FSK
Zentralübung 7

Jannis Limperg

Lehr- und Forschungseinheit für
Theoretische Informatik

Stand: 12. Juli 2023
Folien ursprünglich von PD Dr. David Sabel

Ein Verschlüsselungsproblem...

Eine **Verschlüsselungstabelle** ist eine Tabelle T

Wort	Kodewort
w_1	k_1
$\dots$	$\dots$
w_n	k_n

wobei $w_i, k_i \in \Sigma^+$.
Kodierung von ganzen Zeichenketten:
aus $w = w_{i_1} \cdots w_{i_m}$ wird $k_{i_1} \cdots k_{i_m}$.
Eine Verschlüsselungstabelle ist unsicher, wenn es ein Wort w gibt, dessen Kodierung wieder w ist.

Beispiel:

mit	für
der	das
frau	fru
mann	man
an	nanu
hund	katze
dem	den
ein	nein

aus derhundmitdemmann
wird daskatzefürdenman

Fragen:

- Ist das Beispiel unsicher?
- Ist es entscheidbar, ob eine Verschlüsselungstabelle unsicher ist?

Ein Verschlüsselungsproblem...

Eine **Verschlüsselungstabelle** ist eine Tabelle T

Wort	Kodewort
w_1	k_1
$\dots$	$\dots$
w_n	k_n

wobei $w_i, k_i \in \Sigma^+$.
Kodierung von ganzen Zeichenketten:
aus $w = w_{i_1} \cdots w_{i_m}$ wird $k_{i_1} \cdots k_{i_m}$.
Eine Verschlüsselungstabelle ist unsicher, wenn es ein Wort w gibt, dessen Kodierung wieder w ist.

Beispiel:

mit	für
der	das
frau	fru
mann	man
an	nanu
hund	katze
dem	den
ein	nein

aus derhundmitdemmann
wird daskatzefürdenman

Fragen:

- Ist das Beispiel unsicher? Ja: mannein wird zu mannein
- Ist es entscheidbar, ob eine Verschlüsselungstabelle unsicher ist?

Ein Verschlüsselungsproblem...

Eine **Verschlüsselungstabelle** ist eine Tabelle T

Wort	Kodewort
w_1	k_1
$\dots$	$\dots$
w_n	k_n

wobei $w_i, k_i \in \Sigma^+$.
Kodierung von ganzen Zeichenketten:
aus $w = w_{i_1} \cdots w_{i_m}$ wird $k_{i_1} \cdots k_{i_m}$.
Eine Verschlüsselungstabelle ist unsicher, wenn es ein Wort w gibt, dessen Kodierung wieder w ist.

Beispiel:

mit	für
der	das
frau	fru
mann	man
an	nanu
hund	katze
dem	den
ein	nein

aus derhundmitdemmann
wird daskatzefürdenman

Fragen:

- Ist das Beispiel unsicher? Ja: mannein wird zu mannein
- Ist es entscheidbar, ob eine Verschlüsselungstabelle unsicher ist?
  Nein: Das Problem ist äquivalent zu PCP

Wiederholung: Postsches Korrespondenzproblem

Definition (Postsches Korrespondenzproblem)

Gegeben sei ein Alphabet Σ und eine Folge von Wortpaaren $K = \{(x_1, y_1), \ldots, (x_k, y_k)\}$ mit $x_i, y_i \in \Sigma^+$. Das Postsche Korrespondenzproblem (PCP) ist die Frage, ob es für die gegebene Folge K eine Folge von Indizes $i_1, \ldots, i_m$ mit $i_j \in \{1, \ldots, k\}$ gibt, sodass $x_{i_1} \cdots x_{i_m} = y_{i_1} \cdots y_{i_m}$ gilt.

Beachte:

- PCP verwendet keine Turingmaschinen
- PCP ist unentscheidbar

PCP: Beispiel

$$K = \left[\begin{array}{c} b \\ ca \end{array}\right], \left[\begin{array}{c} a \\ ab \end{array}\right], \left[\begin{array}{c} ca \\ a \end{array}\right], \left[\begin{array}{c} abc \\ c \end{array}\right]$$

Eine Lösung?

PCP: Beispiel

$$K = \left[\begin{array}{c} b \\ ca \end{array}\right], \left[\begin{array}{c} a \\ ab \end{array}\right], \left[\begin{array}{c} ca \\ a \end{array}\right], \left[\begin{array}{c} abc \\ c \end{array}\right]$$

Eine Lösung?

- Lösung kann nur mit Stein 2 beginnen (gleiche Anfänge). $\left[\begin{array}{c} a \\ ab \end{array}\right]$

PCP: Beispiel

$$K = \left[\begin{array}{c} b \\ ca \end{array}\right], \left[\begin{array}{c} a \\ ab \end{array}\right], \left[\begin{array}{c} ca \\ a \end{array}\right], \left[\begin{array}{c} abc \\ c \end{array}\right]$$

Eine Lösung?

- Lösung kann nur mit Stein 2 beginnen (gleiche Anfänge). $\left[\begin{array}{c} a \\ ab \end{array}\right]$
- Danach muss oben ein b kommen: Nur Stein 1 möglich. $\left[\begin{array}{c} a \\ ab \end{array}\right], \left[\begin{array}{c} b \\ ca \end{array}\right]$

PCP: Beispiel

$$K = \left[\begin{array}{c} b \\ ca \end{array}\right], \left[\begin{array}{c} a \\ ab \end{array}\right], \left[\begin{array}{c} ca \\ a \end{array}\right], \left[\begin{array}{c} abc \\ c \end{array}\right]$$

Eine Lösung?

- Lösung kann nur mit Stein 2 beginnen (gleiche Anfänge). $\left[\begin{array}{c} a \\ ab \end{array}\right]$
- Danach muss oben ein b kommen: Nur Stein 1 möglich. $\left[\begin{array}{c} a \\ ab \end{array}\right], \left[\begin{array}{c} b \\ ca \end{array}\right]$
- Oben fehlt ca: Nur Stein 3 möglich. $\left[\begin{array}{c} a \\ ab \end{array}\right], \left[\begin{array}{c} b \\ ca \end{array}\right], \left[\begin{array}{c} ca \\ a \end{array}\right]$

PCP: Beispiel

$$K = \left[\begin{array}{c} b \\ ca \end{array}\right], \left[\begin{array}{c} a \\ ab \end{array}\right], \left[\begin{array}{c} ca \\ a \end{array}\right], \left[\begin{array}{c} abc \\ c \end{array}\right]$$

Eine Lösung?

- Lösung kann nur mit Stein 2 beginnen (gleiche Anfänge). $\left[\begin{array}{c} a \\ ab \end{array}\right]$
- Danach muss oben ein b kommen: Nur Stein 1 möglich. $\left[\begin{array}{c} a \\ ab \end{array}\right], \left[\begin{array}{c} b \\ ca \end{array}\right]$
- Oben fehlt ca: Nur Stein 3 möglich. $\left[\begin{array}{c} a \\ ab \end{array}\right], \left[\begin{array}{c} b \\ ca \end{array}\right], \left[\begin{array}{c} ca \\ a \end{array}\right]$
- Oben fehlt a: Nur Stein 2 oder Stein 4 möglich (wir nehmen 2). $\left[\begin{array}{c} a \\ ab \end{array}\right], \left[\begin{array}{c} b \\ ca \end{array}\right], \left[\begin{array}{c} ca \\ a \end{array}\right], \left[\begin{array}{c} a \\ ab \end{array}\right]$

PCP: Beispiel

$$K = \left[\begin{matrix} b \\ ca \end{matrix}\right], \left[\begin{matrix} a \\ ab \end{matrix}\right], \left[\begin{matrix} ca \\ a \end{matrix}\right], \left[\begin{matrix} abc \\ c \end{matrix}\right]$$

Eine Lösung?

- Lösung kann nur mit Stein 2 beginnen (gleiche Anfänge). $\left[\begin{matrix} a \\ ab \end{matrix}\right]$
- Danach muss oben ein b kommen: Nur Stein 1 möglich. $\left[\begin{matrix} a \\ ab \end{matrix}\right], \left[\begin{matrix} b \\ ca \end{matrix}\right]$
- Oben fehlt ca: Nur Stein 3 möglich. $\left[\begin{matrix} a \\ ab \end{matrix}\right], \left[\begin{matrix} b \\ ca \end{matrix}\right], \left[\begin{matrix} ca \\ a \end{matrix}\right]$
- Oben fehlt a: Nur Stein 2 oder Stein 4 möglich (wir nehmen 2). $\left[\begin{matrix} a \\ ab \end{matrix}\right], \left[\begin{matrix} b \\ ca \end{matrix}\right], \left[\begin{matrix} ca \\ a \end{matrix}\right], \left[\begin{matrix} a \\ ab \end{matrix}\right]$
- Oben fehlt ab: Nur Stein 2 oder Stein 4 möglich (wir nehmen 4). $\left[\begin{matrix} a \\ ab \end{matrix}\right], \left[\begin{matrix} b \\ ca \end{matrix}\right], \left[\begin{matrix} ca \\ a \end{matrix}\right], \left[\begin{matrix} a \\ ab \end{matrix}\right], \left[\begin{matrix} abc \\ c \end{matrix}\right]$

PCP: Beispiel

$$K = \left[\begin{array}{c} b \\ ca \end{array}\right], \left[\begin{array}{c} a \\ ab \end{array}\right], \left[\begin{array}{c} ca \\ a \end{array}\right], \left[\begin{array}{c} abc \\ c \end{array}\right]$$

Eine Lösung?

- Lösung kann nur mit Stein 2 beginnen (gleiche Anfänge). $\left[\begin{array}{c} a \\ ab \end{array}\right]$
- Danach muss oben ein b kommen: Nur Stein 1 möglich. $\left[\begin{array}{c} a \\ ab \end{array}\right], \left[\begin{array}{c} b \\ ca \end{array}\right]$
- Oben fehlt ca: Nur Stein 3 möglich. $\left[\begin{array}{c} a \\ ab \end{array}\right], \left[\begin{array}{c} b \\ ca \end{array}\right], \left[\begin{array}{c} ca \\ a \end{array}\right]$
- Oben fehlt a: Nur Stein 2 oder Stein 4 möglich (wir nehmen 2). $\left[\begin{array}{c} a \\ ab \end{array}\right], \left[\begin{array}{c} b \\ ca \end{array}\right], \left[\begin{array}{c} ca \\ a \end{array}\right], \left[\begin{array}{c} a \\ ab \end{array}\right]$
- Oben fehlt ab: Nur Stein 2 oder Stein 4 möglich (wir nehmen 4). $\left[\begin{array}{c} a \\ ab \end{array}\right], \left[\begin{array}{c} b \\ ca \end{array}\right], \left[\begin{array}{c} ca \\ a \end{array}\right], \left[\begin{array}{c} a \\ ab \end{array}\right], \left[\begin{array}{c} abc \\ c \end{array}\right]$
- Lösung: $2, 1, 3, 2, 4$

Wiederholung: Unentscheidbarkeit mittels Reduktion zeigen

Quiz: Seien $K \subseteq \Sigma^*$ und $L \subseteq \Gamma^*$ Sprachen, wobei K unentscheidbar ist.
Wie können wir zeigen, dass L unentscheidbar ist?

A Zeige, dass es eine totale und berechenbare Funktion f gibt mit $w \in L \iff f(w) \in K$

B Zeige, dass es eine totale und berechenbare Funktion f gibt mit $w \in K \iff f(w) \in L$

C Zeige $K \leq L$

D Zeige $L \leq K$

E Zeige $K \leq_p L$

F Zeige $L \leq_p K$

Wiederholung: Unentscheidbarkeit mittels Reduktion zeigen

Quiz: Seien $K \subseteq \Sigma^*$ und $L \subseteq \Gamma^*$ Sprachen, wobei K unentscheidbar ist.
Wie können wir zeigen, dass L unentscheidbar ist?

A Zeige, dass es eine totale und berechenbare Funktion f gibt mit $w \in L \iff f(w) \in K$ — nein

B Zeige, dass es eine totale und berechenbare Funktion f gibt mit $w \in K \iff f(w) \in L$ — ja

C Zeige $K \leq L$ — ja

D Zeige $L \leq K$ — nein

E Zeige $K \leq_p L$ — ja (unnötig kompliziert)

F Zeige $L \leq_p K$ — nein

Wiederholung: Postsches Korrespondenzproblem

Definition (Postsches Korrespondenzproblem)

Gegeben sei ein Alphabet Σ und eine Folge von Wortpaaren $K = \{(x_1, y_1), \ldots, (x_k, y_k)\}$ mit $x_i, y_i \in \Sigma^+$. Das Postsche Korrespondenzproblem (PCP) ist die Frage, ob es für die gegebene Folge K eine Folge von Indizes $i_1, \ldots, i_m$ mit $i_j \in \{1, \ldots, k\}$ gibt, sodass $x_{i_1} \cdots x_{i_m} = y_{i_1} \cdots y_{i_m}$ gilt.

01-PCP = Σ ist binär, z.B. $\Sigma = \{a, b\}$

Reduktion: 01-PCP auf Schnittproblem für CFGs

Aufgabe

Zeige: Das Schnittproblem für kontextfreie Grammatiken (also die Frage, ob $L(G_1) \cap L(G_2) = \emptyset$ für Grammatiken G_1, G_2 gilt) ist unentscheidbar.

Reduktion: 01-PCP auf Schnittproblem für CFGs

Aufgabe

Zeige: Das Schnittproblem für kontextfreie Grammatiken (also die Frage, ob $L(G_1) \cap L(G_2) = \emptyset$ für Grammatiken G_1, G_2 gilt) ist unentscheidbar.

Reduktionsfunktion

Berechenbare Funktion F, die K in $F(K) = (G_1, G_2)$ übersetzt, wobei K lösbar ist g.d.w. $L(G_1) \cap L(G_2) \neq \emptyset$.

„F: 01-PCP $\rightarrow (G_1, G_2)$“

$G_1 = (\{S, A, B\}, \Sigma \dot{\cup} \{\$, 1, \ldots, k\}, P_1, S)$ wobei P_1 sind:

$$
\begin{aligned}
S &\rightarrow A\$B \\
A &\rightarrow 1Ax_1 \mid \ldots \mid kAx_k \\
A &\rightarrow 1x_1 \mid \ldots \mid kx_k \\
B &\rightarrow \overline{y_1}B1 \mid \ldots \mid \overline{y_k}Bk \\
B &\rightarrow \overline{y_1}1 \mid \ldots \mid \overline{y_k}k
\end{aligned}
$$

wobei $\overline{y_i}$ das umgedrehte Wort y_i ist.

$L(G_1)$ enthält daher genau die Wörter

$$i_n \cdots i_1 x_{i_1} \cdots x_{i_n} \$ \overline{y_{j_m}} \cdots \overline{y_{j_1}} j_1 \cdots j_m,$$

wobei $n, m \geq 1$ und $i_r, j_s \in \{1, \ldots, k\}$.

„F: 01-PCP $\rightarrow (G_1, G_2)$“ (2)

Die Grammatik G_2 ist $G_2 = (\{S, T\}, \Sigma \dot{\cup} \{\$, 1, \ldots, k\}, P_2, S)$, sodass P_2 aus den folgenden Produktionen besteht:

$$S \rightarrow 1S1 \mid \ldots \mid kSk \mid T$$
$$T \rightarrow a_1 T a_1 \mid \ldots \; a_n T a_n \mid \$$$

$L(G_2)$ enthält daher genau die Wörter

$$i_1 \cdots i_n u \$ \overline{u} i_n \cdots i_1$$

mit $u \in \Sigma^*$, $i_j \in \{1, \ldots, k\}$, $n \geq 0$.

Wörter im Schnitt $L(G_1) \cap L(G_2)$

- $L(G_1)$ enthält genau die Wörter

$$i_n \cdots i_1 x_{i_1} \cdots x_{i_n} \$ \overline{y_{j_m}} \cdots \overline{y_{j_1}} j_1 \cdots j_m,$$

wobei $n, m \geq 1$ und $i_r, j_s \in \{1, \ldots, k\}$.

Wörter im Schnitt $L(G_1) \cap L(G_2)$

- $L(G_1)$ enthält genau die Wörter

$$i_n \cdots i_1 x_{i_1} \cdots x_{i_n} \$ \overline{y_{j_m}} \cdots \overline{y_{j_1}} j_1 \cdots j_m,$$

  wobei $n, m \geq 1$ und $i_r, j_s \in \{1, \ldots, k\}$.
- $L(G_2)$ enthält genau die Wörter

$$i_1 \cdots i_n u \$ \overline{u} i_n \cdots i_1$$

  mit $u \in \Sigma^*$, $i_j \in \{1, \ldots, k\}$, $n \geq 0$.

Wörter im Schnitt $L(G_1) \cap L(G_2)$

- $L(G_1)$ enthält genau die Wörter

$$i_n \cdots i_1 x_{i_1} \cdots x_{i_n} \$ \overline{y_{j_m}} \cdots \overline{y_{j_1}} j_1 \cdots j_m,$$

  wobei $n, m \geq 1$ und $i_r, j_s \in \{1, \ldots, k\}$.
- $L(G_2)$ enthält genau die Wörter

$$i_1 \cdots i_n u \$ \overline{u} i_n \cdots i_1$$

  mit $u \in \Sigma^*$, $i_j \in \{1, \ldots, k\}$, $n \geq 0$.
- Wörter in $L(G_1) \cap L(G_2)$:

$$i_n \cdots i_1 x_{i_1} \cdots x_{i_n} \$ \overline{y_{i_n}} \cdots \overline{y_{i_1}} i_1 \cdots i_n$$

  mit $n \geq 1$ und $x_{i_1} \cdots x_{i_n} = y_{i_1} \cdots y_{i_n}$.

Wörter im Schnitt $L(G_1) \cap L(G_2)$

- $L(G_1)$ enthält genau die Wörter

$$i_n \cdots i_1 x_{i_1} \cdots x_{i_n} \$ \overline{y_{j_m}} \cdots \overline{y_{j_1}} j_1 \cdots j_m,$$

  wobei $n, m \geq 1$ und $i_r, j_s \in \{1, \ldots, k\}$.
- $L(G_2)$ enthält genau die Wörter

$$i_1 \cdots i_n u \$ \overline{u} i_n \cdots i_1$$

  mit $u \in \Sigma^*$, $i_j \in \{1, \ldots, k\}$, $n \geq 0$.
- Wörter in $L(G_1) \cap L(G_2)$:

$$i_n \cdots i_1 x_{i_1} \cdots x_{i_n} \$ \overline{y_{i_n}} \cdots \overline{y_{i_1}} i_1 \cdots i_n$$

  mit $n \geq 1$ und $x_{i_1} \cdots x_{i_n} = y_{i_1} \cdots y_{i_n}$.
- Daher $i_1, \ldots, i_n$ ist Lösung von K g.d.w. Wort im Schnitt.

Schnittproblem für CFGs

Aufgabe

Zeige: Das Schnittproblem für kontextfreie Grammatiken (also die Frage, ob $L(G_1) \cap L(G_2) = \emptyset$ für Grammatiken G_1, G_2 gilt) ist unentscheidbar.

Schnittproblem für CFGs

Aufgabe

Zeige: Das Schnittproblem für kontextfreie Grammatiken (also die Frage, ob $L(G_1) \cap L(G_2) = \emptyset$ für Grammatiken G_1, G_2 gilt) ist unentscheidbar.

Beweis: Obige Funktion F zeigt, dass 01-PCP auf das Schnittproblem reduzierbar ist. Die Unentscheidbarkeit von 01-PCP impliziert daher die Unentscheidbarkeit des Schnittproblems.

Weitere Entscheidbarkeitsresultate

Folgende Probleme sind alle unentscheidbar
(die Beweise dazu können im Skript gefunden werden):

- Gilt für kontextfreie Grammatiken G_1, G_2: $|L(G_1) \cap L(G_2)| = \infty$?
- Ist der Schnitt zweier kontextfreier Sprachen kontextfrei?
- Gilt für kontextfreie Grammatiken G_1, G_2: $L(G_1) \subseteq L(G_2)$?
- Gilt für kontextfreie Grammatiken G_1, G_2: $L(G_1) = L(G_2)$?
- Ist eine kontextfreie Grammatik G mehrdeutig?
- Gilt für CFG G, die Sprache $L(G)$ ist regulär?

$\mathcal{P}/\mathcal{NP}$: Wiederholung

Quiz: Für was steht das P der Komplexitätsklasse $\mathcal{P}$?

A Entscheidbare **P**robleme

B **P**olynomielle Zeit

C **P**olynomieller Platz

$\mathcal{P}/\mathcal{NP}$: Wiederholung

Quiz: Für was steht das P der Komplexitätsklasse $\mathcal{P}$?

A Entscheidbare **P**robleme — nein

B **P**olynomielle Zeit — ja

C **P**olynomieller Platz — nein, das wäre die Klasse PSPACE

$\mathcal{P}/\mathcal{NP}$: Wiederholung

Quiz: Für was steht das P der Komplexitätsklasse $\mathcal{P}$?

A Entscheidbare **P**robleme — nein

B **P**olynomielle Zeit — ja

C **P**olynomieller Platz — nein, das wäre die Klasse PSPACE

Quiz: Für was steht das NP der Komplexitätsklasse $\mathcal{NP}$?

A **N**icht in **p**olynomieller Zeit lösbar

B **N**ur **p**olynomielle Laufzeit

C **N**ichtdeterministisch **p**olynomielle Laufzeit

D **N**icht **p**olynomieller Platzbedarf

E **N**ichtdeterministisch **p**olynomieller Platzbedarf

$\mathcal{P}/\mathcal{NP}$: Wiederholung

Quiz: Für was steht das P der Komplexitätsklasse $\mathcal{P}$?

A	Entscheidbare **P**robleme	nein
B	**P**olynomielle Zeit	ja
C	**P**olynomieller Platz	nein, das wäre die Klasse PSPACE

Quiz: Für was steht das NP der Komplexitätsklasse $\mathcal{NP}$?

A	**N**icht in **p**olynomieller Zeit lösbar	nein
B	**N**ur **p**olynomielle Laufzeit	nein
C	**N**ichtdeterministisch **p**olynomielle Laufzeit	ja
D	**N**icht **p**olynomieller Platzbedarf	nein
E	**N**ichtdeterministisch **p**olynomieller Platzbedarf	nein, das wäre NPSPACE

$\mathcal{NP}$-Vollständigkeit: Wiederholung

Eine Sprache L heißt $\mathcal{NP}$-vollständig, wenn gilt

1. $L \in \mathcal{NP}$ und
2. L ist $\mathcal{NP}$-schwer (manchmal auch $\mathcal{NP}$-hart genannt):
   Für alle $L' \in \mathcal{NP}$ gilt $L' \leq_p L$

SAT

$$\text{SAT} := \{code(F) \mid \text{aussagenlogische Formel } F \text{ ist erfüllbar}\}$$

$$\text{3-CNF-SAT} := \{code(F) \mid \text{3-CNF } F \text{ ist erfüllbar}\}$$

In der Vorlesung gezeigt:

- SAT ist $\mathcal{NP}$-vollständig (Satz von Cook)
- 3-CNF-SAT ist $\mathcal{NP}$-vollständig
  ($\mathcal{NP}$-Schwere durch Polynomialzeitreduktion SAT $\leq_p$ 3-CNF-SAT)

1-in-3-SAT

Heute: Sei

$$\text{1-in-3-SAT} := \left\{ code(F) \;\middle|\; \begin{array}{l} \text{3-CNF } F \text{ ist erfüllbar mit Belegung,} \\ \text{die in jeder Klausel genau ein Literal wahr macht} \end{array} \right\}$$

Zeige: 1-in-3-SAT ist $\mathcal{NP}$-vollständig

$\mathcal{NP}$-Vollständigkeit von 1-in-3-SAT

1-in-3-SAT ist in $\mathcal{NP}$:

- Sei M ein NTM, die erst prüft, ob Eingabe $code(F)$, mit F eine 3-CNF, ist (wenn nein: verwirf).
- Berechne alle Variablen, die in F vorkommen.
- Berechne nichtdeterministisch eine der möglichen Variablenbelegungen.
- Verifiziere in Polynomialzeit, ob die Belegung in jeder Klausel genau 1 Literal wahr macht.
- Wenn ja, akzeptiere, wenn nein, verwirf.

1-in-3-SAT ist $\mathcal{NP}$-schwer: Quiz ...

Quiz

Wir möchten die $\mathcal{NP}$-Schwere von 1-in-3-SAT mithilfe der bekannten $\mathcal{NP}$-Vollständigkeit von 3-CNF-SAT zeigen. Was ist zu tun?

A Reduziere 3-CNF-SAT auf 1-in-3-SAT.

B Reduziere 1-in-3-SAT auf 3-CNF-SAT.

C Reduziere 3-CNF-SAT auf 1-in-3-SAT in Polynomialzeit.

D Reduziere 1-in-3-SAT auf 3-CNF-SAT in Polynomialzeit.

Quiz

Wir möchten die $\mathcal{NP}$-Schwere von 1-in-3-SAT mithilfe der bekannten $\mathcal{NP}$-Vollständigkeit von 3-CNF-SAT zeigen. Was ist zu tun?

A	Reduziere 3-CNF-SAT auf 1-in-3-SAT.	nein
B	Reduziere 1-in-3-SAT auf 3-CNF-SAT.	nein
C	Reduziere 3-CNF-SAT auf 1-in-3-SAT in Polynomialzeit.	ja
D	Reduziere 1-in-3-SAT auf 3-CNF-SAT in Polynomialzeit.	nein

Quiz (2)

Was ist für eine Polynomialzeit-Reduktion von 3-CNF-SAT auf 1-in-3-SAT zu tun?

A Gib totales, berechenbares f an, sodass $w \in$ 3-CNF-SAT g.d.w. $f(w) \in$ 1-in-3-SAT.

B Gib totales, berechenbares f an, sodass $w \in$ 1-in-3-SAT g.d.w. $f(w) \in$ 3-CNF-SAT.

C Gib totales, in nichtdeterministischer Polynomialzeit berechenbares f an, sodass $w \in$ 3-CNF-SAT g.d.w. $f(w) \in$ 1-in-3-SAT.

D Gib totales, in nichtdeterministischer Polynomialzeit berechenbares f an, sodass $w \in$ 1-in-3-SAT g.d.w. $f(w) \in$ 3-CNF-SAT.

E Gib totales, in deterministischer Polynomialzeit berechenbares f an, sodass $w \in$ 3-CNF-SAT g.d.w. $f(w) \in$ 1-in-3-SAT.

F Gib totales, in deterministischer Polynomialzeit berechenbares f an, sodass $w \in$ 1-in-3-SAT g.d.w. $f(w) \in$ 3-CNF-SAT.

Quiz (2)

Was ist für eine Polynomialzeit-Reduktion von 3-CNF-SAT auf 1-in-3-SAT zu tun?

A Gib totales, berechenbares f an, sodass $w \in$ 3-CNF-SAT g.d.w. $f(w) \in$ 1-in-3-SAT. nein

B Gib totales, berechenbares f an, sodass $w \in$ 1-in-3-SAT g.d.w. $f(w) \in$ 3-CNF-SAT. nein

C Gib totales, in nichtdeterministischer Polynomialzeit berechenbares f an, sodass $w \in$ 3-CNF-SAT g.d.w. $f(w) \in$ 1-in-3-SAT. nein

D Gib totales, in nichtdeterministischer Polynomialzeit berechenbares f an, sodass $w \in$ 1-in-3-SAT g.d.w. $f(w) \in$ 3-CNF-SAT. nein

E Gib totales, in deterministischer Polynomialzeit berechenbares f an, sodass $w \in$ 3-CNF-SAT g.d.w. $f(w) \in$ 1-in-3-SAT. ja

F Gib totales, in deterministischer Polynomialzeit berechenbares f an, sodass $w \in$ 1-in-3-SAT g.d.w. $f(w) \in$ 3-CNF-SAT. nein

Polynomialzeitreduktion 3-CNF-SAT $\leq_p$ 1-in-3-SAT

Reduktionsfunktion f

- Wenn w kein $code(F)$ ist, dann sei $f(w) = w$
  (und damit ist $f(w)$ auch kein $code(F)$).
- Anderenfalls sei $F = \bigwedge C_i$ eine 3-CNF und f erstellt daraus eine 3-CNF, wobei C_i ersetzt wird durch $C_{i,1} \wedge C_{i,2} \wedge C_{i,3}$:
  Für Klausel $C_i := \{L_{i,1}, L_{i,2}, L_{i,3}\}$ erzeuge Klauseln

$$C_{i,1} := \{\overline{L_{i,1}}, a_i, b_i\}, C_{i,2} := \{L_{i,2}, b_i, c_i\}, C_{i,3} := \{\overline{L_{i,3}}, c_i, d_i\}$$

f ist total und in (deterministischer) Polynomialzeit berechenbar.

Polynomialzeitreduktion 3-CNF-SAT $\leq_p$ 1-in-3-SAT (2)

Erinnerung: $C_i := \{L_{i,1}, L_{i,2}, L_{i,3}\} \quad \overset{f}{\to} \quad C_{i,1} := \{\overline{L_{i,1}}, a_i, b_i\}, C_{i,2} := \{L_{i,2}, b_i, c_i\}, C_{i,3} := \{\overline{L_{i,3}}, c_i, d_i\}$

f ist korrekt ($w \in$ 3-CNF-SAT g.d.w. $f(w) \in$ 1-in-3-SAT)

- $w \notin$ 3-CNF-SAT $\implies f(w) \notin$ 1-in-3-SAT:
  Wenn $w \neq code(F)$, dann klar.

  Wenn $w = code(F)$, dann ist F unerfüllbar.

  Zeige: Dann gibt es keine Belegung B, die jeweils genau ein Literal in $C_{i,1}, C_{i_2}, C_{i_3}$ (für alle i) wahr macht:

  - Da F unerfüllbar macht jede Belegung $L_{i,1}$, $L_{i,2}$ und $L_{i,3}$ falsch.
  - Für jede Belegung versuchen wir nun B für $f(F)$ zu finden: Betrachte $C_{i,2}$: Da $L_{i,2} = 0$ muss B entweder b_i oder c_i wahr machen. Dann macht B aber mindestens 2 Literale wahr in $C_{i,1}$ oder in $C_{i,2}$.

Polynomialzeitreduktion 3-CNF-SAT $\leq_p$ 1-in-3-SAT (3)

Erinnerung: $C_i := \{L_{i,1}, L_{i,2}, L_{i,3}\} \quad \xrightarrow{f} \quad C_{i,1} := \{\overline{L_{i,1}}, a_i, b_i\}, C_{i,2} := \{L_{i,2}, b_i, c_i\}, C_{i,3} := \{\overline{L_{i,3}}, c_i, d_i\}$

- $w \in$ 3-CNF-SAT $\implies f(w) \in$ 1-in-3-SAT:
  - Sei $w = code(F)$, F erfüllbar und I Belegung mit $I(F) = 1$.
  - Wir zeigen: Es gibt Erweiterung B von I (d.h. $B(L) = I(L)$ für alle $L_{i,j}$), die jeweils genau ein Literal in $C_{i,1}, C_{i_2}, C_{i_3}$ wahr macht.
  - Fall $I(L_{i,2}) = 1$: Setze $B(b_i) = 0, B(c_i) = 0$, $B(a_i) = I(L_{i,1})$ und $B(d_i) = I(L_{i,3})$.
  - Fall $I(L_{i,2}) = 0$, $I(L_{i,1}) = 1$:
    Setze $B(a_i) = 0$, $B(b_i) = 1$, $B(c_i) = 0$, $B(d_i) = I(L_{i,3})$.
  - Fall $I(L_{i,2}) = 0$, $I(L_{i,1}) = 0$, $I(L_{i,3}) = 1$:
    Setze $B(a_i) = 0$, $B(b_i) = 0, B(c_i) = 1$, $B(d_i) = 0$.
- Damit gilt 3-CNF-SAT $\leq_p$ 1-in-3-SAT.
- Da 3-CNF-SAT $\mathcal{NP}$-schwer, ist auch 1-in-3-SAT $\mathcal{NP}$-schwer.