

Berechenbarkeit und Komplexitätstheorie

Wintersemester 2021/2022

Aufgabenblatt 6

Abgabe: 17. Dezember 2021 um 12 Uhr

Aufgabe 6.1 (3 + 4 Punkte)

Es seien die beiden Folgen von Tupeln gegeben:

- $K_1 = (\oplus \dot{\vee} \dot{\vee}, \dot{\vee} \dot{\vee} \oplus), (\dot{\vee} \dot{\vee} \oplus, \dot{\vee}), (\oplus \dot{\vee} \dot{\vee} \oplus, \dot{\vee}), (\dot{\vee} \dot{\vee}, \oplus \dot{\vee} \dot{\vee}), (\oplus, \oplus \oplus \dot{\vee})$
- $K_2 = (101, 10), (1, 01), (010, 10), (10, 0)$

Zeigen oder widerlegen Sie die folgende Aussagen:

- (i) $K_1 \in PCP$
- (ii) $K_2 \in PCP$

Lösung

- a) $K_1 \in PCP$ mit $i_1 = 5, i_2 = 1, i_3 = 3, i_4 = 5, i_5 = 2$ und $i_6 = 4$. Es ergibt sich damit:

$\oplus \oplus \dot{\vee} \dot{\vee} \oplus \dot{\vee} \dot{\vee} \oplus \oplus \oplus \dot{\vee} \dot{\vee} \oplus \dot{\vee} \dot{\vee}$
 $\oplus \oplus \dot{\vee} \dot{\vee} \oplus \dot{\vee} \dot{\vee} \oplus \oplus \oplus \dot{\vee} \dot{\vee} \oplus \dot{\vee} \dot{\vee}$

- b) $K_2 \notin PCP$. Der einzig mögliche Anfang der Folge kann nur das erste Tupel $(101, 10)$ sein, da sich bei allen anderen Tupeln die beiden Komponenten im ersten Symbol unterscheiden. Darauf muss ein Tupel folgen, dessen zweite Komponente mit 1 beginnt. Mit dem Tupel $(101, 10)$ ergibt sich allerdings $101\underline{1}01 \cdot w \neq 101\underline{1}0 \cdot v$, d.h. die beiden Wörter stimmen an der vierten Position nicht überein. Also muss das zweite gewählte Tupel $(010, 10)$ sein. Somit ergibt sich die folgende Situation, die wir (*) nennen:

$101010 \cdot w$
 $1010 \cdot v$

Beobachtung: Es muss auf jeden Fall das Tupel $(1, 01)$ verwendet werden, da bei allen anderen Tupeln die erste Komponente länger als die zweite ist und die beiden Wörter somit nie gleichlang werden können. Weiter kann das genannte Tupel höchstens zweimal hintereinander verwendet werden, da es keine Möglichkeit gibt "111" mit den zweiten Komponenten zu erzeugen. (Aus dem gleichen Grund kann Tupel $(101, 10)$ auch nicht auf die genannte Tupelkombination folgen.)

Wenn wir die Tupel $(101, 10), (010, 10), (101, 10)$ verwenden, dann können wir nicht als nächstes Tupel $(101, 10)$ nutzen, da dann das nächste Tupel in der zweiten Komponente mit "11" beginnen müsste. Dies ist nicht möglich. Ebenfalls können wir die Tupel $(1, 01)$ und $(10, 0)$ nicht nutzen, da die zweiten Komponenten nicht mit 1 beginnen. Wenn wir als viertes Tupel $(010, 10)$ verwenden, dann müssen wir danach wieder das Tupel $(101, 10)$ verwenden. Dies wiederholt sich immer wieder und das erste Wort wächst schneller als das zweite Wort. Es ist also nicht möglich, $(101, 10)$ als drittes Tupel zu verwenden. Das heißt, das dritte gewählte Tupel muss $(010, 10)$ sein. Wenn wir nun auf dieses Tupel zweimal $(1, 01)$ folgen lassen (*Einzig sinnvolle Wahl.*), dann befinden wir uns wieder in einer Situation (*).

Zusammengefasst bedeutet das, dass $K_2 \notin PCP$.

Aufgabe 6.2 (4 Punkte)

Sei 01-PCP die Variante des Postschen Korrespondenzproblems, bei der die Eingabetupel auf das Alphabet $\{0, 1\}$ beschränkt sind (d. h. die Wörter der Wortpaarfolge sind nicht-leere Binärstrings). Zeigen Sie, dass diese Variante *unentscheidbar* ist.

Lösung

Die Idee bei dieser Aufgabe ist es, das allgemeine PCP auf 01-PCP zu reduzieren; dann gilt, dass 01-PCP nicht entscheidbar sein kann, da PCP nicht entscheidbar ist. Gegeben sei also eine PCP-Instanz über einem beliebigen Alphabet Σ . Wir müssen nun jedes Symbol $\alpha \in \Sigma$ als Binärstring kodieren. Allerdings muss man bei der Kodierung aufpassen, dass keine “falschen” Lösungen entstehen. Würde bspw. ein Symbol α mit 0 kodiert werden und ein zweites Symbol β mit 00, dann wäre $0 \cdot 0 = 00$, allerdings $\alpha \cdot \text{alpha} \neq \beta$. Um diese Problematik zu umgehen, kann man die Kodierung so wählen, dass alle Kodierungen die gleiche Länge haben oder, dass bei allen Kodierungen der “Rand” eines Symbols explizit erkennbar ist. Prinzipiell funktioniert hier jede präfixfreie (oder suffixfreie) Kodierung.

Eine (für die Reduktion passende) Kodierung könnte bspw. wie folgt konstruiert werden: Sei $\Sigma = \{\alpha_1, \dots, \alpha_m\}$. Wir ersetzen jedes Symbol α_i durch den Binärstring 01^i (also eine Null gefolgt von i Einsen). Außerdem ersetzen wir in jedem Wort der Wortpaarfolge der gegebenen Problem Instanz die Symbole passend zur Kodierung. Diese Funktion (wenn auch nicht formal als Funktion aufgeschrieben) ist total und berechenbar und erfüllt somit die Eigenschaften einer Reduktionsfunktion. Da die Funktion bijektiv ist, gilt auch offensichtlich, dass die gegebene Problem Instanz genau dann eine Lösung hat, wenn auch die kodierte Instanz eine Lösung besitzt und somit gilt $\text{PCP} \leq 01\text{-PCP}$.

Aufgabe 6.3 (4 Punkte)

Sei 1-PCP die Variante des Postschen Korrespondenzproblems, bei dem man die Eingabetupel auf das Alphabet $\{1\}$ beschränkt. Zeigen Sie, dass 1-PCP *entscheidbar* ist.

Lösung

Wenn das Alphabet nur aus einem Zeichen besteht, dann kommt es nur auf die Längen der einzelnen Teilworte an. Darum kann man das ganze “rechnerisch” entscheiden. Folgender Algorithmus entscheidet das unäre PCP:

1. Wenn für alle Paare (x_i, y_i) gilt: $|x_i| > |y_i|$, dann lehne ab.
2. Wenn für alle Paare (x_i, y_i) gilt: $|x_i| < |y_i|$, dann lehne ab.
3. Sonst akzeptiere.

Im letzten Schritt kann der Algorithmus problemlos akzeptieren, denn: Entweder es gibt ein Paar (x_i, y_i) mit $|x_i| = |y_i|$. Dann ist dieses Paar eine gültige Lösung. Oder es existieren zwei Paare (x_i, y_i) und (x_j, y_j) , so dass gilt: $|x_i| > |y_i|$ und $|x_j| < |y_j|$. Um nun mit den beiden Paaren zwei gleiche Wörter zu bilden müssen wir $n \cdot |x_i| + m \cdot |x_j| = n \cdot |y_i| + m \cdot |y_j|$ lösen. Diese Gleichung ist äquivalent zu $n \cdot (|x_i| - |y_i|) = m \cdot (|y_j| - |x_j|)$. Diese hat eine Lösung mit $n = (|y_j| - |x_j|)$ und $m = (|x_i| - |y_i|)$. Wenn man nun n -mal Paar (x_i, y_i) und

m -mal Paar (x_j, y_j) nimmt, ergeben sich gleiche Wörter:

$$\begin{aligned}
 & x_i^n \cdot x_j^m &= y_i^n \cdot y_j^m \\
 \equiv x_i^{|y_j|-|x_j|} \cdot x_j^{|x_i|-|y_i|} &= y_i^{|y_j|-|x_j|} \cdot y_j^{|x_i|-|y_i|} \\
 \equiv (|y_j|-|x_j|) \cdot |x_i| + (|x_i|-|y_i|) \cdot |x_j| &= (|y_j|-|x_j|) \cdot |y_i| + (|x_i|-|y_i|) \cdot |y_j| &> \text{Kodierung ausgewertet} \\
 \equiv |x_i||y_j|-|x_i||x_j|+|x_i||x_j|-|x_j||y_i| &= |y_i||y_j|-|x_j||y_i|+|x_i||y_j|-|y_i||y_j| &> \text{ausmultipliziert} \\
 \equiv |x_i||y_j|-|x_j||y_i| &= -|x_j||y_i|+|x_i||y_j| &> \text{Zusammengefasst} \\
 \equiv |x_i||y_j|-|x_j||y_i| &= |x_i||y_j|-|x_j||y_i| &> \text{Umsortiert} \\
 \equiv 1^{|x_i||y_j|-|x_j||y_i|} &= 1^{|x_i||y_j|-|x_j||y_i|} &> \text{Kodierung angewandt}
 \end{aligned}$$

Da $|x_i| > |x_j| > 0$ und $|y_j| > |y_i| > 0$ und somit $|x_i||y_j| > |x_j||y_i|$, ist $1^{|x_i||y_j|-|x_j||y_i|}$ ein gültiges Wort.