

Separation oder Gleichheit von $\mathcal{P}$ und $\mathcal{NP}$

Satz

Sei L eine $\mathcal{NP}$ -vollständige Sprache und $L \in \mathcal{P}$. Dann gilt $\mathcal{P} = \mathcal{NP}$.

- Müssen für beliebiges $A \in \mathcal{NP}$ zeigen, dass $A \in \mathcal{P}$.
- Da $A \in \mathcal{NP}$ und L $\mathcal{NP}$ -vollständig ist, gibt es eine polynomiell berechenbare Funktion f mit $w \in A \Leftrightarrow f(w) \in L$ für alle $w \in A$.
- Sei M_f eine DTM, die die polynomielle Reduktion f berechnet.
- $L \in \mathcal{P} \Rightarrow \exists$ DTM M_L , die L in Polynomialzeit entscheidet.

Algorithmus DTM M_A für A

Eingabe: w

- 1 Berechne $f(w)$ mittels M_f .
 - 2 Falls M_L Eingabe $f(w)$ akzeptiert, akzeptiere. Sonst lehne ab.
- Laufzeit $T(M_A) = \mathcal{O}(T(M_f) + T(M_L))$, d.h. polynomiell in $|w|$.

$\mathcal{NP}$ Vollständigkeits-Beweise

Satz $\mathcal{NP}$ -Vollständigkeit mittels Reduktion

Seien A, L Sprachen. Sei L $\mathcal{NP}$ -vollständig, $A \in \mathcal{NP}$ und $L \leq_p A$.
Dann ist auch A $\mathcal{NP}$ -vollständig.

Beweis: Müssen zeigen, dass $B \leq_p A$ für alle $B \in \mathcal{NP}$.

- Da L $\mathcal{NP}$ -vollständig ist, gilt $B \leq_p L$ für beliebiges $B \in \mathcal{NP}$.
- Ferner gilt nach Voraussetzung $L \leq_p A$.
- Aus der Transitivität von $\leq_p$ folgt: $B \leq_p A$.
- Damit ist A ebenfalls $\mathcal{NP}$ -vollständig.

Problem: Wir benötigen ein *erstes* $\mathcal{NP}$ -vollständiges Problem.

Satz von Cook-Levin (1971)

Satz von Cook-Levin

SAT ist $\mathcal{NP}$ -vollständig.

Beweis: Müssen zeigen

- 1 SAT $\in \mathcal{NP}$ (bereits gezeigt)
- 2 Für alle $L \in \mathcal{NP}$ existiert polyn. berechenbare Reduktion f :

$$w \in L \Leftrightarrow f(w) \in \text{SAT}.$$

Beweisidee: Sei $L \in \mathcal{NP}$ beliebig.

- $\exists$ NTM N mit polynomieller Laufzeit n^k mit
$$w \in L \Leftrightarrow N \text{ akzeptiert } w.$$
- Konstruieren aus (N, w) eine Formel ϕ mit
 - 1 N akzeptiert $w \Leftrightarrow f(w) = \phi \in \text{SAT}$
 - 2 f ist in Zeit polynomiell in $|w| = n$ berechenbar.
- Betrachten dazu $(n^k + 1) \times (n^k + 1)$ Berechnungstabelle von N .

Berechnungstabelle T von N auf w

q_0	$\triangleright$	w_1	$\dots$	w_n	$\sqcup$	$\dots$	$\sqcup$
$\triangleright$	q_i	w_1	$\dots$	w_n	$\sqcup$	$\dots$	$\sqcup$
		$\vdots$				$\vdots$	

- Tabelle T entspricht einem Pfad im Berechnungsbaum.
- Erste Zeile enthält die Startkonfiguration.
- $(i + 1)$ -te Zeile ist mögliche Nachfolgekonfiguration der i -ten Zeile.
- In Laufzeit n^k können höchstens n^k Zellen besucht werden.
- T akzeptierend $\Leftrightarrow T$ enthält eine akzeptierende Konfiguration.
- Konstruieren ϕ derart, dass ϕ erfüllbar ist gdw. T akzeptierend ist.

Struktur der Formel für ϕ

- Sei $T(i, j)$ der Eintrag in der i -ten Zeile und j -ten Spalte von T .
- $T(i, j) \in Q \cup \Gamma$ für alle i, j .
- Definieren ϕ über den Booleschen Variablen $x_{i,j,s}$ mit

$$x_{i,j,s} = 1 \Leftrightarrow T(i, j) = s \quad \text{für } s \in Q \cup \Gamma.$$

Formel für ϕ : $\phi = \phi_{\text{Start}} \wedge \phi_{\text{accept}} \wedge \phi_{\text{Eintrag}} \wedge \phi_{\text{move}}$ mit

ϕ_{Start} : T beginnt mit Startkonfiguration.

ϕ_{accept} : T muss Eintrag q_a besitzen.

ϕ_{Eintrag} : T enthält Einträge aus $Q \cup \Gamma$.

ϕ_{move} : T besitzt gültige Nachfolgekongfigurationen.

Definition von ϕ_{Start} , ϕ_{accept} und $\phi_{Eintrag}$

ϕ_{Start} : Kodieren die Startkonfiguration $q_0 \triangleright w_1 \dots w_n$

$$x_{1,1,q_0} \wedge x_{1,2,\triangleright} \wedge x_{1,3,w_1} \wedge \dots \wedge x_{1,n+2,w_n} \wedge x_{1,n+3,\sqcup} \wedge \dots \wedge x_{1,n^k+1,\sqcup}$$

ϕ_{accept} : ϕ ist erfüllend gdw T eine erfüllende Konfiguration enthält

$$\phi_{accept} = \bigvee_{1 \leq i,j \leq n^k+1} x_{i,j,q_a}$$

$\phi_{Eintrag}$: $T(i,j) \in Q \cup \Gamma$, d.h. es gibt ein $s \in Q \cup \Gamma$ mit $x_{i,j,s} = 1$.

- $T(i,j)$ enthält mindestens einen Eintrag $s \in Q \cup \Gamma$:

$$\phi_{\geq 1} = \bigvee_{s \in Q \cup \Gamma} x_{i,j,s}.$$

- $T(i,j)$ enthält höchstens einen Eintrag $s \in Q \cup \Gamma$:

$$\phi_{\leq 1} = \bigwedge_{s,t \in Q \cup \Gamma, s \neq t} \neg(x_{i,j,s} \wedge x_{i,j,t}).$$

- Liefert insgesamt $\phi_{Eintrag} = \bigwedge_{1 \leq i,j \leq n^k+1} (\phi_{\geq 1} \wedge \phi_{\leq 1})$.

Definition von ϕ_{move}

Ziel: Zeile $i + 1$ muss Nachfolgekongfiguration von Zeile i sein.

- Definieren Fenster F der Größe 2×3 .
- (i, j) -Fenster besitzt Einträge $(i, j - 1), (i, j), (i, j + 1)$ und $(i + 1, j - 1), (i + 1, j), (i + 1, j + 1)$.
- Tabelle T besitzt (i, j) -Fenster für $i = 1, \dots, n^k, j = 2, \dots, n^k$.
- Fenster F heißt legal gwd F's Einträge δ nicht widersprechen.

Beispiele für legale Fenster

Sei δ wie folgt definiert

- $\delta(q_1, a) = \{(q_1, b, R)\}$.
- $\delta(q_1, b) = \{(q_2, c, L), (q_2, a, R)\}$.

a	q_1	b
q_2	a	c

legal

a	q_1	b
a	a	q_2

legal

a	b	b
a	a	b

nicht legal

a	a	q_1
a	a	b

legal

a	q_1	b
q_1	a	a

nicht legal

$\sqcup$	b	a
$\sqcup$	b	a

legal

a	q_1	b
q_2	b	q_2

nicht legal

a	b	a
a	b	q_2

legal

b	b	b
c	b	b

legal

Korrektheit der Konstruktion

Lemma Korrektheit Berechnungstabelle

Sei T eine Tabelle mit den folgenden Eigenschaften.

- 1 Die erste Zeile ist die Startkonfiguration von N auf w .
- 2 Jedes Fenster ist legal.

Dann ist T eine Berechnungstabelle von N auf Eingabe w .

- $T(i, j) \neq T(i + 1, j)$ ist nur dann möglich, falls einer der Einträge $T(i, j - 1)$, $T(i, j)$ oder $T(i, j + 1)$ einen Zustand enthält.
- Falls die obere Zeile einen Zustand ändert, muss sich die untere Zeile gemäß δ ändern.
- D.h. jede Zeile ist eine Nachfolgekonfiguration der Vorgängerzeile.
- Damit ist T eine Berechnungstabelle.

Konstruktion von ϕ_{move}

- Informal gilt: $\phi_{move} = \bigwedge_{1 \leq i \leq n^k, 2 \leq j \leq n^k}$ Fenster (i, j) ist legal.
- Die Anzahl legaler Fenster hängt nur von den möglichen Übergängen in N ab, nicht von der Eingabe w .
- D.h. es gibt eine Menge F von 6-Tupeln $(f_1, \dots, f_6)$, so dass F alle legalen Fenster beschreibt.
- Damit können wir das Prädikat [Fenster (i, j) ist legal] formalisieren

$$\bigvee_{(f_1, \dots, f_6) \in F} (x_{i,j-1,f_1} \wedge x_{i,j,f_2} \wedge x_{i,j+1,f_3} \wedge x_{i+1,j-1,f_4} \wedge x_{i+1,j,f_5} \wedge x_{i+1,j+1,f_6}).$$

Reduktion ist polynomiell

Lemma Länge von ϕ

Sei N eine NTM mit Laufzeit n^k bei Eingabe w , $|w| = n$. Dann besitzt die Formel $\phi = \phi_{\text{Start}} \wedge \phi_{\text{accept}} \wedge \phi_{\text{Eintrag}} \wedge \phi_{\text{move}}$ Länge $\mathcal{O}(n^{2k})$, d.h. Länge polynomiell in n .
Zudem ist ϕ bei Eingabe (N, w) in Zeit $\mathcal{O}(n^{2k})$ berechenbar.

- ϕ_{Start} :
 - Anzahl Literale: $\mathcal{O}(n^k)$, Berechnung direkt aus w
- ϕ_{accept} :
 - Anzahl Literale: $\mathcal{O}(n^{2k})$
- ϕ_{Eintrag} :
 - Anzahl Literale in $\phi_{\geq 1}, \phi_{\leq 1}$: $\mathcal{O}(1)$, unabhängig von w .
 - Anzahl Literale in ϕ_{Eintrag} : $\mathcal{O}(n^{2k})$.
- ϕ_{move} :
 - Anzahl legaler Fenster $|F|$: $\mathcal{O}(1)$, unabhängig von w .
 - Anzahl Literale in ϕ_{move} : $\mathcal{O}(n^{2k})$.

Von SAT zu 3SAT

Satz

3SAT ist $\mathcal{NP}$ -vollständig.

- Modifizieren zunächst vorigen Beweis derart, dass ϕ in KNF ist.
- ϕ_{start} und ϕ_{accept} sind bereits in KNF.
- $\phi_{Eintrag} = \bigwedge_{i,j} (\phi_{\geq 1} \wedge \phi_{\leq 1}) = \bigwedge_{i,j} \phi_{\geq 1} \wedge \bigwedge_{i,j} \phi_{\leq 1}$
 - ▶ $\phi_{\geq 1}$ besteht aus einer Klausel.
 - ▶ Schreiben $\phi_{\leq 1}$ als Konjunktion von Klauseln:

$$\phi_{\leq 1} = \bigwedge_{s \neq t} (\neg x_{i,j,s} \vee \neg x_{i,j,t}).$$

- ϕ_{move} : Wandle disjunktive Normalform des Prädikats für legale Fenster

$$\bigvee_{(f_1, \dots, f_6) \in F} (x_{i,j-1,f_1} \wedge x_{i,j,f_2} \wedge \dots \wedge x_{i+1,j+1,f_6}).$$

in KNF um. Umwandlung in $\mathcal{O}(1)$, da $|F|$ unabhängig von $|w| = n$.

Umwandlung von KNF in 3-KNF

Sei $\phi = k_1 \wedge \dots \wedge k_m$ eine KNF-Formel, wobei $k_i = a_1 \vee \dots \vee a_n$ eine Klausel mit $n > 3$ Literalen ist.

- Führen neue Variablen $z_1, \dots, z_{n-3}$ ein.
- Ersetzen Klausel k_i durch die 3-KNF Formel

$$(a_1 \vee a_2 \vee z_1) \wedge (\neg z_1 \vee a_3 \vee z_2) \wedge (\neg z_2 \vee a_4 \vee z_3) \wedge \dots \wedge (\neg z_{n-3} \vee a_{n_1} \vee a_n).$$

- B ist eine erfüllende Belegung für k gdw ein Literal a_i wahr ist.
- Dann ist aber k' erfüllbar mit $a_i = 1$ und

$$z_j = 1 \text{ für } j < i - 1 \quad \text{und} \quad z_j = 0 \text{ für } j \geq i - 1.$$

- Sei andererseits k' erfüllbar.
- Dann muss ein Literal a_i wahr sein, und damit ist k erfüllbar.
- Können ϕ in KNF bzw. in 3-KNF in $\mathcal{O}(|\phi|)$ Schritten umwandeln.

$\mathcal{NP}$ -Vollständigkeit von CLIQUE

Satz

CLIQUE ist $\mathcal{NP}$ -vollständig.

Beweis: zu zeigen

- 1 CLIQUE $\in \mathcal{NP}$
 - ▶ Übung
- 2 $\exists$ $\mathcal{NP}$ -vollständige Sprache L mit $L \leq_p$ CLIQUE
 - ▶ Bereits gezeigt: 3SAT ist $\mathcal{NP}$ -vollständig.
 - ▶ Bereits gezeigt: 3SAT $\leq_p$ CLIQUE.

Knotenüberdeckung

Definition k -Knotenüberdeckung

Sei $G = (V, E)$ ein ungerichteter Graph. Eine Knotenmenge $U \subseteq V$, $|U| = k$ heißt k -Knotenüberdeckung, falls

$$e \cap U \neq \emptyset \text{ für alle } e \in E.$$

Wir definieren die folgende Sprache.

KNOTENÜBERDECKUNG := $\{(G, k) \mid G \text{ besitzt eine } k\text{-Knotenüberdeckung.}\}$

Satz

KNOTENÜBERDECKUNG ist $\mathcal{NP}$ -vollständig.

Beweis: zu zeigen

- 1 KNOTENÜBERDECKUNG $\in \mathcal{NP}$ (Übung)
- 2 3-SAT $\leq_p$ KNOTENÜBERDECKUNG, d.h. es gibt berechenbares f :
$$\phi \in 3\text{SAT} \Leftrightarrow f(\phi) = (G, k) \in \text{KNOTENÜBERDECKUNG}$$

Die Reduktion f

Idee der Reduktion f :

- Konstruieren für jedes Literal x_i Knotenpaar mit Labeln x_i und $\neg x_i$.
- Knotenlabel einer Überdeckung bilden erfüllende Belegung.

Algorithmus M_f

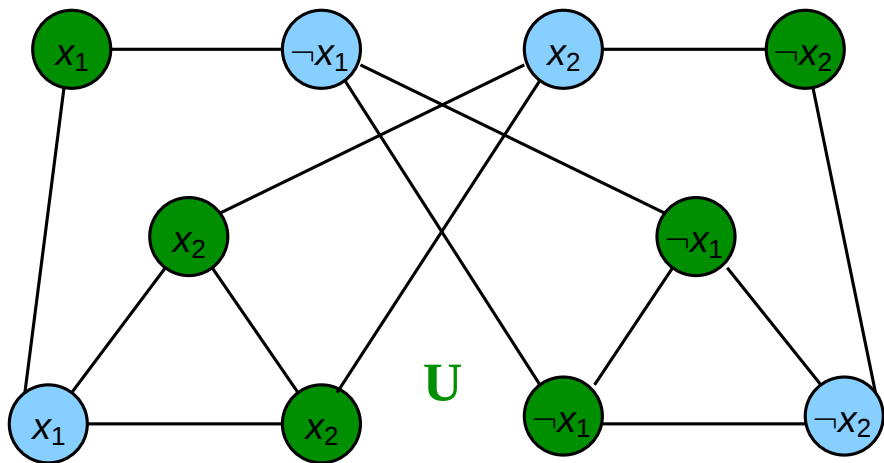
Eingabe: $\phi(x_1, \dots, x_n) = K_1 \wedge \dots \wedge K_m$ mit $K_j = \ell_{j1} \vee \ell_{j2} \vee \ell_{j3}$.

- 1 Variablenknoten: Für $i = 1 \dots n$:
 - ▶ Konstruiere zwei verbundene Knoten mit Labeln x_i und $\neg x_i$.
- 2 Klauselknoten: Für $j = 1 \dots m$:
 - ▶ Konstruiere 3 paarweise verbundene Knoten mit Labeln $\ell_{j1}, \ell_{j2}, \ell_{j3}$.
- 3 Verbinde Variablen- und Klauselknoten mit denselben Labeln.
- 4 Setze $k = n + 2m$.

Ausgabe: (G, k)

- Schritt 1: $\mathcal{O}(n)$, Schritt 2: $\mathcal{O}(m)$, Schritt 3: $\mathcal{O}(m)$, Schritt 4: $\mathcal{O}(1)$.
- $|\phi| = \mathcal{O}(n + m) = \mathcal{O}(m)$, d.h. die Laufzeit ist polynomiell in $|\phi|$.

Reduktion für $\phi = (x_1 \vee x_2 \vee x_2) \wedge (\neg x_1 \vee \neg x_1 \vee \neg x_2)$



$\phi \in 3SAT \Rightarrow f(\phi) \in \text{KNOTENÜBERDECKUNG}$

Sei $\phi(x_1, \dots, x_n) \in 3SAT$

- Dann gibt es eine erfüllende Belegung der Variablen $x_1, \dots, x_n$.
- In die Menge U werden die folgenden Knoten aufgenommen.
 - ▶ n Variablenknoten:
Falls $x_i = 1$, ist Knoten mit Label x_i in U . Sonst Knoten mit $\neg x_i$.
 - ▶ $2m$ Klauselknoten:
Für jede Klausel ist mindestens ein Knoten mit einem Variablenknoten aus U verbunden. Die *anderen beiden Knoten* sind in U .
- U ist eine $n + 2m$ -Knotenüberdeckung:
 - ▶ Die Kanten zwischen Variablenknoten $x_i, \neg x_i$ sind überdeckt durch einen Variablenknoten.
 - ▶ Kanten zwischen Klauselknoten $\ell_{j1}, \ell_{j2}, \ell_{j3}$ sind überdeckt durch zwei Klauselknoten.
 - ▶ Kanten zwischen Variablen- und Klauselknoten sind überdeckt:
Entweder der Variablenknoten überdeckt die Kante oder einer der beiden Klauselknoten.
- D.h. $f(\phi) = (G, n + 2m) \in \text{KNOTENÜBERDECKUNG}$

Korrektheit: Rückrichtung

Sei $f(\phi) = (G, n + 2m) \in \text{KNOTENÜBERDECKUNG}$:

- Dann gibt es eine $(n + 2m)$ -Knotenüberdeckung U mit:
 - ▶ Mindestens ein Variablenknoten x_i oder $\neg x_i$ ist in U für alle i .
 - ▶ Mindestens 2 von 3 Klauselknoten $\ell_{j1}, \ell_{j2}, \ell_{j3}$ sind in U für alle j .
 - ▶ Da $|U| = n + 2m$:
Jeweils *genau ein* Variablenknoten und *genau zwei* Klauselknoten.
- Sei B die Belegung, die die Variablenknoten aus U auf wahr setzt.
 - ▶ B ist eine konsistente Belegung.
 - ▶ Für alle Klauseln K_j mit Knoten $\ell_{j1}, \ell_{j2}, \ell_{j3}$ ist ein $\ell_{jk}, k \in [3]$ nicht in U .
 - ▶ Die Kante vom Klausel- zum Variablenknoten ℓ_{jk} wird überdeckt.
 - ▶ D.h. der Variablenknoten ℓ_{jk} ist in U . Damit erfüllt ℓ_{jk} die Klausel K_j .
- D.h. B ist eine erfüllende Belegung für ϕ .
- Damit gilt $\phi \in 3\text{SAT}$.