Polynomielle Reduktion

TECHNISCHE UNIVERSITÄT DARMSTADT

$A \subseteq \Sigma^*$ heißt reduzierbar auf $B \subseteq \Sigma^*$, falls es eine berechenbare, totale Funktion $f:\Sigma^* \to \Sigma^*$ gibt so dass für alle $\underline{x} \in \Sigma^*$ gilt: $\underline{x} \in A \Leftrightarrow f(\underline{x}) \in B$.
Wir schreiben: $A \preceq B$ (mittels f). „$\preceq$“ ist transitiv.
Falls B (semi-) entscheidbar ist und $A \preceq B$ gilt, dann ist auch A (semi-) entscheidbar.

A heißt *polynomiell* reduzierbar auf B („$A \preceq_{\mathbf{p}} B$”), falls es eine in *polynomieller* Zeit berechenbare totale Funktion $f:\Sigma^* \to \Sigma^*$ gibt mit $\underline{x} \in A \Leftrightarrow f(\underline{x}) \in B \quad \forall \underline{x} \in \Sigma^*$.

Lemma: a) $A \preceq_{\mathbf{p}} B$ und $B \in \mathbf{P} \Rightarrow A \in \mathbf{P}$

b) $A \preceq_{\mathbf{p}} B$ und $B \preceq_{\mathbf{p}} C \Rightarrow A \preceq_{\mathbf{p}} C$ (*Transitivität*)

Beispiel: $\mathbf{CLIQUE} \preceq_{\mathbf{p}} \mathbf{IS} \preceq_{\mathbf{p}} \mathbf{CLIQUE}$, $f(\langle G,k\rangle) := \langle \overline{G},k\rangle$

Weiteres Beispiel: Boole‘sche Erfüllbarkeit

- Eine Boole’sche Variable x kann Werte 0 und 1 (**`falsch`** und **`wahr`**) annehmen.
- Eine Boole’sche Formel Φ ist eine Verknüpfung von Boole’schen Variablen durch Boole’sche Operatoren, z.B. **`AND`** ($\wedge$) , **`OR`** ($\vee$), **`NOT`** ($\neg$).

Beispiel: $\Phi = (\neg x \vee y) \wedge (x \vee \neg z)$

ist eine Boole’sche Formel mit Variablen x, y, z.

- Φ ist erfüllbar, falls es eine Belegung der Variablen mit Werten 0, 1 gibt, die Φ **`wahr`** macht.

Beispiel: Φ ist erfüllbar, z. b. durch $x:=1$, $y:=1$, $z:=0$.

Anwendung: Optimierung von Logik-Schaltkreisen

Konjunktive Normalform (KNF)

- Literal: Variable oder negierte Variable: y_i, $\neg y_i$
- Klausel: Disjunktion („**or**“) von Literalen:

  $C = x_1 \vee \ \ldots \vee x_m$, $\quad x_i$ Literale
- Formel Φ in Konjunktiver Normalform (KNF): Konjunktion („**and**“) von Klauseln:

  $\Phi = C_1 \wedge \ldots \wedge C_\ell$, $\quad C_i$ Klauseln
- **Value**:= { $\langle \Phi, \underline{x} \rangle$: Φ wird **wahr** bei Belegung $\underline{x}$ }
- **SAT** := { $\langle \Phi \rangle$: Φ erfüllbare Formel in KNF }

Beispiel: $\langle\ (\neg x \vee\ y) \wedge (x \vee\ \neg z\ )\ ,\ (1,1,0)\ \rangle \quad \in$ **Value**

$\langle\ (\neg x \vee\ y) \wedge (x \vee\ \neg z\ )\ ,\ (0,0,1)\ \rangle \quad \notin$ **Value**

$\langle\ (\neg x \vee\ y) \wedge (x \vee\ \neg z\ )\ \rangle \quad \in$ **SAT**

$\langle\ (\neg x \vee\ y) \wedge (x \vee\ \neg z\ ) \wedge (z \vee\ \neg y\ ) \wedge x \wedge (\neg y)\ \rangle \quad \notin$ **SAT**

Erfüllbarkeit (Satisfiability)

Konjunktive Normalform (KNF) =
Konjunktion ($\wedge$) von Disjunktion ($\vee$) von Literalen (x,$\neg x$)
Value = { $\langle \Phi, \underline{x} \rangle$: Formel Φ wird **wahr** bei Belegung $\underline{x}$ }
SAT = { $\langle \Phi \rangle$: KNF-Formel Φ ist erfüllbar }

- *k*-KNF Formel: Formel in KNF,
  in der jede Klausel aus $\leq k$ Literalen besteht.
- ***k*-SAT** := { $\langle \Phi \rangle$: Φ erfüllbare Formel in *k*-KNF }

Übung: 2-**SAT** $\in$ **P**

Fragen: **SAT** $\in$ **P** ? 3-**SAT** $\in$ **P** ?

Zweites Beispiel für Polynomielle Reduktion

a) $3\mathbf{SAT}$ ist polynomiell auf $\mathbf{SAT}$ reduzierbar:

$$3\mathbf{SAT} \leqslant_p \mathbf{SAT}$$

b) $\mathbf{SAT}$ ist polynomiell auf $3\mathbf{SAT}$ reduzierbar:

$$\mathbf{SAT} \leqslant_p 3\mathbf{SAT}$$

Folge: Beide Probleme sind im Wesentlichen '*gleich schwer*'; ein polynomialzeit-Algorithmus für das eine würde einen ebensolchen für das andere liefern:

$$\mathbf{SAT} \in \mathcal{P} \Leftrightarrow 3\mathbf{SAT} \in \mathcal{P}.$$

Die gleiche Situation wie bei $\mathbf{IS}$ und $\mathbf{CLIQUE}$…

a) 3SAT $\leqslant_p$ SAT b) SAT $\leqslant_p$ 3SAT

TECHNISCHE UNIVERSITÄT

Und was ist *f*(Eingaben, die keine 3-KNF Formel darstellen) ?

Beweis: *Was ist zu tun*?

a) Beschreibe eine in polynom… Funktion *f*, die aus einer 3-KNF … KNF-Formel Φ' macht so dass gilt:

Φ ist genau dann erfüllbar, wenn Φ' es ist. √

b) Gegeben KNF-Formel $\Phi = (a \vee b \vee c \vee d \vee e) \wedge \Phi_1$ mit Literalen *a,b,c,d,e.*

Betrachte neue Variablen *x*,*y* und Formel

$\Phi' = (a \vee b \vee x) \wedge (\neg x \vee c \vee y) \wedge (\neg y \vee d \vee e) \wedge \Phi'_1$

Klausel mit *k* Lit. → *k*-2 Klauseln a 3 Lit. & *k*-3 neue Var.

Φ' ist 3-KNF; und erfüllbar (d.h. in 3**SAT**)
genau dann, wenn Φ erfüllbar (d.h. in **SAT**) ist!

Reduktion $\mathrm{IS} \leqslant_p \mathrm{SAT}$

Zu tun: Bei Eingabe von Graph G und $k \in \mathbb{N}$,
berechne in polynomieller Zeit eine KNF Formel Φ, so dass
Φ erfüllbar ist g.d.w. es k unabhängige Knoten in G gibt.

G habe Knoten $V=\{1,..,n\}$ und Kanten E.

Betrachte Bool'sche Variablen $x_{v,i}$, $v \in V$, $i=1...k$

Knoten v ist i-ter der k unabhängigen.

Es gibt einen i-ten unabhängigen.

dazu Klauseln $K_i := \bigvee_{v \in V} x_{v,i}$, $i=1...k$

Knoten v kann nicht i-ter und j-ter sein.

und $\neg x_{v,i} \vee \neg x_{v,j}$, $v \in V$, $1 \le i < j \le k$

2 verbundene Knoten sind nicht unabhängig.

und $\neg x_{u,i} \vee \neg x_{v,j}$, $\{u,v\} \in E$, $1 \le i < j \le k$

Größe von Φ: $O(k \cdot n + n \cdot k^2 + n^2 k^2) = O(n^2 k^2)$

Berechnungsaufwand von $(G,k) \to \Phi$: polynomiell in $n+\log k$
da $k \le n$.

Reduktion $3SAT \leqslant_p IS$

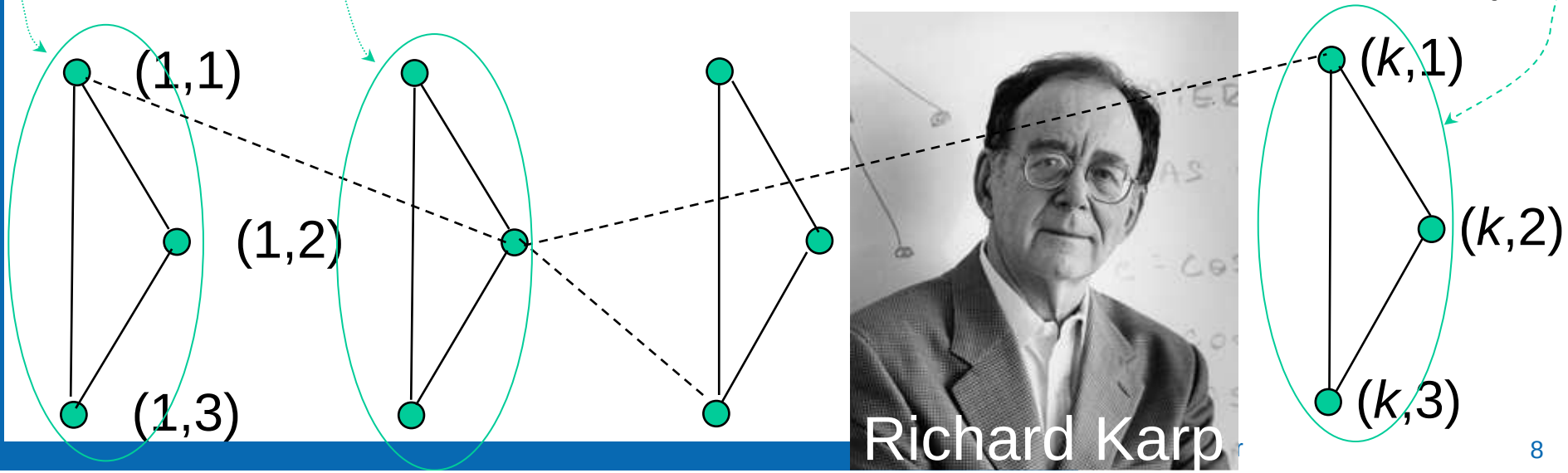
TECHNISCHE UNIVERSITÄT DARMSTADT

Berechne in polynomieller Zeit aus einer 3-KNF Formel Φ einen Graphen G und eine Zahl k, so dass gilt: Φ *ist genau dann erfüllbar, wenn es in* G k *unabhängige Knoten gibt.*

z.B. $(u \vee .. \vee ..) \wedge (.. \vee \neg u \vee ..) \wedge (.. \vee .. \vee u) \wedge (u \vee .. \vee ..)$

$\Phi = C_1 \wedge C_2 \ldots \wedge C_k$, $C_i = x_{i1} \vee x_{i2} \vee x_{i3}$, x_{is} Literale

$V := \{ (i,1), \ldots (i,3) : i \leq k \}$, $E := \{ \{(i,s),(j,t)\} : i=j$ oder $\overline{x}_{is} = x_{jt} \}$



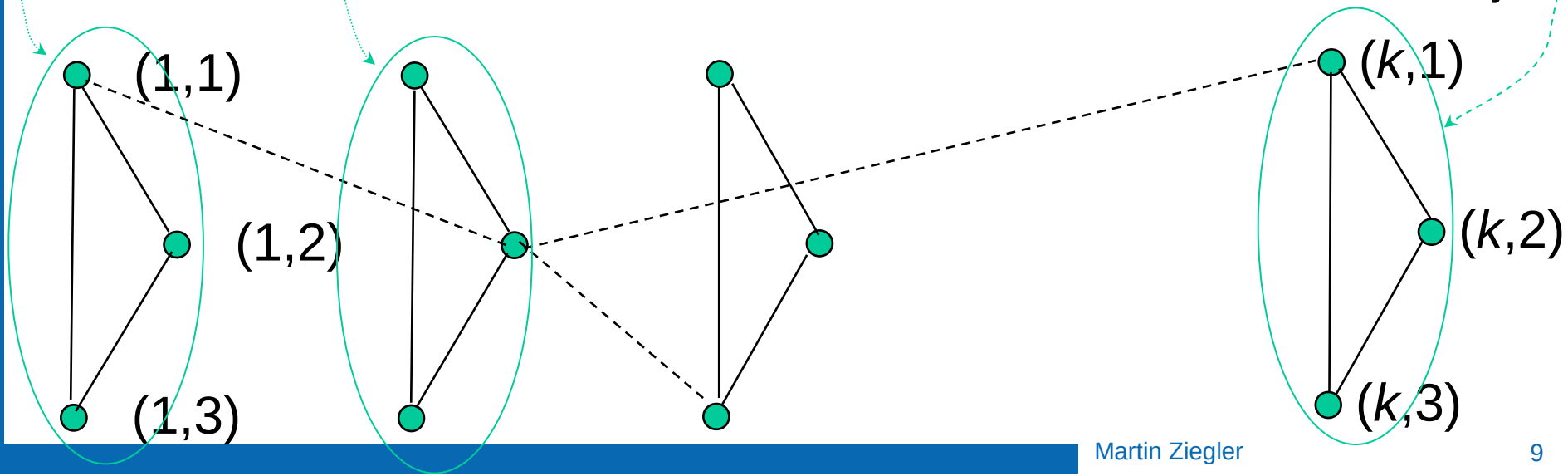
Reduktion $3SAT \leqslant_p IS$

„$\Rightarrow$“ Sei x erfüllende Belegung von Φ. Dann ist in jeder Klausel mindestens ein Literal **wahr**. Die zugehörigen Knoten in G sind nicht miteinander verbunden.

„$\Leftarrow$“ Seien $(v_1,..,v_k)$ unabhängig in G. Dann gehört zu jedem v_i ein Literal x_{is} in Φ. Diese haben alle gleiches Vorzeichen. Belege sie mit **wahr**: konsistent!

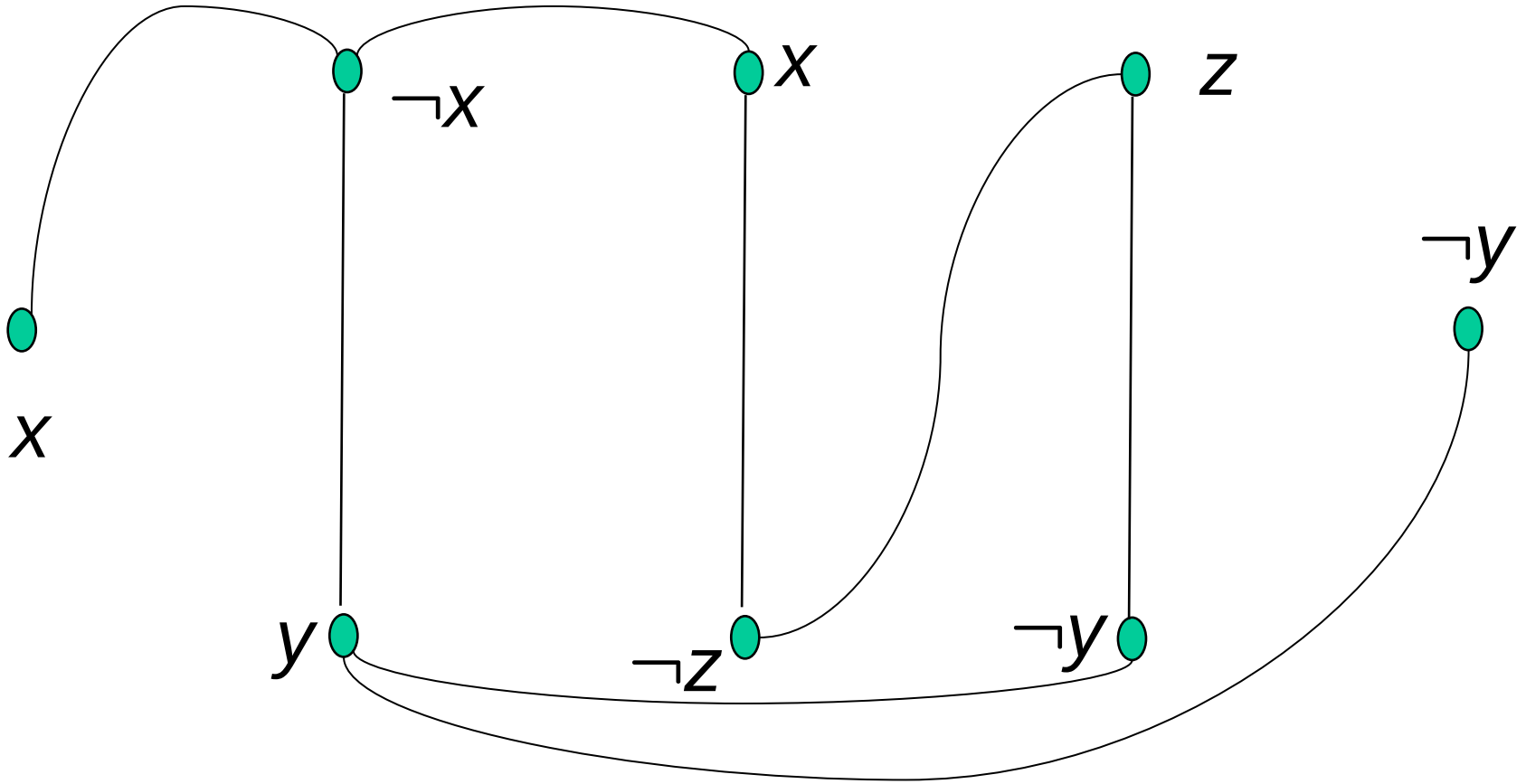
$\Phi = C_1 \wedge C_2 \ldots \wedge C_k$, $C_i = x_{i1} \vee x_{i2} \vee x_{i3}$, x_{is} Literale

$V := \{ (i,1),\ldots(i,3): i \leq k \}$, $E := \{ \{(i,s),(j,t)\} : i=j$ oder $\overline{x}_{is} = x_{jt} \}$



Beispiel-Instanz von 3SAT *reduziert auf* IS

$$x \wedge (\neg x \vee y) \wedge (x \vee \neg z) \wedge (z \vee \neg y) \wedge (\neg y)$$



Viele gleich schwere Probleme

Gezeigt: $\mathbf{CLIQUE} \equiv_p \mathbf{IS} \leqslant_p \mathbf{SAT} \equiv_p \mathbf{3SAT} \leqslant_p \mathbf{IS}$.

Diese 4 Probleme sind *alle* ungefähr gleich schwer:

Entweder alle sind in $\mathcal{P}$ oder keines.

Wir werden noch zeigen: auch **TSP**, **HC**, **VC** und viele weitere Probleme in $\mathcal{NP}$ gehören zu dieser Klasse, genannt $\mathcal{NP}$c.

Und wir werden zeigen: Dies sind die ‚schwersten' Probleme in $\mathcal{NP}$: Für jedes $L \in \mathcal{NP}$ gilt: $L \leqslant_p \mathbf{SAT}$. (*Satz von Cook*)

D.h. wenn a) irgendwer einen polynomialzeit-Algorithmus für irgendein Problem aus $\mathcal{NP}$c fände, so folgte $\mathcal{P}=\mathcal{NP}$:

Eine DTM könnte jede NTM in Polynomialzeit simulieren!

Und umgekehrt: Aus einem Beweis b), daß irgendeines dieser Probleme nicht in Polynomialzeit lösbar ist, folgt $\mathcal{P}\neq\mathcal{NP}$:

kein Problem in $\mathcal{NP}$c ließe sich in Polynomialzeit lösen.

die Klasse $\mathcal{NP}$

Def: Eine Sprache $L \subseteq \Sigma^*$ gehört zur Klasse $\mathcal{NP}$, wenn es $K \in \mathcal{P}$ und $p(N) \in \mathbb{N}[N]$ gibt mit:

$$L = \{ x : \exists\, y \in \Sigma^{\leq p(|x|)} : \langle x,y \rangle \in K \}$$

Beispiele:

- $\mathcal{P} \subseteq \mathcal{NP}$
- $\mathbf{SAT} \in \mathcal{NP}$
- $\mathbf{HC} \in \mathcal{NP}$
- $\mathbf{VC} \in \mathcal{NP}$
- $\mathbf{TSP} \in \mathcal{NP}$
- $\mathbf{IS} \in \mathcal{NP}$

Übung: $\mathcal{NP}$ via nichtdeterministische Turingmaschinen