

Berechenbarkeit und Komplexitätstheorie

Wintersemester 2021/2022

Aufgabenblatt 4

Abgabe: 3. Dezember 2021 um 12 Uhr

Definition(en)

Seien $L_1, L_2 \subseteq \Sigma^*$ zwei Sprachen. Wir definieren die *Konkatenation von Sprachen* als $L_1 \circ L_2 := \{w_1 \cdot w_2 : w_1 \in L_1 \wedge w_2 \in L_2\}$.

Aufgabe 4.1 (3 Punkte)

Seien $L_1, L_2 \subseteq \Sigma^*$ entscheidbare Sprachen. Zeigen Sie, dass $L_1 \circ L_2$ ebenfalls entscheidbar ist.

Lösung

$L_1 \circ L_2$ entscheidbar, gdw. $\chi_{L_1 \circ L_2}$ berechenbar. Sei $w = \alpha_1 \cdots \alpha_n$ mit $\alpha_1, \dots, \alpha_n \in \Sigma$. Es gilt:

$$\begin{aligned} w \in L_1 \circ L_2 &\equiv \varepsilon \in L_1 \wedge \alpha_1 \cdots \alpha_n \in L_2 \\ &\vee \alpha_1 \in L_1 \wedge \alpha_2 \cdots \alpha_n \in L_2 \\ &\vee \dots \\ &\vee \alpha_1 \cdots \alpha_{n-1} \in L_1 \wedge \alpha_n \in L_2 \\ &\vee \alpha_1 \cdots \alpha_n \in L_1 \wedge \varepsilon \in L_2 \\ &\equiv \chi_{L_1}(\varepsilon) = 1 \wedge \chi_{L_2}(\alpha_1 \cdots \alpha_n) = 1 \\ &\vee \dots \\ &\vee \chi_{L_1}(\alpha_1 \cdots \alpha_n) = 1 \wedge \chi_{L_2}(\varepsilon) = 1 \\ &\equiv \chi_{L_1}(\varepsilon) \cdot \chi_{L_2}(\alpha_1 \cdots \alpha_n) + \dots + \chi_{L_1}(\alpha_1 \cdots \alpha_n) \cdot \chi_{L_2}(\varepsilon) \geq 1 \end{aligned}$$

Daraus folgt: $\chi_{L_1 \circ L_2}$ ist berechenbar, da es eine Verknüpfung berechenbarer Funktionen ist.

Alternativ kann man auch einen Algorithmus angeben der die Sprache entscheidet. Am einfachsten läuft man mit einer Schleife über die Eingabe w und teilt das Wort in zwei Teilwörter auf. Dann kann man in jedem Schleifendurchlauf testen, ob das linke Teilwort in L_1 und das rechte Teilwort in L_2 liegt.

Aufgabe 4.2 (4 Punkte)

Seien $A, \bar{A} \subseteq \Sigma^*$ semi-entscheidbare Sprachen (wobei $\bar{A} = \Sigma^* \setminus A$ das Komplement von A bezeichnet).

Zeigen Sie: A ist entscheidbar.

(*Hinweis:* Für ein beliebiges $w \in \Sigma^*$ hält mindestens eine der beiden Turingmaschinen, die die jeweilige charakteristische Funktion berechnen. Überlegen Sie sich, wie Sie diese beiden Turingmaschinen “parallel” laufen lassen können.)

Lösung

Die Idee ist, wie im Hinweis steht, beide Maschinen gleichzeitig laufen zu lassen. D. h. man führt jeweils einen Rechenschritt der einen Maschine und dann einen Rechenschritt der anderen Maschine durch. Da mindestens eine der beiden Maschinen hält (für ein $w \in \Sigma^*$ gilt entweder $w \in A$ oder $w \notin A \equiv w \in \bar{A}$), erkennt man, ob das Wort in der Menge oder im Komplement ist. Etwas genauer: Sei M_1 die TM, die χ'_A berechnet und M_2 die TM, die $\chi'_{\bar{A}}$ berechnet. Wie konstruieren eine neue TM M wie folgt:

(Eingabe: $w \in \Sigma^*$ beliebig)

- Simuliere *einen* Rechenschritt (bzw. Zustandsübergang) von M_1 .
- Simuliere *einen* Rechenschritt (bzw. Zustandsübergang) von M_2 .
- Wiederhole die (abwechselnd ausgeführten) Rechenschritte der beiden TMs bis eine der beiden Maschinen in den Endzustand übergeht.
- Falls M_1 hält, so wechsele in den Endzustand und gib das Ergebnis der Simulation von M_1 aus.
- Falls M_2 hält, so wechsele in den Endzustand und gib das Ergebnis der Simulation von M_2 aus.

Für die (parallele) Simulation beider Maschinen kann man sich bspw. eine Mehrband-TM mit zwei Bändern vorstellen. Das erste Band simuliert M_1 und das zweite Band M_2 . Falls A semi-entscheidbar aber *nicht* entscheidbar ist, hält *genau* eine der beiden TMs M_1 und M_2 . Dann gilt $w \in A$, falls M_1 hält und $w \in \bar{A}$, falls M_2 hält.

Aufgabe 4.3 (4 Punkte)

Seien $A, B \subseteq \mathbb{N}^k$ semi-entscheidbare Mengen. Welche der Mengen $A \cup B$, $A \cap B$, $\bar{A}$ und $A \setminus B$ sind ebenfalls semi-entscheidbar? Und welche nicht? Begründen Sie Ihre Antworten.

Lösung

$A \cup B$ und $A \cap B$ sind semi-entscheidbar: A ist semi-entscheidbar $\Rightarrow A$ ist rekursiv aufzählbar $\Rightarrow \exists f_A : \mathbb{N} \rightarrow A$, $A = \{f(1), f(2), f(3), \dots\}$. (Analog existiert f_B für B .) Folgendes Programm ist ein Semi-Entscheidungsverfahren für $A \cup B$:

```
1: function  $\chi_{A \cup B}(x)$ 
2:    $i \leftarrow 1$ 
3:    $inA, inB \leftarrow \text{FALSE}$ 
4:   while  $\neg(inA \vee inB)$  do
5:     if  $f_A(i) = x$  then
6:        $inA \leftarrow \text{TRUE}$ 
7:     fi
8:     if  $f_B(i) = x$  then
9:        $inB \leftarrow \text{TRUE}$ 
10:    fi
11:     $i \leftarrow i + 1$ 
12:  od
13:  return 1
14: end
```

Wenn man die Bedingung in Zeile 4 zu “ $\neg(inA \wedge inB)$ ” ändert, ist der Algorithmus ein Semi-Entscheidungsverfahren für $A \cap B$. Wichtig bei der Vereinigung ist, dass man die beiden Mengen A und B “parallel” testet. Wenn man sie nacheinander testet, dann hat man ein Problem, falls $x \notin A$ aber $x \in B$; Da A semi-entscheidbar, läuft der Algorithmus eventuell in eine Endlosschleife und erkennt nicht, dass x in B und somit auch in der Vereinigung ist.

$\bar{A}$ ist im Allgemeinen nicht semi-entscheidbar: Sei A semi-entscheidbar aber *nicht entscheidbar*. Angenommen $\bar{A}$ wäre semi-entscheidbar. Da A nach Voraussetzung semi-entscheidbar ist, wäre A entscheidbar (Vorige Aufgabe bzw. Satz 8.2). Dies ist ein Widerspruch.

$A \setminus B$ ist ebenfalls nicht semi-entscheidbar (da das Komplement nicht semi-entscheidbar ist). Wäre $A \setminus B$ semi-entscheidbar, dann wäre insbesondere auch $\mathbb{N}^k \setminus A = \bar{A}$ semi-entscheidbar, was ein Widerspruch zum vorigen Punkt darstellt.

Aufgabe 4.4 (2 + 2 Punkte)

Zeigen Sie die folgenden Aussagen:

- a) $\mathbb{P} := \{p \in \mathbb{N} : p \text{ ist eine Primzahl.}\}$ ist entscheidbar.
- b) $\{n \in \mathbb{N} : \text{Es gibt eine Primzahl } p, \text{ sodass } p+1, \dots, p+n \text{ keine Primzahlen sind.}\}$ ist semi-entscheidbar.

Lösung

- a) $\{p \in \mathbb{N} : p \text{ ist eine Primzahl.}\}$ kann man mit folgendem Algorithmus entscheiden:

```

    ▷ Eingabe:  $p \in \mathbb{N}$ 
1: if  $p \leq 1$  then
2:   return FALSE
3: fi
4:  $i \leftarrow 2$ 
5: while  $i < p$  do                                ▷ Als Bedingung reicht auch  $i \cdot i \leq p$ 
6:   if  $p \equiv 0 \pmod{i}$  then                          ▷  $i$  teilt  $p$ 
7:     return FALSE
8:   fi
9:    $i \leftarrow i + 1$ 
10: od
11: return TRUE

```

Der Algorithmus testet alle potentiellen Teiler und gibt FALSE zurück, wenn ein Teiler gefunden wird. Wenn kein Teiler (in $\{2, \dots, p-1\}$) gefunden wird, dann ist die Zahl prim und der Algorithmus liefert TRUE zurück.

- b) Ein einfaches Semi-Entscheidungsverfahren ist das folgende:

```

1: for each  $p \in \mathbb{P}$  do                                ▷  $\mathbb{P}$  ist die Menge aller Primzahlen
2:   if  $\{p+1, \dots, p+n\} \cap \mathbb{P} = \emptyset$  then
3:     return TRUE
4:   fi
5: od

```

Der Algorithmus testet zu jeder Primzahl p ob $p+1$ bis $p+n$ auch Primzahlen sind. Vielleicht ist die Menge sogar entscheidbar, da es zu jeder Zahl $n \in \mathbb{N}$ mit $\{n!+2, \dots, n!+n\}$ eine Menge von $n-1$ aufeinanderfolgenden Zahlen gibt, die nicht prim sind. Der Algorithmus könnte in dem Fall also immer TRUE zurückgeben, da man beliebig große Lücken zwischen Primzahlen generieren kann. (Hier müsste man zahlentheoretisch etwas genauer argumentieren, oder die ursprüngliche Menge anders formulieren, denn $n!+1$ ist nicht immer eine Primzahl. Allerdings wird vermutet, dass es unendlich viele Primzahlen dieser Form gibt.)