Prof. T. Nipkow
K. Kappelmann, J. Rädle, L. Stevens

KEINE ABGABE

Technische Universität München
Lehrstuhl für Logik und Verifikation

Einführung in die Theoretische Informatik

Sommersemester 2020 – Übungsblatt Lösungsskizze 10

AUFGABE 10.1. (*Wichtige Begriffe*) Stufe A

Überprüfen Sie, dass Sie die Folgenden Begriffe korrekt definieren können.

- PCP
- unentscheidbare PDA Probleme
- unentscheidbare CFG Probleme
- Zertifikat
- (polynomiell beschränkter) Verifikator
- $time_M(w)$ und $ntime_M(w)$
- $TIME(f(n))$ und $NTIME(f(n))$
- P und NP

AUFGABE 10.2. (*PCP*) Stufe B - D

Wir betrachten in dieser Aufgabe das Post'sche Korrespondenzproblem (PCP).

(a) Bestimmen Sie *alle* Lösungen für das folgende PCP: $P_1 = ((d, cd), (d, d), (abc, ab))$.
(b) Zeigen Sie, dass die folgende Instanz des PCPs keine Lösung hat: $P_2 = ((ab, aba), (baa, aa), (aba, baa))$.
(c) Zeigen Sie, dass das Post'sche Korrespondenzproblem über einem Alphabet mit nur einem Symbol entscheidbar ist, indem Sie einen Algorithmus angeben. Begründen Sie auch dessen Korrektheit.

Lösungsskizze

(a) Die Menge aller Lösungen ist: $\mathsf{L}((2 \mid (31))^*) \setminus \{\varepsilon\}$
(b) Zu Beginn kann nur die Karte (ab, aba) verwendet werden, da 2 und 3 als Startkarte ungeeignet sind und somit alle Lösungen mit 1 beginnen müssen. Wir müssen deshalb mit dem Überhang (ε, a) fortfahren. Offensichtlich kann nun weder (ab, aba) (Karte 1) noch (baa, aa) (Karte 2) angewendet werden. Durch (aba, baa) (Karte 3) erhalten wir aber $(aba, abaa)$, was wiederum zu dem Überhang (ε, a) führt. Wir können somit keinen Abschluss finden und damit kann diese Instanz des Post'schen Korrespondenzproblems keine Lösung besitzen.
(c) In diesem Fall haben alle Karten c die Form $c = (a^i, a^j)$ für $\Sigma = \{a\}$ und $i, j \geq 0$.
- Falls $i = j$, ist die Karte c alleine eine Lösung.
- Wenn alle Karten oben länger als unten ($i > j$) sind, gibt es keine Lösung.
- Analog für den umgekehrten Fall ($i < j$).
- Wenn es zwei Karten c_1, c_2 mit $c_1 = (a^{j_1}, a^{k_1})$, $c_2 = (a^{j_2}, a^{k_2})$, $j_1 > k_1$ und $j_2 < k_2$ gibt, sei i_1 der Index von c_1 und sei i_2 der von c_2. Dann ist z.B. $i_1^{k_2 - j_2} i_2^{j_1 - k_1}$ eine Lösung des PCP.
  Darauf kann man kommen, indem man eine Lösung für das folgende Gleichungssystem sucht:

$$\begin{aligned} j_1 \cdot \#_{c1} + j_2 \cdot \#_{c2} &= k_1 \cdot \#_{c1} + k_2 \cdot \#_{c2} \\ j_1 &> k_1 \\ j_2 &< k_2 \end{aligned}$$

  Eine TM kann diese Vorbedingungen prüfen und somit bestimmen, ob es eine Lösung gibt.

AUFGABE 10.3. (*Entscheidbarkeit und kontextfreie Grammatiken*) Stufe D

Seien $\mathsf{G}_1, \mathsf{G}_2$ CFGs. Beweisen Sie die folgenden beiden Aussagen:

(a) $\mathsf{L}(\mathsf{G}_1) \not\subseteq \mathsf{L}(\mathsf{G}_2)$ ist semi-entscheidbar.
(b) $\mathsf{L}(\mathsf{G}_1) \subseteq \mathsf{L}(\mathsf{G}_2)$ ist unentscheidbar.
Hinweis: Betrachten Sie den Beweis für die Unentscheidbarkeit von $\mathsf{L}(\mathsf{G}_1) \cap \mathsf{L}(\mathsf{G}_2) = \emptyset$ aus der Vorlesung.

Lösungsskizze

(a) Es gilt $\mathsf{L}(\mathsf{G}_1) \not\subseteq \mathsf{L}(\mathsf{G}_2)$ genau dann, wenn es ein $w \in \mathsf{L}(\mathsf{G}_1) \setminus \mathsf{L}(\mathsf{G}_2)$ gibt.
Für $w \in \Sigma^*$: Für jedes w testet man mittels CYK (o.B.d.A. sind G_1 und G_2 in CNF), ob $w \in \mathsf{L}(\mathsf{G}_1)$ und ob $w \in \mathsf{L}(\mathsf{G}_2)$ gilt. Sobald man das erste $w \in \mathsf{L}(\mathsf{G}_1) \setminus \mathsf{L}(\mathsf{G}_2)$ gefunden hat, stoppt man und gibt 1 aus. Offensichtlich stoppt der Algorithmus im Fall $\mathsf{L}(\mathsf{G}_1) \not\subseteq \mathsf{L}(\mathsf{G}_2)$ stets, im Fall $\mathsf{L}(\mathsf{G}_1) \subseteq \mathsf{L}(\mathsf{G}_2)$ terminiert der Algorithmus allerdings nie. Damit ist das Problem semi-entscheidbar.

(b) Siehe Vorlesung Folie 344. Hier nochmal etwas ausführlicher:
Es gilt $\mathsf{L}(\mathsf{G}_1) \subseteq \mathsf{L}(\mathsf{G}_2)$ **gdw** $\mathsf{L}(\mathsf{G}_1) \setminus \mathsf{L}(\mathsf{G}_2) = \emptyset$ **gdw** $\mathsf{L}(\mathsf{G}_1) \cap \overline{\mathsf{L}(\mathsf{G}_2)} = \emptyset$. (O.B.d.A. verwenden G_1 und G_2 dasselbe Alphabet Σ.)
Sei $(x_1, y_1), \ldots, (x_l, y_l) \in \Gamma^* \times \Gamma^*$ eine PCP-Instanz. Sei $\Sigma = \{a_1, \ldots, a_l\}$, o.B.d.A. $\Gamma \cap \Sigma = \emptyset$, da wir Σ frei wählen können. Dann sind die in der Vorlesung verwendeten Grammatik $\mathsf{G}_1, \mathsf{G}_2$ deterministisch. Da DCFL unter Komplement nach Vorlesung abgeschlossen sind, kann man aus G_2 bzw. dem entsprechenden DPDA eine CFG G_2' mit $\mathsf{L}(\mathsf{G}_2') = \overline{\mathsf{L}(\mathsf{G}_2)}$ konstruieren.
Damit gilt: $\mathsf{L}(\mathsf{G}_1) \subseteq \mathsf{L}(\mathsf{G}_2')$ **gdw** $\mathsf{L}(\mathsf{G}_1) \cap \overline{\mathsf{L}(\mathsf{G}_2')} = \emptyset$ **gdw** $\mathsf{L}(\mathsf{G}_1) \cap \mathsf{L}(\mathsf{G}_2) = \emptyset$ **gdw** die gegebene PCP-Instanz hat keine Lösung. Somit ist die Teilmengenrelation über CFGs unentscheidbar.

AUFGABE 10.4. (*Abschlusseigenschaften von NP*)
Seien $A, B \subseteq \Sigma^*$ Sprachen in NP. Zeigen Sie, dass $A \cap B$ ebenfalls in NP liegt. Geben Sie hierfür an, wie geeignete polynomiell verifizierbare Zertifikate hierfür aussehen. Stufe C

Lösungsskizze
Da A und B in NP liegen, können wir annehmen, dass es polynomiell beschränkte Verifikatoren M_A und M_B für A bzw. B gibt. Für jedes $w \in A$ gibt es also ein Zertifikat c_w^A, das "beweist", dass $w \in A$ liegt und dies in Polynomialzeit von M_A überprüft werden kann. Formaler: Es existiert ein Polynom $p_A(\cdot) : \mathbb{N} \to \mathbb{N}$, sodass für jedes $w \in A$ das Wort $w\#c_w^A$ in höchstens $p_A(|w|)$ Schritten von M_A akzeptiert wird. Analoges gilt für $w \in B$, c_w^B, $p_B(\cdot)$ und M_B.
Wir bauen nun einen polynomiellen Verifikator $M_{A\cap B}$ für $A \cap B$ wie folgt: Ein Zertifikat für $w \in A \cap B$ besteht aus einem Paar $c_w^A \# c_w^B$ von Zertifikaten: eines für $w \in A$ und eines für $w \in B$. Dabei benutzen wir wieder # als Trennzeichen. Der Verifikator $M_{A\cap B}$ führt dann bei Eingabe $w\#c_1\#c_2$ zunächst den Verifikator M_A auf $w\#c_1$ aus. Falls dieser akzeptiert, führt $M_{A\cap B}$ anschließend den Verifikator für M_B auf $w\#c_2$ aus. Falls das Eingabewort nicht im richtigen Format vorliegt (beispielsweise das Zertifikat nicht die Form $c_1\#c_2$ besitzt), wird die Eingabe ohne Simulation abgelehnt. Für den Verifikator $M_{A\cap B}$ gilt nun

$$w\#c \in L(M_{A\cap B}) \iff \exists c_w^A, c_w^B.\ c = c_w^A \# c_w^B \wedge w\#c_w^A \in L(M_A) \wedge w\#c_w^B \in L(M_B)$$

Wir sind allerdings noch nicht fertig: Es fehlt noch zu zeigen, dass $M_{A\cap B}$ in polynomieller Zeit terminiert. Hierbei ergibt sich eine Subtilität: Laut Definition 6.7 der Vorlesung kann ein polynomiell beschränkter Verifikator auch "lange" Zertifikate (beispielsweise exponentiell lange in $|w|$) akzeptieren, obwohl er nur ein polynomiell langes Präfix lesen kann. Das Suchen nach einem Trennzeichen bzw. kopieren der Eingabe würde aber für solche langen Zertifikate nicht mehr in polynomieller Zeit funktionieren.
Für solche Zertifikate existiert jedoch stets ein kürzeres, polynomiell beschränktes Zertifikat, das einfach aus den tatsächlich gelesenen Zeichen des Verfikators besteht. Wir erweitern $M_{A\cap B}$ also um einen Zähler beim Parsen des Eingabezertifikats: sobald $M_{A\cap B}$ mehr als $p_A(|w|) + p_B(|w|)$ Zeichen im Zertifikat feststellt, lehnt die Maschine die Eingabe ab. Somit existiert weiterhin für jedes $w \in A \cap B$ ein geeignetes Zertifikat $c_w^A \# c_w^B$ und $M_{A\cap B}$ terminiert in Zeit $\mathcal{O}(p_A(|w|) + p_B(|w|))$, also in polynomieller Zeit bezüglich $|w|$.