13. Woche: NP-Vollständigkeit
Satz von Cook-Levin
Anwendungen in der Kryptographie

$\mathcal{NP}$-Vollständigkeit

Definition $\mathcal{NP}$-vollständig

Sei L eine Sprache. Wir bezeichnen L als $\mathcal{NP}$-vollständig, falls

1. $L \in \mathcal{NP}$
2. Für **jede** Sprache $A \in \mathcal{NP}$ gilt: $A \leqslant_p L$.

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

Satz

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

Beweis:

- Wir zeigen für ein beliebiges $A \in \mathcal{NP}$, dass $A \in \mathcal{P}$.
- Da $A \in \mathcal{NP}$ und L $\mathcal{NP}$-vollständig ist, gilt $A \leqslant_p L$.
- Nach Voraussetzung gilt $L \in \mathcal{P}$.
- $\mathcal{P}$-Reduktionssatz: Aus $A \leqslant_p L$, $L \in \mathcal{P}$ folgt $A \in \mathcal{P}$.
- Da dies für ein beliebiges $A \in \mathcal{NP}$ gilt, folgt $\mathcal{NP} \subseteq \mathcal{P}$.
- Wegen $\mathcal{P} \subseteq \mathcal{NP}$ gilt schließlich $\mathcal{P} = \mathcal{NP}$.

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

Die $\mathcal{NP}$-vollständige Probleme bilden die Menge der schwierigsten Probleme in $\mathcal{NP}$.

Der Satz von Richard Ladner (1975)

Falls $\mathcal{P} \neq \mathcal{NP}$, dann existieren Probleme in $\mathcal{NP}$, die weder $\mathcal{NP}$-vollständig sind, noch in $\mathcal{P}$ liegen – sie bilden die Klasse $\mathcal{NPI}$ (Nicht-deterministisch Polynomiell, Intermediate).

Für den Beweis des Satzes wurde von Ladner ein künstliches Problem generiert, welches keinerlei praktische Relevanz besitzt.

Es ist nicht bekannt, ob auch „natürliche“ Probleme in $\mathcal{NPI}$ liegen. Es wird jedoch vermutet, dass dies z.B. für die Primfaktorzerlegung gilt.

Ladner, Richard: On the Structure of Polynomial Time Reducibility. Journal of the ACM (JACM) 22 (1): 155–171, 1975.

$\mathcal{NP}$ Vollständigkeits-Beweise

Satz $\mathcal{NP}$-Reduktionssatz

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

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

- Da L $\mathcal{NP}$-vollständig ist, gilt $A \leqslant_p L$ für beliebiges $A \in \mathcal{NP}$.
- Ferner gilt nach Voraussetzung $L \leqslant_p B$.
- Aus der Transitivität von $\leqslant_p$ folgt: $A \leqslant_p B$.
- Damit ist B 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 polynomiell 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 eine $(n^k + 1) \times (n^k + 1)$ Berechnungstabelle von N.

Berechnungstabelle T von N auf w

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

- Tabelle T entspricht einem Pfad im Berechnungsbaum.
- Erste Zeile enthält die Startkonfiguration.
- $(i+1)$-te Zeile ist *mögliche* Nachfolgekonfiguration der i-ten Zeile: wenn die NTM doch nicht-deterministisch ist, dann gibt es mehrere Möglichkeiten, die Tabelle zu belegen.
- 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. eine mögliche Belegung von 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 $x_{i,j,\sigma}$ mit

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

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

ϕ_{Start}: T beginnt mit Startkonfiguration.

ϕ_{accept}: T muss Eintrag q_a besitzen.

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

ϕ_{move}: T besitzt gültige Nachfolgekonfigurationen.

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

ϕ_{Start}: Codieren 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

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

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

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

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

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

$$\phi_{\leqslant 1} = \bigwedge_{\sigma,\tau \in Q \cup \Gamma, \sigma \neq \tau} \neg(x_{i,j,\sigma} \wedge x_{i,j,\tau}).$$

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

Definition von ϕ_{move}

Ziel: Zeile $i+1$ muss mögliche Nachfolgekonfiguration 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 $i = 1,\ldots,n^k$, $j = 2,\ldots,n^k$.
- Fenster F heißt legal gwd F's Einträge δ *nicht widersprechen*.

Beispiele für legale Fenster

Sei δ wie folgt definiert

$$\delta := \{((q_1, a), (q_1, b, R)), ((q_1, b), (q_2, c, L), (q_2, a, R)) \\ ((q_1, b), (q_2, c, L), (q_2, a, R))\} \\ \subseteq (Q \setminus \{q_a, q_r\} \times \Gamma) \times (Q \times \Gamma \times \{L, N, R\})$$

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 und entspricht somit einem Berechnungspfad von N.

- $T(i,j) \neq T(i+1,j)$ 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: $\phi_{move} = \bigwedge_{1 \leqslant i \leqslant n^k, 2 \leqslant j \leqslant n^k}$ 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 $(f_1, \ldots, f_6)$, so dass F alle legalen Fenster beschreibt.
- Damit können wir das Prädikt [Fenster (i, j) ist legal] formalisieren

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

Reduktion ist polynomiell

Lemma Länge von ϕ

Sei N eine NTM mit Laufzeit n^k bei Eingabe w, $|w| = n$. Dann besitzt die Formel $\phi = \phi_{Start} \wedge \phi_{accept} \wedge \phi_{Eintrag} \wedge \phi_{move}$ 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: $\mathcal{O}(n^{2k})$

$\phi_{Eintrag}$:
- Anzahl Literale in $\phi_{\geqslant 1}, \phi_{\leqslant 1}$: $\mathcal{O}(1)$, unabhängig von w.
- Anzahl Literale in $\phi_{Eintrag}$: $\mathcal{O}(n^{2k})$.

ϕ_{move}:
- Anzahl legaler Fenster $|F|$: $\mathcal{O}(1)$, unabhängig von w.
- Anzahl Literale in ϕ_{move}: $\mathcal{O}(n^{2k})$.

Fazit

Es ist klar, dass es eine $1-1$ Korrespondenz gibt zwischen korrekten (legalen) Berechnungstabellen von N (also, von Berechnungspfaden von N) und Belegungen der Variablen von ϕ, und eine akzeptierende Berechnungstabelle existiert gdw eine erfüllbare Belegung von ϕ existiert.

Die Konstruktion von T ist quadratisch in der Laufzeit von der NTM N, also polynomiell in der Eingabelänge n.

Somit ist der Satz von Cook-Levin bewiesen.

Kryptographische Anwendungen.

Diffie-Hellman Schlüsselaustausch (1976)

Öffentliche Parameter: Primzahl p, Generator g von $\mathbb{Z}_p^*$

Protokoll Diffie-Hellman Schlüsselaustausch

EINGABE: p, g

1. Alice wählt $\alpha \in_R \mathbb{Z}_{p-1}$ und schickt $g^\alpha \bmod p$ an Bob.
2. Bob wählt $\beta \in_R \mathbb{Z}_{p-1}$ und schickt $g^\beta \bmod p$ an Alice.
3. Alice berechnet $\left(g^\beta\right)^\alpha = g^{\alpha\beta}$, Bob analog $\left(g^\alpha\right)^\beta = g^{\alpha\beta}$.

Gemeinsamer geheimer DH-Schlüssel: $g^{\alpha\beta}$.

- Angreifer Eve erhält g, g^α, g^β.
- **Sicherheit:** Eve kann $g^{\alpha\beta}$ nicht von g^y, $y \in_R \mathbb{Z}_{p-1}$ unterscheiden.

Definition Decisional Diffie-Hellman (DDH)

Sei p prim, g Generator von $\mathbb{Z}_p^*$. Wir definieren die Sprache

$$\mathrm{DDH} := \{(g^\alpha, g^\beta, g^y) \mid g^y = g^{\alpha\beta}\}.$$

Das ElGamal Kryptosystem (1984)

Parameter des ElGamal Kryptosystems:

öffentlich: p prim, g Generator von $\mathbb{Z}_p^*$, g^a

geheim: $a \in \mathbb{Z}_{p-1}$

Algorithmus ElGamal Ver- und Entschlüsselung

- Verschlüsselung von $m \in \mathbb{Z}_p$ unter Verwendung von p, g, g^a.
  - Wähle $r \in_R \mathbb{Z}_{p-1}$.
  - Berechne $Enc(m) = (\gamma, \delta) = (g^r, m \cdot (g^a)^r) \in \mathbb{Z}_p^* \times \mathbb{Z}_p$.
- Entschlüsselung von $Enc(m)$ unter Verwendung von p, a.
  - Berechne $Dec(Enc(m)) = \frac{\delta}{\gamma^a} = \frac{m \cdot g^{ar}}{g^{ar}} = m$.

Laufzeit:

- Verschlüsselung: $\mathcal{O}(\log r \cdot \log^2 p) = \mathcal{O}(\log^3 p)$
- Entschlüsselung: $\mathcal{O}(\log a \cdot \log^2 p) = \mathcal{O}(\log^3 p)$

Sicherheit von ElGamal

Intuitiv: Eve soll $\delta = m \cdot g^{ab}$ nicht von $x \in_R \mathbb{Z}_p$ unterscheiden können.

Protokoll Unterscheider

EINGABE: p, g, g^a

1. Eve wählt $m \in \mathbb{Z}_p^*$ und schickt m an Alice. (Man beachte: $m \neq 0$.)
2. Alice wählt $b \in \{0, 1\}$:
   - Falls $b = 0$: Sende $Enc(m) = (g^r, m \cdot g^{ar})$ an Eve zurück.
   - Falls $b = 1$: Sende $(g^r, x) \in_R \mathbb{Z}_p^* \times \mathbb{Z}_p^*$ an Eve zurück.

Eves AUSGABE: $b' \in \{0, 1\}$

- Eve gewinnt das Spiel gdw $b' = b$.
- D.h. Eve muss δ von einer Zufallszahl x unterscheiden.

Definition Sprache ElGamal

Sei p prim und g ein Generator von $\mathbb{Z}_p^*$. Wir definieren

$$\text{ELGAMAL} := \{g^a, g^r, m, x \mid x = m \cdot g^{ar} \bmod p\}.$$

Sicherheitsbeweis per Reduktion

Satz Sicherheit von ElGamal unter DDH

Das ElGamal Kryptosystem ist sicher gegen polynomielle Angreifer unter der Annahme, dass DDH nicht effizient entscheidbar ist.

Logik des Beweises:

- Zeigen: DDH $\leqslant_p$ ELGAMAL
- D.h. jeder polynomielle Algorithmus für ELGAMAL liefert einen polynomiellen Algorithmus für DDH.
- **Annahme:** Es existiert ein polynomieller Angreifer A, der Verschlüsselungen von Zufallszahlen unterscheiden kann.
- Dann gibt es einen Algorithmus, der in polynomieller Zeit DH-Schlüssel $g^{\alpha\beta}$ von Zufallszahlen unterscheidet.
- **Widerspruch:** Nach Annahme gibt es keinen effizienten Algorithmus zum Entscheiden von DH-Schlüsseln $g^{\alpha\beta}$.
- Daher kann es auch keinen polynomiellen Angreifer A geben.

Reduktion f

Algorithmus M_f

EINGABE: $g^\alpha, g^\beta, g^y \in \mathbb{Z}_p^*$

1. Setze $g^a \leftarrow g^\alpha$ und $g^r \leftarrow g^\beta$.
2. Wähle $m \in_R \mathbb{Z}_p^*$.
3. Berechne $x = m \cdot g^y \bmod p$.

AUSGABE: g^a, g^r, m, x

Laufzeit:

- Eingabelänge: $\Omega(\log p)$
- Gesamtlaufzeit: $\mathcal{O}(\log^2(p))$

Korrektheit Reduktion: $w \in \text{DDH} \leqslant_p f(w) \in \text{ElGamal}$

Sei $(g^\alpha, g^\beta, g^\gamma) \in \text{DDH}$.

- Dann gilt $g^\gamma = g^{\alpha\beta} = g^{ar}$.
- Damit ist $x = m \cdot g^\gamma = m \cdot g^{ar}$ korrekte Verschlüsselung von m.
- D.h. $(g^a, g^r, m, x) \in \text{ElGamal}$

Sei $f(g^\alpha, g^\beta, g^\gamma) = (g^a, g^r, m, x) \in \text{ElGamal}$.

- Dann ist $x = m \cdot g^\gamma$ eine korrekte Verschlüsselung von m.
- D.h. $d(m) = \frac{m \cdot g^\gamma}{g^{ar}} = m$ und damit $g^\gamma = g^{ar} = g^{\alpha\beta}$.
- Dann ist $(g^\alpha, g^\beta, g^\gamma) \in \text{DDH}$.

Brechen von ElGamal ist nicht schwerer als DDH

Satz

ELGAMAL $\leq_p$ DDH

Beweis: Wir definieren die folgende Reduktion f.

Algorithmus M_f

EINGABE: $g^a, g^r, m, x \in \mathbb{Z}_p^*$

1. Setze $g^\alpha \leftarrow g^a$ und $g^\beta \leftarrow g^r$.
2. Berechne $g^y = \frac{x}{m}$.

AUSGABE: g^α, g^β, g^y

Laufzeit:

- Eingabelänge: $\Omega(\log p)$
- Laufzeit: $\mathcal{O}(\log^2 p)$

Korrektheit von f: $w \in$ ElGamal $\Leftrightarrow f(w) \in$ DDH

Sei $(g^a, g^r, m, x) \in$ ElGamal.

- Dann ist $x = m \cdot g^{ar}$ korrekte Verschlüsselung von m.
- Damit gilt $\frac{x}{m} = g^{ar} = g^{\alpha\beta} = g^y$.
- D.h. $(g^\alpha, g^\beta, g^y) \in$ DDH.

Sei $f(g^a, g^r, m, x) = (g^\alpha, g^\beta, g^y) \in$ DDH.

- Dann gilt $g^y = g^{\alpha\beta} = g^{ar}$.
- Damit folgt $x = m \cdot g^y = m \cdot g^{ar}$ ist Verschlüsselung von m.
- D.h. $(g^a, g^r, m, x) \in$ ElGamal.