Automaten und Berechenbarkeit

Wintersemester 2022/2023

Jana Grajetzki (jana.grajetzki@uni-jena.de)

11. Übungsserie

Abgabe: Montag, 23.1.2023, bis 12 Uhr in Moodle.

Bitte begründen Sie Ihre Antworten immer, auch wenn dies nicht explizit gefordert wird.

Aufgabe 1:
Zeigen Sie, dass folgende Sprachen entscheidbar sind:

(a) $E_{NFA} = \{\langle A\rangle \mid A \,\text{ist NFA und}\, L(A) = \emptyset\}$, die Menge aller Kodierungen nichtdeterministischer endlicher Automaten, die kein Wort erzeugen.

(b) $EQ_{REG} = \{\langle G_1, G_2\rangle \mid G_1, G_2$ sind reguläre Grammatiken und $L(G_1) = L(G_2)\}$, die Menge aller Paare von Kodierungen regulärer Grammatiken für die gilt, dass die von ihnen erzeugten Sprachen gleich sind.

(c) $A = \{\langle G\rangle \mid G \,\text{ist kfG über}\, \{0,1\}$ und $(1)^* \cap L(G) \neq \emptyset\}$, die Menge aller Kodierungen kontextfreier Grammatiken über dem Alphabet $\{0,1\}$, die mindestens ein Wort erzeugen, das nur aus Einsen besteht.

(d) $B = \{\langle G\rangle \mid G$ ist reguläre Grammatik und $L(G) = \{w \in \{0,1\}^* \mid \#_0(w) \equiv_3 1\}\}$

(e) $C = \{\langle G\rangle \mid G$ ist reguläre Grammatik und $\exists w \in L(G), w$ beginnt mit 1 und endet auf $1\}$

(23 Punkte)

Aufgabe 2:
Beweisen Sie folgende Aussagen:

(a) Wenn eine Sprache $L \subseteq \Sigma^*$ entscheidbar ist und eine Funktion $f : \Sigma^* \to \Sigma^*$ total definiert und berechenbar ist, dann ist auch $f^{-1}(L)$ entscheidbar.

(b) Wenn eine Sprache $L \subseteq \Sigma^*$ rekursiv aufzählbar ist und eine Funktion $f : \Sigma^* \to \Sigma^*$ berechenbar ist, dann ist auch $f^{-1}(L)$ rekursiv aufzählbar.

(12 Punkte)

AuB: 11.Hausaufgabe (23.01.23) - Cora Zeitler

Montag, 23. Januar 2023 00:01

Aufgabe 1:
Zeigen Sie, dass folgende Sprachen entscheidbar sind:

(a) $E_{NFA} = \{\langle A\rangle \mid A \text{ ist NFA und } L(A) = \emptyset\}$, die Menge aller Kodierungen nichtdeterministischer endlicher Automaten, die kein Wort erzeugen.

(b) $EQ_{REG} = \{\langle G_1, G_2\rangle \mid G_1, G_2 \text{ sind reguläre Grammatiken und } L(G_1) = L(G_2)\}$, die Menge aller Paare von Kodierungen regulärer Grammatiken für die gilt, dass die von ihnen erzeugten Sprachen gleich sind.

(c) $A = \{\langle G\rangle \mid G \text{ ist kfG über } \{0,1\} \text{ und } (1)^* \cap L(G) \neq \emptyset\}$, die Menge aller Kodierungen kontextfreier Grammatiken über dem Alphabet $\{0,1\}$, die mindestens ein Wort erzeugen, das nur aus Einsen besteht.

(d) $B = \{\langle G\rangle \mid G \text{ ist reguläre Grammatik und } L(G) = \{w \in \{0,1\}^* \mid \#_0(w) \equiv_3 1\}\}$

(e) $C = \{\langle G\rangle \mid G \text{ ist reguläre Grammatik und } \exists w \in L(G), w \text{ beginnt mit 1 und endet auf 1}\}$

(23 Punkte)

1) (a) $E_{NFA} = \{\langle A\rangle \mid A \text{ ist NFA und } L(A) = \emptyset\}$, die Menge aller Kodierungen nichtdeterministischer endlicher Automaten, die kein Wort erzeugen.

```
Eingabe: NFA
for i in Endzustände
        makiere
for j in (Zustände, die in makierte Zustände überführen)
        makiere j
for k in Startzustände
        if (k == markiert)
                return  0
return 1
```

Erklärung	- erst markiert man jeden Endzustand - dann werden alle Zustände markiert, die in einen bereits markierte Zustände überführt werden können --> wird solange Wiederholt bis kein Zustand mehr markiert werden kann - nach der Schleife schaut man ob ein Startzustand markiert ist --> wenn ja , dann gebe 0 aus, sonst 1

(b) $EQ_{REG} = \{\langle G_1, G_2\rangle \mid G_1, G_2 \text{ sind reguläre Grammatiken und } L(G_1) = L(G_2)\}$, die Menge aller Paare von Kodierungen regulärer Grammatiken für die gilt, dass die von ihnen erzeugten Sprachen gleich sind.

Eingabe: REG

(c) $A = \{\langle G\rangle \mid G \text{ ist kfG über } \{0,1\} \text{ und } (1)^* \cap L(G) \neq \emptyset\}$, die Menge aller Kodierungen kontextfreier Grammatiken über dem Alphabet $\{0,1\}$, die mindestens ein Wort erzeugen, das nur aus Einsen besteht.

- die TM sucht ob es mindestens ein Wort gibt das nur aus 1 besteht

```
in Chomsky-Normalform überführen
for  i in Terminale          // i ist Nichtterminal
        if ( i == "1")
                markiere i
for j in markierte Nichtterminale
        for k in Nichtterminale
                if (delta (überführt "1") existiert)
                        markiere k
for m  in Startzustände
        if (m == markiert)
                return 1
return 0
```

(d) $B = \{\langle G\rangle \mid G \text{ ist reguläre Grammatik und } L(G) = \{w \in \{0,1\}^* \mid \#_0(w) \equiv_3 1\}\}$

$G' = (\{0,1\}, \{S, E, Z\}, S, R)$

$R = \{ S \to 0E \mid 1S \mid 0$;

$E \to 0Z \mid 1E \mid 1$;

$Z \to 0S \mid 1Z \}$

↳ die Grammatik G' erzeugt die Sprache $\{w \in \{0,1\}^* \mid \#_0(w) \equiv_3 1\}$

↳ mit B überprüft man, ob die eingegebene Grammatik G, die gleiche Sprache, wie G' erzeugt

↳ wenn Sie übereinstimmen, gebe 1 aus, sonst 0

(e) $C = \{\langle G\rangle \mid G \text{ ist reguläre Grammatik und } \exists w \in L(G), w \text{ beginnt mit 1 und endet auf 1}\}$

```
-in Chomsky-Normalform überführen
for i in Terminale          // i ist Nichtterminal
        if (i == "1")
                markiere i
for j in markierte Nichtterminale
        for k in  Nichtterminale
                if (delta(k,j) existert)
                        markiere k
for m in Startterminale
        for n in makierte Nichtterminale
                if (m == markiert UND delta("1", m, n, irgendwohin) existiert)
                        return 1
return 0
```

Aufgabe 2:
Beweisen Sie folgende Aussagen:

(a) Wenn eine Sprache $L \subseteq \Sigma^*$ entscheidbar ist und eine Funktion $f : \Sigma^* \to \Sigma^*$ total definiert und berechenbar ist, dann ist auch $f^{-1}(L)$ entscheidbar.

(b) Wenn eine Sprache $L \subseteq \Sigma^*$ rekursiv aufzählbar ist und eine Funktion $f : \Sigma^* \to \Sigma^*$ berechenbar ist, dann ist auch $f^{-1}(L)$ rekursiv aufzählbar.

(12 Punkte)

2)a)	Eine Möglichkeit, dies zu beweisen, ist zu zeigen, dass man einen Algorithmus verwenden kann, um die Funktion f^{1} (L) zu berechnen, indem man die bereits gegebenen Eigenschaften von L und f verwendet. Da L entscheidbar ist, kann man einen Algorithmus verwenden, um jeden String in Σ* zu testen, ob er in L enthalten ist. Da f total definiert und berechenbar ist, kann man jeden String in Σ* in einen anderen String in Σ* umwandeln. Um f^{1} (L) zu berechnen, kann man nun jeden String in Σ* verwenden, um ihn mithilfe der Funktion f umzuwandeln, und dann zu testen, ob das Ergebnis in L enthalten ist. Wenn es in L enthalten ist, gehört der ursprüngliche String zu f^{1} (L). Wenn nicht, gehört er nicht dazu.

	Da man einen Algorithmus verwenden kann, um f^{1} (L) zu berechnen, ist es entscheidbar

2)b)	Um zu beweisen, dass f^{1} (L) rekursiv aufzählbar ist, müssen wir zeigen, dass es eine rekursive Funktion g gibt, die jeden String in f^{1} (L) auf eine natürliche Zahl abbildet und umgekehrt jede natürliche Zahl auf einen String in f^{1} (L) abbildet. Da L rekursiv aufzählbar ist, gibt es eine rekursive Funktion h, die jeden String in L auf eine natürliche Zahl abbildet und umgekehrt jede natürliche Zahl auf einen String in L abbildet. Da f berechenbar ist, gibt es eine berechenbare Funktion f, die jeden String in Σ* in einen anderen String in Σ* abbildet. Um f^{1} (L) zu berechnen, können wir eine neue Funktion g definieren: $g(n) = h^{1}(f(h(n)))$. Da h und f berechenbar sind, ist g auch berechenbar. Da h und h^{1} rekursive Funktionen sind, ist g eine rekursive Funktion. Da h eine Abbildung von L auf ℕ ist, ist g eine Abbildung von f (L) auf ℕ. Da die Funktion g die Eigenschaft hat jeden String in f^{1} (L) auf eine natürliche Zahl abzubilden und jede natürliche Zahl wiederum auf einen String in f^{1} (L), ist f^{1} (L) rekursiv aufzählbar.

AuB: 11.Hausaufgabe (23.01.23) - Cora Zeitler

Montag, 23. Januar 2023 00:01

Aufgabe 1:
Zeigen Sie, dass folgende Sprachen entscheidbar sind:

(a) $E_{NFA} = \{\langle A\rangle \mid A \text{ ist NFA und } L(A) = \emptyset\}$, die Menge aller Kodierungen nichtdeterministischer endlicher Automaten, die kein Wort erzeugen.

(b) $EQ_{REG} = \{\langle G_1, G_2\rangle \mid G_1, G_2$ sind reguläre Grammatiken und $L(G_1) = L(G_2)\}$, die Menge aller Paare von Kodierungen regulärer Grammatiken für die gilt, dass die von ihnen erzeugten Sprachen gleich sind.

(c) $A = \{\langle G\rangle \mid G \text{ ist kfG über } \{0,1\} \text{ und } (1)^* \cap L(G) \neq \emptyset\}$, die Menge aller Kodierungen kontextfreier Grammatiken über dem Alphabet $\{0,1\}$, die mindestens ein Wort erzeugen, das nur aus Einsen besteht.

(d) $B = \{\langle G\rangle \mid G \text{ ist reguläre Grammatik und } L(G) = \{w \in \{0,1\}^* \mid \#_0(w) \equiv_3 1\}\}$

(e) $C = \{\langle G\rangle \mid G \text{ ist reguläre Grammatik und } \exists w \in L(G), w \text{ beginnt mit 1 und endet auf } 1\}$

(23 Punkte)

1) (a) $E_{NFA} = \{\langle A\rangle \mid A \text{ ist NFA und } L(A) = \emptyset\}$, die Menge aller Kodierungen nichtdeterministischer endlicher Automaten, die kein Wort erzeugen.

```
Eingabe: NFA
for i in Endzustände
        makiere
for j in (Zustände, die in makierte Zustände überführen)
        makiere j
for k in Startzustände
        if (k == markiert)
                return  0
return 1
```

Ich nehme an nur nicht markierte Zustände

Erklärung	- erst markiert man jeden Endzustand - dann werden alle Zustände markiert, die in einen bereits markierte Zustände überführt werden können --> wird solange Wiederholt bis kein Zustand mehr markiert werden kann - nach der Schleife schaut man ob ein Startzustand markiert ist --> wenn ja , dann gebe 0 aus, sonst 1

Den Ansatz kurz begründen

(✓)3

(b) $EQ_{REG} = \{\langle G_1, G_2\rangle \mid G_1, G_2$ sind reguläre Grammatiken und $L(G_1) = L(G_2)\}$, die Menge aller Paare von Kodierungen regulärer Grammatiken für die gilt, dass die von ihnen erzeugten Sprachen gleich sind.

Eingabe: REG

(c) $A = \{\langle G\rangle \mid G \text{ ist kfG über } \{0,1\} \text{ und } (1)^* \cap L(G) \neq \emptyset\}$, die Menge aller Kodierungen kontextfreier Grammatiken über dem Alphabet $\{0,1\}$, die mindestens ein Wort erzeugen, das nur aus Einsen besteht.

- die TM sucht ob es mindestens ein Wort gibt das nur aus 1 besteht

```
in Chomsky-Normalform überführen
for  i in Terminale          // i ist Nichtterminal
        if ( i == "1")
                markiere i
for j in markierte Nichtterminale
        for k in Nichtterminale
                if (delta (überführt "1") existiert)
                        markiere k
for m  in Startzustände
        if (m == markiert)
                return 1
return 0
```

?

Ansatz gut, Pseudocode ist fragwürdig

(✓)3

(d) $B = \{\langle G\rangle \mid G$ ist reguläre Grammatik und $L(G) = \{w \in \{0,1\}^* \mid \#_0(w) \equiv_3 1\}\}$

$G' = (\{0,1\}, \{S, E, Z\}, S, R)$

$R = \{ S \to 0E \mid 1S \mid 0;$

$E \to 0Z \mid 1E \mid 1;$

$Z \to 0S \mid 1Z \}$

↳ die Grammatik G' erzeugt die Sprache $\{w \in \{0,1\}^* \mid \#_0(w) \equiv_3 1\}$

↳ mit B überprüft man, ob die eingegebene Grammatik G, die gleiche Sprache, wie G' erzeugt

↳ wenn Sie übereinstimmen, gebe 1 aus, sonst 0

B ist eine Menge

(✓)2

Man kann DFAs vergleichen, für Grammatiken müsstest du G angeben

(e) $C = \{\langle G\rangle \mid G$ ist reguläre Grammatik und $\exists w \in L(G), w$ beginnt mit 1 und endet auf 1$\}$

```
-in Chomsky-Normalform überführen
for i in Terminale          // i ist Nichtterminal
        if (i == "1")
                markiere i
for j in markierte Nichtterminale
        for k in  Nichtterminale
                if (delta(k,j) existert)
                        markiere k
for m in Startterminale
        for n in makierte Nichtterminale
                if (m == markiert UND delta("1", m, n, irgendwohin) existiert)
                        return 1
return 0
```

siehe c)

(✓)2

10/23

Aufgabe 2:
Beweisen Sie folgende Aussagen:

(a) Wenn eine Sprache $L \subseteq \Sigma^*$ entscheidbar ist und eine Funktion $f : \Sigma^* \to \Sigma^*$ total definiert und berechenbar ist, dann ist auch $f^{-1}(L)$ entscheidbar.

(b) Wenn eine Sprache $L \subseteq \Sigma^*$ rekursiv aufzählbar ist und eine Funktion $f : \Sigma^* \to \Sigma^*$ berechenbar ist, dann ist auch $f^{-1}(L)$ rekursiv aufzählbar.

(12 Punkte)

2)a)	Eine Möglichkeit, dies zu beweisen, ist zu zeigen, dass man einen Algorithmus verwenden kann, um die Funktion f^{-1} (L) zu berechnen, indem man die bereits gegebenen Eigenschaften von L und f verwendet. Da L entscheidbar ist, kann man einen Algorithmus verwenden, um jeden String in Σ* zu testen, ob er in L enthalten ist. Da f total definiert und berechenbar ist, kann man jeden String in Σ* in einen anderen String in Σ* umwandeln. Um f^{-1} (L) zu berechnen, kann man nun jeden String in Σ* verwenden, um ihn mithilfe der Funktion f umzuwandeln, und dann zu testen, ob das Ergebnis in L enthalten ist. Wenn es in L enthalten ist, gehört der ursprüngliche String zu f^{-1} (L). Wenn nicht, gehört er nicht dazu. ✓

kurz begründen

	Da man einen Algorithmus verwenden kann, um f^{1} (L) zu berechnen, ist es entscheidbar
2)b)	Um zu beweisen, dass f^{1} (L) rekursiv aufzählbar ist, müssen wir zeigen, dass es eine rekursive Funktion g gibt, die jeden String in f^{1} (L) auf eine natürliche Zahl abbildet und umgekehrt jede natürliche Zahl auf einen String in f^{1} (L) abbildet. Da L rekursiv aufzählbar ist, gibt es eine rekursive Funktion h, die jeden String in L auf eine natürliche Zahl abbildet und umgekehrt jede natürliche Zahl auf einen String in L abbildet. Da f berechenbar ist, gibt es eine berechenbare Funktion f, die jeden String in Σ* in einen anderen String in Σ* abbildet. Um f^{1} (L) zu berechnen, können wir eine neue Funktion g definieren: $g(n) = h^{-1}(f(h(n)))$. Da h und f berechenbar sind, ist g auch berechenbar. Da h und h^{1} rekursive Funktionen sind, ist g eine rekursive Funktion. Da h eine Abbildung von L auf ℕ ist, ist g eine Abbildung von f (L) auf ℕ. Da die Funktion g die Eigenschaft hat jeden String in f^{1} (L) auf eine natürliche Zahl abzubilden und jede natürliche Zahl wiederum auf einen String in f^{1} (L), ist f^{1} (L) rekursiv aufzählbar.

5/12

15/35