Prof. T. Nipkow
M. Eberl, J. Rädle, S. Wimmer
Technische Universität München
HANDSCHRIFTLICHE ABGABE **8 Punkte**
Abgabefrist: Montag, 22.07.2019, 14 Uhr

Einführung in die theoretische Informatik

Sommersemester 2019 – Bonusblatt Lösungsskizze 12

Handschriftliche Abgabe

Formale Kriterien zu handschriftlichen Abgaben entnehmen Sie bitte der Website `https://www21.in.tum.de/teaching/theo/SS19`.

Bonusblatt

Die Aufgaben auf diesem Blatt sind *Bonusaufgaben*. Die Punkte zählen wie normale Hausaufgabenpunkte; es ist aber auch ohne Bearbeitung dieser Aufgaben möglich, die Maximalpunktzahl von 60 Punkten zu erreichen. Es ist nicht möglich, mehr als 60 Punkte insgesamt zu erreichen, da die Punktezahl bei 60 abgeschnitten wird.

AUFGABE 12.1. (*Wahr oder falsch?*) 1.5 Punkte

Im folgenden seien f und g Funktionen $\Sigma^* \to \Sigma^*$. Das Symbol $\circ$ steht für die Komposition zweier Funktionen, d.h. $(f \circ g)(x) = f(g(x))$.

Einen Punkt gibt es grundsätzlich *nur* mit einer korrekten Begründung.

(a) Wenn f und g berechenbare Funktionen sind, ist auch $f \circ g$ berechenbar.

(b) Wenn g und $f \circ g$ berechenbare Funktionen sind, ist auch f berechenbar.

(c) Sei $A \in \mathsf{P}$ und sei M eine deterministische Turingmaschine mit $\mathsf{L}(M) = A$. Dann ist $\text{time}_M(x)$ polynomiell beschränkt in $|x|$ (d.h. es gibt ein Polynom $p : \mathbb{N} \to \mathbb{N}$ mit $\text{time}_M(x) \leq p(|x|)$ für alle $x \in \Sigma^*$).

Lösungsskizze

(a) Korrekt. Wir können die Turingmaschinen für g und f hintereinanderschalten.

(b) Falsch. Beispiel: Sei f irgendeine nicht berechenbare Funktion (z.B. χ_{H_0}) und $g(x) = \varepsilon$. Dann ist $f \circ g$ eine konstante Funktion und damit berechenbar.

(c) Falsch. $A \in \mathsf{P}$ heißt nur, dass es eine DTM *gibt*, für die das der Fall ist. Es heißt *nicht*, dass das für alle DTMs gilt, die A entscheiden. Wir können z.B. eine beliebige TM für A nehmen und vor den Startzustand neue Zustände einfügen, die $2^{|w|}$ Schritte lang nichts tun und dann in den Startzustand von M übergehen. Diese neue Maschine akzeptiert offensichtlich ebenfalls A, aber definitiv nicht in Zeit polynomiell in $|w|$.

AUFGABE 12.2. (k-*Färbbarkeit*) 1 Punkt

Für $k \in \mathbb{N}$ sei k-COL wie in der Vorlesung die Menge der Graphen, deren Knoten sich mit k verschiedenen Farben einfärben lassen, sodass keine zwei adjazenten Knoten die gleiche Farbe erhalten. Formal: Gegeben einen Graphen $G = (V, E)$, gibt es eine Funktion $col : V \to \{1, \ldots, k\}$ sodass für alle $\{u, v\} \in E$ gilt $col(u) \neq col(v)$?

Zeigen Sie: Für jedes feste $k \geq 3$ gilt $\mathsf{3\text{-}COL} \leq_\mathsf{p} k\text{-}\mathsf{COL}$.

Lösungsskizze

Sei $G = (V, E)$ ein Graph. Wir bilden diesen auf einen neuen Graphen $f(G) = (V \uplus V', E \cup \{\{u, v\} \mid u \in V', v \in V \cup V'\})$ mit $V' = \{1, \ldots, k-3\}$.

In anderen Worten: Für jede "überschüssige" Farbe ($k-3$ Stück) fügen wir einen Dummy-Knoten hinzu, der mit allen anderen Knoten verbunden ist und daher eine Farbe komplett "blockiert" (da kein anderer Knoten die gleiche Farbe bekommen kann). Dadurch erzwingen wir, dass $k-3$ der k Farben nur dafür verwendet werden, die Dummy-Knoten zu färben und für die ursprünglichen Knoten aus V nur noch 3 Farben zur Verfügung stehen.

Falls $G \in \mathsf{3\text{-}COL}$ ist, ist offensichtlich auch $f(G) \in k\text{-}\mathsf{COL}$, da wir jede Färbung von G mit höchstens 3 Farben zu einer Färbung von $f(G)$ mit höchstens k Farben erweitern können, indem wir $k-3$ der unbenutzten Farben beliebig auf die Dummy-Knoten verteilen.

In der Gegenrichtung folgt aus $f(G) \in k\text{-}\mathsf{COL}$ auch $G \in \mathsf{3\text{-}COL}$: Seien col eine k-Färbung von $f(G)$. Sei $C_1 = col(V')$ die Menge der Farben, die in V' vorkommen und $C_2 = col(V)$ die restlichen Farben. Da die Knoten aus V' mit

allen anderen verbunden sind, muss *col* injektiv auf V' sein und $C_1 \cap C_2 = \emptyset$. Es folgt $|C_1| = |V'| = k-3$ und mit $k \geq |\text{Bild}(col)| = |C_1 \cup C_2| = |C_1| + |C_2| = k - 3 + |C_2|$ folgt $|C_2| \leq 3$. Folglich ist die Einschränkung von *col* auf V eine 3-Färbung von G.

AUFGABE 12.3. (*Reguläre Ausdrücke mit Schnitt*) 1 + 0,5 + 1.5 + 1.5 + 1 Punkte

In dieser Aufgabe betrachten wir die regulären Ausdrücke über einem Alphabet Σ ohne den "Kleene-Stern"-Operator, aber dafür mit einem Schnitt-Operator $\cap$. Die Menge dieser Ausdrücke nennen wir $\text{RE}(\Sigma, |, \cdot, \cap)$.

Für diese Klasse von regulären Ausdrücken definieren wir die folgenden Probleme:

Wortproblem: Gegeben einen regulären Ausdruck $r \in \text{RE}(\Sigma, |, \cdot, \cap)$ und ein Wort $w \in \Sigma^*$, ist $w \in \mathsf{L}(r)$?

Nicht-Leerheitsproblem: Gegeben einen regulären Ausdruck $r \in \text{RE}(\Sigma, |, \cdot, \cap)$, ist $\mathsf{L}(r) \neq \emptyset$?

Die regulären Ausdrücke sind dabei auf die übliche Weise als Wörter kodiert (über dem Alphabet $\Sigma \cup \{(,), |, \cap, \boldsymbol{\varepsilon}, \boldsymbol{\emptyset}\}$). Beachten Sie, dass beim Wortproblem sowohl der reguläre Ausdruck als auch das Wort die Eingabe bilden.

(a) Begründen Sie, dass das Wortproblem für $\text{RE}(\Sigma, |, \cdot, \cap)$ in NP ist.

(b) Begründen Sie, dass das Nicht-Leerheitsproblem für $\text{RE}(\Sigma, |, \cdot, \cap)$ in NP ist (sie dürfen hierbei Aufgabe (a) ohne Beweis verwenden).

(c) Zeigen Sie, dass das Nicht-Leerheitsproblem für $\text{RE}(\Sigma, |, \cdot, \cap)$ mit $\Sigma = \{0, 1\}$ NP-schwer ist.

Tipp: Reduzieren Sie SAT auf das Nicht-Leerheitsproblem.

(d) Funktionieren ihre Konstruktionen aus den Aufgaben (a) bis (c) auch für $\text{RE}(\Sigma, |, \cdot, \cap, ^*)$, also wenn man zusätzlich den Kleene-Stern erlaubt?

Hinweis: Ihre Begründungen können hier relativ kurz sein. Insb. wenn ein Argument nicht mehr funktioniert, reicht es, zu erklären, warum es zumindest nicht offensichtlich ist, ob das Argument noch funktioniert. Für ein korrektes Beispiel von Probleminstanzen, wo die Konstruktion nicht mehr funktioniert sowie für einen Beweis, dass sie nicht funktioniert, vergeben wir aber u.U. sogar zusätzliche Bonuspunkte.

(e) **Bonus-Knobelaufgabe (schwierig):** Zeigen Sie: Das Wortproblem für $\text{RE}(\Sigma, |, \cdot, \cap, ^*)$ ist sogar in P.

Es reicht hierbei, ein entsprechendes Verfahren anzugeben, mit einer Begründung für die Korrektheit und polynomielle Laufzeit, aber ohne rigorosen Beweis.

Hinweis: Die Lösung ist recht kurz und benötigt kein Wissen, das über die Vorlesung hinausgeht. Es kann dennoch schwierig sein, darauf zu kommen. Sie sollten daher selbst abwägen, ob Sie die Zeit und Energie haben, sich an dieser Aufgabe zu versuchen.

Lösungsskizze

Hinweis: Den RE einfach in einem NFA umzuwandeln (mit Produktkonstruktion für $\cap$) funktioniert grundsätzlich *nicht*. Das Verfahren aus der Vorlesung liefert für einen RE der Größe n einen Automaten mit $\Theta(n)$ Zuständen; bei der Produktkonstruktion bekommt man aber für zwei Automaten mit m und n Zuständen im schlimmsten Fall $m \cdot n$ Zustände. Wenn man nun z.B. reguläre Ausdrücke der Form $r_1 \cap \ldots \cap r_k$ betrachtet wobei die r_i jeweils die Größe k haben, muss man $k-1$ mal die Produktkonstruktion machen und die bekannte worst-case-Abschätzung liefert nur, dass der resultierende NFA höchstens etwa die Größe k^k hat, was nicht polynomiell ist. Man müsste also begründen, warum dieser worst case nicht auftreten kann.

Dass der Ansatz "RE in NFA umwandeln" vermutlich grundsätzlich nicht funktionieren kann, kann man auch folgendermaßen sehen: Wenn man (ohne Verwendung von Nichtdeterminismus) den RE in einen NFA umwandelt, kann man danach die Leerheit in polynomieller Zeit prüfen (da dies für NFAs einfach ist). Damit wäre das Leerheitsproblem für $\text{RE}(\Sigma, |, \cdot, \cap)$ in P. Laut Aufgabe c) ist es außerdem NP-schwer, und damit wäre dann P = NP (was für eine Hausaufgabe mit einem Punkt doch etwas schwierig wäre).

Wir verlangen von Ihnen keine derart detaillierte Analyse. Es wird jedoch erwartet, dass Sie sehen, dass durch die iterierte Produktkonstruktion nicht klar ist, ob der resultierende Automat noch polynomielle Größe hat und dass daher RE $\to$ NFA hier nicht funktioniert. Daher gab es für solche Lösungen keine Punkte bzw. starken Punktabzug.

Nun zur Lösung der Aufgaben an sich:

(a) Wir definieren eine rekursive Prozedur $P(r, w)$, die $w \in \mathsf{L}(r)$ entscheidet, folgendermaßen:

$$
\begin{aligned}
P(r \cap s, w)&\text{: gib } P(r,w) \wedge P(s,w) \text{ zurück}\\
P(r \mid s, w)&\text{: gib } P(r,w) \vee P(s,w) \text{ zurück}\\
P(r\, s, w)&\text{: wähle nichtdeterministisch } i \in \{0, \ldots, |w|\},\ \text{gib zurück } P(r, w_{1\ldots i}) \wedge P(s, w_{(i+1)\ldots|w|})
\end{aligned}
$$

Die anderen Fälle sind trivial. Da der "Aufteilungspunkt" i immer $\leq$ der Länge des Eingabeworts ist und der Algorithmus sonst eine einfache strukturelle Rekursion über r, ist dieser Algorithmus nichtdeterministisch

polynomiell in seiner Eingabe.

(b) Da die regulären Ausdrücke sternfrei sind, können die akzeptierten Wörter maximal so lang sein wie der Ausdruck selbst (einfache strukturelle Induktion). Folglich können wir für Eingabe $r \in \mathrm{RE}(\Sigma, |, \cdot, \cap, ^*)$ einfach ein Wort $w \in \Sigma^*$ mit $|w| \leq |r|$ raten und dann die Konstruktion aus Aufgabe a) verwenden, um $w \in \mathsf{L}(r)$ zu entscheiden.

(c) Wir zeigen die NP-Schwere des Problems, indem wir das NP-vollständige Problem SAT auf unser Problem reduzieren. Sei φ eine aussagenlogische Formel in KNF mit n Variablen $x_1, \ldots, x_n$.

Die Idee der Reduktion ist, dass eine Belegung der $x_1, \ldots, x_n$ als Wort aus $\{0,1\}^n$ geschrieben werden kann, sodass wir für eine korrekte Reduktion nur einen entsprechenden regulären Ausdruck für die erfüllenden Belegungen bauen müssen.

Die Eingabe von SAT ist eine Formel in KNF (d.h. eine Konjunktion von Klauseln, wobei jede Klausel eine Disjunktion von Literalen der Form x_i oder $\neg x_i$ ist). Hierzu bauen wir eine Reduktionsfunktion f folgendermaßen:

- $f(x_i) = \underbrace{(0|1)\ldots(0|1)}_{i-1 \text{ mal}} 1 \underbrace{(0|1)\ldots(0|1)}_{n-i \text{ mal}}$ für jedes Literal x_i
- $f(\neg x_i) = \underbrace{(0|1)\ldots(0|1)}_{i-1 \text{ mal}} 0 \underbrace{(0|1)\ldots(0|1)}_{n-i \text{ mal}}$ für jedes Literal $\neg x_i$
- $f(l_1 \vee \ldots \vee l_k) = f(l_1) \mid \ldots \mid f(l_k)$ für jede Klausel $l_1 \vee \ldots \vee l_k$
- $f(c_1 \wedge \ldots \wedge c_m) = f(c_1) \cap \ldots \cap f(c_m)$ für die gesamte Formel in KNF

Die Größe des Ausgabe-Ausdrucks ist $O(mn)$, wobei m die Anzahl der Literale in φ ist. Damit ist die Reduktionsfunktion offensichtlich total und polynomiell.

(d) (a) funktioniert genauso, nur dass man im Fall r^* eine Aufteilung des Wortes w in höchstens $|w|$ nicht-leere Teilwörter raten muss. So eine Aufteilung lässt sich in $O(|w|)$ Zeichen repräsentieren, sodass der Algorithmus weiterhin nichtdeterministisch polynomiell bleibt.

(b) funktioniert in dieser Form nicht, da wir nun a priori keine obere Schranke mehr für das kürzeste Wort in $\mathsf{L}(r)$ haben, sodass nicht klar ist, wie lang das zu ratende Wort sein kann (siehe unten für mehr Details).

(c) funktioniert, da $\mathrm{RE}(\Sigma, |, \cdot, \cap, ^*)$ eine Obermenge von $\mathrm{RE}(\Sigma, |, \cdot, \cap)$ ist, sodass wir das Nicht-Leerheitsproblem für letztere trivial (mit $f(w) = w$) auf das für erstere polynomiell reduzieren können. (NB: man muss bei der Reduktion streng genommen noch aufpassen, was mit Eingaben passiert, die keinen gültigen RE kodieren)

(e) Gegeben ein Wort w mit $|w| = n$ und einen regulären Ausdruck r definieren wir die folgende Relation $R(r) \subseteq \{0 \ldots n\}^2$:

$$(i,j) \in R(r) \longleftrightarrow w_{i+1} \ldots w_j \in \mathsf{L}(r)$$

Die Relation enthält also alle Paare von Start- und Endindizes von w, sodass das entsprechende Teilwort in $\mathsf{L}(r)$ ist (wobei für ein leeres Teilwort $i = j$ ist, d.h. wir haben immer $i \leq j$). Wenn wir diese Relation berechnen können, können wir auch das Wortproblem $w \in \mathsf{L}(r)$ lösen, indem wir $(0, n) \in R(r)$ testen.

Die Relation lässt sich für fixes w per Rekursion über r berechnen:

$$\begin{aligned}
R(\emptyset) &= \emptyset \\
R(\varepsilon) &= \{(i,i) \mid 0 \leq i \leq n\} \\
R(x) &= \{(i-1,i) \mid 1 \leq i \leq n \wedge w_i = x\} \\
R(r \mid s) &= R(r) \cup R(s) \\
R(r \cap s) &= R(r) \cap R(s) \\
R(r\, s) &= R(r) \circ R(s) \\
R(r^*) &= R(r)^*
\end{aligned}$$

Hierbei steht $\circ$ für Komposition von Relationen und * für die reflexive transitive Hülle einer Relation. Beides lässt sich offensichtlich in polynomieller Zeit berechnen (durch Wahl einer geeigneten Repräsentation für die Relationen lässt sich die gesamte Berechnung in $O(|r|\, n^3)$ Schritten durchführen, unter der Annahme, dass elementare Operationen wie Array-Zugriffe konstante Zeit benötigen).

Alternativ kann man auch mit einer sehr ähnlichen Rekursion die Menge

$$A(r) := \{u \in \mathsf{L}(r) \mid u \text{ ist Teilwort von } w\}$$

berechnen, da $|A(r)| \leq \frac{1}{2} n(n+1) \in O(n^2)$ ist, und dann $w \in A(r)$ prüfen.

Ergänzung zu d): (Nur relevant für den Bonus)
In der Tat ist das kürzeste Wort, das von einem regulären Ausdruck aus $\mathrm{RE}(\Sigma, |, \cdot, \cap, ^*)$ akzeptiert wird i.A. von exponentieller Länge. Man kann z.B. für das Alphabet $\Sigma = \{a\}$ für jede Zahl n den regulären Ausdruck

$$r_n := aa^* \cap \bigcap_{i=1}^{n} (a^{p_i})^* = aa^* \cap (aa)^* \cap (aaa)^* \cap (aaaaa)^* \cap \dots$$

definieren (wobei p_i für die i-te Primzahl steht). Dann gilt:

- Das kürzeste Wort in $\mathsf{L}(r_n)$ hat die Länge $\mathrm{lcm}\{p_1, \dots, p_n\} = \prod_{i=1}^{n} p_i =: n\#$ (das sogenannte Primorial). Offensichtlich ist $p_i \geq 2$, d.h. $n\# \geq 2^n$ (genauer: $n\# \in e^{(1+o(1))n \ln n}$).
- $|r_n| = 3 + 4n + \sum_{i=1}^{n} p_i \leq 3 + 4n + np_n \in O(n^2 \log n)$ da $p_n \in O(n \log n)$.
- Es gibt offensichtlich kein k, sodass $2^n \in O((n^2 \log n)^k)$ wäre.

Es gibt also reguläre Ausdrücke (im Sinne der Aufgabe), für die das kürzestes erkannte Wort nicht polynomiell beschränkt bzgl. der Länge des Ausdrucks selbst ist. Damit kann der Ansatz "Wort raten" hier prinzipiell nicht funktionieren.

Man kann sogar zeigen, dass das Leerheitsproblem für $\mathrm{RE}(\Sigma, |, \cdot, \cap, ^*)$ PSPACE-vollständig ist, was es eher unwahrscheinlich macht, dass das Nicht-Leerheitsproblem in NP ist, da dann NP = PSPACE wäre und die meisten Experten auf dem Gebiet stark vermuten, dass NP eine echte Teilmenge von PSPACE ist.