**Universität des Saarlandes**
**Fachbereich 14 – Informatik**
Prof. Dr. G. Hotz / Dr. E. Schömer / T. Warken

# Musterlösung zum 12. Übungsblatt
# Informatik IV

1. **Aufgabe**:
   Seien $L_1 \subseteq \Sigma_1^*$, $L_2 \subseteq \Sigma_2^*$ und $L_3 \subseteq \Sigma_3^*$. Es gelte $L_1 \leq_p L_2$ und $L_2 \leq_p L_3$. Es ist zu zeigen, daß $L_1 \leq_p L_3$ gilt.
   Nach Voraussetzung exitsiteren polynomialzeit-berechenbare Funktionen $f_1 : \Sigma_1^* \to \Sigma_2^*$ und $f_2 : \Sigma_2^* \to \Sigma_3^*$ mit den Eigenschaften $w \in L_1 \Leftrightarrow f_1(w) \in L_2$ für alle $w \in \Sigma_1^*$, und $w \in L_2 \Leftrightarrow f_2(w) \in L_3$ für alle $w \in \Sigma_2^*$. Definieren wir nun $h := f_1 \circ f_2$, so gilt offensichtlich $w \in L_1 \Leftrightarrow h(w) \in L_3$ für alle $w \in \Sigma_1^*$. Es bleibt noch zu zeigen, daß $h$ polynomialzeit-berechenbar ist.
   Da $f_1$ polynomialzeit-berechenbar ist, gibt es ein Polynom $p_1$ mit $|f_1(w)| \leq p_1(|w|)$, denn es gibt eine TM, die $f_1(w)$ in höchstens $p_1(|w|)$ Schritten berechnet, und die in jedem Rechenschritt ihren Schreib/Lesekopf um höchstens eine Position bewegen kann.
   Da $f_2$ polynomialzeit-berechenbar ist, folgt hieraus, daß es ein Polynom $p_2$ gibt, so daß zur Berechnung von $h$ höchstens Zeit $p_2(p_1(|w|))$ benötigt wird. Da $p_1$ und $p_2$ Polynome sind, ist auch $p_2 \circ p_1$ ein Polynom, und die Behauptung ist bewiesen.

2. **Aufgabe**:
   (i) $\Rightarrow$ (ii):
   Es gibt eine NP-vollständige Sprache $L$ (z.B. $L = SAT$). Dann gilt nach Definition von NP-vollständig $L \in NP$. Ist $P = NP$, so ist dies gleichbedeutend mit $L \in P$.

   (ii) $\Rightarrow$ (i):
   Sei $L$ NP-vollständig, und es gelte $L \in P$. Daß $P \subseteq NP$ ist, ist klar. Sei also $L_1 \in NP$ beliebig. Da $L$ NP-vollständig ist, gilt $L_1 \leq_p L$, d.h. es gibt eine polynomialzeit-berechenbare Funktion $f : \Sigma^* \to \Sigma^*$ mit $w \in L_1 \Leftrightarrow f(w) \in L$ für alle $w \in \Sigma^*$. Es gibt also eine deterministische TM $M_f$, die $f(w)$ in Zeit höchstens $p_f(|w|)$ für ein Polynom $p_f$ berechnet. Da $L \in P$ gilt, gibt es eine deterministische TM $M$, die die Frage, ob $w \in L$ gilt, in Zeit höchstens $p(|w|)$ für ein Polynom $p$ entscheidet.
   Um nun zu $w \in \Sigma^*$ zu entscheiden, ob $w \in L_1$ gilt, berechnet man mit Hilfe von $M_f$ zuerst $f(w)$ in Zeit $p_f(|w|)$ und entscheidet dann mittls $M$ in Zeit $p(|f(w)|)$, ob $f(w) \in L$ gilt. Dazu braucht man insgesamt Zeit höchstens $p_f(|w|) + p(|f(w)|) \leq p_f(|w|) + p(p_f(|w|))$. Dies ist polynomielle Zeit, denn da $p_f$ und $p$ Polynome sind, ist auch $p_f + p \circ p_f$ ein Polynom.

3. **Aufgabe**:

   1. CLIQUE $\in$ NP:
      Zu einem Paar $(G, k)$ mit $G = (V, E)$ rate man nichtdeterministisch eine Teilmenge $V' \subseteq V$ der Größe $k$. Dazu braucht man Zeit $\mathcal{O}(k)$. Dann prüfe man für alle $u, v \in V'$ mit $u \neq v$, ob $\{u, v\} \in E$ gilt. Falls ja, gebe man ja aus, ansonsten nein. Für diese Verifikation muß man $\mathcal{O}(k^2)$ Knotenpaare betrachten und für jedes dieser Paare $\mathcal{O}(\sharp E)$ Vergleiche machen. Die Gesamtzeit ist also $\mathcal{O}(n^3)$, wenn $n$ die Länge der Eingabe ist.
   2. CLIQUE ist NP-hart:
      Wir zeigen, daß $SAT \leq_p$ CLIQUE gilt. Sei also eine beliebige Formel $F$ in KNF gegeben. $F$ habe die Form
      $$F = \bigwedge_{i=1}^{m} c_i,$$
      und jede Klausel $c_i$ habe die Form
      $$c_i = \bigvee_{j=1}^{n_i} l_{ij}.$$
      Dabei seien die $l_{ij}$ Literale. OBdA komme keine Klausel mehrfach vor, und in jeder Klausel komme kein Literal mehrfach vor. Wir konstruieren zu $F$ einen Graph $G = (V, E)$:
      $$\begin{aligned} V &:= \{(i,j) \mid i = 1, \ldots, m \text{ und } j = 1, \ldots, n_i\} \\ E &:= \{\{(i,j),(i',j')\} \mid i \neq i' \text{ und die Literale } l_{ij} \text{ und } l_{i'j'} \text{ sind nicht komplementär}\}. \end{aligned}$$

(bitte wenden)

Diese Konstruktion ist offensichtlich in polynomieller Zeit durchführbar.

**Beh.:** $F$ ist genau dann erfüllbar, wenn $G$ eine Clique der Größe $m$ besitzt.

**Beweis:** Sei $F$ erfüllbar. Dann gibt es eine zu $F$ passende Belegung $\alpha$ mit $\alpha(F) = 1$. Daraus folgt $\alpha(c_i) = 1$ für alle $i = 1, \dots, m$. Hieraus wiederum folgt, daß es zu jedem $i = 1, \dots, m$ ein $j_i \in \{1, \dots, n_i\}$ gibt mit $\alpha(l_{ij_i}) = 1$. Für $i \neq i'$ sind die Literale $l_{ij_i}$ und $l_{i'j'_i}$ nicht komplementär zueinander, da sonst nicht $\alpha(l_{ij_i}) = \alpha(l_{i'j'_i})$ gelten könnte. Daher existiert eine Kante $\{(i, j_i), (i', j'_i)\} \in E$. Definieren wir also die Menge $V'$ durch

$$V' := \{(i, j_i) \mid i = 1, \dots, m\}$$

so bildet $V'$ eine Clique der Größe $m$.

Sei umgekehrt $V' \subseteq V$ eine Clique der Größe $m$. Dann definieren wir wie folgt eine Belegung $\alpha$ für $F$:

$$\alpha(x) := \begin{cases} 1 & \text{falls es } (i,j) \in V' \text{ gibt mit } l_{ij} = x \\ 0 & \text{falls es } (i,j) \in V' \text{ gibt mit } l_{ij} = \neg x \\ \text{beliebig} & \text{sonst} \end{cases}$$

Man beachte, daß sich die ersten beiden Fälle in der Definition von $\alpha$ gegenseitig ausschließen, da es keine Kante zwischen komplementären Literalen gibt. Also ist $\alpha$ wohldefiniert. Ferner müssen nach Definition von $E$ alle $l_{ij}$ mit $(i, j) \in V'$ in verschiedenen Klauseln liegen. Damit gibt es aufgrund der Definition von $\alpha$ zu jedem $i = 1, \dots, m$ ein $j \in \{1, \dots, n_i\}$, so daß $\alpha(l_{ij}) = 1$ gilt. Daraus folgt $\alpha(F) = 1$, d.h. die Formel $F$ ist erfüllbar.

4. **Aufgabe**:

   (a) 1. $3 - SAT^* \in NP$: Klar, da $SAT \in NP$.
   2. $3 - SAT^*$ ist NP-hart:
   Wir zeigen $3 - SAT \leq_p 3 - SAT^*$. Sei dazu $F$ eine beliebige Formel in KNF, in der jede Klausel höchstens drei Literale enthält. OBdA können wir davon ausgehen, daß in keiner Klausel ein Literal mehrfach vorkommt. Ist $c$ eine Klausel in $F$, die nur zwei Literale enthält, etwa $c = (x_1 \vee x_2)$, dann wählen wir eine bisher in $F$ nicht vorkommende Variable $y$ und ersetzen $c$ durch die beiden Klauseln $(x_1 \vee x_2 \vee y) \wedge (x_1 \vee x_2 \vee \neg y)$. Ist dagegen $c$ eine Klausel, die nur ein Literal enthält, wtwa $c = x_1$, dann wählen wir zwei bisher in $F$ nicht vorkommende Variablen $y_1$ und $y_2$ und ersetzen $c$ durch die vier Klauseln $(x_1 \vee y_1 \vee y_2) \wedge (x_1 \vee \neg y_1 \vee y_2) \wedge (x_1 \vee y_1 \vee \neg y_2) \wedge (x_1 \vee \neg y_1 \vee \neg y_2)$. Man sieht sofort, daß man diese Konstruktion in Linearzeit durchführen kann und daß sich an der Erfüllbarkeit der Formel durch die Konstruktion nichts ändert.

   (b) Sei $F$ eine Formel in KNF, in der jede Klausel drei verschiedene Literale enthält, und in der jede Variable insgesamt höchstens dreimal auftaucht. Wir können oBdA davon ausgehen, daß keine Variable in einer Klausel mehrfach vorkommt (sonst können wir die Klausel einfach weglasen). Die Klauseln von $F$ seien $c_1, \dots, c_m$, und die Variablen seien $x_1, \dots, x_n$. Wir konstruieren einen bipartiten Graph $G = (V, E)$ mit

$$\begin{aligned} V &= \{c_1, \dots, c_m\} \cup \{x_1, \dots, x_n\} \\ E &= \{\{c_i, x_j\} \mid x_j \text{ oder } \neg x_j \text{ kommt in } c_i \text{ vor}\}. \end{aligned}$$

   Sei nun für ein $k \in \{1, \dots, m\}$ eine $k$-elementige Teilmenge $C$ der Klauseln gegeben. Da jede Klausel drei verschiedene Literale (und damit auch drei verschiedene Variablen) enthält, gehen im Graph $3k$ Kanten von $C$ aus. Da jede Variable in höchstens drei verschiedenen Klauseln vorkommt, müssen diese Kanten bei mindestens $k$ verschiedenen Variablen enden. Damit ist die Voraussetzung für das Heiratsproblem gegeben. Es existiert also ein Matching, das die Menge aller Klauseln umfaßt. D.h. es existieren paarweise verschiedene $x_{i_1}, \dots, x_{i_m}$, so daß

$$E' := \{\{c_1, x_{i_1}\}, \dots, \{c_m, x_{i_m}\}\} \subseteq E$$

   gilt. Wir definieren

$$\alpha(x_{i_j}) = \begin{cases} 1 & \text{falls das Literal } x_{i_j} \text{ in } c_j \text{ vorkommt} \\ 0 & \text{falls das Literal } \neg x_{i_j} \text{ in } c_j \text{ vorkommt,} \end{cases}$$

   und $\alpha(x)$ beliebig für alle übrigen Variablen. Man sieht, daß $\alpha$ jede Klausel von $F$ erfüllt, und damit ist $\alpha(F) = 1$.

(bitte wenden)

Damit haben wir gezeigt, daß jede Formel von der angegebenen Form erfüllbar ist. Also ist trivialerweise $3-\mathrm{SAT}^{**} \in \mathrm{P}$, denn ein Polynomialzeit-Algorithmus für $3-\mathrm{SAT}^{**}$ kann folgendermaßen vorgehen: Gegeben eine Formel $F$ von der angegebenen Form. Ausgabe des Algorithmus: erfüllbar. Die Ausgabe des Wortes "erfüllbar" ist offensichtlich in polynomieller Zeit machbar.

(c) Daß $\mathrm{SAT3} \in \mathrm{NP}$ ist, ist klar, da $\mathrm{SAT} \in \mathrm{NP}$ ist. Um zu zeigen, daß SAT3 NP-hart ist, beweisen wir $\mathrm{SAT} \leq_p \mathrm{SAT3}$. Sei also $F$ eine beliebige Formel in KNF, und sei $x$ eine Variable von $F$, die $k$ mal vorkommt mit $k \geq 4$.

**Fall 1:** Das Literal $x$ kommt mindestens zweimal vor. Dann wählen wir eine bisher in $F$ nicht vorkommende Variable $y$, ersetzen zwei der Vorkommen von $x$ durch $y$ und fügen die Klausel $\neg y \vee x$ (d.h. $y \to x$) hinzu. In diesem Fall sagen wir, die neue Variable $y$ ist vom Typ 1.
**Fall 2:** Das Literal $x$ kommt weniger als zweimal vor. Dann kommt das Literal $\neg x$ mindestens zweimal vor. Wir wählen eine bisher in $F$ nicht vorkommende Variable $y$, ersetzen zwei der Vorkommen von $\neg x$ durch $y$ und fügen die Klausel $\neg y \vee \neg x$ (d.h. $y \to \neg x$) hinzu. In diesem Fall sagen wir, die neue Variable $y$ ist vom Typ 2.
Nach diesem Konstruktiosschritt kommt die Varialbe $x$ nur noch $k-1$ mal und $y$ drei mal vor. Wiederholen wir diesen Schritt solange, bis alle Variablen nur noch höchstens dreimal vorkommen, erhalten wir in Polynomialzeit eine Formel von der gewünschten Bauart. Sei $F'$ diese Formel. Wir müssen zeigen, daß $F$ genau dann erfüllbar ist, wenn $F'$ erfüllbar ist.
Sei $F$ erfüllbar, und sei $\alpha$ eine erfüllende Belegung. Sei weiter $y$ eine Variable, die in $F'$, nicht aber in $F$ vorkommt. Dann setzen wir

$$\alpha(y) = \begin{cases} \alpha(x) & \text{falls } y \text{ vom Typ 1 ist} \\ \neg\alpha(x) & \text{falls } y \text{ vom Typ 2 ist.} \end{cases}$$

Damit gilt offensichtlich $\alpha(F') = 1$, und $F'$ ist erfüllbar.
Sei umgekehrt $F'$ erfüllbar und $\alpha$ eine erfüllende Belegung. Sei $y$ eine Varible, die in $F'$, nicht aber in $F$ vorkommt. Ferner sei $c'$ eine Klausel von $F'$, in der $y$ vorkommt. Wegen $\alpha(F') = 1$ gilt auch $\alpha(c') = 1$.

**Fall 1:** $y$ ist vom Typ 1. Sei zunächst $\alpha(y) = 0$. Da $\alpha(c') = 1$ ist und $y$ nur nicht negiert in $c'$ vorkommt, muß es ein von $y$ verschiedenes Literal in $c'$ geben, das unter $\alpha$ Eins wird. Die Klausel $c'$ behält also ihren Wert unter $\alpha$, wenn man $y$ wieder durch $x$ ersetzt. Sei nun $\alpha(y) = 1$. Dann gilt wegen $\alpha(y \to x) = 1$ auch $\alpha(x) = 1$. Daher behält auch in diesem Fall die Klausel $c'$ ihren Wert unter $\alpha$, wenn man $y$ wieder durch $x$ ersetzt.
**Fall 2:** $y$ ist vom Typ 2. Sei zunächst $\alpha(y) = 0$. Dann gilt nach der selben Begründung wie im Fall 1, daß die Klausel $c'$ ihren Wert unter $\alpha$ behält, wenn man $y$ wieder durch $\neg x$ ersetzt. Sei also $\alpha(y) = 1$. Dann gilt wegen $\alpha(y \to \neg x) = 1$ auch $\alpha(\neg x) = 1$. Daher behält die Klausel $c'$ auch in diesem Fall ihren Wert unter $\alpha$, wenn man $y$ wieder durch $\neg x$ ersetzt.
Rekonstruiert man also $F$ aus $F'$, indem man die neuen Klauseln entfernt und die neuen Variablen wieder durch Literale der Form $x$ oder $\neg x$ ersetzt, bleibt in jedem Schritt der Wert unter $\alpha$ gleich. Also gilt $\alpha(F) = 1$, und somit ist $F$ erfüllbar.

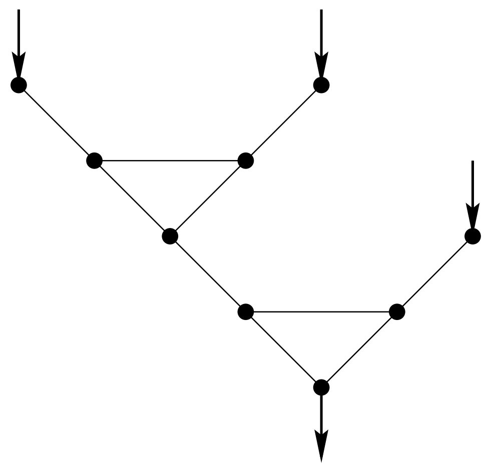
5. **Aufgabe**:

(a)
- Alle Literalknoten sind mit 1 bzw. 2 gefärbt, wobei komplementäre Literale unterschiedlich gefärbt sind.
- Die Knoten $s$ und $t$ sind mit 1 bzw. 2 gefärbt. Dabei tragen $s$ und $t$ unterschiedliche Farben.
- Alle Klauselknoten haben die gleiche Farbe wie $t$.

(b) Da als Farben von Literalknoten nur zwei Farben auftreten, und komplementäre Literale verschieden gefärbt sind, kann man eine dieser beiden Farben mit dem Wahrheitswert 0 und die andere mit 1 identifizieren.

(bitte wenden)

(c)



(d) Wir erweitern $G$ nun für jede Klausel $c_i$ um jeweils eine Kopie des Graphen aus Teil (c), wobei wir jeweils die drei Eingangsknoten mit den drei entsprechenden Literalknoten identifizieren und den Ausgangsknoten mit dem Klauselknoten $c_i$.

**Beh.:** $F$ ist genau dann erfüllbar, wenn der so konstruierte Graph 3-färbbar ist.

**Beweis:** Sei $\alpha$ eine Belegung mit $\alpha(F) = 1$. Dann gibt es in jeder Klausel mindestens ein Literal $l$ mit $\alpha(l) = 1$. Wir wählen in jeder Klausel ein solches Literal und färben den entsprechenden Knoten mit Farbe 1. Alle Anderen Literalknoten färben wir beliebig mit 1 oder 2, wobei wir komplementäre Literalknoten unterschiedlich färben. Die in Teil (c) angegebenen Teilgraphen lassen nun eine Färbung zu, bei der alle Klauselknoten mit 1 gefärbt sind (Eigenschaft (i) in Teil (c)). Färben wir nun noch $y$ mit Farbe 3, $s$ mit 2 und $t$ mit 1, so ist der Graph 3-gefärbt.

Sei umgekehrt eine 3-Färbung des konstruierten Graphen gegeben. Dann tragen alle Klauselknoten dieselbe Farbe, etwa 1. Dann tragen die die Knoten $y$ und $s$ die Farben 2 und 3. OBdA sei $y$ mit 3 und $s$ mit 2 gefärbt. Dann hat $t$ die Farbe 1, und alle Literalknoten sind mit 1 bzw. 2 gefärbt, wobei komplementäre Literale unterschiedlich gefärbt sind. Wir definieren eine Belegung $\alpha$ durch

$$\alpha(x) = \begin{cases} 1 & \text{falls der Knoten } x \text{ die Farbe 1 hat} \\ 0 & \text{falls der Knoten } x \text{ die Farbe 2 hat.} \end{cases}$$

**Annahme:** $\alpha(F) = 0$.
Dann ist mindestens eine Klausel $c_i$ nicht erfüllt, d.h. die drei Literale in $c_i$ tragen den Wert $0$, d.h. die entsprechenden Literalknoten sind mit der Farbe 2 gefärbt. Dann gilt aber wegen Eigenschaft (ii) der Teilgraphen aus Teil (c), daß auch der zu $c_i$ gehörige Knoten die Farbe 2 hat. Dies ist ein Widerspruch, da die Klauselknoten mit 1 gefärbt sind. Also ist $\alpha(F) = 1$ und daher $F$ erfüllbar.

(e) Es ist noch zu zeigen, daß $3COL \in NP$ gilt und daß die angegebene Konstruktion in Polynomialzeit durchführbar ist. Dies ist aber leicht zu zeigen.