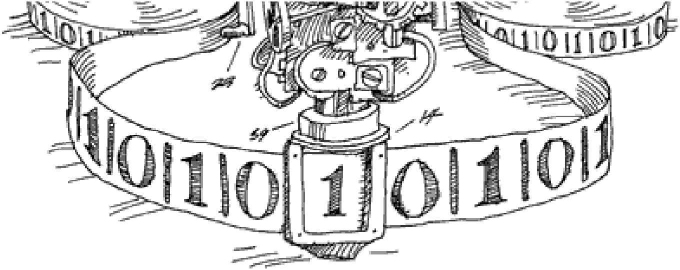
Theoretische Grundlagen der Informatik

Tutorium 8

Institut für Kryptographie und Sicherheit



Reduktion

Karlsruher Institut für Technologie

Aufgabe: Ist ein gegebenes *Problem* A *attribut*?

- Nehme an, A ist *attribut*
- Suche ein geeignetes *Problem* B, das bekanntermaßen (laut Vorlesung) *nicht attribut* ist
- Zeige: Wenn A *attribut* ist, dann wäre B auch *attribut*
- Transformiere **alle** Instanzen von B zu Instanzen von A, wobei diese Transformation *attribut* **nicht verändern** darf.
- Widerspruch!

Ist die Sprache $L = \{\langle M \rangle \mid \text{TM M hat mind. einen nicht erreichbaren Zustand}\}$ entscheidbar?

- Annahme: L entscheidbar ($\Leftrightarrow \overline{L}$ entscheidbar)
- Bekannt: Das Halteproblem ist nicht entscheidbar
- Transformation f von (allen) Instanzen $\in$ Halt zu Instanzen von $\overline{L}$
  $f : (\langle M \rangle, w) \rightarrow \langle M' \rangle$
- Konstruiere M': M' hat folgende Funktionsweise:
  1. Leere das Band
  2. Schreibe w auf das Band
  3. Simuliere M
  4. Gehe in einen zusätzlichen Zustand q_s
- Folgerung:
  - $\langle M' \rangle = f((\langle M \rangle, w)) \in \overline{L}$
  - $\Leftrightarrow$ M' hat keinen nicht erreichbaren Zustand
  - $\Leftrightarrow$ M' geht in Zustand q_s
  - $\Leftrightarrow$ M hält bei Eingabe w
  - $\Leftrightarrow$ $(\langle M \rangle, w) \in HALT$
- Also: L entscheidbar $\Rightarrow \overline{L}$ entscheidbar $\Rightarrow$ HALT entscheidbar $\lightning$

SAT (SATisfiability = Erfüllbarkeitsproblem)

Problem

Gegeben: Formel in konjunktiver Normalform

- Menge U von Variablen
- Menge C von Klauseln (Disjunktionen) über U

$$\underbrace{(\overset{\text{Literale}}{\overset{\downarrow}{a}} \vee \overset{\downarrow}{b} \vee \overset{\downarrow}{\bar{c}})}_{\text{Klausel}} \wedge \underbrace{(b \vee c)}_{\text{Klausel}} \wedge \underbrace{(\bar{a} \vee \bar{b} \vee \bar{c})}_{\text{Klausel}}$$

Frage: Existiert eine (alle Klauseln) erfüllende Variablenbelegung?

$\mathcal{P}, \mathcal{NP}$

$\mathcal{P}$ ist die Klasse aller Sprachen, die von einer deterministischen Turingmaschine in Polynomialzeit erkannt werden.

$\mathcal{NP}$ ist die Klasse aller Sprachen, die von einer **nicht**deterministischen Turingmaschine in Polynomialzeit erkannt werden.

Anmerkungen:

- $\mathcal{P} \subseteq \mathcal{NP}$.
- Die Frage ob $\mathcal{P} = \mathcal{NP}$ gilt ist ein großes, ungeklärtes Problem.

Gedanklicher Trick für $\mathcal{NP}$

Ein Problem liegt in $\mathcal{NP}$ falls man eine mögliche Lösung in Polynomialzeit von einer **deterministischen** Turingmaschine verifizieren lassen kann.

Beispiel

Zeige: $SAT \in \mathcal{NP}$

1. Es werden nichtdeterministisch alle möglichen Variablenbelegungen aufs Band geschrieben.
2. Es gibt nun eine deterministische Turingmaschine welche die Variablenbelegung in Polynomialzeit überprüft.

Aufgabe zu P, co-P (B8 A2)

Die Komplexitätsklasse **co-P** sei definiert als die Menge der Sprachen $\mathcal{L}$, deren Komplementsprache $\mathcal{L}^C$ in der Komplexitätsklasse **P** liegt.

Erinnerung: Zu einer Sprache $\mathcal{L}$ über einem Alphabet Σ ist die Komplementsprache $\mathcal{L}^C = \Sigma^* \setminus \mathcal{L}$.

Beweisen Sie: **co-P** = **P**

Graph-n-Färbbarkeit

Kurzdefinition

Gegeben ist ein ungerichteter Graph den man mit n Farben einfärben soll ohne das zwei benachbarte (mit einer Kante verbundenen) Knoten die gleiche Farbe haben.

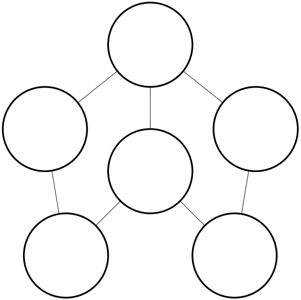
Formal

Gegeben: Ein ungerichteter Graph $G = (V, E)$ mit Knoten $v \in V$ und Kanten $e = (v_1, v_2) \in E$ mit $v_1, v_2 \in V$ und n Farben $F_1, F_2, ..., F_n \in F$.
Gesucht: Eine totale Funktion

$$g : V \to F : \forall (v_1, v_2) \in E : g(v_1) \neq g(v_2)$$

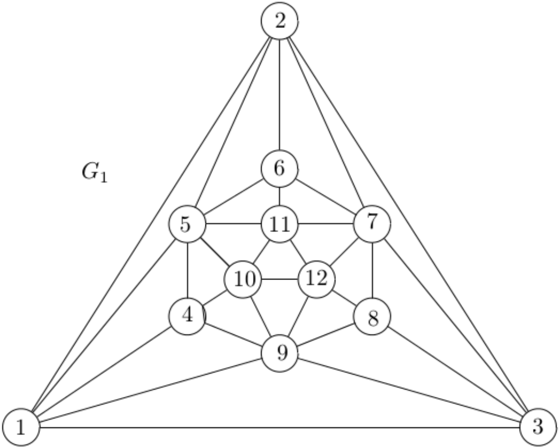
Beispiel Graph-Färbbarkeit

Färbe diesen Graph mit 2 Farben



Beispiel Graph-Färbbarkeit

Ist dieser Graph 3-Färbbar?



Aufgabe zu Graphfärbbarkeit (B8 A3)

Gegeben seien ein ungerichteter Graph $G = (V, E)$ mit Knoten $v \in V$
und Kanten $e = (v_1, v_2) \in E$ mit $v_1, v_2 \in V$
und zwei Farben A und B.
Beweisen Sie: Die Sprache 2-COLOR $=$
$\{G \mid G = (V, E)$ ungerichteter Graph mit $\exists$ totale Funktion $g : V \to \{A, B\} : \forall\, (v_1, v_2) \in E : g(v_1) \neq g(v_2)\}$
liegt in der Komplexitätsklasse **P**.

n-SAT

Kurzdefinition

Das selbe wie SAT allerdings enthalten ALLE Klauseln exakt n Literale.

Formal

Gegeben: Formel in konjunktiver Normalform

- Menge U von Variablen
- Menge C von Klauseln (Disjunktionen) über U mit je exakt n Literalen

Frage: Existiert eine (alle Klauseln) erfüllende Variablenbelegung?

Beispiele

3-Sat

$$U = \{a, b, c, d\}$$

$$C = \{\{\neg a, \neg b, d\}, \{\neg b, \neg a, c\}, \{\neg c, \neg d, a\}, \{b, c, d\}\}$$

2-Sat

$$U = \{a, b, c, d, e\}$$

$$C = \{\{a, \neg b\}, \{\neg b, \neg d\}, \{b, \neg d\}, \{\neg c, d\}, \{d, e\}, \{d, \neg e\}\}$$

Reduktionsaufgabe (B8 A4)

Das Problem 2-SAT ist folgendermaßen definiert:

2-SAT

Gegeben eine in ihrer Größe polynomiell beschränkte aussagenlogische Formel F in konjuktiver Normalform, wobei jede Klausel genau 2 Literale enthält. F hat also die Form

$$F = \Lambda_{i=1}^{n} (L_i \vee N_i),$$

wobei L_i und N_i Literale sind, also von der Form X oder $\neg X$ für eine Variable X sind.

Gibt es eine erfüllende Belegung für F?

Geben Sie eine polynomielle Reduktion von 2-COLOR auf 2-SAT an!

Aufgabe B8 A1

Ein Algorithmus, der eine Zahl $n \in \mathbb{N}$ als Eingabe erhält und prüft, ob n prim ist, sei gegeben durch die
nachfolgende Beschreibung einer Turingmaschine $\mathcal{M}$, die die Funktion

$$f : \mathbb{N} \to \{0, 1\}, n \mapsto \begin{cases} 1 & , \; n \text{ prim} \\ 0 & , \; \text{sonst} \end{cases}$$

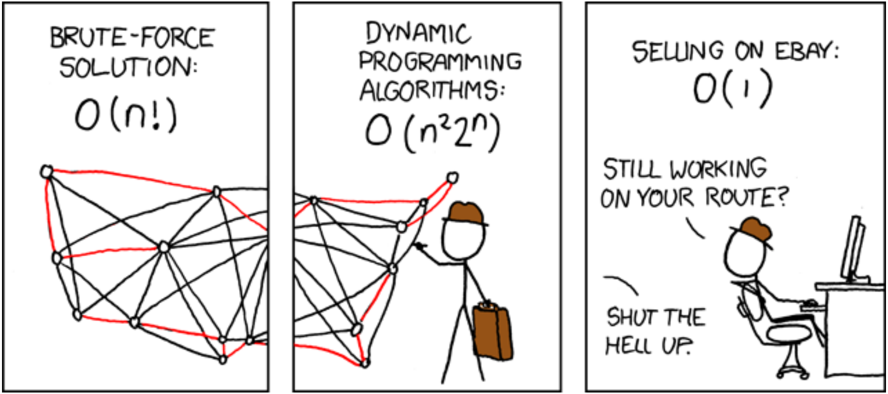
realisiert.

Aufgabe B8 A1

1. Initialisiere Zähler z mit 2 und Ausgabe a mit 1
2. Berechne $n \,\%\, z$
3. Prüfe, ob $n \,\%\, z = 0$ gilt:
- Falls ja: Setze a auf 0 und gehe zu Schritt 4.
- Falls nein: Gehe zu Schritt 4.
4. Prüfe, ob $z < n$ gilt:
- Falls ja: Erhöhe z um 1 und gehe zu Schritt 2.
- Falls nein: Lösche das Band, schreibe a auf das Band und stoppe

Geben Sie die Komplexität des oben angegebenen Algortihmus bezüglich der Eingabe n und auch bezüglich der Länge der Binärdarstellung von n an!

Bis zum nächsten Mal!



Lizenzen