Reductionen in der Berechenbarkeitstheorie

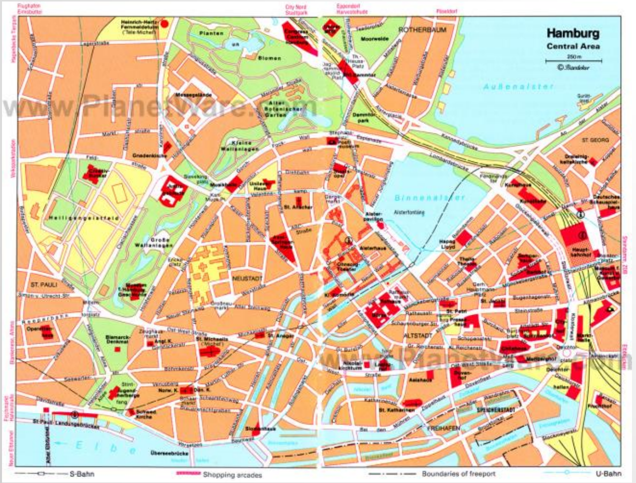
Problem: Wie komme ich
von hier zum Hamburger Hbf?

Beispiel

P1 Wie komme ich von hier zum Hamburger Hbf?

kann ich reduzieren auf

verwende für



Finde jemand, der den Weg kennt!

Alternativ: Finde eine Stadtkarte!

Alternativ: Finde ein Taxi!

…

P2

Beispiel

P1 Wie komme ich von hier zum Hamburger Hbf?

kann ich reduzieren auf

verwende für

Finde je… d, der den Weg kennt!

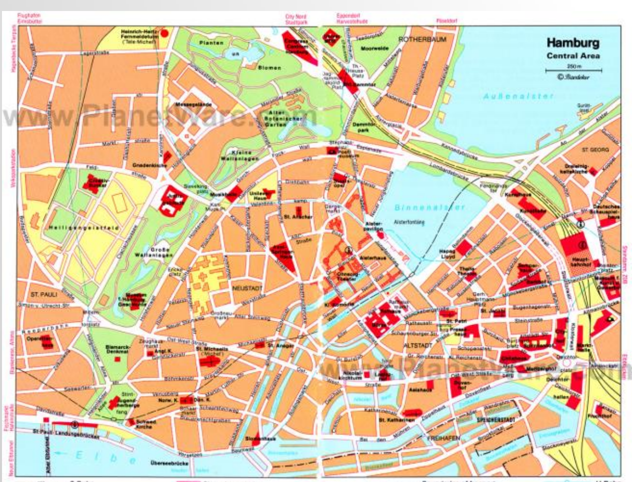
Alternat… de eine Stadtkarte!

Alte…

…

P2

Reduziere P1 auf P2!



Reduktionen

Ein mächtiges Konzept in der Theoretischen Informatik.

Informelle Definition: Reduktion

Problem P1 kann in ein Problem P2 transformiert werden (Funktion *f*), sodass Lösung von P2 zur Lösung von P1 verwendet werden kann.

$$\begin{tikzcd} \mathrm{P1} \arrow[r, bend left, "\text{Transformation } f"] & \mathrm{P2} \arrow[l, bend left, "\text{verwende Lösung}"] \end{tikzcd}$$

Löse!

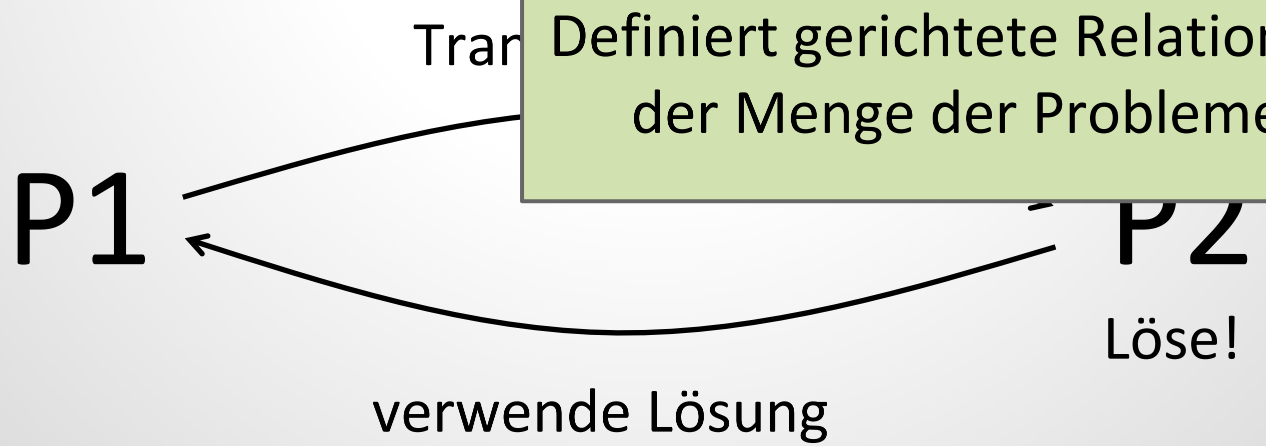
Reduktionen

Ein mächtiges Konzept in der Theoretischen Informatik.

Informelle Definition: Reduktion

Problem P1 kann in ein Problem P2 transformiert werden (Funktion *f*), sodass Lösung von P2 zur Lösung von P1 verwendet werden kann.

Definiert gerichtete Relation auf der Menge der Probleme!



Reduktionen

Ein mächtiges Konzept in der Theoretischen Informatik.

Informelle Definition: Reduktion

… roblem P2 transformiert … ass Lösung von P2 zur Lösung von … endet w… rden kann.

Einfach, aber Vorsicht: Richtung der Relation, Eigenschaften der Reduktion, etc. wichtig!

Definiert gerichtete Relation auf der Menge der Probleme!

Tran…

P1

P2

Löse!

verwende Lösung

Beispiele

- Beispiele im Alltag: Rekursiv
  - Das Problem von **Berlin nach Hamburg** zu kommen reduziert sich auf das Problem, eine **Fahrkarte zu kaufen**
  - … das wiederum reduziert sich darauf, **Geld für die Fahrkarte** zu verdienen
  - … reduziert sich auf das Problem, einen **Job zu finden**

Beispiele

- Beispiele im Alltag: Rekursiv
  - Das Problem von **Berlin nach Hamburg** zu kommen reduziert sich auf das Problem, eine **Fahrkarte zu kaufen**
  - … das wiederum reduziert sich darauf, **Geld für die Fahrkarte** zu verdienen
  - … reduziert sich auf das Problem, einen **Job zu finden**

Fläche?

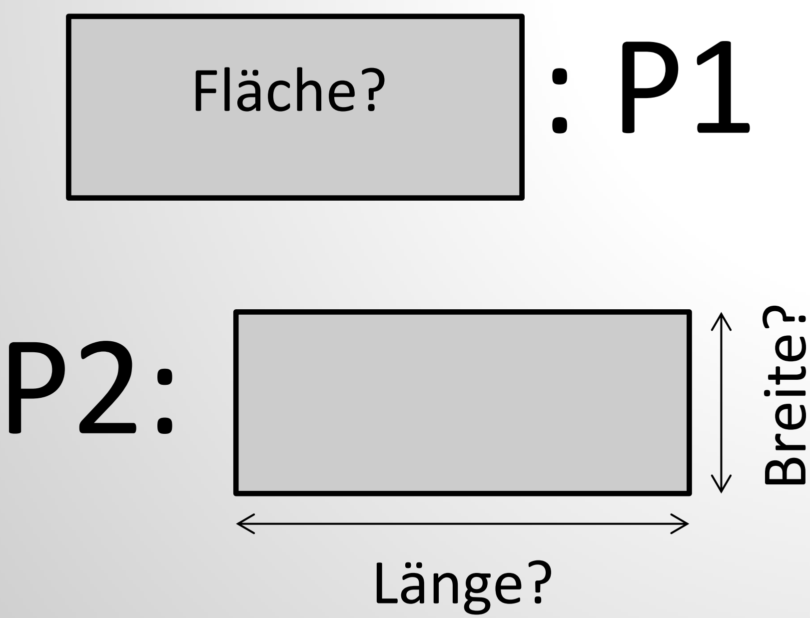
: P1

Wie muss ich vorgehen, um **Fläche des Rechtecks** zu bestimmen?

- Beispiele in der Mathematik:

Beispiele

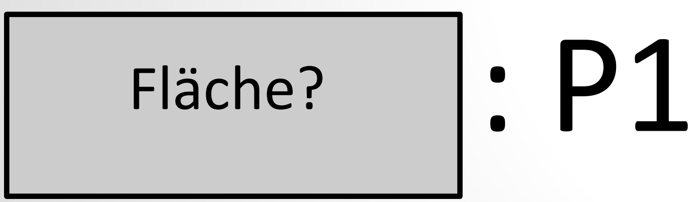
- Beispiele im Alltag: Rekursiv
  - Das Problem von **Berlin nach Hamburg** zu kommen reduziert sich auf das Problem, eine **Fahrkarte zu kaufen**
  - … das wiederum reduziert sich darauf, **Geld für die Fahrkarte** zu verdienen
  - … reduziert sich auf das Problem, einen **Job zu finden**

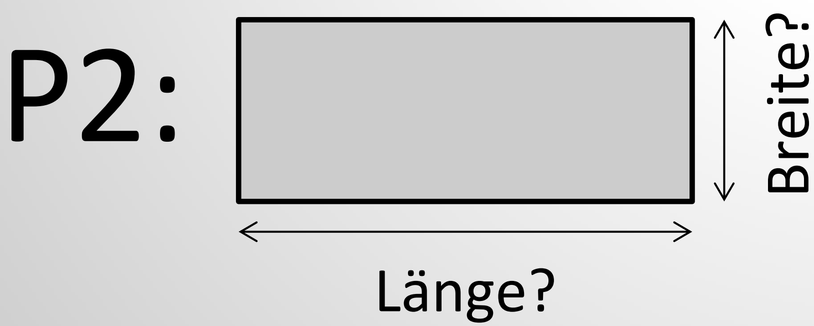


- Beispiele in der Mathematik:
  - Das Problem die **Fläche eines Rechtecks** zu bestimmen reduziert sich auf das Problem, dessen **Seitenlängen zu bestimmen.** (Einfach!)

Beispiele

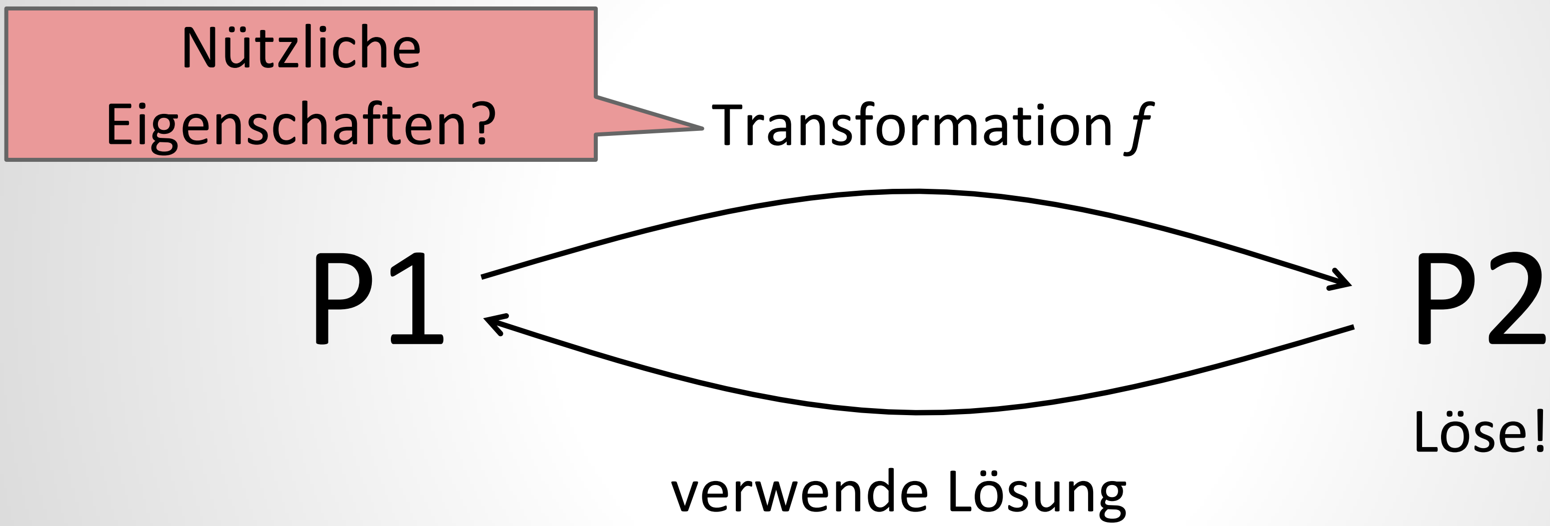
- Beispiele im Alltag: Rekursiv
  - Das Problem von **Berlin nach Hamburg** zu kommen reduziert sich auf das Problem, eine **Fahrkarte zu kaufen**
  - … das wiederum reduziert sich darauf, **Geld für die Fahrkarte** zu verdienen
  - … reduziert sich auf das Problem, einen **Job zu finden**





- Beispiele in der Mathematik:
  - Das Problem die **Fläche eines Rechtecks** zu bestimmen reduziert sich auf das Problem, dessen **Seitenlängen zu bestimmen.** (Einfach!)
  - Das Problem ein **lineares Gleichungssystem** zu lösen reduziert sich auf das Problem, eine **Matrix zu invertieren**.

Eigenschaften der Reduktion



Beispiel: Sei

P1: Finde Weg zum Hamburger Hbf.

P2: Finde eine City Map.

Eigenschaften der Reduktion

Nützliche Eigenschaften?

Transformation *f*

P1 P2

Löse!

verwende Lösung

Beispiel: Sei

P1: Finde Weg zum Hamburger Hbf.

P2: Finde eine City Map.

Möchte nicht zuerst zu einem Shop am Hbf gehen, um P2 zu lösen!

Eigenschaften der Reduktion

"It depends!": Welches Ziel / welche Anwendung?

Nützliche Eigenschaften?

Transformation *f*

P1 → P2

P1 ← P2

Löse!

verwende Lösung

Beispiel: Sei

P1: Finde Weg zum Hamburger Hbf.

P2: Finde eine City Map.

Möchte nicht zuerst zu einem Shop am Hbf gehen, um P2 zu lösen!

Komplexität vs Berechenbarkeit

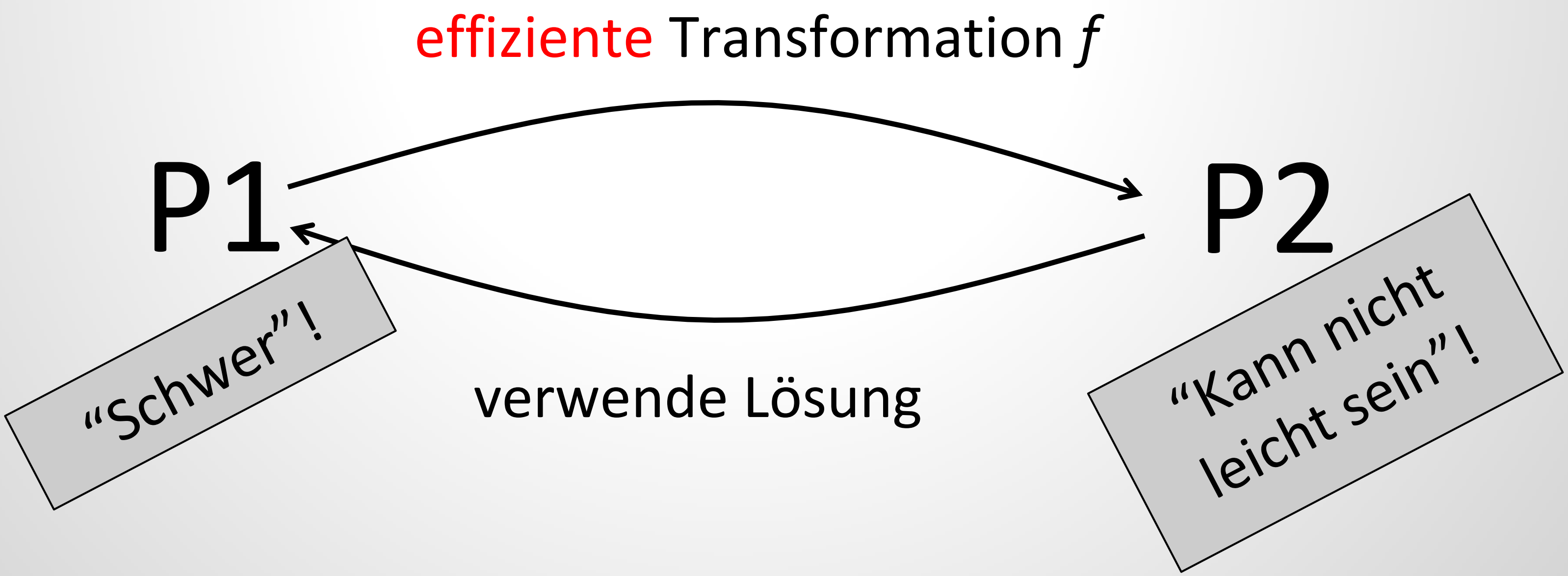
- Reduktionen in der Komplexitätstheorie:
  - Will effiziente Lösung für P1 mittels Lösung von P2
  - Auch Reduktion ("Umweg") sollte Zeit/Speicher effizient sein
- Reduktionen in der Berechenbarkeitstheorie
  - Will zeigen, dass es möglich ist, P1 zu lösen (mit P2)
  - Reduktion muss "nur" "berechenbar" sein

$$\begin{tikzcd} \mathrm{P1} \arrow[rr, bend left, "\text{Transformation } f"] & & \mathrm{P2} \arrow[ll, bend left, "\text{verwende Lösung}"] \end{tikzcd}$$

Anwendungen (1)

- Beispiel Komplexitätstheorie: Reduktionen für untere Schranken (“Lower Bounds”)

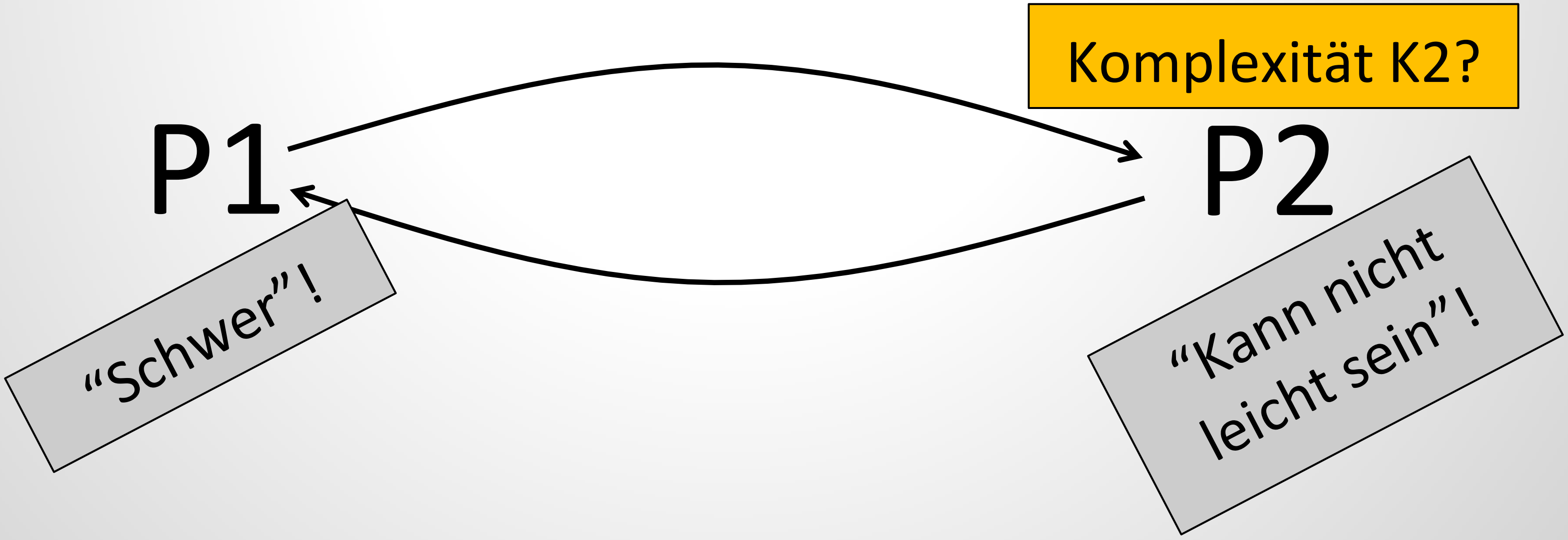
Beispiel: Wenn ich weiss, dass P1 (z.B. SAT) NP-schwer, kann ich durch Reduktion zeigen, dass viele weitere Probleme P2 auch NP-schwer sind.



Anwendungen (1)

- <u>Beispiel Komplexitätstheorie:</u> Reduktionen für untere Schranken (“Lower Bounds”)

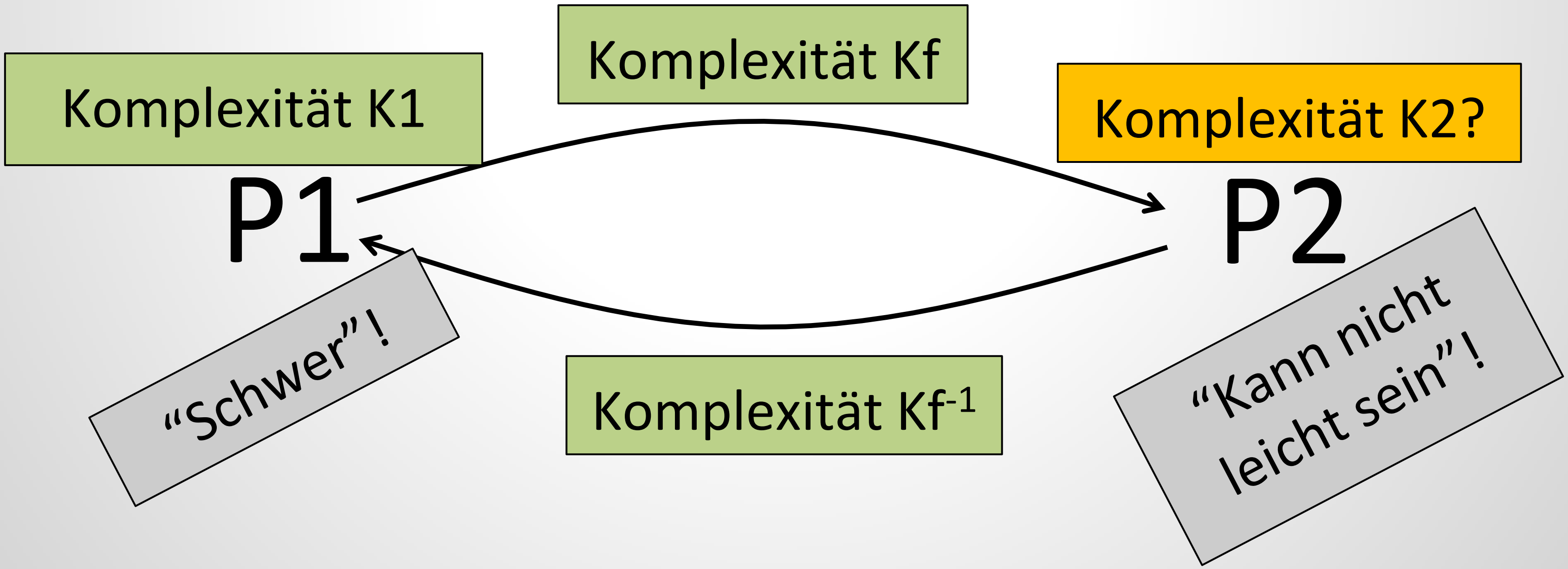
Beispiel: Wenn ich weiss, dass P1 (z.B. SAT) NP-schwer, kann ich durch Reduktion zeigen, dass viele weitere Probleme P2 auch NP-schwer sind.



Anwendungen (1)

- <u>Beispiel Komplexitätstheorie:</u> Reduktionen für untere Schranken (“Lower Bounds”)

Beispiel: Wenn ich weiss, dass P1 (z.B. SAT) NP-schwer, kann ich durch Reduktion zeigen, dass viele weitere Probleme P2 auch NP-schwer sind.

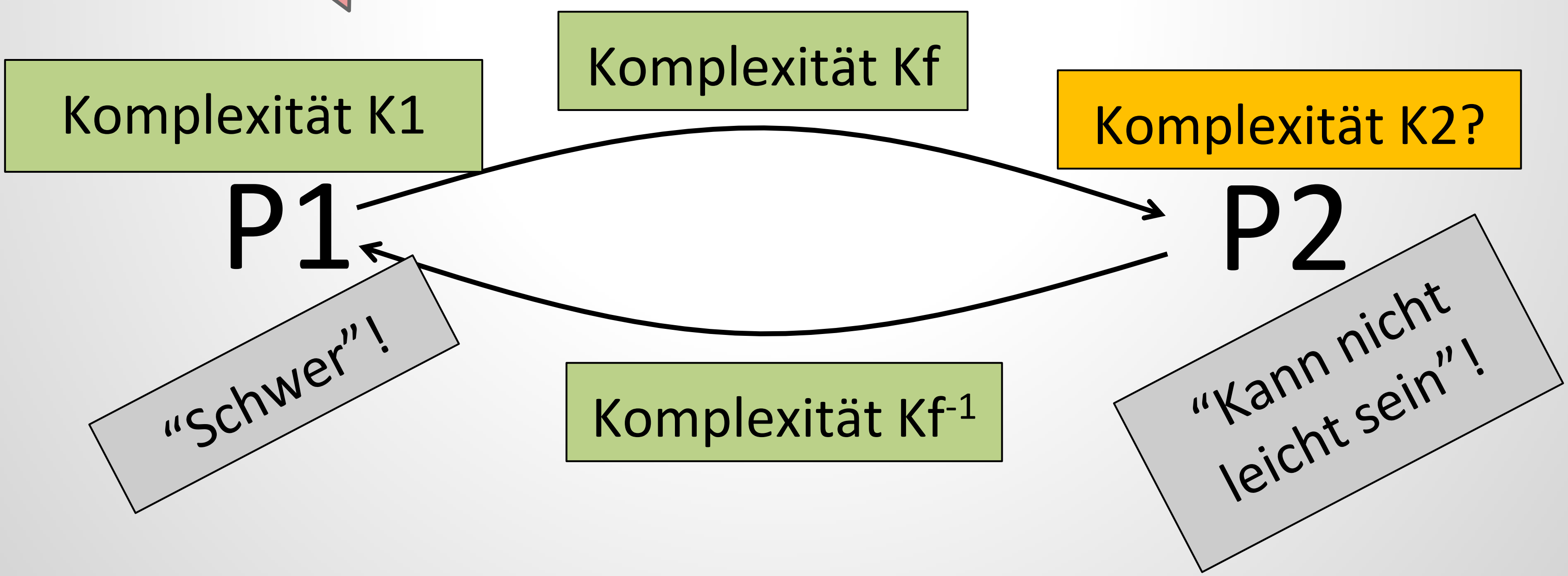


Anwendungen (1)

- Beispiel Komplexitätstheorie: Reduktionen für untere

Wenn K1 schwer und Kf und Kf^{-1} leicht, dann auch K2 schwer.

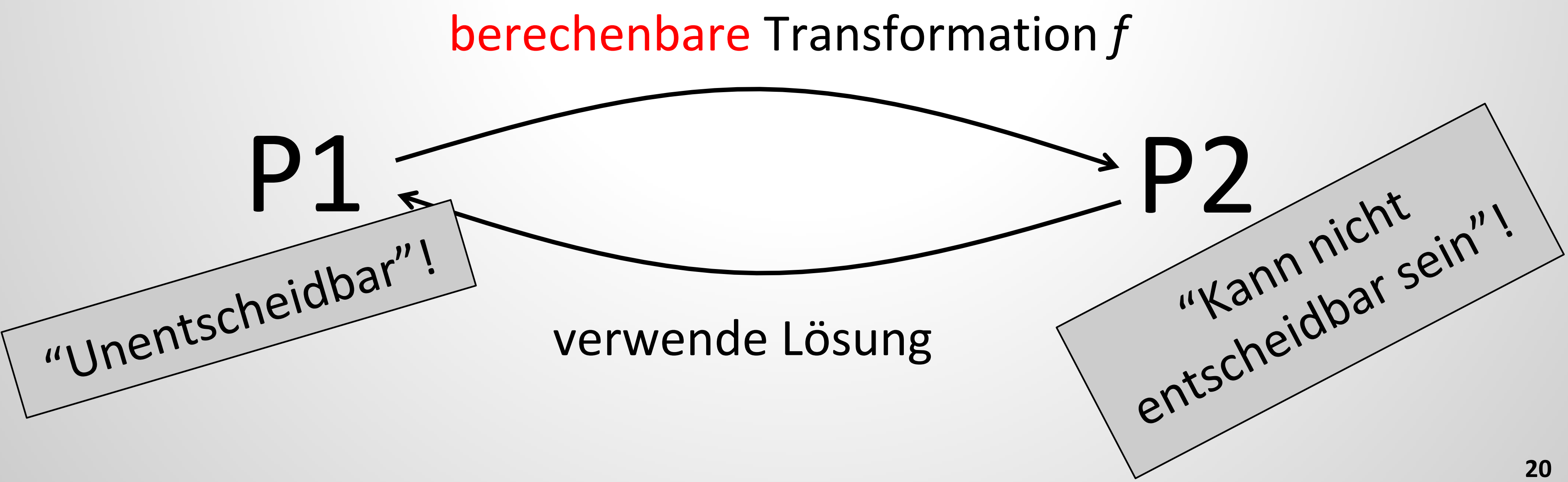
Beispie… …ch weiss, dass P1 (z.B. SAT) NP-schwer, kann ich durch Reduktion z… …, dass viele weitere Probleme P2 auch NP-schwer sind.



Anwendungen (2)

- <u>Beispiel Berechenbarkeitstheorie:</u> Reduktionen für negative Resultate (“Impossibility Results”)

Beispiel: Wenn ich weiss, dass P1 unentscheidbar ist (z.B. Sprache der durch TMs *M* akzeptierte Wörter *w*, A_{TM}: Diagonalisierung), kann ich durch Reduktion zeigen, dass andere Probleme P2 auch unentscheidbar sind.

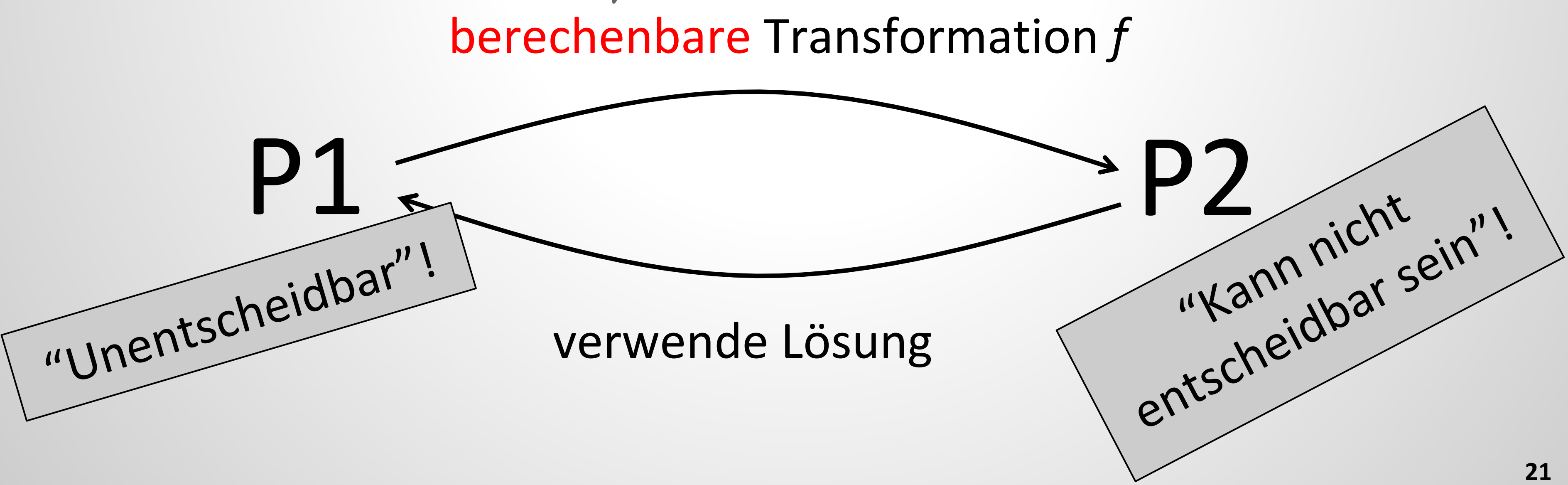


Anwendungen (2)

- Beispiel Berechenbarkeitstheorie: Reduktionen für negative Resulta ts”)

Wichtiges Konzept:
Many-One Reduktion

Beispiel: Wenn ich w st (z.B. Sprache der durch TMs *M* akzeptierte W er *w*, A_{TM}: Diagonalisierung), kann ich durch Reduktion zeigen, d andere Probleme P2 auch unentscheidbar sind.



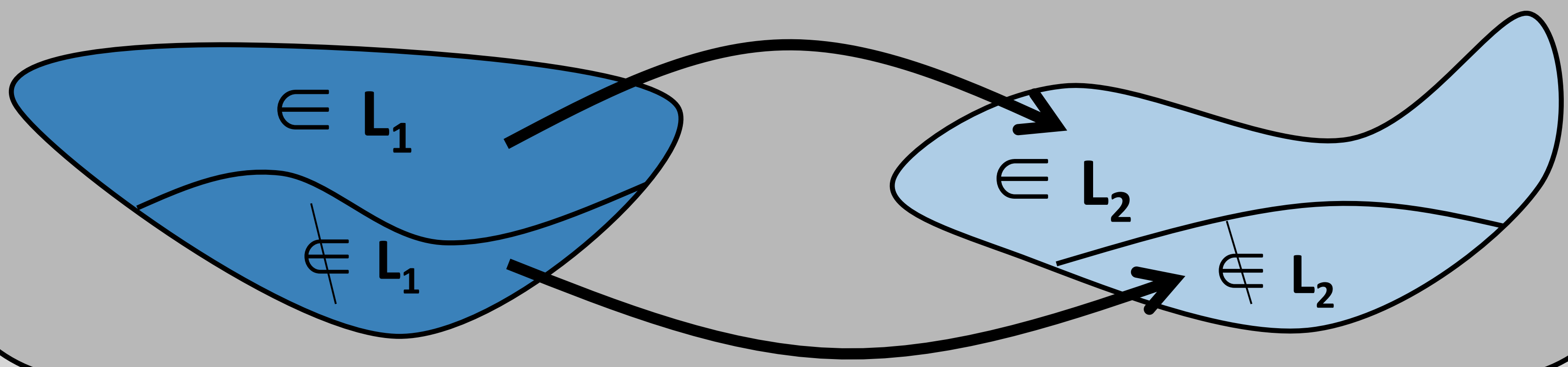
Reduktionen in Berechenbarkeitstheorie

- Formalisierung der Konzepte: Fokus auf Entscheidungsprobleme
- In Sprachformalismus: Probleme P der Form $w \in L$?

Definition: Many-One Reduktion *f* von L_1 auf L_2

Eine Funktion *f* heisst many-one Reduktion von Sprache $L_1 \subseteq \Sigma^*$ auf Sprache $L_2 \subseteq \Sigma^*$ gdw.:

1. *f* ist berechenbar und total
2. $\forall w \in \Sigma^*$ gilt dass $w \in L_1 \Leftrightarrow f(w) \in L_2$



Reduktionen in Berechenbarkeitstheorie

- Formalisierung der Konzepte: Fokus auf Entscheidungsprobleme
- In Sprachformalismus: Probleme P der Form $w \in L$?

Definition: Many-One Reduktion f von L_1 auf L_2

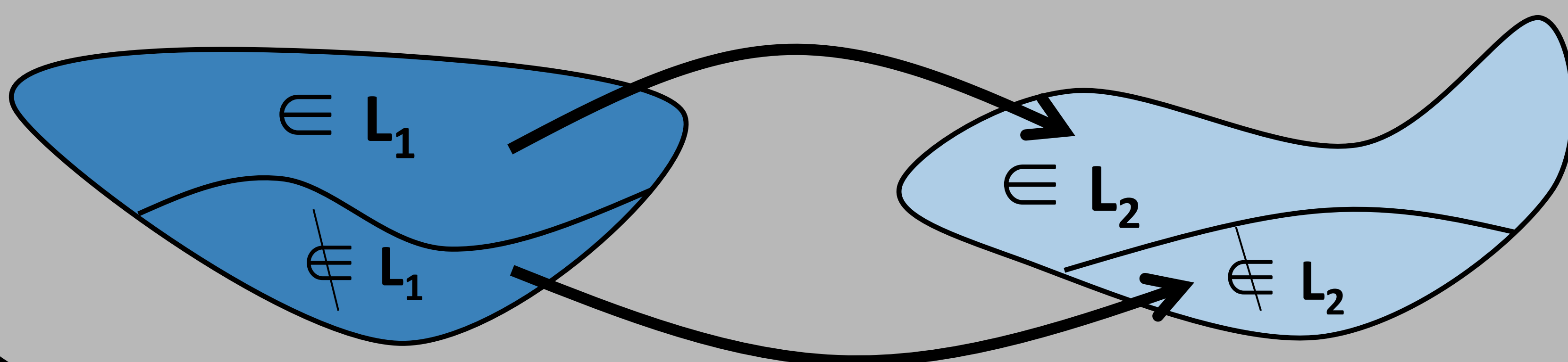
D.h. es gibt eine TM M_f, die f berechnet.

total = für alle Wörter definiert

… unktion f heisst …

$L_1 \subseteq \Sigma^*$ auf Sprache $L_2 \subseteq \Sigma^*$ …

1. f ist berechenbar und total
2. $\forall w \in \Sigma^*$ gilt dass $w \in L_1 \Leftrightarrow f(w) \in L_2$



Reduktionen in Berechenbarkeitstheorie

- Formalisierung der Konzepte: Fokus auf Entscheidungsprobleme
- In Sprachformalismus: Probleme P der Form $w \in L$?

Definition: Many-One Reduktion f von L_1 auf L_2

D.h. es gibt eine TM M_f, die f berechnet.

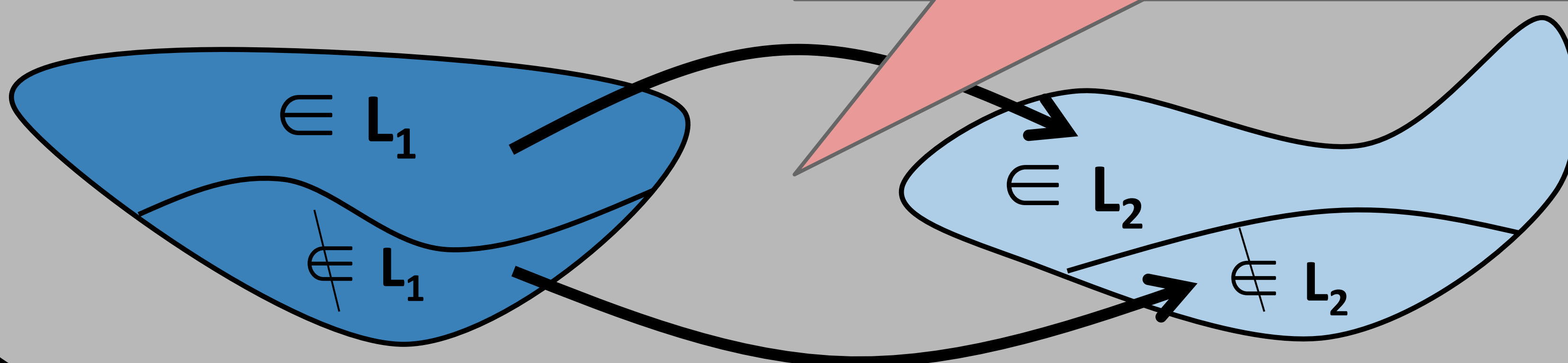
total = für alle Wörter definiert

... heisst ...

$L_1 \subseteq$ ... auf Sprache L_2 ...

1. f ist berechenbar und total
2. $\forall w \in \Sigma^*$ gilt dass $w \in$ ...

Funktion muss weder injektiv noch surjektiv sein!



Definition und Repetition

Definition: Many-One Reduzierbare Sprachen

Die Sprache L_1 heisst many-one reduzierbar auf Sprache L_2 gdw. es eine many-one Reduktion von L_1 auf L_2 gibt. Kurz: $L_1 \leq_m L_2$

Definition und Repetition

Definition: Many-One Reduzierbare Sprachen

Die Sprache L_1 heisst many-one reduzierbar auf Sprache L_2 gdw. es eine many-one Reduktion von L_1 auf L_2 gibt. Kurz: $L_1 \leq_m L_2$

Repetition: Akzeptierbare Sprache $\mathcal{A}$

Die Sprache L ist akzeptierbar, wenn es eine TM gibt, die für jedes Eingabewort $w \in L$ mit «Ja» hält.

Repetition: Entscheidbare Sprache $\mathcal{E}$

Die Sprache L ist entscheidbar, wenn es eine TM gibt, die für jedes $w \in L$ mit «Ja» hält und für jedes $w \notin L$ mit «Nein».

Sehr mächtiges Theorem:

Theorem: Implikationen der Reduktion

Sei $L_1 \leq_m L_2$, dann gilt: $L_2 \in \mathbb{E} \Rightarrow L_1 \in \mathbb{E}$.

Sehr mächtiges Theorem:

Theorem: Implikationen der Reduktion

Sei $L_1 \leq_m L_2$, dann gilt: $L_2 \in \mathbf{E} \Rightarrow L_1 \in \mathbf{E}$.

Beispiel: Wie kann ich damit zeigen, dass eine Sprache $L \notin \mathbf{E}$?

Sehr mächtiges Theorem:

Theorem: Implikationen der Reduktion

Sei $L_1 \leq_m L_2$, dann gilt: $L_2 \in \mathbb{E} \Rightarrow L_1 \in \mathbb{E}$.

Beispiel: Wie kann ich damit zeigen, dass eine Sprache $L \notin \mathbb{E}$?

Zeige, dass es 1. eine many-one Reduktion *f* von einer 2. unentscheidbaren Sprache $L' \notin \mathbb{E}$ ($=L_1$) auf L ($=L_2$) gibt:

$$\mathbf{L' \leq_m L}$$

Dann folgt aus Theorem: falls L ($=L_2$) entscheidbar wäre, Widerspruch zu Annahme dass L' ($=L_1$) unentscheibar:

$$\mathbf{L} \in \mathbb{E} \Rightarrow \mathbf{L'} \in \mathbb{E} : \text{Widerspruch zu } \mathbf{L'} \notin \mathbb{E}$$

Theorem: Implikationen der Reduktion

Sei $L_1 \leq_m L_2$, dann gilt: $L_2 \in \mathbf{E} \Rightarrow L_1 \in \mathbf{E}$.

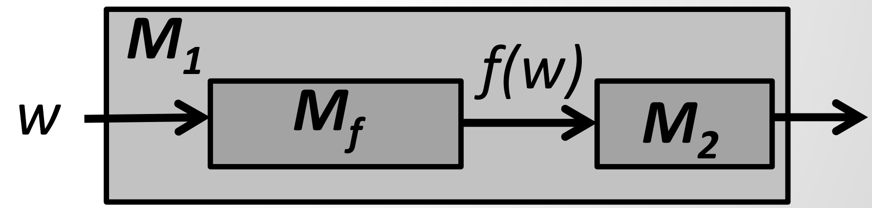
Wie kann man ein solches Theorem beweisen?

Theorem: Implikationen der Reduktion

Sei $L_1 \leq_m L_2$, dann gilt: $L_2 \in \mathbb{E} \Rightarrow L_1 \in \mathbb{E}$.

Beweis. Konstruktiv!

- Nehme an $L_2 \in \mathbb{E}$: also gibt es eine TM M_2, die L_2 entscheidet
- Sei f eine many-one Reduktion $\leq_m$ von L_1 auf L_2: berechenbar!
- Sei M_f die TM, die f berechnet
- Konstruiere die TM M_1:
  eine Hintereinanderschaltung
  von M_f und M_2.



- Es gilt: $M_1(w)$ akz. $\Leftrightarrow$ $M_2(f(w))$ akz. $\Leftrightarrow$ $f(w) \in L_2 \Leftrightarrow w \in L_1$
- Also entscheidet M_1 L_1, und somit: $L_1 \in \mathbb{E}$.

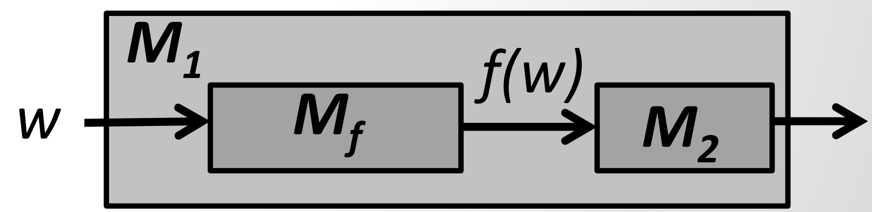
Theorem: Implikationen der Reduktion

Sei $L_1 \leq_m L_2$, dann gilt: $L_2 \in \mathbb{E} \Rightarrow L_1 \in \mathbb{E}$.

Eine solche TM muss existieren, da *f* per Definition berechenbar.

Beweis. Konstruktiv!

- Nehme an $L_2 \in \mathbb{E}$, also gibt es eine TM M_2, die L_2 entscheidet
- Sei *f* eine many-one Reduktion $\leq_m$ von L_1 auf L_2: berechenbar!
- Sei M_f die TM, die *f* berechnet
- Konstruiere die TM M_1:
  eine Hintereinanderschaltung
  von M_f und M_2.



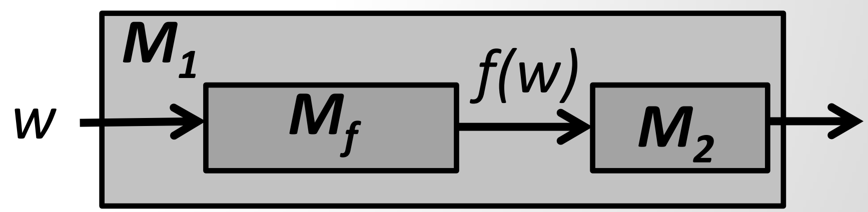
- Es gilt: $M_1(w)$ akz. $\Leftrightarrow$ $M_2(f(w))$ akz. $\Leftrightarrow$ $f(w) \in L_2 \Leftrightarrow w \in L_1$
- Also entscheidet M_1 L_1, und somit: $L_1 \in \mathbb{E}$.

Theorem: Implikationen der Reduktion

Sei $L_1 \leq_m L_2$, dann gilt: $L_2 \in \mathbb{E} \Rightarrow L_1 \in \mathbb{E}$.

Beweis. Konstruktiv!

- Nehme an $L_2 \in \mathbb{E}$: also gibt es eine TM M_2, die L_2 entscheidet
- Sei f eine many-one Reduktion $\leq_m$ von L_1 auf L_2: berechenbar!
- Sei M_f die TM, die f berechnet
- Konstruiere die TM M_1: eine Hintereinanderschaltung von M_f und M_2.



- Es gilt: $M_1(w)$ akz. $\Leftrightarrow$ $M_2(f(w))$ akz. $\Leftrightarrow$ $f(w) \in L_2 \Leftrightarrow w \in L_1$
- Also entscheidet M_1 L_1, und somit: $L_1 \in \mathbb{E}$.

Theorem: Implikationen der Reduktion

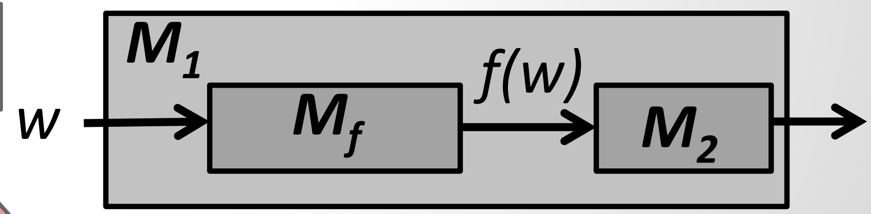
Sei $L_1 \leq_m L_2$, dann gilt: $L_2 \in \mathbb{E} \Rightarrow L_1 \in \mathbb{E}$.

Beweis. Konstruktiv!

- Nehme an $L_2 \in \mathbb{E}$: also gibt es eine TM M_2, die L_2 entscheidet
- Sei f eine many-one Reduktion $\leq_m$ von L_1 auf L_2: berechenbar!
- Sei M_f die TM, die f berechnet
- Konstruiere d

  eine Hintereinanderschaltu

Definition von M_2



Konstruktion von M_1

- Es gilt: $M_1(w)$ akz. $\Leftrightarrow$ $M_2(f(w))$ akz. $\Leftrightarrow$ $f(w) \in L_2 \Leftrightarrow w \in L_1$
- Also entscheidet M_1 L_1, und som

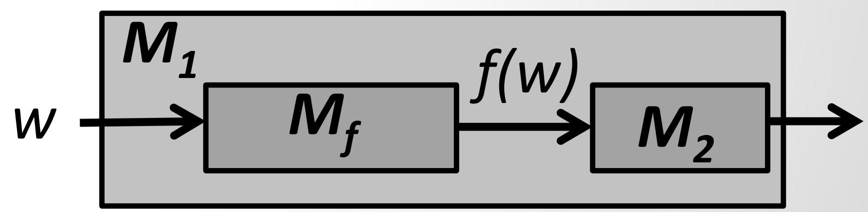
Definition Many-One Reduktion

Theorem: Implikationen der Reduktion

Sei $L_1 \leq_m L_2$, dann gilt: $L_2 \in \mathbb{E} \Rightarrow L_1 \in \mathbb{E}$.

Beweis. Konstruktiv!

- Nehme an $L_2 \in \mathbb{E}$: also gibt es eine TM M_2, die L_2 entscheidet
- Sei f eine many-one Reduktion $\leq_m$ von L_1 auf L_2: berechenbar!
- Sei M_f die TM, die f berechnet
- Konstruiere die TM M_1:
  eine Hintereinanderschaltung
  von M_f und M_2.



- Es gilt: $M_1(w)$ akz. $\Leftrightarrow$ $M_2(f(w))$ akz. $\Leftrightarrow$ $f(w) \in L_2$ $\Leftrightarrow$ $w \in L_1$
- Also entscheidet M_1 L_1, und somit: $L_1 \in \mathbb{E}$.

Beispiel: Das Halteproblem

Definition: Sprache A_{TM}

Sprache des Problems «Akzeptiert eine gewisse Turing Maschine *M* einen gegebenen Input *w*?»: $A_{TM} = \{\langle M,w\rangle \mid M \text{ akzeptiert } w\}$

Definition: Sprache $HALT_{TM}$

Sprache des Problems «Hält eine gewisse Turing Maschine *M* bei gegebenem Input *w*?»: $HALT_{TM} = \{\langle M,w\rangle \mid M \text{ hält auf } w\}$

Beispiel: Das Halteproblem

Definition: Sprache A_{TM}

Sprache des Problems «Akzeptiert eine gewisse Turing Maschine *M* einen gegebenen Input *w*?»: A_{TM} = {<*M*,*w*> | *M* akzeptiert *w*}

A_{TM} ist unentscheidbar: $A_{TM} \notin E$

Definition: Sprache $HALT_{TM}$

Sprache des Problems «Hält eine gewisse Turing Maschine *M* bei gegebenem Input *w*?»: $HALT_{TM}$ = {<*M*,*w*> | *M* hält auf *w*}

Auch unentscheidbar: wie beweisen?

Beispiel: Das Halteproblem

Definition: Sprache A_{TM}

Sprache des Problems «Akzeptiert eine gewisse Turing Maschine *M* einen gegebenen Input *w*?»: $A_{TM} = \{<M,w> \mid M \text{ akzeptiert } w\}$

A_{TM} ist unent-scheidbar: $A_{TM} \notin E$

Definition: Sprache $HALT_{TM}$

Sprache des Problems «Hält eine gewisse Turing Maschine *M* bei gegebenem Input *w*?»: $HALT_{TM} = \{<M,w> \mid M \text{ hält auf } w\}$

Auch unent-scheidbar: wie beweisen?

Zeige: $A_{TM} \leq_m HALT_{TM}$

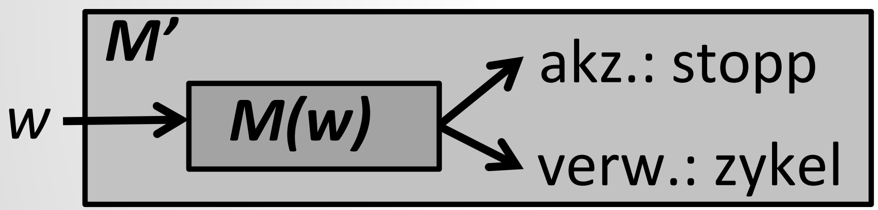
Theorem: $\mathbf{HALT}_{TM} \notin \mathbf{E}$

Das Problem $\mathbf{HALT}_{TM}$ ist unentscheidbar.

Beweis:

- Zeige Reduktion *f*: $\mathbf{A}_{TM} \leq_m \mathbf{HALT}_{TM}$. Daraus folgt das Theorem!
- Idee: Reduziere P1=«Akzeptiert M w?» auf P2=«Hält M’ auf w?»

  wobei *M’(w)*:



1. Simuliere M(w)
2. Wenn M(w) verwirft, dann zykel
3. Sonst stopp

- Reduktion $f(\langle M,w\rangle):=\langle M',w\rangle$ ist berechenbar und total, ausserdem:
- $\langle M,w\rangle \in \mathbf{A}_{TM} \Leftrightarrow$ *M* akz. *w* $\Leftrightarrow$ *M’(w)* hält $\Leftrightarrow f(\langle M,w\rangle) \in \mathbf{HALT}_{TM}$
- Also: $\mathbf{A}_{TM} \leq_m \mathbf{HALT}_{TM} \Rightarrow \mathbf{HALT}_{TM} \notin \mathbf{E}$

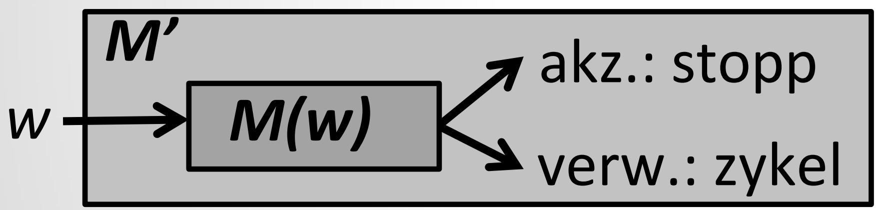
Theorem: $\mathrm{HALT}_{TM} \notin \mathrm{E}$

Das Problem HALT_{TM} ist unentscheidbar.

Beweis:

- Zeige Reduktion *f*: $\mathrm{A}_{TM} \leq_m \mathrm{HALT}_{TM}$. Daraus folgt das Theorem!
- Idee: Reduziere P1=«Akzeptiert M w?» auf P2=«Hält M' auf w?»

  wobei *M'(w)*:



1. Simuliere M(w)
2. Wenn M(w) verwirft, dann zykel
3. Sonst stopp

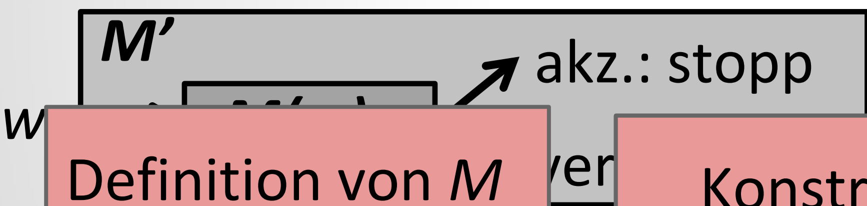
- Reduktion *f(<M,w>):=<M',w>* ist berechenbar und total, ausserdem:
- $\langle M,w\rangle \in \mathrm{A}_{TM} \Leftrightarrow M$ akz. $w \Leftrightarrow M'(w)$ hält $\Leftrightarrow f(\langle M,w\rangle) \in \mathrm{HALT}_{TM}$
- Also: $\mathrm{A}_{TM} \leq_m \mathrm{HALT}_{TM} \Rightarrow \mathrm{HALT}_{TM} \notin \mathrm{E}$

Theorem: $\mathrm{HALT}_{TM} \notin \mathrm{E}$

Das Problem HALT_{TM} ist unentscheidbar.

Beweis:

- Zeige Reduktion f: $\mathrm{A}_{TM} \leq_m \mathrm{HALT}_{TM}$. Daraus folgt das Theorem!
- Idee: Reduziere P1=«Akzeptiert M w?» auf P2=«Hält M' auf w?» wobei $M'(w)$:



1. Simuliere M(w)
2. Wenn M(w) verwirft, dann zykel
3. Sonst stopp

Definition von M

Konstruktion

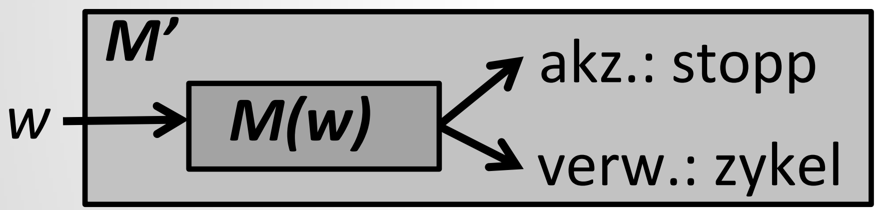
- Reduktion $f(<M,w>):=<M',w>$ ist berechenbar und total, ausserdem:
- $<M,w> \in \mathrm{A}_{TM} \Leftrightarrow M$ akz. $w \Leftrightarrow M'(w)$ hält $\Leftrightarrow f(<M,w>) \in \mathrm{HALT}_{TM}$
- Also: $\mathrm{A}_{TM} \leq_m \mathrm{HALT}_{TM} \Rightarrow \mathrm{HALT}_{TM} \notin \mathrm{E}$

Theorem: $\mathrm{HALT}_{TM} \notin \mathrm{E}$

Das Problem HALT_{TM} ist unentscheidbar.

Beweis:

- Zeige Reduktion *f*: $\mathrm{A}_{TM} \leq_m \mathrm{HALT}_{TM}$. Daraus folgt das Theorem!
- Idee: Reduziere P1=«Akzeptiert M w?» auf P2=«Hält M' auf w?»
  wobei *M'(w)*:



1. Simuliere M(w)
2. Wenn M(w) verwirft, dann zykel
3. Sonst stopp

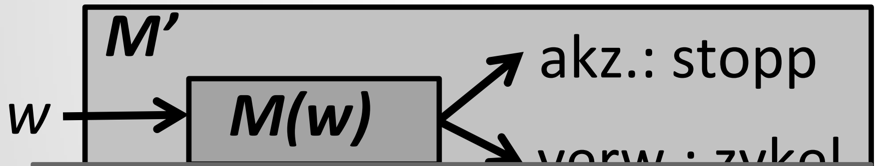
- Reduktion *f(<M,w>):=<M',w>* ist berechenbar und total, ausserdem:
- $<M,w> \in \mathrm{A}_{TM} \Leftrightarrow M$ akz. $w \Leftrightarrow M'(w)$ hält $\Leftrightarrow f(<M,w>) \in \mathrm{HALT}_{TM}$
- Also many-one Reduktion: $\mathrm{A}_{TM} \leq_m \mathrm{HALT}_{TM} \Rightarrow \mathrm{HALT}_{TM} \notin \mathrm{E}$

Theorem: $HALT_{TM} \notin E$

Das Problem $HALT_{TM}$ ist unentscheidbar.

Beweis:

- Zeige Reduktion f: $A_{TM} \leq_m HALT_{TM}$. Daraus folgt das Theorem!
- Idee: Reduziere P1=«Akzeptiert M w?» auf P2=«Hält M' auf w?» wobei $M'(w)$:



1. Simuliere M(w)
2. Wenn M(w) verwirft, dann zykel
3. Sonst stopp

Widerspruch zu Unentscheidbarkeit von A_{TM}

- F berechenbar und total, ausserdem:
- $<M,w> \in A_{TM}$ ⇔ M akz. w ⇔ $M'(w)$ hält ⇔ $f(<M,w>) \in HALT_{TM}$
- Also many-one Reduktion: $A_{TM} \leq_m HALT_{TM} \Rightarrow HALT_{TM} \notin E$

Ende

Anwendungen: Weiteres Beispiel

- **Komplexitätstheorie:**
  - Viele negative Resultate ("Lower Bounds")

Beispiel: Lower Bound des Konvexe Hülle Problems **CH** ist auch eine Lower Bound für das Sortierungsproblem **SORT**

Transformation: O(n)

SORT $\rightleftarrows$ **CH**

Sortiere zahlen $\{x_1, \ldots, x_n\}$!

Finde konvexe Hülle von Punkten $\{(x_1, x_1^2), \ldots, \{(x_n, x_n^2)\}$!

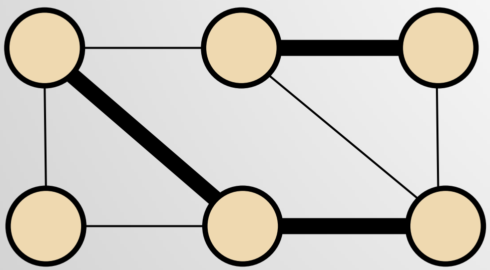
Parabel

Beispiel: Matching (1)

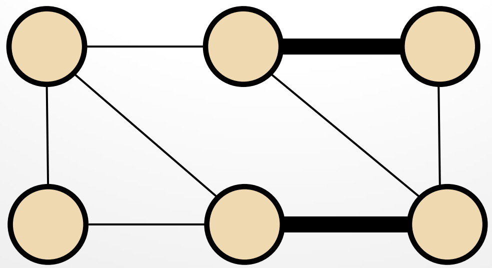
Definition: Maximales Matching

Input: Ungerichteter Graph $G=(V,E)$

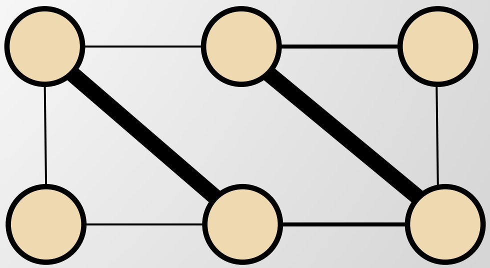
Output: Eine Teilmenge an Kanten $M \subseteq E$ ist ein gültiges Matching, gdw. keine zwei Kanten in M einen gemeinamen Knoten haben. M ist maximal gdw. es gültig ist und es keine Kante $e \in E \setminus M$ gibt, sodass $\{e\} \cup M$ auch ein gültiges Matching ist.



Gültig? Maximal?



Gültig? Maximal?



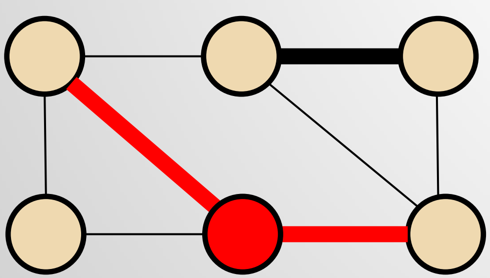
Gültig? Maximal?

Beispiel: Matching (1)

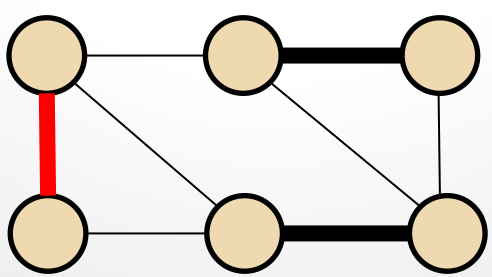
Definition: Maximales Matching

<u>Input:</u> Ungerichteter Graph $G=(V,E)$

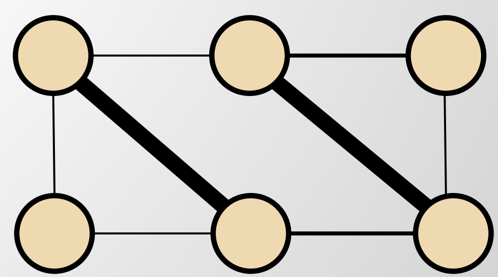
<u>Output:</u> Eine Teilmenge an Kanten $M \subseteq E$ ist ein gültiges Matching, gdw. keine zwei Kanten in M einen gemeinamen Knoten haben. M ist maximal gdw. es gültig ist und es keine Kante $e \in E \setminus M$ gibt, sodass $\{e\} \cup M$ auch ein gültiges Matching ist.



Ungültig!



Gültig, nicht maximal!



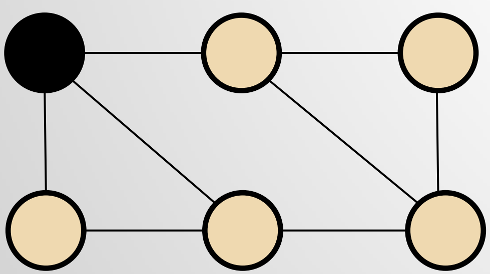
Gültig und maximal

Beispiel: Matching (2)

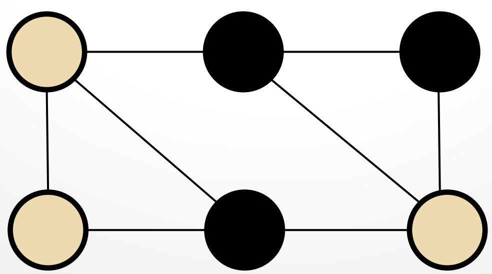
Definition: Maximales Independent Set

<u>Input:</u> Ungerichteter Graph *G=(V,E)*

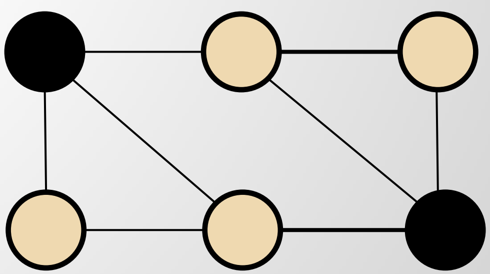
<u>Output:</u> Eine Teilmenge an Knoten $I \subseteq V$ ist ein gültiges Independent Set, gdw. es keine zwei Knoten in *I* gibt, die benachbart sind. *I* ist maximal gdw. es gültig ist und es keinen Knoten $v \in V \setminus I$ gibt, sodass $\{v\} \cup I$ auch ein gültiges Independent Set ist.



Gültig? Maximal?



Gültig? Maximal?



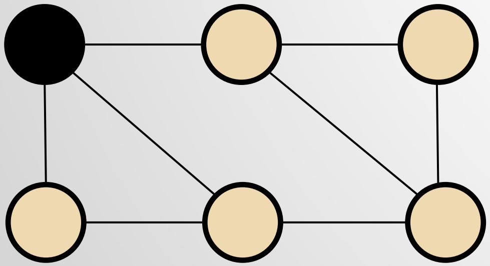
Gültig? Maximal?

Beispiel: Matching (2)

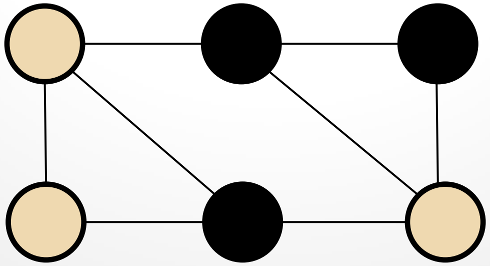
Definition: Maximales Independent Set

<u>Input:</u> Ungerichteter Graph $G=(V,E)$

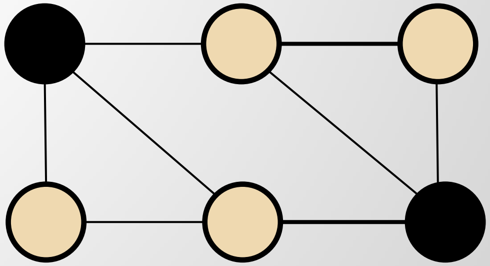
<u>Output:</u> Eine Teilmenge an Knoten $I \subseteq V$ ist ein gültiges Independent Set, gdw. es keine zwei Knoten in I gibt, die benachbart sind. I ist maximal gdw. es gültig ist und es keinen Knoten $v \in V \setminus I$ gibt, sodass $\{v\} \cup I$ auch ein gültiges Independent Set ist.



Gültig, nicht maximal!



Ungültig!



Gültig und maximal

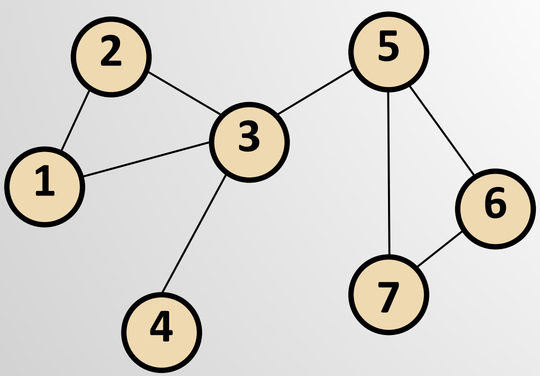
Beispiel: Matching (3)

- Beispiel
  - Gegeben: Algorithmus für maximales Independent Set
  - Gesucht: Algorithmus für maximales Matching
  - Methode: Effiziente Reduktion

Maximales Matching?

Transformation f

Maximales Independent Set



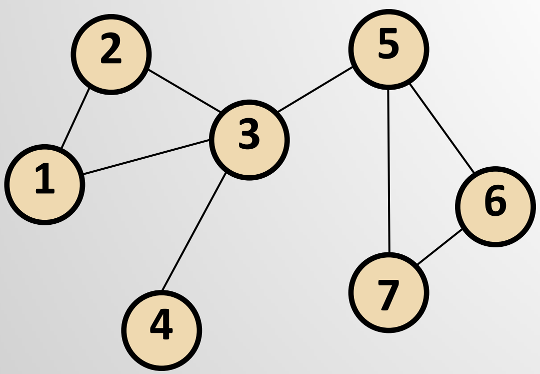
verwende Lösung

Beispiel: Matching (3)

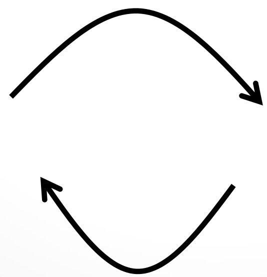
- Beispiel
  - Gegeben: Algorithmus für maximales Independent Set
  - Gesu … ing
  - Meth

1. Ersetze Kanten durch Knoten
2. Verbinde Knoten, deren Kanten inzident sind im urspr. Graphen

Maximales Matching?

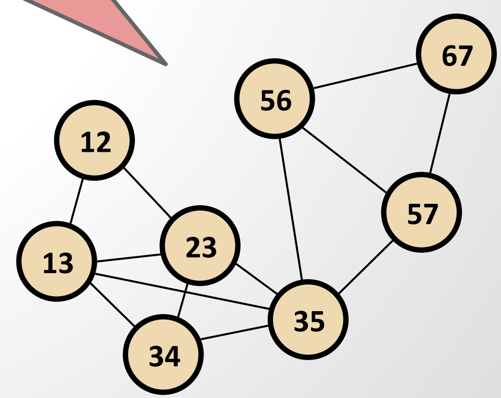


Transformation



verwende Lösung

Maximales Independent Set

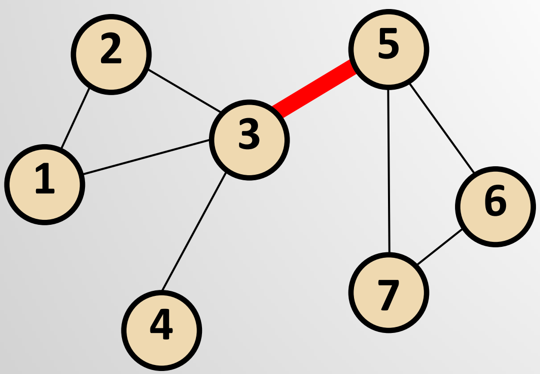


Beispiel: Matching (3)

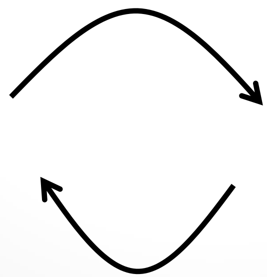
- Beispiel
  - Gegeben: Algorithmus für maximales Independent Set
  - Gesu … ing
  - Meth

1. Ersetze Kanten durch Knoten
2. Verbinde Knoten, deren Kanten inzident sind im urspr. Graphen

Maximales Matching?

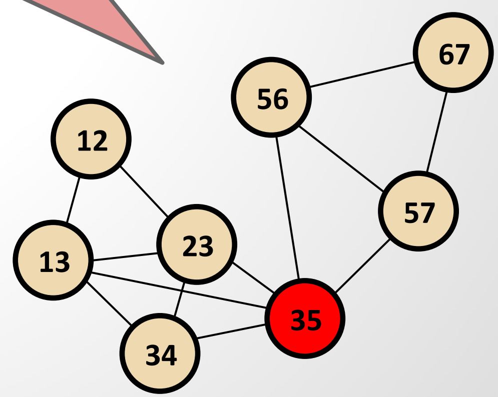


Transformatio



verwende Lösung

Maximales Independent Set

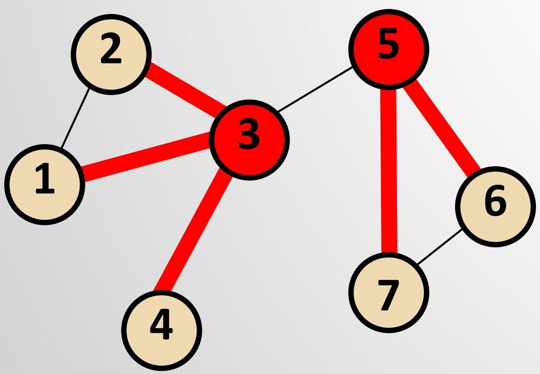


Beispiel: Matching (3)

- Beispiel
  - Gegeben: Algorithmus für maximales Independent Set
  - Gesu ... ing
  - Meth

1. Ersetze Kanten durch Knoten
2. Verbinde Knoten, deren Kanten inzident sind im urspr. Graphen

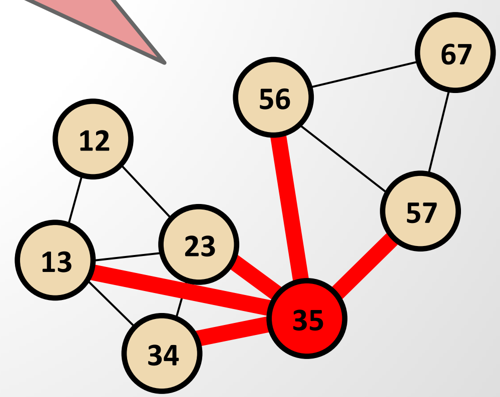
Maximales Matching?



Transformation

verwende Lösung

Maximales Independent Set



Beispiel: Matching (3)

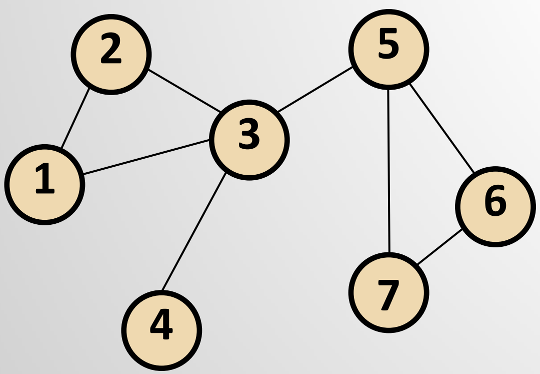
- Beispiel
  - Gegeben: Algorithmus für maximales Independent Set
  - Gesucht: Algorithmus für maximales Matching
  - Methode: Effiziente R

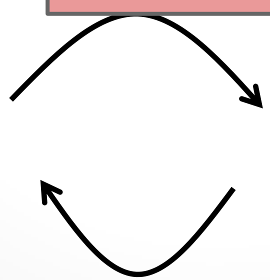
Berechne maximales Independent Set!

Maximales Matching?

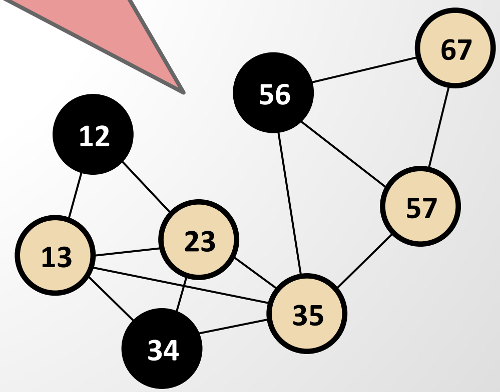
Trans

Maximales Independent Set



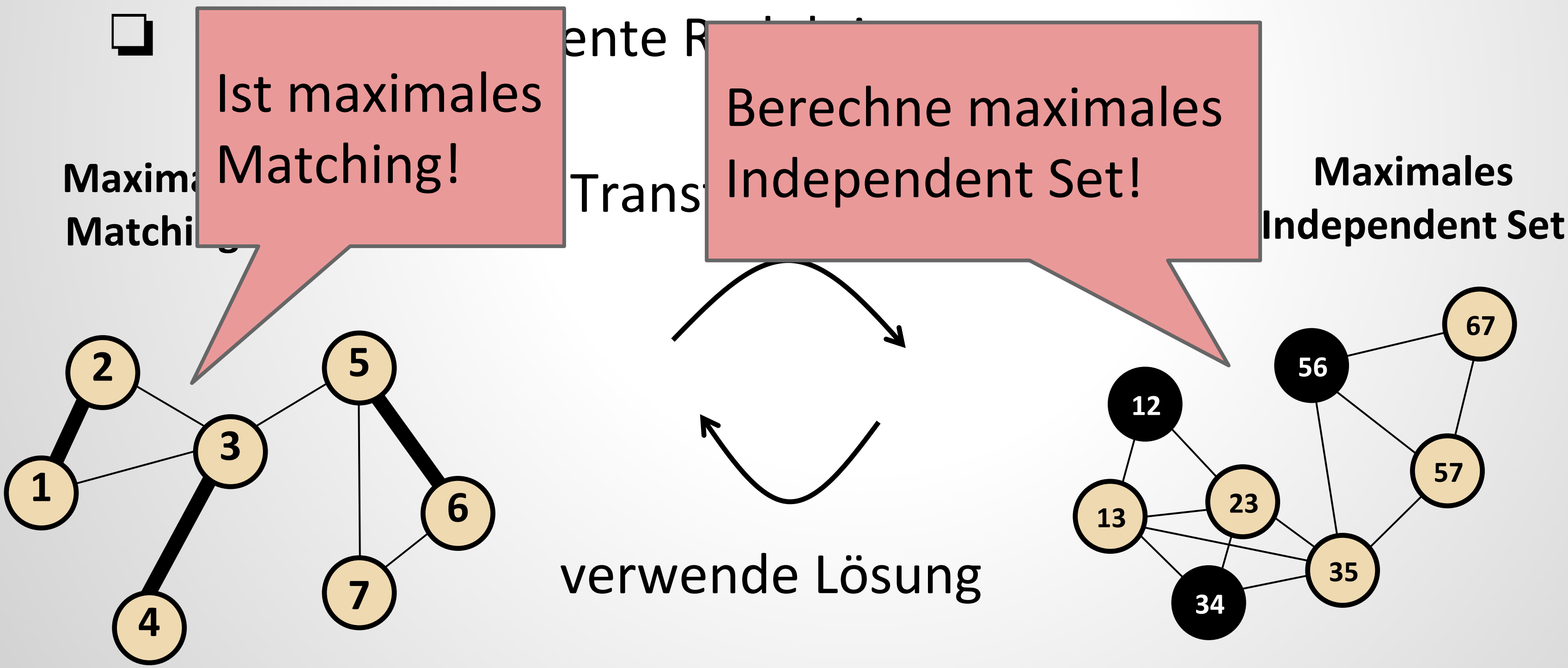


verwende Lösung



Beispiel: Matching (3)

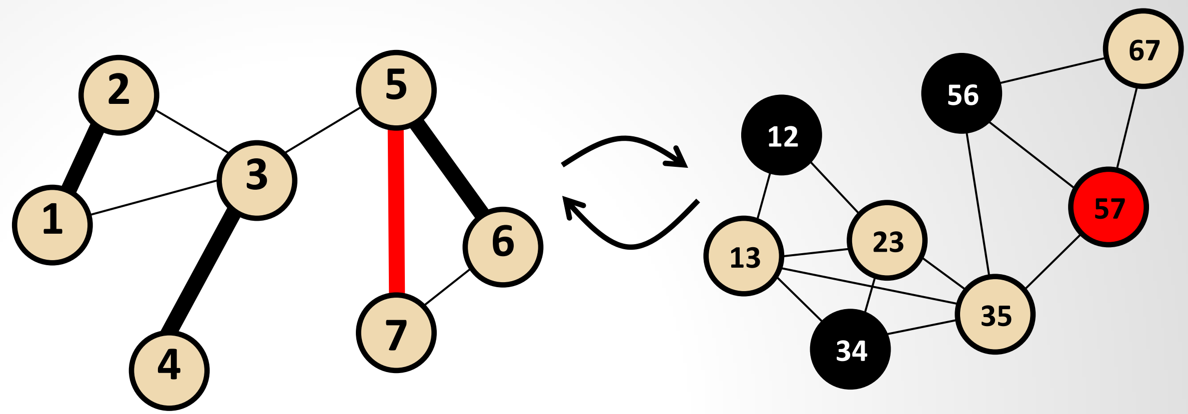
- Beispiel
  - Gegeben: Algorithmus für maximales Independent Set
  - Gesucht: Algorithmus für maximales Matching
  - ente R



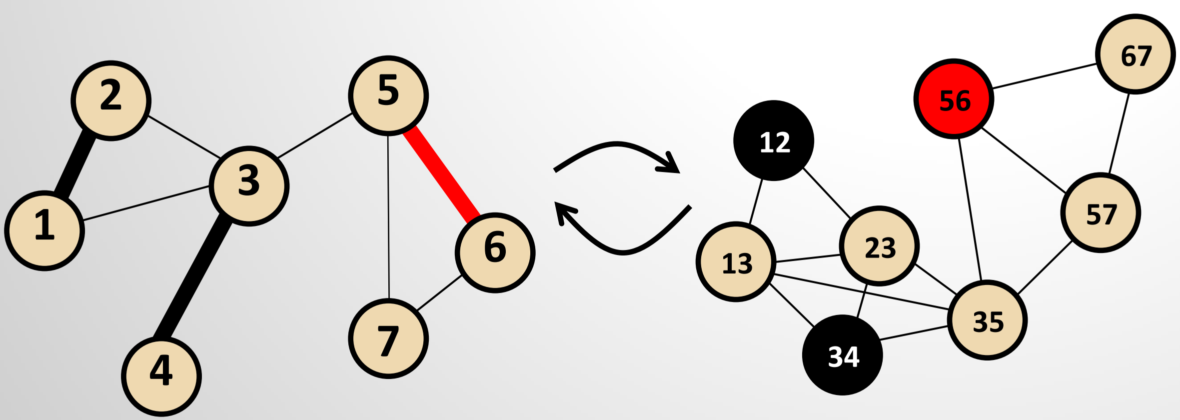
Beweisidee

<u>Gültig:</u> Durch Widerspruch. Falls Matching ungültig, existiert Knote mit zwei anliegenden Kanten. Dann sind aber auch die entsprechenden Kantenknoten nicht independent.

Matching **Independent Set**



Max. Matching **Max. Independent Set**

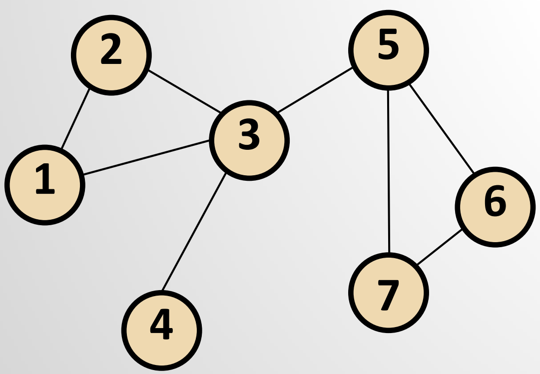


<u>Maximal:</u> Durch Widerspruch. Falls eine weitere Kante hinzugefügt werden kann, muss ein weiterer unabhängiger Knote existieren im transformierten Graphen. Verletzt Maximalität.

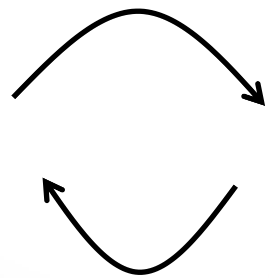
Bemerkung

- Transformation ist nicht nur effizient, sondern auch lokal
  - kann in einem verteilten System (z.B. Computer Netzwerk) lokal emuliert werden
  - Verteilter Algorithmus für Independent Set als verteilter Algorithmus für Matching benutzbar

Maximales Matching

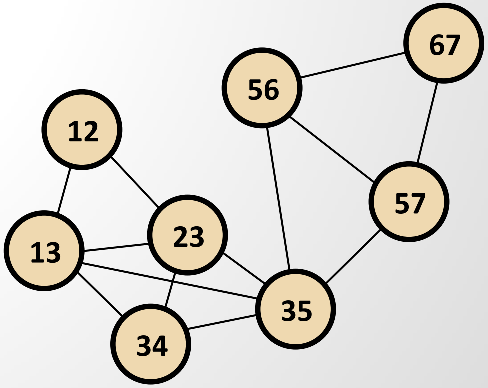


effiziente und lokale
Transformation f



verwende Lösung

Maximales Independent Set



Reduktionen in Berechenbarkeitstheorie

- Formalisierung mit Fo
  Objekt eine gewünsch

Funktion muss weder injektiv noch surjektiv sein: ein Element in Zielmenge kann beliebig viel mal angenommen werden!

Definition: Many-One-Reduktion von L_1 auf L_2

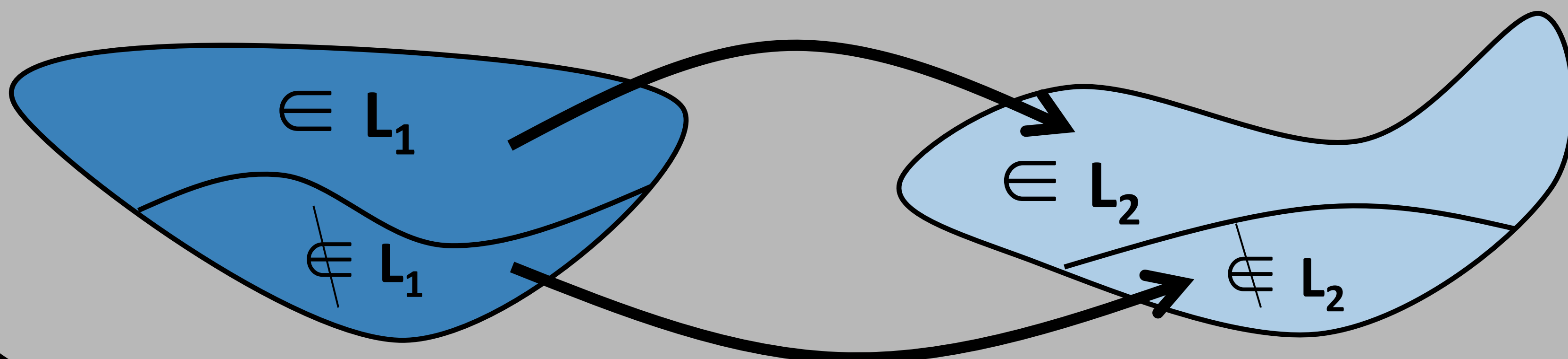
D.h. Es gibt eine TM M_f, die f berechnet.

Funktion f heißt m

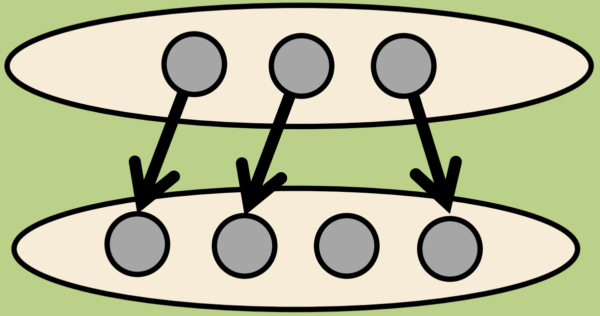
total = für alle Wörter definiert

$L_1 \subseteq \Sigma^*$ auf Sprache $L_2 \subseteq \Sigma^*$

1. f ist berechenbar und total
2. $\forall w \in \Sigma^* : w \in L_1 \Leftrightarrow f(w) \in L_2$



Injektiv (“linkseindeutig”): Jedes Element der Zielmenge höchstens einmal angenommen.

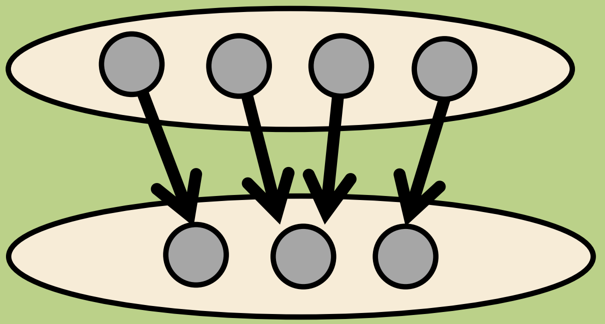


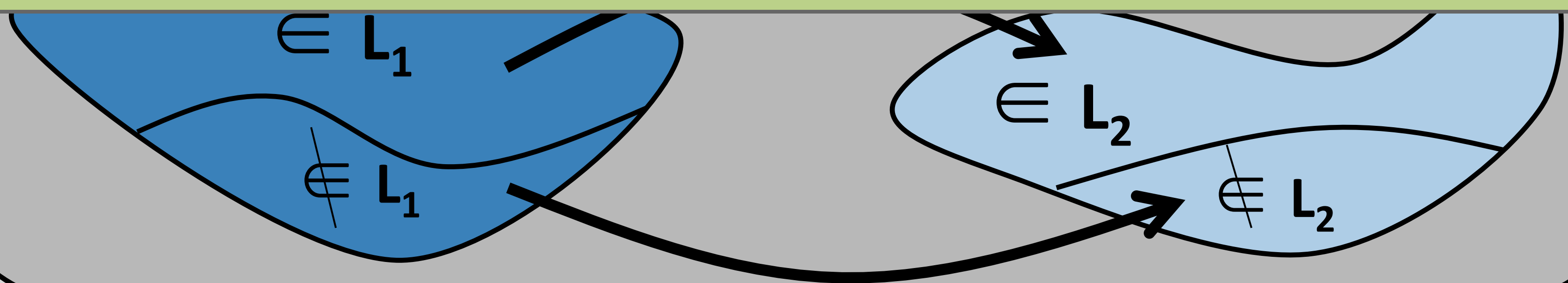
werden!

Definition: Many-One-Reduktion von L_1 auf L_2

D.h. Es gibt eine TM M_f, die f berechnet.

Surjektiv (“rechtstotal”): Jedes Element der Zielmenge wird mindestens einmal angenommen.





Implikationen der Reduktion

Theorem: Implikationen der Reduktion

Sei $L_1 \leq_m L_2$, dann gilt

1. $L_2 \in \mathbf{E} \Rightarrow L_1 \in \mathbf{E}$
2. $L_2 \in \mathbf{A} \Rightarrow L_1 \in \mathbf{A}$
3. $L_2 \in \mathbf{A}^{co} \Rightarrow L_1 \in \mathbf{A}^{co}$

Beweis?