Prof. Dr. Rolf Wanka
Bernd Bassimir, Alexander Raß, Jonas Schreiner,
Stefan Schwanecke, Jeanette Wohnig

Erlangen, 18. Dezember 2018

Abgabe bis Di., 8.1.19, 23:59 Uhr

Übungen zur Vorlesung

Berechenbarkeit und Formale Sprachen

WS 2018/2019

Blatt 8

Je mehr Plus-Zeichen $+$, desto wichtiger, je mehr Sterne $\star$, desto schwieriger.

AUFGABE 41:
[Präsenzaufgabe, $++++,\star$] Sieben Ankreuzfragen aus einer alten Klausur. Anders als dort: kreuzen Sie an, ob die Aussage links stimmt (+) oder falsch ($-$) ist **und** begründen Sie die Antwort. In der Klausur im Frühjahr (der noch nicht verbindliche Termin: **11.04.2019**) werden Sie einige kurze Begründungen selbst schreiben müssen, da es keine Multiple-Choice-Aufgaben (im Juristen-Deutsch: Antwort-Wahl-Verfahren) mehr geben wird!

Frage: Stimmt diese Aussage?	+ − ☐☐
1. Alle Sprachen in NP sind entscheidbar.	☐☐
2. $L = \{\langle M\rangle \mid M$ ist nichtdeterministische 2-Band-Turingmaschine, für die es w_1 und w_2, $w_1 \neq w_2$, gibt, so daß M die Eingabe w_1 und die Eingabe w_2 akzepiert$\}$ ist rekursiv aufzählbar.	☐☐
3. Seien L_1 und L_2 rekursiv aufzählbare Sprachen. Dann ist $L_1 \cap L_2$ auch rekursiv aufzählbar.	☐☐
4. Seien $A \subseteq \Sigma^*$, $B \subseteq \Sigma^*$. B ist reduzierbar auf A ($B \leq A$), wenn gilt: es gibt eine totale Funktion $f : \Sigma^* \to \Sigma^*$ mit: $w \in B \iff f(w) \in A$	☐☐
5. Sei CLIQUE das Cliquen-Problem, und sei H das allgemeine Halteproblem. Dann ist CLIQUE $\leq H$.	☐☐
6. Sei $G = (V,E)$ ein ungerichteter Graph, und sei $V' \subseteq V$ gegeben. Es gibt keine Registermaschine, die in polynomieller Zeit entscheidet, ob die Knoten von V' eine CLIQUE bilden.	☐☐
7. Die Sprache $L = \{\langle M\rangle \mid \langle M\rangle$ ist die Gödelnummer einer deterministischen 1-Band-Turingmaschine M, die gestartet mit leerem Band nach höchstens $\lvert\langle M\rangle\rvert^2$ Schritten hält$\}$ ist unentscheidbar.	☐☐

Der **Lese- und Versteh-Aufwand** der beiden folgenden Aufgaben ist definitv größer als der Aufwand zu ihrer Lösung ... ☺
Schauen Sie sich auch **vorher** das am Ende des Blatts präsentierte Beispiel zur Reduktion SAT $\leq_{\mathrm{p}}$ BP an. Es ist dasselbe wie das in der Vorlesung durchgeführte Beispiel in aller Ausführlichkeit.

AUFGABE 42 (4 Punkte)**:**
[++,⋆⋆] Das Problem *Binary Programming* BP ist die folgende Menge:

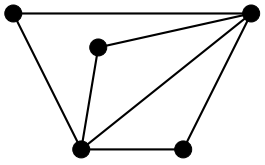
$$\text{BP} = \{\langle A, \vec{b}\rangle \mid A \text{ ist eine } m \times n\text{-Matrix mit ganzzahligen Einträgen, } \vec{b} \text{ ist ein Vektor mit } m \text{ ganzzahligen Einträgen, und es gibt einen 0-1-Vektor } \vec{x} \in \{0,1\}^n \text{ mit } A \cdot \vec{x} \leq \vec{b}\}$$

Das Problem IS (*Independent Set Problem*, vgl. Aufgabe 30 auf Blatt 6, am besten dort noch einmal nachlesen) ist die folgende, diesmal sehr kompakt beschriebene Menge:

$$\text{IS} = \{\langle G, k\rangle \mid G = (V,E) \text{ ist ein Graph, } k \in \mathbb{N}, \exists U \subseteq V, |U| = k\text{: } \forall u, v \in U : \{u,v\} \notin E\}$$

(a) Zeigen Sie: IS $\leq_{\text{p}}$ BP

(b) Stellen Sie $\langle A, \vec{b}\rangle$ für den rechts dargestellten Graphen G und $k = 3$ auf und geben Sie eine Lösung für $\vec{x}$ an.



G

Hinweise (vgl. auch die Beispielkonstruktion am Ende **dieses** Blattes): (i) Die Reduktion besteht darin, daß Sie zum Graphen $G = (V,E)$ und der Zahl k ein lineares Ungleichungssystem bestimmen, das genau dann 0-1-Lösungsvektoren hat, wenn G eine unabhängige Menge U aus k Knoten besitzt. Interpretieren Sie dazu den 0-1-Vektor $\vec{x} = (x_1, \ldots, x_{|V|})$ folgendermaßen: $x_i = 1 \iff$ Knoten u_i ist einer der Knoten in U. Mit dieser Interpretation im Hinterkopf: Was muß für jede Kante $\{u_i, u_j\} \in E$ gelten?

(ii) $\sum_{i=1}^{n} x_i = k$ kann man durch zwei Bedingungen beschreiben: $\sum_{i=1}^{n} x_i \leq k$ und $\sum_{i=1}^{n} -x_i \leq -k$

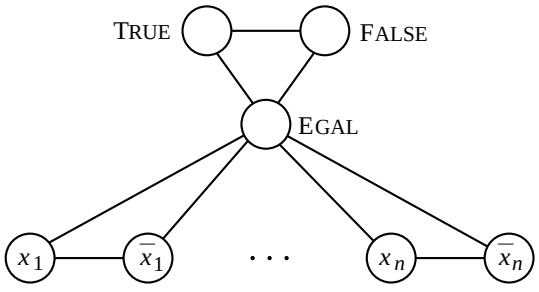
AUFGABE 43 (8 Punkte)**:**
[+++,⋆] *Die folgende Aufgabe ist eine sehr schöne Bastelaufgabe. Sie zeigt, wie man mit Hilfe von kleinen Graphbausteinen regelrecht „programmieren" kann. Und es gilt: Viel Text, aber vergleichsweise einfach!*

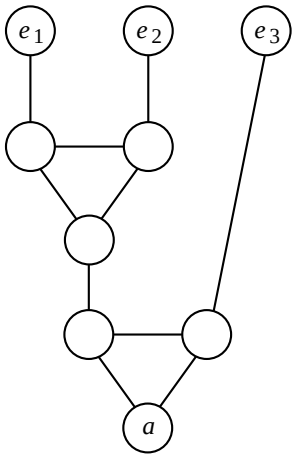
Sei $k \in \mathbb{N}$. Eine *k-Färbung* eines Graphen $G = (V,E)$ ist eine Abbildung $c : V \to \{1, \ldots, k\}$, so daß für alle Kanten $\{u,v\} \in E$ gilt: $c(u) \neq c(v)$.
Das Dreifärbungsproblem ist die Sprache

$$\text{3COL} = \{\langle G\rangle \mid G \text{ ist 3-färbbarer Graph}\} \ .$$

Wir wollen in dieser Aufgabe zeigen: 3SAT $\leq_{\text{p}}$ 3COL. Da offensichtlich 3COL $\in$ NP ist, ist 3COL also NP-vollständig.

Betrachten Sie die beiden dargestellten Graphen, die als Bausteine (in der Fachliteratur *gadgets* genannt) in der Konstruktion eines größeren Graphen benutzt werden sollen.





Beachten Sie, daß im Graphen, der das Ergebnis der Reduktion sein muß, alle Knoten ungefärbt sind, also keine „Vorfärbung“ von Knoten gegeben ist bzw. gegeben werden kann. D. h., bei der Korrektheitsbegründung färben Sie den links dargestellten Graphen und nennen dann (!) die Farbe des Knotens oben links TRUE, usw.

(a) Sie haben also die drei Farben $\{\text{TRUE}, \text{FALSE}, \text{EGAL}\}$. Was stellt der linke Graph sicher?

(b) Betrachten Sie nun den rechten Graphen. Welche Farbe bekommt „Ausgang“ a, wenn alle drei „Eingänge“ e_1, e_2 und e_3 die Farbe FALSE haben? Wenn mindestens einer der Eingänge die Farbe TRUE hat, kann man dann die Farbe TRUE auf den Ausgang bringen?

(c) Benutzen Sie die beiden Bausteine, um zur KNF Φ mit Klauseln der Länge 3 einen Graphen G_Φ zu konstruieren, so daß G_Φ genau dann 3-färbbar ist, wenn Φ erfüllbar ist (denken Sie daran, daß sie eine „genau dann, wenn“-Beziehung zeigen müssen).

Hinweis: Benutzen Sie für jede Klausel eine Kopie des rechten Bausteins. Legen Sie die Eingänge und Ausgänge auf Knoten des linken Graphen. Z. B. sollten *alle* Ausgänge auf dem Knoten TRUE liegen (Warum?).

Führen Sie ihre Konstruktion für $\Phi = (x_1 \vee \bar{x}_2 \vee x_3) \wedge (\bar{x}_1 \vee \bar{x}_2 \vee \bar{x}_3)$ und eine erfüllende Belegung durch.

(d) Φ möge eine KNF über n Variablen sein und aus t Klauseln bestehen. Wieviele Knoten und Kanten hat G_Φ und wie groß ist der Grad?

Zur Erinnerung: Sei $G = (V, E)$ ein Graph. Der Grad $\delta_G(v)$ eines Knotens $v \in V$ ist die Anzahl der Kanten, die er berührt (Fachsprache: zu denen er *inzident* ist). Der Grad $\Delta(G)$ des Graphen ist der größte vorkommende Knotengrad, also $\Delta(G) = \max\{\delta_G(v) \mid v \in V\}$.

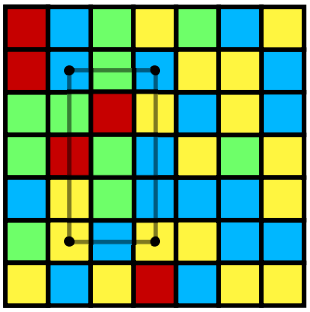
AUFGABE 44 (8 **Bonus**- Punkte)**:**
$[+++, \star\star\star]$ **Abgabe bis 15.01.19**
In dem in Aufgabe 33 auf Blatt 6 erwähnten Aufsatz schreibt Martin Grötschel über das NP-schwere TSP:

> Exakte und approximative Lösungsverfahren werden an Beispielen skizziert, und es wird angedeutet, dass man, obwohl TSPs zu den theoretisch schweren Problemen zählen, in der Praxis TSPs von atemberaubender Größe optimal lösen kann.

Ziel dieser Aufgabe ist es, ein ziemlich **kleines** Problem kennenzulernen, das sich gegen Holzhammeransätze hartnäckig zu wehren versteht, die sog. 17×17 *Challenge*.

Das *Grid Coloring Problem* GCP kann sehr einfach beschrieben werden: „The $n \times n$ grid is 4-colorable if there is a way to 4-color the vertices of the $n \times n$ grid so that there is no rectangle with all four corners the same color.“ Wir schreiben dann: GCP $:= \{n \mid$ das $n \times n$-Gitter ist 4-färbbar$\}$.

Z. B. ist $7 \in$ GCP, da das 7×7-Gitter 4-färbbar ist, wie die folgende Abbildung links zeigt:



```
1324234
1323443
2214343
2123424
3423334
2434433
4341344
```

Eines der Rechtecke ist markiert, es geht bei diesem Problem nur um die Farben in den vier Ecken der Rechtecke. Es müssen mindestens zwei verschiedene Farben sein. Eine mögliche Kodierung dieser Färbung sehen Sie rechts.
Einfach mit Computer-Power zu zeigen ist, daß $\{2,\ldots,16\} \subseteq \text{GCP}$, und daß $19 \notin \text{GCP}$ und damit $n \notin \text{GCP}$ für alle $n \geq 19$ kann man durch Nachdenken beweisen. **Die Fälle *n*=17 und *n*=18 waren lange unbekannt.** Erst seit Februar 2012 weiß man, daß $18 \in \text{GCP}$ (und damit auch $17 \in \text{GCP}$) (veröffentlicht in: B. Steinbach, C. Posthoff. *Extremely Complex 4-Colored Rectangle-Free Grids Solution of Open Multiple-Valued Problems*. doi:10.1109/ISMVL.2012.12). Nachdenken und die computergestützte Suche nach Symmetrien führte erst zum Erfolg.
GCP ist also eine endliche Menge.

(a) Modellieren Sie die Aufgabe, eine 4-Färbung für das $n \times n$-Gitter zu berechnen, durch eine KNF Φ_n, so daß gilt:

$$n \in \text{GCP} \iff \Phi_n \in \text{SAT}$$

Aus einer erfüllenden Belegung soll also eine Färbung abgeleitet werden können und umgekehrt. Mit anderen Worten: Zeigen Sie $\text{GCP} \leq_\text{p} \text{SAT}$. Dabei sollen Sie nicht einfach so fest verdrahtete Lösungen in die KNF hineinkodieren, sondern Φ_n soll die Problembeschreibung widerspiegeln!

Wieviele Variablen benötigen Sie? Geben Sie $\texttt{size}(\Phi_n)$ an.

(b) Implementieren Sie ein Programm, das für $n = 8$, $n = 10$ und $n = 16$ Färbungen berechnet (wie in (a): keine fest verdrahteten Lösungen in das Programm einbauen, sondern Lösungen wirklich berechnen!). Und wie ist es mit 17 und 18 ?

Auf der Webseite der Vorlesung finden Sie die Datei `GCP.txt`, in die sie Ihre mutmaßlichen Lösungen eingeben können. Schicken Sie die Dateien (Lösung und Programm) per Email an Alexander Raß (`Alexander.Rass@fau.de`) mit dem Betreff: `GridColoring` (kein Blank)

Die Datei enthält noch einige Erklärungen und Anweisungen, lesen Sie diese bitte. Insbesondere sei noch einmal betont: Keine „fest verdrahteten“ Lösungen.

Zusatzinformation zu AUFGABE 35:
In Aufgabe 35(b) auf Blatt 7 hatten wir ja bereits das 10. Hilbertsche Problem kurz erwähnt. Insbesondere ging es um die Sprache DIOGL $:= \{\langle p\rangle \mid p$ ist Polynom (auch über mehreren Variablen) mit ganzzahligen Koeffizienten und ganzzahligen Nullstellen$\}$ der Diophantischen Gleichungen. Yuri Matiyasevich konnte 1970 zeigen, daß DIOGL unentscheidbar ist (konkret zeigte er: $H \leq$ DIOGL). In Aufgabe 35 hatten wir einen nichtdeterministischen Algorithmus angeschaut, der eine Belegung der Variablen errät und dann überprüft, ob dies eine Nullstelle ist. Dort war die Frage, ob dies nicht beweist, daß DIOGL in NP liegt. Jetzt, da wir wissen, daß alle Probleme in NP entscheidbar sind, ist die Antwort NEIN ☺, aber das ist natürlich nicht das, was dort in der Aufgabe gefunden werden sollte. Dort ging es darum, daß trotz kleinem Polynom niemand garantiert, daß die ganzzahligen Nullstellen (wenn sie existieren), auch klein sind.

Daß das auch wirklich nicht der Fall sein muß, zeigt das folgende, vielleicht beeindruckende Beispiel.

Sei

$$p_k(x,y) = x^2 - 5^{2k+1} \cdot y^2 + 1 \quad .$$

$p_k(x,y)$ kann mit $n = \Theta(k)$ Bits kodiert werden, was damit die Eingabelänge ist.
Wir fragen für $k \in \mathbb{N}_0$ nach *ganzzahligen* x_k und y_k mit $p(x_k,y_k) = 0$. p hat derartige Nullstellen, und es ist bekannt [1], daß für das kleinste Paar (x_k,y_k) gilt:

$$x_k + y_k \cdot 5^k \cdot \sqrt{5} = (2+\sqrt{5})^{5^k} \quad .$$

D. h. das Hinschreiben einer Lösung benötigt $\Omega(5^k) = \Omega(2^n)$ Bits, was damit exponentiell in der Eingabelänge ist.

Beispiele:

$k=0$: $p_0(x,y) = x^2 - 5y^2 + 1$, $x_0 = 2$, $y_0 = 1$
$k=1$: $p_1(x,y) = x^2 - 125y^2 + 1$, $x_1 = 682$, $y_1 = 61$
$k=2$: $p_2(x,y) = x^2 - 3125y^2 + 1$, $x_2 = 2.360.712.083.917.682$, $y_2 = 42.229.701.559.561$
$k=3$: $p_3(x,y) = x^2 - 78125y^2 + 1$,
$x_3 = 1173100170951472281886354528716834541556511612763081435530825222412631169342682$,
$y_3 = 4197010762662585870122528925098478873640815186220492590579494904540039297061$

[1] J. C. Lagarias, On the computational complexity of determining the solvability or unsolvability of the equation $X^2 - DY^2 = -1$. *Transactions of the American Mathematical Society* 260 (1980) 485–508. doi:10.1090/S0002-9947-1980-0574794-0

Da es vielleicht doch einige Gewöhnungsschwierigkeiten bei der Beschreibung kombinatorischer „Bastel"-Probleme durch „binäre Programme" (vgl. auch Aufgabe 42) gibt, möchten wir hier einmal die gesamte Konstruktion aus der Reduktion von SAT auf BP an einem Beispiel vorführen. Dabei ist BP die Sprache der „binären Programme":

$$\text{BP} = \{\langle A, \vec{b}\rangle \mid A \text{ ist eine } (n \times m)\text{-Matrix mit ganzzahligen Einträgen, } \vec{b} \text{ ist ein Vektor mit } m \text{ ganzzahligen Einträgen, und es gibt einen 0-1-Vektor } \vec{x} \in \{0,1\}^n \text{ mit } A \cdot \vec{x} \leq \vec{b}\}$$

„Binär" deswegen, weil die Einträge des unbekannten(!) Vektors $\vec{x}$ nur aus $\{0,1\}$ sein dürfen. $\langle A, \vec{b}\rangle$ nennt man ein *binäres Programm*. Sei

$$\Phi = (x_1 \vee x_2) \wedge (x_1 \vee \bar{x}_2) \wedge (\bar{x}_1 \vee \bar{x}_2)$$

eine KNF über $v = 2$ Variablen und mit $k = 3$ Klauseln. Die einzige erfüllende Belegung ist $x_1 = \text{TRUE}$ und $x_2 = \text{FALSE}$. Gemäß der Reduktion der Vorlesung wird diese KNF durch das links dargestellte Ungleichungssystem beschrieben. Rechts sehen Sie die äquivalente Darstellung, bei der nur noch die $\leq$-Relation verwendet wird.

$$\left|\begin{array}{rcr} z_1 + z_1' & = & 1 \\ z_2 + z_2' & = & 1 \\ z_1 + z_2 & \geq & 1 \\ z_1 + z_2' & \geq & 1 \\ z_1' + z_2' & \geq & 1 \end{array}\right| \iff \left|\begin{array}{rcr} -1 \cdot z_1 - 1 \cdot z_1' & \leq & -1 \\ 1 \cdot z_1 + 1 \cdot z_1' & \leq & 1 \\ -1 \cdot z_2 - 1 \cdot z_2' & \leq & -1 \\ 1 \cdot z_2 + 1 \cdot z_2' & \leq & 1 \\ -1 \cdot z_1 - 1 \cdot z_2 & \leq & -1 \\ -1 \cdot z_1 - 1 \cdot z_2' & \leq & -1 \\ -1 \cdot z_1' - 1 \cdot z_2' & \leq & -1 \end{array}\right|$$

Eine *binäre* Lösung ist $z_1 = 1$, $z_1' = 0$, $z_2 = 0$, $z_2' = 1$, was man auch als $\vec{x}^{\text{T}} = (1,0,0,1)$ schreiben kann (T steht für „transponiert"). Machen Sie sich den Zusammenhang zwischen der erfüllenden Belegung und dem Lösungsvektor klar! Machen Sie sich auch klar (indem Sie z. B. eine Klausel wegnehmen), daß das resultierende Ungleichungssystem mehr als eine binäre Lösung haben kann, und wie dann aus einer solchen Lösung eine erfüllende Belegung bestimmt werden kann.
Das Ungleichungssystem muß nun noch in die „richtige" Matrix-Form gebracht werden:

$$\underbrace{\begin{pmatrix} -1 & -1 & 0 & 0 \\ 1 & 1 & 0 & 0 \\ 0 & 0 & -1 & -1 \\ 0 & 0 & 1 & 1 \\ -1 & 0 & -1 & 0 \\ -1 & 0 & 0 & -1 \\ 0 & -1 & 0 & -1 \end{pmatrix}}_{A_\Phi} \cdot \underbrace{\begin{pmatrix} z_1 \\ z_1' \\ z_2 \\ z_2' \end{pmatrix}}_{\vec{x}} \leq \underbrace{\begin{pmatrix} -1 \\ 1 \\ -1 \\ 1 \\ -1 \\ -1 \\ -1 \end{pmatrix}}_{\vec{b}_\Phi}$$

A_Φ ist also eine (7×4)-Matrix mit Einträgen aus $\{-1,0,1\}$, $\vec{x}$ eine (4×1)-„Matrix" (also ein 4-Vektor) und $\vec{b}_\Phi$ ein 7-Vektor mit Einträgen aus $\{-1,1\}$. Und nach allem Gesagten gilt: $\langle A_\Phi, \vec{b}_\Phi\rangle \in \text{BP}$. Allgemein ist A_Φ eine $(2v+k) \times (2v)$-Matrix und $\vec{b}_\Phi$ ein $(2v+k)$-Vektor, also alles zusammen polynomiell groß in $|\langle\Phi\rangle|$.

Firmen wie die französische ILOG (http://www.ilog.com/), die im Januar 2009 von IBM gekauft worden ist, mit einem Jahresumsatz von über 133 Millionen US-Dollar entwickeln sog. *Solver* für solche und verwandte Probleme. Das Flaggschiff von ILOG ist das bekannte Programm cplex (offizieler Name: IBM ILOG CPLEX). Dieser Firmenbereich stellt konsequent Leute ein, die einen deutlich erkennbaren Hintergrund in der Theoretischen Informatik haben, und nach dem Kapitel über die NP-Vollständigkeit wird Ihnen vielleicht klar, warum das so ist.

Das BFS-Team wünscht allen Studentinnen und Studenten der Vorlesung „Berechenbarkeit und Formale Sprachen" eine schöne Weihnachtszeit, einen guten Rutsch und ein erfolgreiches Jahr 2019.