Prof. T. Nipkow — Technische Universität München
K. Kappelmann, J. Rädle, L. Stevens — Lehrstuhl für Logik und Verifikation

ABGABE 06.07.2020, 23:59 (Σ 5,5 P.)

Einführung in die Theoretische Informatik

Sommersemester 2020 – Übungsblatt Lösungsskizze 10

AUFGABE 10.1. (*Die Korrespondenz hält sich in Grenzen*) 1,5 Punkte

Für Instanzen $(x_1, y_1), \dots, (x_m, y_m)$ des Post'schen Korrespondenzproblems bezeichnen wir eine Lösung $i_1, \dots, i_n$ als k-begrenzt, wenn für alle $j \leq n$ gilt:

$$\big|\big|x_{i_1} \dots x_{i_j}\big| - \big|y_{i_1} \dots y_{i_j}\big|\big| \leq k.$$

Zeigen Sie, dass für fixes k entscheidbar ist, ob eine PCP-Instanz eine k-begrenzte Lösung hat. **Hinweis:** Konstruieren Sie einen DFA über dem Alphabet $\Sigma = \{1, \dots, m\}$, der ein Wort $i_1 \dots i_n$ genau dann akzeptiert, wenn es eine k-begrenzte Lösung ist. Einen Korrektheitsbeweis für die Konstruktion ist nicht notwendig.

Lösungsskizze

Wir zeigen, dass die Lösungen einer Instanz des k-beschränkten-PCPs eine reguläre Sprache bilden. Schnitt und Leerheitsproblem sind auf regulären Sprachen entscheidbar. Deshalb können wir entscheiden, ob die Sprache ein nichtleeres Wort beziehungsweise eine Lösung der k-beschränkten-PCP-Instanz enthält.
Sei Σ' das Alphabet, über dem die PCP-Instanz definiert ist. Wir definieren

$$Q = \{(\varepsilon, y) \mid |y| \leq k\} \cup \{(x, \varepsilon) \mid |x| \leq k\} \cup \{*\}.$$

Die Anwendung einer Karte der PCP-Instanz auf einen solchen Zustand $\oplus_k : Q \times (\Sigma'^* \times \Sigma'^*) \to Q$ ist nun wie folgt definiert (wir schreiben die Karten für bessere Lesbarkeit senkrecht):

$$\begin{pmatrix} \varepsilon \\ w \end{pmatrix} \oplus_k \begin{pmatrix} x \\ y \end{pmatrix} = \begin{cases} \binom{\varepsilon}{y'} & \text{wenn } wy = xy' \text{ und } |y'| \leq k, \\ \binom{x'}{\varepsilon} & \text{wenn } x = wyx' \text{ und } |x'| \leq k, \\ * & \text{sonst,} \end{cases}$$

$$\begin{pmatrix} v \\ \varepsilon \end{pmatrix} \oplus_k \begin{pmatrix} x \\ y \end{pmatrix} = \begin{cases} \binom{x'}{\varepsilon} & \text{wenn } vx = yx' \text{ und } |x'| \leq k, \\ \binom{\varepsilon}{y'} & \text{wenn } y = vxy' \text{ und } |y'| \leq k, \\ * & \text{sonst,} \end{cases}$$

$$* \oplus_k \begin{pmatrix} x \\ y \end{pmatrix} = *.$$

So erhalten wir einen DFA $M = (Q, \Sigma, \delta, (\varepsilon, \varepsilon), \{(\varepsilon, \varepsilon)\})$ mit $\delta(q, i) = q \oplus_k \binom{x_i}{y_i}$. Nun hat das PCP-Problem $(x_1, y_1), \dots, (x_m, y_m)$ genau dann eine k-begrenzte Lösung, wenn M ein nichtleeres Wort akzeptiert. Dieses Problem ist entscheidbar.

AUFGABE 10.2. 2 Punkte

Sei $A \subseteq \Sigma^*$ eine Sprache in P. Zeigen Sie, dass die Sprache $\{w_1 \dots w_l \mid w_1, \dots, w_l \in A \wedge l \leq k\}$ für jede Konstante $k \in \mathbb{N}$ ebenfalls in P liegt.

Lösungsskizze

Wir lösen die Aufgabe mit dynamischer Programmierung. Für ein gegebenes Wort w und Obergrenze k ist dabei die Idee, sich für Präfixe von w zu merken ob sie in A^* liegen und in wie viele Teile man sie dazu mindestens zerlegen muss. Im folgenden Code speichern wir diese Information in der Liste L; in $L[n]$ steht also die minimale Anzahl an Teilworten in A in die man $w_1 \dots w_n$ zerlegen kann, bzw. ∞ falls dies nicht möglich ist.
In Pseudocode:

- **Eingabe**: Wort $w = w_1 \dots w_n$, k
- Lege eine Liste L der Länge $n + 1$ an, wobei anfangs $L[0] = 0$ und $L[i] = \infty$ für alle $i > 0$
- Für alle $i \in \{1, \dots, n\}$
  - Für alle $j \in \{0, \dots, n - 1\}$
    - Wenn $L[j] < \infty \wedge (w_j \dots w_{i-1}) \in A$, dann setze $L[i] = \min(A[j] + 1, A[i])$
- Gebe $A[n] \leq k$ zurück

Da $A \in \mathsf{P}$, können wir $w \in A$ in $\mathcal{O}(p(n))$ für ein gewisses Polynom p entscheiden. Damit läuft der obige Algorithmus in $\mathcal{O}(n^2 p(n))$, was wiederum polynomiell ist.
Eine andere Möglichkeit ist es, einfach alle möglichen Zerlegungen von $w = w_1 \dots w_n$ in höchstens k Teilwörter zu betrachten. Für die asymptotische Laufzeit reicht es aus Zerlegungen in *genau* k Teilwörter zu betrachten, für die es $\binom{n-1}{k-1}$ Möglichkeiten gibt. Dann gilt:

$$\binom{n-1}{k-1} \leq \binom{n}{k} = \frac{n * \dots * (n-k+1)}{(n-k)!} \leq \frac{n^k}{1} = n^k$$

Der Algorithmus ist also polynomiell.

AUFGABE 10.3. ($\Pr[$*Tutor*in akzeptiert Beweis*$] > 0$) 1 + 1 Punkte

Bis hierhin haben wir nur deterministische oder nichtdeterministische Algorithmen betrachtet. Ein bisher von uns unbetrachtetes Feld ist das der randomisierten Algorithmen. Dies soll sich nun ändern:
Eine *probabilistische* Turingmaschine ist ein 8-Tupel $M = (Q, \Sigma, \Gamma, \delta_1, \delta_2, q_0, \square, F)$, wobei

(a) $\delta_1 : Q \times \Gamma \to Q \times \Gamma \times \{L, N, R\}$ die erste Übergangsfunktion und
(b) $\delta_2 : Q \times \Gamma \to Q \times \Gamma \times \{L, N, R\}$ die zweite Übergangsfunktion

der Maschine ist. Die restlichen Komponenten sind analog zu dem klassischen, bereits bekannten Turingmaschinen-Modell.
Bei jedem Schritt der Maschine wird nun zufällig, mit jeweils Wahrscheinlichkeit $1/2$, entweder ein Schritt der ersten oder der zweiten Übergangsfunktion durchgeführt. Das Auswählen der Transitionsfunktion spiegelt somit das Prinzip eines Münzwurfes wieder.
Das zufällige Auswählen der Übergangsfunktion kann zu Uneinstimmigkeiten führen: Es kann passieren, dass in einem Lauf die Maschine eine Wort akzeptiert, in einem weiteren Lauf das Wort jedoch abgelehnt wird. Eine probabilistische Turingmaschine akzeptiert somit Sprachen mit einer gewissen Wahrscheinlichkeit.
Sei $\varepsilon_+, \varepsilon_- \in [0,1] \subset \mathbb{R}$ und L eine Sprache. Wir sagen, dass die probabilistische Turingmaschine M die Sprache L mit Fehler $(\varepsilon_+, \varepsilon_-)$ akzeptiert falls

(a) $\forall w \in L.\ \Pr[M \text{ akzeptiert } w] \geq 1 - \varepsilon_+$
(b) $\forall w \notin L.\ \Pr[M \text{ lehnt } w \text{ ab }] \geq 1 - \varepsilon_-$

Wir definieren nun die Klasse $\mathsf{RP}_{(\varepsilon_+, \varepsilon_-)}$ als die Menge aller Sprachen, die von einer probabilistischen Turingmaschine mit Fehler $(\varepsilon_+, \varepsilon_-)$ in polynomieller Zeit akzeptiert werden. Zusätzlich setzen wir $\mathsf{RP} \coloneqq \mathsf{RP}_{(1/2,0)}$.

(a) Zeigen Sie: $\mathsf{P} \subseteq \mathsf{RP} \subseteq \mathsf{NP}$.

(b) Zeigen Sie: $\mathsf{RP} = \mathsf{RP}_{(\varepsilon_+,0)}$ für alle $\varepsilon_+ \in (0,1) \subset \mathbb{R}$.

Bemerkung: Halten Sie sich in beiden Teilaufgaben strikt an die in der Vorlesung bzw. hier präsentierten Maschinenmodelle und Definitionen der Zeitklassen. Argumentationen beispielsweise anhand von Javaprogrammen werden nicht gewertet.

Lösungsskizze

(a) Sei $M = (Q, \Sigma, \Gamma, \delta, q_0, \square, F)$ eine TM für eine Sprache in P. Wir definieren die probabilistische TM $M' = (Q, \Sigma, \Gamma, \delta, \delta, q_0, \square, F)$. Dann simuliert M' in jedem Schritt M und akzeptiert somit $L(M)$ mit Fehler $(0,0)$ in polynomieller Zeit. Somit folgt $L(M) \in \mathsf{RP}$.
Sei $M = (Q, \Sigma, \Gamma, \delta_1, \delta_2, q_0, \square, F)$ eine probabilistische TM für eine Sprache L in RP. Wir definieren die nichtdeterministische TM $N \coloneqq (Q, \Sigma, \Gamma, \delta, q_0, \square, F)$, wobei

$$\delta(q, A) \ni (q', A', D) \quad :\overset{\text{Def.}}{\iff} \quad (q', A', D) \in \{\delta_1(q, A), \delta_2(q, A)\}.$$

Dann simuliert N in jedem Schritt beide möglichen Übergänge der Maschine M. Jeder Lauf in N ist daher ein möglicher Lauf in M und somit weiterhin polynomiell.
Sei nun $w \in L$. Da M das Wort mit Wahrscheinlichkeit > 0 akzeptiert, gibt es mindestens einen akzeptierenden Lauf in M für w. Dieser Lauf existiert weiterhin in N und somit $w \in L(N)$. Sei nun $w \notin L$. Dann gibt es keinen akzeptierenden Lauf in M für w. Damit existiert auch kein akzeptierender Lauf in N für w und somit $w \notin L(N)$. Somit gilt $L = L(N) \in \mathsf{NP}$.

(b) Sei $u \coloneqq \min\{1/2, \varepsilon_+\}$ und $l \coloneqq \max\{1/2, \varepsilon_+\}$. Es folgt sofort $\mathsf{RP}_{(u,0)} \subseteq \mathsf{RP}_{(l,0)}$.
Sei nun M eine probabilistische TM für eine Sprache L in $\mathsf{RP}_{(l,0)}$. Dann akzeptiert M ein Wort $w \in L$ mit Wahrscheinlichkeit $\geq 1 - l$. Betrachte nun eine TM M_k die bei Eingabe w die Maschine M k-mal simuliert und akzeptiert, falls mindestens ein Lauf akzeptiert. Dann akzeptiert M_k ein Wort $w \in L$ mit Wahrscheinlichkeit $1 - l^k$ und lehnt jedes Wort $w \notin L$ weiterhin ab. Nun gilt

$$1 - l^k \geq 1 - u \quad \iff \quad u \geq l^k \quad \overset{\log(\cdot) \text{ monoton}}{\iff} \quad \log(u) \geq k \log(l) \quad \overset{\log(l) < 0}{\iff} \quad \log(u)/\log(l) \leq k.$$

Setze also $n \coloneqq \lceil \log(u)/\log(l) \rceil$. Dann akzeptiert M_n ein Wort $w \in L$ mit Wahrscheinlichkeit $\geq 1 - u$. Da n konstant und unabhängig von der Eingabe ist und M in polynomieller Zeit terminiert, terminiert auch M_n weiterhin in polynomieller Zeit für jede Eingabe. Somit folgt $L \in \mathsf{RP}_{(u,0)}$.

> Nearly every example of faulty reasoning that has been published is accompanied by the phrase "of course" or its equivalent.
>
> — Donald Knuth