.0001}$.

Zwei extreme Beispiele (2)

- Sei $\epsilon = 1/10^{10^{100}}$.
- Es gilt $\log(n) \in o(n^{\epsilon})$.
- Seien k und n so gewählt, dass $\epsilon > 10^{-k}$ und $n = 10^{10^{2k}}$.
- Dann gilt:

$$\log n = 10^{2k},$$

$$n^{\varepsilon} > \left(10^{10^{2k}}\right)^{10^{-k}} = 10^{10^{k}}$$

- Also: $\forall\, n > 10^{10^{2k}} : \log(n) < n^{\varepsilon}$.

Einfache Folgerungen

- Es gilt wegen des *speed-up*-Satzes:

$$\mathcal{P} = \bigcup_{i \geq 1} \mathcal{D}Time(n^i) \text{ und } \mathcal{NP} = \bigcup_{i \geq 1} \mathcal{N}Time(n^i)$$

- Ähnlich definiert man polynomiale Platz-Klassen:

$$\begin{aligned} \mathcal{P}Space &:= \bigcup_{i \geq 1} \mathcal{D}Space(n^i) \\ \mathcal{NP}Space &:= \bigcup_{i \geq 1} \mathcal{N}Space(n^i) \end{aligned}$$

Einfache Inklusionen

Korollar:

Die sich direkt aus den Definitionen ergebenden trivialen Beziehungen zwischen diesen Komplexitätsklassen sind offensichtlich die folgenden:

$$
\begin{aligned}
\mathcal{D}Space(f) &\subseteq \mathcal{N}Space(f) \\
\mathcal{D}Time(f) &\subseteq \mathcal{N}Time(f) \\
\mathcal{D}Time(f) &\subseteq \mathcal{D}Space(f) \\
\mathcal{N}Time(f) &\subseteq \mathcal{N}Space(f) \\
\mathcal{P} &\subseteq \mathcal{NP} \\
\mathcal{P}Space &\subseteq \mathcal{NP}Space
\end{aligned}
$$

$\mathcal{P}$ und die Praxis

- Von Problemen aus der Komplexitätsklasse $\mathcal{P}$ sagt man, dass sie **effizient lösbar** seien.
- Probleme in $\mathcal{P}$ sind anscheinend wirklich in der Praxis noch mit vertretbarem Aufwand lösbar.
- Nehmen wir also an, dass tatsächlich alle Probleme in $\mathcal{P}$ in der Praxis vernünftig gelöst werden können.
  - Was ist dann mit $\mathcal{NP}$?
  - Sind Probleme aus $\mathcal{NP}$ nicht mehr praktisch verwendbar/lösbar?
  - Handelt es sich nur um praktisch nicht relevante Probleme?

Die Klasse $\mathcal{NP}$

- Jedes Problem $L \in \mathcal{NP}$, ist mit exponentiellem Zeitaufwand (in $\mathcal{D}Time(2^{p(n)})$ für ein Polynom p) deterministisch lösbar.
  - Bessere Transformationen sind i.a. nicht bekannt.
- Viele der *praktisch wichtigen* Fragestellungen haben Lösungen mit Algorithmen, die **nur dann in Polynomzeit arbeiten, wenn sie nicht-deterministisch sind**, jedenfalls kennt man zur Zeit noch nicht Besseres!
- Eine **Implementierung** erfordert ein **deterministisches Verfahren**.
- Unterscheiden sich $\mathcal{P}$ und $\mathcal{NP}$ überhaupt?!

Einige Probleme

- Ähnliche Probleme liegen trotzdem oft in (vielleicht wirklich) unterschiedlichen Komplexitätsklassen!

Gegeben: Matrix $A \in \mathbb{Z}^{m \times n}$ und Vektor $b \in \mathbb{Z}^m$	
Frage: Gibt es $x \in \mathbb{Z}^n$ mit $Ax = b$	**Frage:** Gibt es $x \in \mathbb{N}^n$ mit $Ax = b$
$\mathcal{P}$	$\mathcal{NP}$

Einige Probleme (2.1)

Gegeben: Ungerichteter, kantenbewerteter Graph $G = (V, E)$, eine Gewichtsfunktion $g : E \to \mathbb{N}$, zwei Knoten $s, t \in V$ und eine Schranke $k \in \mathbb{N}$	
Frage: Gibt es einen einfachen Pfad p von s nach t mit $g(p) \leq k$? („Kürzester Weg zwischen zwei Knoten“)	**Frage:** Gibt es einen einfachen Pfad p von s nach t mit $g(p) \geq k$? („Längster Weg zwischen zwei Knoten“)
$\mathcal{P}$	$\mathcal{NP}$

Einige Probleme (2.2)

- $L_w := \{\langle G, s, t, k\rangle \mid G$ besitzt einen Pfad p von S nach t mit $g(p) \geq k\}$ ist die dem Problem „Längster Weg zwischen zwei Knoten“ zugeordnete Sprache.
- **Theorem:** Das Problem zu L_w ist nichtdeterministisch in Polynomzeit zu lösen, d.h. $K_w \in \mathcal{NP}$.
  Idee: Raten eines geeigneten Pfades.
- $K_w := \{\langle G, s, t, k\rangle \mid G$ besitzt einen Pfad p von s nach t mit $g(p) \leq k\}$ ist die Sprache, die zu dem Problem „Kürzester Weg zwischen zwei Knoten“ gehört.
- **Theorem:** Das Problem zu K_w ist in $\mathcal{P}$.
  Idee: Systematische Suche der minimalen Pfadlängen.

Einige Probleme (3)

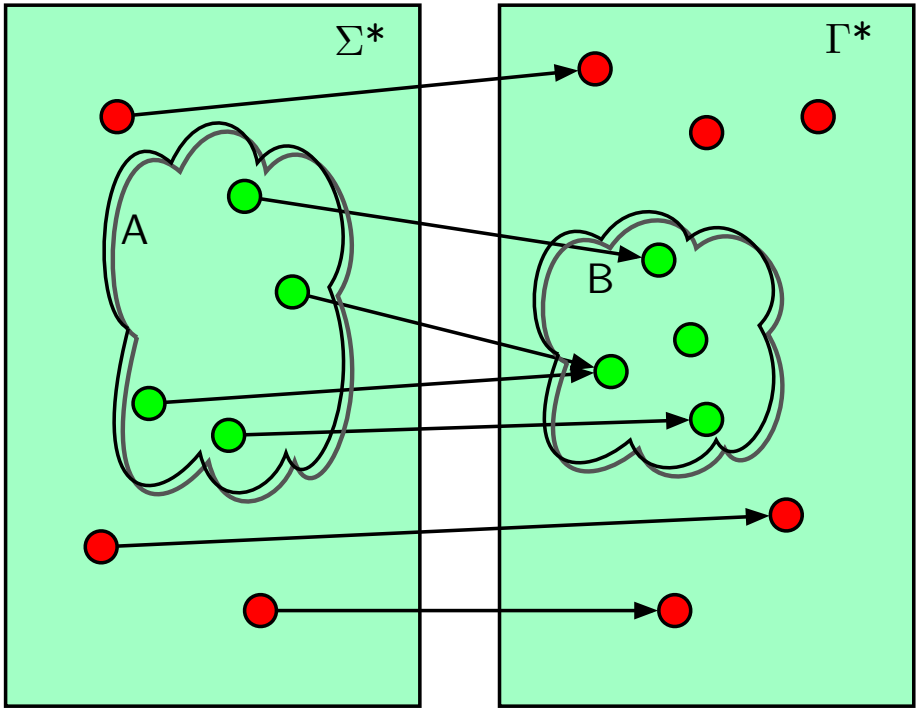
Gegeben: Ungerichteter Graph $G = (V, E)$	
Frage: Gibt es einen geschlossenen Kreis, in dem jede Kante genau einmal auftritt? („Euler-Kreis“)	**Frage:** Gibt es einen geschlossenen Kreis, in dem jeder Knoten genau einmal auftritt? („Hamilton-Kreis“)
$\mathcal{P}$	$\mathcal{NP}$

- (Leider) sind häufig Probleme aus $\mathcal{NP}$ die (für die Praxis) interessantesten!
- Manchmal kann man das aber auch ausnutzen. Z.B. für die Kryptographie.

Reduktion

- Seien $A \subset \Sigma^*$ und $B \subset \Gamma^*$.
- Man sagt „A ist **reduzierbar** auf B“ ($A \leq B$) gdw.

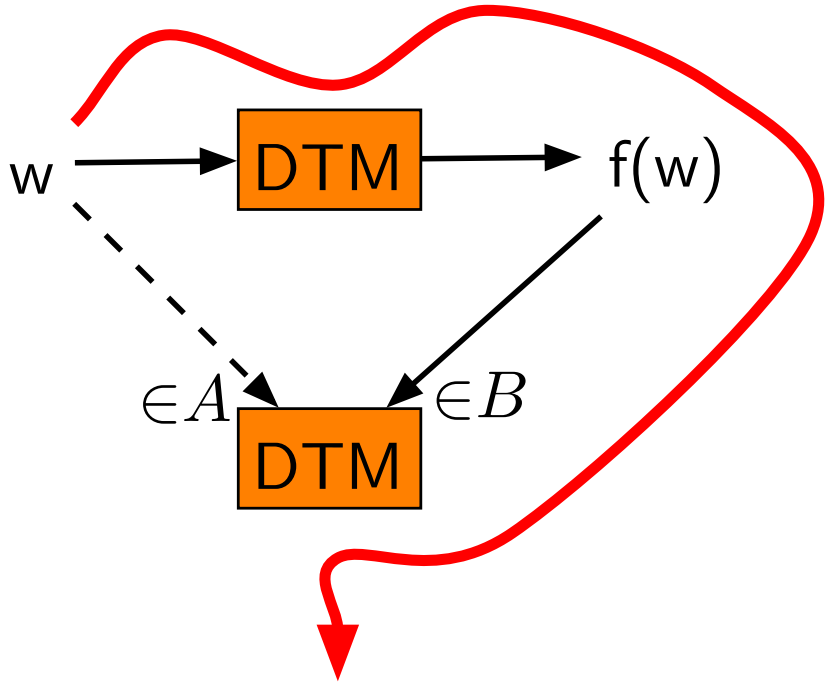
$$\exists f : \Sigma^* \to \Gamma^* : \forall x \in \Sigma^* : x \in A \iff f(x) \in B$$



von speziellem Interesse:
Polynomialzeitreduktion
($\leq_{\text{pol}}$),
logarithmische-Platz-Reduktion
($\leq_{\log}$).

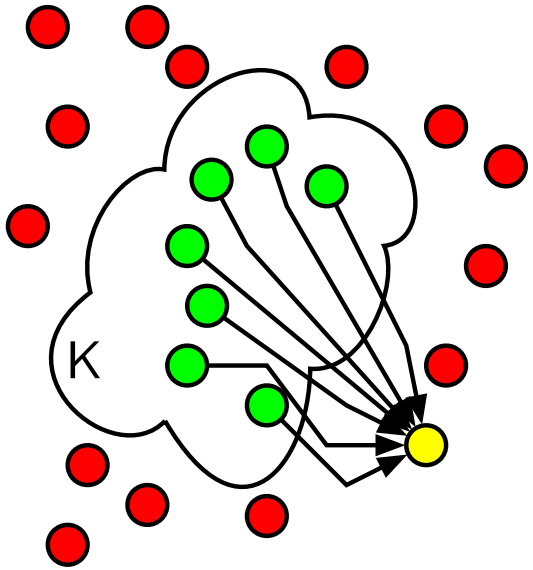
Reduktion (2)

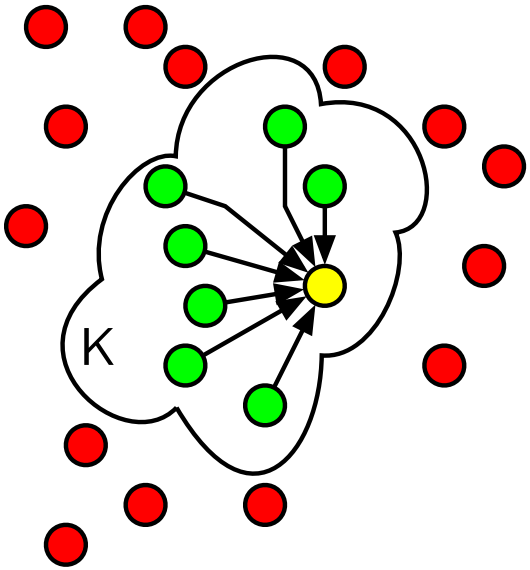
- Zurückführen der Entscheidbarkeit von A auf die Entscheidbarkeit von B:
- Voraussetzung $A \leq B$.



$\mathcal{K}$-Vollständigkeit

- Eine Menge $A \subseteq \Sigma^*$ heißt **hart** (oder besser: **schwer**) für eine Klasse $\mathcal{K}$ gdw. $\quad \forall B \in \mathcal{K} : B \leq A$
- Eine Menge $A \subseteq \Sigma^*$ heißt **vollständig** für eine Klasse $\mathcal{K}$ gdw. $\quad A \in \mathcal{K} \ \wedge \ \forall B \in \mathcal{K} : B \leq A$





$\mathcal{NP}$-Vollständigkeit

- besonders interessannter Spezialfall der Vollständigkeit:
- Eine Menge $A \subseteq \Sigma^*$ heißt $\mathcal{NP}$**-vollständig** gdw.

$$A \in \mathcal{NP} \ \wedge \ \forall B \in \mathcal{NP} : B \leq_{\text{pol}} A$$

- Ein $\mathcal{NP}$-vollständiges Problem ist somit eines der **schwersten** bzw. **umfassendsten** Probleme innerhalb der Klasse $\mathcal{NP}$.
- Wichtige Eigenschaft: Transitivität von $\leq_{\text{pol}}$.
- Ist A ein $\mathcal{NP}$-vollständiges Problem und gilt $A \leq_{\text{pol}} B$, so ist auch B $\mathcal{NP}$-vollständig.

Ausblick

- Existenz eines $\mathcal{NP}$-vollständigen Problems
  - Konstruktion am Beispiel des Dominoproblems (Parkettierung)
- Einfache Folgerungen
- das $\mathcal{P}=\mathcal{NP}$-Problem