

Blatt 4

Soriano Ortiz, Karime (469451)

Grez Leisinger, Karl Nicolas (471888)

Galytskyy, Danylo (469032)

9. Mai 2025

Summe: 13/15

Aufgabe 1

Überprüfen Sie die Korrektheit der Aussage „ $f(n)$ ist in $\mathcal{O}(g(n))$ “ für die folgenden Funktionen f und g . Nutzen Sie dazu entweder die in der Vorlesung vorgestellte Definition oder die folgende äquivalente Formulierung. (Wir gehen dabei davon aus, dass g nur an endlich vielen Stellen 0 ist, und ignorieren diese Stellen bei der Zahlenfolge für den \limsup .)

$$\text{„}f(n) \text{ ist } \mathcal{O}(g(n))\text{“} \iff \limsup_{n \rightarrow \infty} \frac{|f(n)|}{|g(n)|} = c < \infty$$

Begründen Sie jeweils Ihre Antwort.

(a) $f(n) = 500$ und $g(n) = 1$

(b) $f(n) = 13 \cdot n^2 + n$ und $g(n) = (n+2)^3$

(c) $f(n) = \sqrt{n^5}$ und $g(n) = n^2 \log(n^5)$

(d) $f(n) = \begin{cases} n^2 & \text{falls } n \text{ durch 2 teilbar ist} \\ n & \text{sonst} \end{cases}$ und $g(n) = \sqrt{n^3 + 2n}$

(e) $f(n) = \begin{cases} 2^n & \text{falls } 0 \leq n \leq 10 \text{ oder } n = 2025, \\ n & \text{sonst} \end{cases}$ und $g(n) = n^2$

a) $\frac{|f(n)|}{|g(n)|} = \frac{500}{1} \Rightarrow \limsup_{n \rightarrow \infty} \left| \frac{f(n)}{g(n)} \right| = 500 < \infty \Leftrightarrow \text{„}f(n) \text{ ist } \mathcal{O}(g(n))\text{“}$ +1

b) $\frac{|f(n)|}{|g(n)|} \sim \frac{13n^2}{n^3} = \frac{13}{n} \Rightarrow \limsup_{n \rightarrow \infty} \left| \frac{f(n)}{g(n)} \right| = 0 < \infty \Leftrightarrow \text{„}f(n) \text{ ist } \mathcal{O}(g(n))\text{“}$ +1

c) $\frac{|f(n)|}{|g(n)|} = \frac{n^{2.5}}{5n^2 \log n} = \frac{n^{0.5}}{5 \log n} \Rightarrow \limsup_{n \rightarrow \infty} \left| \frac{f(n)}{g(n)} \right| = \infty \Rightarrow \text{„}f(n) \text{ ist nicht } \mathcal{O}(g(n))\text{“}$ +1

d) Für gerade n : $\frac{f(n)}{g(n)} = \frac{n^2}{n^{1.5}} = n^{0.5} \rightarrow \infty$

Für ungerade n : $\frac{f(n)}{g(n)} = \frac{n}{n^{1.5}} = \frac{1}{n^{0.5}} \rightarrow 0$

$\Rightarrow \limsup_{n \rightarrow \infty} \left| \frac{f(n)}{g(n)} \right| = \infty \Rightarrow \text{„}f(n) \text{ ist nicht } \mathcal{O}(g(n))\text{“}$ +1

e) $f(n) = \begin{cases} 2^n & \text{falls } 0 \leq n \leq 10 \text{ oder } n = 2025 \\ n & \text{sonst} \end{cases}, \quad g(n) = n^2$

$\frac{f(n)}{g(n)} = \frac{n}{n^2} = \frac{1}{n} \rightarrow 0 \quad (\text{außer an endlich vielen Stellen})$

$$\Rightarrow \limsup_{n \rightarrow \infty} \left| \frac{f(n)}{g(n)} \right| = 0 < \infty \Leftrightarrow „f(n) ist \mathcal{O}(g(n))“$$

+1

Aufgabe 2

Das *Alle-Wörter-Problem* über dem Alphabet $\Sigma = \{a, b, c\}$ ist das Entscheidungsproblem, ob ein gegebener NEA alle Wörter über dem Alphabet akzeptiert. Formal:

- **Gegeben:** NEA $\mathcal{A} = (Q, \Sigma, q_0, \Delta, F)$
- **Frage:** Gilt für alle Wörter $w \in \Sigma^*$, dass $w \in L(\mathcal{A})$?

- (a) Benutzen Sie die in der Vorlesung vorgestellten Algorithmen und Methoden, um einen Algorithmus für das Alle-Wörter-Problem zu bekommen. Geben Sie Ihren Algorithmus in Form von Pseudocode an. Wenn Sie einen Algorithmus der Vorlesung benutzen, machen Sie dies entsprechend kenntlich.
- (b) Leiten Sie für Ihren Algorithmus schrittweise eine obere Laufzeitschranke in der \mathcal{O} -Notation her.

a)

```

# Hauptfunktion: Löse das Alle-Wörter-Problem für einen NEA
Funktion ALLE_WÖRTER(NEA A):
    # Schritt 1: Konvertiere NEA zu DEA
    DEA A_dea = POTENZMENGEN_KONSTRUKTION(A)

    # Schritt 2: Komplementiere den DEA
    DEA A_komplement = KOMPLEMENT_DEA(A_dea)

    # Schritt 3: Prüfe, ob das Komplement leer ist
    return IST_LEER(A_komplement) +2
Ende Funktion

# --- Hilfsfunktionen --- (hätten nicht angegeben werden müssen)

# Konvertiert NEA zu DEA mittels Potenzmengenkonstruktion
Funktion POTENZMENGEN_KONSTRUKTION(NEA A):
    # Initialisierung
    startzustand = _HÜLLE({A.q0}, A)
    DEA_Zustände = {startzustand}
    DEA_Übergänge = leere_Übergangstabelle()
    warteschlange = [startzustand]
    DEA_Endzustände = leere_Menge()

    # Startzustand auf Endzustand prüfen

```

```

falls ist_Endzustand(startzustand, A):
    DEA_Endzustände.füge_hinzu(startzustand)

# Hauptverarbeitung
solange warteschlange nicht leer:
    aktueller_Zustand = warteschlange.entferne_erstes()

    für jedes Symbol a in A.:
        # Berechne nächsten Zustand
        nächster_Zustand = leere_Menge()
        für jeden q in aktueller_Zustand:
            für jeden (q, a, q') in A.:
                nächster_Zustand.füge_hinzu(q')

    nächster_Zustand = _HÜLLE(nächster_Zustand, A)

# Füge neuen Zustand hinzu falls neu
falls nächster_Zustand nicht in DEA_Zustände:
    DEA_Zustände.füge_hinzu(nächster_Zustand)
    warteschlange.füge_hinzu(nächster_Zustand)

# Prüfe auf Endzustand
falls ist_Endzustand(nächster_Zustand, A):
    DEA_Endzustände.füge_hinzu(nächster_Zustand)

# Speichere Übergang
DEA_Übergänge[aktueller_Zustand, a] = nächster_Zustand

return DEA(
    Q = DEA_Zustände,
    = A.,
    q0 = startzustand,
    = DEA_Übergänge,
    F = DEA_Endzustände
)
Ende Funktion

# Berechnet die -Hülle einer Zustandsmenge
Funktion _HÜLLE(Zustandsmenge S, NEA A):
    hülle = kopiere(S)
    stack = liste_aus(S)

    solange stack nicht leer:
        q = stack.entferne_letztes()
        für jeden (q, , q') in A.:
            falls q' nicht in hülle:

```

```

        hülle.füge_hinzu(q')
        stack.füge_hinzu(q')

    return hülle
Ende Funktion

# Prüft, ob ein DEA-Zustand (Menge von NEA-Zuständen) ein Endzustand ist
Funktion ist_Endzustand(Zustandsmenge S, NEA A):
    für jeden q in S:
        falls q in A.F:
            return wahr
        return falsch
Ende Funktion

# Komplementiert einen DEA
Funktion KOMPLEMENT_DEA(DEA A):
    return DEA(
        Q = A.Q,
        = A.,
        q0 = A.q0,
        = A.,
        F = A.Q \ A.F # Alle nicht-Endzustände werden zu Endzuständen
    )
Ende Funktion

# Prüft, ob ein DEA die leere Sprache akzeptiert
Funktion IST_LEER(DEA A):
    besucht = leere_Menge()
    warteschlange = [A.q0]

    solange warteschlange nicht leer:
        zustand = warteschlange.entferne_erstes()

        falls zustand in A.F:
            return falsch # Sprache nicht leer

        falls zustand nicht in besucht:
            besucht.füge_hinzu(zustand)
            für jedes a in A.:
                nächster = A.(zustand, a)
                warteschlange.füge_hinzu(nächster)

    return wahr # Sprache ist leer
Ende Funktion

```

b) Laufzeitanalyse des Alle-Wörter-Problems

Schritt	Komplexität	Begründung
NEA \rightarrow DEA	$O(2^n \cdot \Sigma)$	Potenzmengenkonstruktion
DEA-Komplement	$O(2^n)$	Einfache Zustandsinvertierung
Leerheitsprüfung	$O(2^n \cdot \Sigma)$	BFS auf DEA-Graph
Gesamt	$O(2^n)$	Exponentiell in NEA-Zuständen

+1

Aufgabe 3

Startzustände fehlen -1

