# Lösungsblatt 10

## Aufgabe 1

Analog zur Definition von $M_w$ bezeichne in dieser Aufgabe $F_w$ die durch das Wort $w \in \{0,1\}^*$ codierte boolesche Formel.

1. Zeigen Sie, dass folgende Sprache coNP-vollständig bezüglich $\leq_p$ ist:

$$\text{UNSAT} = \{w \mid F_w \text{ ist unerfüllbar}\}.$$

2. Die Komplexitätsklasse DP (*Difference Polynomial-Time*) sei wie folgt definiert:

$$\mathsf{DP} = \{A \setminus B \mid A, B \in \mathsf{NP}\}.$$

   (a) In welcher Beziehung stehen die den Klassen DP, NP und coNP zueinenader?

   (b) Geben Sie eine bezüglich $\leq_p$ DP-vollständige Sprache an und beweisen Sie die Korrektheit Ihrer Antwort.

## Lösung

1. Sei $L \in \mathsf{coNP}$ beliebig. Dann gilt $\overline{L} \in \mathsf{NP}$ und somit $\overline{L} \leq_p \text{SAT}$. Da $A \leq_p B$ genau dann gilt, wenn $\overline{A} \leq_p \overline{B}$ gilt, erhalten wir:

$$L = \overline{\overline{L}} \leq_p \overline{\text{SAT}} = \text{UNSAT}.$$

2. (a) Es gilt:

   (1) $\mathsf{NP} \subseteq \mathsf{DP}$

   (2) $\mathsf{coNP} \subseteq \mathsf{DP}$

   (3) $\mathsf{NP} = \mathsf{coNP} \iff \mathsf{DP} = \mathsf{NP}$

   Beweise:

   (1) Sei $L \in \mathsf{NP}$ beliebig. Aus $L, \emptyset \in \mathsf{NP}$ folgt $L = L \setminus \emptyset \in \mathsf{DP}$.

   (2) Sei $L \in \mathsf{coNP}$ beliebig (o. B. d. A. über dem Alphabet $\Sigma$). Aus $\Sigma^*, \overline{L} \in \mathsf{NP}$ folgt $L = \Sigma^* \setminus \overline{L} \in \mathsf{DP}$.

   (3) Unter der Annahme $\mathsf{NP} = \mathsf{coNP}$ muss nur noch die Inklusion $\mathsf{DP} \subseteq \mathsf{NP}$ gezeigt werden. Diese gilt, da NP unter Schnitt abgeschlossen ist. Unter der Annahme $\mathsf{DP} = \mathsf{NP}$ gilt

$$\mathsf{coNP} \subseteq \mathsf{DP} = \mathsf{NP}$$

   und demnach auch

$$\mathsf{NP} = \mathsf{co}(\mathsf{coNP}) \subseteq \mathsf{coNP}.$$

- $\mathsf{coNP} \subseteq \mathsf{DP} = \mathsf{NP}$
- $\mathsf{NP} \subseteq \mathsf{coNP}$

(b) Wir zeigen, dass die Sprache

$$\text{SAT-UNSAT} = \{u\#v \mid u \in \text{SAT} \wedge v \in \text{UNSAT}\}$$

DP-vollständig ist.

SAT-UNSAT $\in$ DP

Gesucht sind zwei Sprachen $A, B \in \mathsf{NP}$ mit SAT-UNSAT $= A \setminus B$. Betrachte hierzu die Sprachen $A = \{u\#v \mid u \in \text{SAT}\}$ und $B = \{u\#v \mid v \in \text{SAT}\}$. Diese sind in NP, da man eine TM konstruieren kann, die zuerst den Suffix $\#v$ bzw. den Präfix $u\#$ von $u\#v$ löscht (lineare Zeit) und dann die Zugehörigkeit von $u$ bzw. $v$ in SAT testet. Dann gilt für alle $u, v \in \{0,1\}^*$:

$$\begin{aligned} u\#v \in \text{SAT-UNSAT} &\iff u \in \text{SAT} \wedge v \in \text{UNSAT} \\ &\iff u \in \text{SAT} \wedge v \notin \text{SAT} \\ &\iff u\#v \in A \wedge u\#v \notin B \\ &\iff u\#v \in A \setminus B. \end{aligned}$$

Wörter, die nicht von der Form $u\#v$ sind, sind weder in SAT-UNSAT noch in $A \setminus B$ enthalten, woraus die Gleichheit beider Mengen folgt.

SAT-UNSAT ist DP-hart

Zu zeigen ist:

$$\forall L \in \mathsf{DP}\colon L \leq_p \text{SAT-UNSAT}.$$

Sei hierfür $L \in \mathsf{DP}$ eine beliebige Sprache über einem Alphabet $\Sigma$. Dann ist $L = A \setminus B$ für Sprachen $A, B \in \mathsf{NP}$. Da SAT NP-hart ist, gibt es zwei totale, in polynomieller Zeit berechenbare Funktionen $f, g\colon \Sigma^* \to \{0,1\}^*$ mit

$$w \in A \iff f(w) \in \text{SAT} \qquad \text{und} \qquad w \in B \iff g(w) \in \text{SAT}.$$

Betrachte die Funktion $h\colon \Sigma^* \to \{0,1,\#\}$ mit $h(w) = f(w)\#g(w)$. Dann ist $h$ total und in polynomieller Zeit berechenbar, da $f$ und $g$ es jeweils sind, und für alle $w \in \Sigma^*$ gilt:

$$\begin{aligned} w \in L &\iff w \in A \wedge w \notin B \\ &\iff f(w) \in \text{SAT} \wedge g(w) \notin \text{SAT} \\ &\iff f(w) \in \text{SAT} \wedge g(w) \in \text{UNSAT} \\ &\iff f(w)\#g(w) \in \text{SAT-UNSAT} \\ &\iff h(w) \in \text{SAT-UNSAT}. \end{aligned}$$

## Aufgabe 2

Eine *starke Zusammenhangskomponente* (engl. *strongly-connected component*) eines gerichteten Graphen ist eine maximale Teilmenge $C$ der Knoten, sodass jeder Knoten aus $C$ von jedem anderen Knoten aus $C$ erreichbar ist.

Zeigen Sie, dass das folgende Entscheidungsproblem NL-vollständig bezüglich $\leq_{\log}$ ist:

SCC
**Eingabe:** Ein gerichteter Graph $G = (V, E)$ und eine Zahl $k \in \mathbb{N}$.
**Frage:** Enthält $G$ genau $k$ starke Zusammenhangskomponenten?

*Hinweis*: Sie dürfen die NL-Vollständigkeit des folgenden Problems annehmen:

GAP
**Eingabe:** Ein gerichteter Graph $G = (V, E)$ und zwei Knoten $s, t \in V$.
**Frage:** Gibt es in $G$ einen Pfad von $s$ nach $t$?

Diese wird in Vorlesungseinheit 39 bewiesen.

### Lösung

Wir zeigen, dass SCC in NL liegt und NL-hart ist.

SCC $\in$ NL

SCC kann von einer TM entschieden werden, die wie folgt arbeitet:

**Eingabe:** Codierung einer SCC-Instanz $(G, k)$

```
1  count := 0
2  for i = 1, ..., m
3       new := true
4       for j = 1, ..., i − 1
5            if v_i und v_j sind in G voneinenader erreichbar
6                 new := false
7       if new
8            count := count +1
9  if count = k
10      akzeptiere
```

Dabei haben wir O. B. d. A. angenommen, dass $G$ die Knotenmenge $V = \{v_1, \ldots, v_m\}$ besitzt.

Der Algorithmus zählt die Anzahl der starken Zusammenhangskomponenten mit der Variable count. Diese wird im $i$-ten Schleifendurchlauf genau dann um eins erhöht, wenn $v_i$ der Knoten mit dem kleinsten Index in seiner starken Zusammenhangskomponente ist. Da dies für jeden Knoten $v_i$ genau einmal überprüft wird, ist der Wert von count nach dem äußeren Schleifendurchlauf genau die Anzahl der starken Zusammenhangskomponenten.

Die Variablen $i$, $j$ und count können jeweils in ein Arbeitsband gespeichert werden. Da sie nur Werte aus $\{1, \ldots, m\}$ annehmen können und $m$ höchstens so groß sein kann wie die

Länge $n$ der Eingabe, benötigen ihre Binärdarstellungen jeweils Platz $\mathcal{O}(\log n)$. Da die Variable new nur konstant viele Werte annehmen kann, können diese in den Zuständen gespeichert werden.

Dass die Anweisung auf Zeile 5 von einer nichtdeterministischen Turingmaschine in logarithmischen Platz ausgeführt werden kann folgt aus GAP $\in$ NL, was wir ohne Beweis annehmen dürfen. Man beachte, dass diese die einzige Stelle im Algorithmus ist, in der der Nichtdeterminismus ausgenutzt wurde.

*Wichtige Bemerkung*:

In der Ergänzung wurde der folgende (kompliziertere) Algorithmus vorgestellt:

**Eingabe:** Codierung einer SCC-Instanz $(G, k)$

```
 1  min := 0
 2  sum := 0
 3  for i = 1, ..., k
 4       min := guess_{min+1,...,m}()
 5       sum := sum +1
 6       for j = 1, ..., min -1
 7            if v_j und v_min sind in G voneinenader erreichbar
 8                 gehe in eine Endlosschleife
 9       for j = min +1, ..., m
10            if v_j und v_min sind in G voneinenader erreichbar
11                 sum := sum +1
12  if sum = m
13       akzeptiere
```

Zur Anweisung auf Zeile 4 wurde gesagt, dass bei $\text{guess}_A()$ die Menge $A$ lediglich endlich sein muss. Dies ist nicht ausreichend, da die Menge $A$ nach der Konvention aus Übungsblatt 6 eine **feste** endliche Menge sein soll, d. h. sie darf weder von der Eingabegröße $n$ noch von irgendeiner Variable (hier: min) abhängig sein. Man könnte dieses Problem beheben indem man stattdessen zuerst den Wert von min um eins erhöht und dann $m - \min$ mal die Anweisung

$$\min := \min + \text{guess}_{\{0,1\}}()$$

ausführt. Dieser Verbesserungsvorschlag hat den Vorteil, dass man sich keine Gedanken über die Binärdarstellung von Zahlen machen muss. Da die Menge $\{0, 1\}$ in obigem Sinne fest ist, ist die Funktion $\text{guess}_{\{0,1\}}()$ zulässig.

SCC ist NL-hart

Wir zeigen GAP $\leq_{\log}$ SCC. Sei $x$ eine Kodierung einer GAP-Instanz $(G, s, t)$ bestehend aus einem Graph $G = (V, E)$ und zwei Knoten $s, t \in V$. Wähle $f(x)$ als Kodierung einer SCC-Instanz $(G', k)$ bestehend aus dem Graph $G' = (V, E')$ mit

$$E' = E \cup \{(v, s) \mid v \in V\} \cup \{(t, v) \mid v \in V\}$$

und $k = 1$.

$f$ ist total und in logarithmischem Platz berechenbar und für alle Codierungen $x$ von GAP-Instanzen gilt:

$$\begin{aligned} x \in \text{GAP} &\implies \text{es gibt in } G \text{ einen Pfad } (s, \ldots, t) \\ &\implies \text{für je zwei Knoten } u, v \in V \text{ gibt es in } G' \text{ einen Pfad } (u, s, \ldots, t, v) \\ &\implies \text{in } G' \text{ ist jeder Knoten von jedem anderen aus erreichbar} \\ &\implies f(x) \in \text{SCC} \end{aligned}$$

und

$$\begin{aligned} x \notin \text{GAP} &\implies \text{es gibt in } G \text{ keinen Pfad von } s \text{ nach } t \\ &\implies \text{es gibt in } G' \text{ keinen Pfad von } s \text{ nach } t \\ &\implies G' \text{ besitzt mindestens zwei starke Zusammenhangskomponenten} \\ &\implies f(x) \notin \text{SCC}, \end{aligned}$$

d. h. $x \in \text{GAP} \iff f(x) \in \text{SCC}$.