Universität Leipzig
Institut für Informatik

Prof. Dr. Manfred Droste
Philipp Riemer

Automatentheorie

Projekt zur Programmerstellung für Entscheidbarkeitsresultate und Konstruktionen in der Automatentheorie

(Einzureichen bis 11.01.2026 um 23:59 im Moodle-Kurs)

- Ziel des Projektes ist es zu verstehen, inwieweit sich derzeitige LLMs für das Erstellen von Programmen für ausgewählte Entscheidbarkeitsresultate der Automatentheorie verwenden lassen.

  Als Grundlage hierzu dienen die Entscheidungsprobleme aus der Vorlesung in Theorem 1.13. Sie können ein LLM und eine (verständliche[1]) Programmiersprache Ihrer Wahl benutzen. Bitte vermerken Sie Ihre Entscheidungen auf der Abgabe.

- Die Aufgabenstellungen sind bewusst offen gehalten. Bei Fragen oder Problemen können Sie jederzeit im Moodle-Kurs sich mit Ihren Kommilitonen austauschen oder sich über Moodle bei Philipp Riemer melden.

- Die Aufgaben sollen in Gruppen (ca. 2–3 Personen) bearbeitet werden. Jede Abgabe soll nur von einer Person aus der Gruppe abgegeben werden und alle Gruppenmitglieder sind auf der Abgabe zu vermerken.

  Alle Gruppenmitglieder sollen in der Lage sein, die Ergebnisse kurz mündlich zu präsentieren.

- Mit Einreichung der Aufgaben können bis zu 10 Bonuspunkte (BP) für die Klausur erreicht werden. Die erreichten Punkte werden allen Gruppenmitgliedern als Prozentpunkte in der Klausur angerechnet.

[1] Zum Beispiel: Python, C oder Rust.

Aufgabe B1 (4 BP)
Entwickeln Sie mithilfe des LLMs ein Programm zur Lösung des Leerheitsproblems für endliche Automaten. Das Programm soll zur folgenden Beschreibung passen:

IN: Endlicher Automat $\mathcal{A}$.

OUT: Falls $L(\mathcal{A}) \neq \varnothing$, ein Beispielwort $w \in L(\mathcal{A})$.
Sonst $\perp$ (bel. Symbol für $L(\mathcal{A}) = \varnothing$).

Gehen Sie hierfür wie folgt vor:

(a) Beschreiben Sie einen Algorithmus zur Lösung des Leerheitsproblem mathematisch exakt (z.B. als mathematische Konstruktion, Pseudocode, Flussdiagramm o.Ä.).

(b) Lassen Sie das LLM Ihre mathematische Beschreibung in Quellcode übersetzen.

Wichtig: Das Sprachmodell sollte vorher *nicht* wissen oder erkennen, dass es um das Leerheitsproblem geht — sonst besteht die Gefahr, dass das Programm einfach im Internet gesucht und verwendet wird. Um dies sicherzustellen sollten Sie auch immer einen neuen Chat öffnen.

(c) Überprüfen Sie das resultierende Programm auf Korrektheit, indem Sie den Quellcode analysieren und kleine bis große Beispiel-Eingaben testen. Halten Sie Ihre Beobachtungen kurz fest.

Hinweis: Eventuell hilft es, bei (a) zunächst das reine Entscheidungsproblem (ja/nein) zu lösen und noch keine Beispielwörter für $L(\mathcal{A}) \neq \varnothing$ zu liefern.

Aufgabe B2 (2 BP)
Entwickeln Sie einen Algorithmus, der das Leerheitsproblem für das Komplement der Sprache $L(\mathcal{A})^c$ löst. Das Programm soll zur folgenden Beschreibung passen:

IN: Endlicher Automat $\mathcal{A}$.

OUT: Falls $L(\mathcal{A})^c \neq \varnothing$, ein Beispielwort $w \in L(\mathcal{A})^c$.
Sonst $\perp$ (bel. Symbol für $L(\mathcal{A})^c = \varnothing$).

Gehen Sie hierbei wie in Aufgabe B1 vor und passen Sie den Algorithmus an. Orientieren Sie sich an Theorem 1.7 aus der Vorlesung, vermeiden Sie aber den exponentiellen Konstruktionsaufwand und wenden Sie die Konstruktion in Ihrem Algorithmus „*on the fly*“ an.

Aufgabe B3 (2 BP)
Entwickeln Sie einen Algorithmus, der das Leerheitsproblem für den Schnitt der Sprachen von zwei gegebenen Automaten $\mathcal{A}_1$ und $\mathcal{A}_2$ löst (also für die Sprache $L(\mathcal{A}_1) \cap L(\mathcal{A}_2)$). Das Programm soll zur folgenden Beschreibung passen:

IN: Endliche Automaten $\mathcal{A}_1, \mathcal{A}_2$.

OUT: Falls $L(\mathcal{A}_1) \cap L(\mathcal{A}_2) \neq \varnothing$, ein Beispielwort $w \in L(\mathcal{A}_1) \cap L(\mathcal{A}_2)$.
Sonst $\perp$ (bel. Symbol für $L(\mathcal{A}_1) \cap L(\mathcal{A}_2) = \varnothing$).

Gehen Sie hierbei wie in Aufgabe B1 vor und passen Sie den Algorithmus an. Orientieren Sie sich an der Konstruktion von „Produktautomaten“ aus der Vorlesung und den Übungen (1.4c). Wenden Sie die Konstruktion in Ihrem Algorithmus „*on the fly*“ an.

Aufgabe B4 (2 BP)
Entwickeln Sie einen Algorithmus, der für gegebene Automaten $\mathcal{A}_1$ und $\mathcal{A}_2$ das Inklusionsproblem beantwortet.

IN: Endliche Automaten $\mathcal{A}_1, \mathcal{A}_2$.

OUT: Falls $L(\mathcal{A}_1) \not\subseteq L(\mathcal{A}_2)$, ein Beispielwort $w \in L(\mathcal{A}_1) \setminus L(\mathcal{A}_2)$.
Sonst $\perp$ (bel. Symbol für $L(\mathcal{A}_1) \subseteq L(\mathcal{A}_2)$).

Gehen Sie hierbei wie in Aufgabe B1 vor und passen Sie den Algorithmus an. Setzen Sie dabei Ihre Algorithmen aus den Aufgaben B2 und B3 zusammen und nutzen Sie:

$$L(\mathcal{A}_1) \subseteq L(\mathcal{A}_2) \iff L(\mathcal{A}_1) \cap L(\mathcal{A}_2)^c = \varnothing$$