UNIKASSEL VERSITÄT | ELEKTROTECHNIK INFORMATIK

Berechnung einer Gewichtung regulärer Sprachen

Bachelorarbeit
zur Erlangung des Grades Bachelor of Science

von

Maurice Herwig

Matrikelnummer:	35362863
Vorgelegt im:	Fachgebiet Theoretische Informatik/Formale Methoden
Gutachter:	Prof. Dr. Martin Lange Prof. Dr. Claudia Fohry
Betreuer:	Dr. Florian Bruse
Bearbeitungszeit:	von 25.08.2021 bis 27.10.2021

Erklärung

Hiermit versichere ich, dass ich diese Arbeit selbstständig und nur mit den nach der Prüfungsordnung der Universität Kassel zugelassenen Hilfsmittel angefertigt habe. Sowie keine anderen als die im Literaturverzeichnis aufgelisteten Quellen verwendet habe.

Kassel, Oktober 2021

Maurice Herwig

Inhaltsverzeichnis

1 Einleitung

1.1 Motivation

Eine gute Bewertung von studentischen Abgaben ist ein wichtiger Bestandteil, um die Leistungen eines Studierenden einordnen zu können und dem Studierenden ein hilfreiches Feedback bezüglich der Aufgabe zu geben. Für Aufgabenstellungen, die beinhalten, dass die Studierenden eine reguläre Sprache angeben sollen, gestalte sich die Bewertung der Abgabe durchaus schwierig, vor allem, wenn die Korrektur automatisch erfolgen soll.
Ein existierendes Lerntool ist der „Automata Tutor“ TU München [4], dieser bietet den Studierenden das Bearbeiten verschiedenster Übungsaufgaben aus dem Bereich der formalen Sprachen, wie Aufgabenstellungen zum CYK-Algorithmus, der Chromsky-Normalform und vielen mehr, an. Unter anderem werden auch Aufgabenstellungen zur Konstruktion von regulären Sprachen gestellt. Für alle Aufgabenstellungen bekommt der Studierende ein automatisch erstelltes Feedback und eine Bewertung seiner Abgabe zurück. Die Bewertung für deterministischer endliche Automaten wird dabei durch den „Grading Algorithmus“[2] vorgenommen. Dabei wir die Bewertung aus der Kombination von drei Faktoren berechnet. Der erste Faktor gibt die Anzahl der syntaktischen Änderungen an, die benötigt werden, damit die Abgabe des Studierenden Äquivalent zur Abgabe der Musterlösung ist. Der zweite Faktor schätzt mithilfe der Entropie der symmetrischen Differenz den Anteil der Wörter, die sich in der symmetrischen Differenz befinden. Der dritte Faktor beschreibt Fehler, die durch falsches lesen oder verstehen der Aufgabenstellung entstehen.
Auch das Fachgebiet Fachgebiet Theoretische Informatik/Formale Methoden der Universität Kassel besitzt durch vorangegangen Arbeiten ein automatisierten Bewertungstool, dabei werden die Teilmengenbeziehungen der Sprache einer Abgabe mit einer Sprache, die die Musterlösung charakterisiert, untersucht. Dabei wird mittels eines Verfahrens zunächst betrachtet, ob es sich bei der abgegebenen Sprache um eine Teilmenge der Musterlösungssprache handelt, anschließend wird die Obermengenbeziehung der Abgabe zur Musterlösung untersucht. Liegen beide Teilmengenbeziehungen vor, so handelt es sich bei der Sprache der Abgabe um eine äquivalente Sprache zur Musterlösungssprache und somit ist diese Abgabe als eine korrekte Lösung zu betrachten. Die bisher implementierten Verfahren zur Bestimmung der Teilmengenbeziehungen beziehungsweise zum Lösen des Äquivalenzproblems zwischen zwei regulären Sprachen sind das betrachteten des Leerheitsproblems der symmetrischen Differenz [9], die Antiketten-Methode [13] sowie die Ramsey-Methode [5], wobei sich durch vorangegangene Arbeiten herausgestellt hat, dass die Antiketten-Methode bezüglich ihrer Laufzeit am effizientesten ist. Der verwendete Bewertungsmaßstab vergibt dann automatisiert pro vorhandene Teilmengenbeziehung die Hälfte der Punkte.
Allerdings ist dieser Bewertungsmaßstab nur für den Fall geeignet, wenn die Sprache der Abgabe äquivalent zu Musterlösungssprache ist und damit die volle Punktzahl vergeben werden kann. In allen anderen Fällen ist diese Bewertung kein guter Maßstab, da so den trivialen Abgaben der vollen Sprache Γ^* [1] und der leeren Sprache $\emptyset$ mindestens die Hälfte der Punkte zugeordnet wird, während abgegebene Automaten, die nur ein einziges Wort der Musterlösungssprache nicht akzeptieren und zusätzlich noch ein Wort mehr akzeptieren, keine Punkte zugeordnet wird. Daher ist es bisher unausweichlich, alle nicht korrekten Abgaben händisch nachzukontrollieren, um noch angemessene Teilpunkte vergeben zu können.

1.2 Ziel

Das Ziel dieser Bachelorarbeit ist es, eine Gewichtsfunktion ω zu entwickeln, die jeder formalen Sprache ein Gewicht zuordnet, wobei die konkrete Berechnung nur für die Sprachklasse der regulären Sprachen angestrebt wird. Dabei soll das Gewicht der leeren Sprache 0 betragen und das Gewicht der vollen Sprache den Wert 1 annehmen. Allen anderen Sprachen soll ein Gewicht zwischen 0 und 1 zugeordnet werden, wie hoch dieses Gewicht ist, hängt davon ab, wie hoch der Anteil der akzeptierenden Wörter der gegebenen Sprache ist. Allerdings soll die Wortlänge bestimmen, wie stark ein akzeptierendes Wort gewichtet wird, dabei sollen kürzere Wörter höher als längere Wörter gewichtet werden.
Durch diese Gewichtung einer Sprache soll das Bewertungstool so erweitert werden, dass eine weitgehend automatisierte Bewertung von studentischen Abgaben möglich ist. Dazu soll jeder Abgabe das Gewicht der symmetrischen Differenz zwischen der Sprache der studentischen Abgabe und der Sprache der Musterlösung zugeordnet werden, sodass dieses Gewicht beschreibt, wie weit die Sprache der Abgabe von der Sprache der Musterlösung entfernt ist. Hier bedeutet, dass wenn einer Sprache ein höheres Gewicht

[1] Anstatt der herkömmlichen Notation Σ für das Alphabet wird in dieser Arbeit Γ verwendet.

als eine andere Sprache aufweist sie auch weiter von der Musterlösung entfernt ist und die Abgabe auf keinen Fall mehr Punkte bekommen sollte als eine Abgabe, die eine Sprache beschreibt, die näher an der Musterlösung liegt und damit auch ein geringeres Gewicht aufweist. Dabei bedeutet das Gewicht 0, dass eine äquivalente Lösung zur Musterlösung abgegeben wurde, während das Gewicht 1 besagt, dass genau das Komplement zur Musterlösung abgegeben wurden ist. Durch diese Erweiterung soll der Korrigierende in der Lage sein, eine feinere und fairere Bewertung vornehmen zu können.

1.3 Aufbau der Arbeit

Die Arbeit ist so aufgebaut, dass wir uns in Kapitel 2 zunächst die regulären Sprachen, insbesondere deren Charakterisierung durch deterministische endliche Automaten, anschauen, um die Grundlage für den weiteren Verlauf dieser Arbeit zu schaffen. Darauf aufbauend zeigen wir in Kapitel 3, wie wir jeder (regulären) Sprache ein Gewicht zuordnen können, dafür benötigen wir in den meisten Fällen den Anteil der enthaltenden Wörter einer Sprache für eine bestimmte Wortlänge. Darum geben wir in Kapitel 4 zwei Verfahren an, mit denen wir den Anteil der enthaltenden Wörter für deterministische endliche Automaten effizient bestimmen können. In Kapitel 5 gehen wir dann noch auf die praktische Anwendung der Gewichtung einer Sprache ein, um damit studentische Abgaben bewerten zu können, indem wir das Gewicht auf der symmetrischen Differenz zwischen der Sprache der Musterlösung und der Sprache der studentischen Abgabe bestimmen. Außerdem ordnen wir experimentell gewonnene Gewichte ein, bevor wir im abschließenden Kapitel 6 ein Fazit dieser Arbeit ziehen und mögliche Erweiterungen vorstellen.

2 Reguläre Sprachen

Die Menge der regulären Sprachen sind die Sprachen, die durch einen nichtdeterministischen endlichen Automaten oder einem deterministischen endlichen Automaten akzeptiert werden. Daher schauen wir uns in diesem Kapitel die Charakterisierung einer regulären Sprache durch nichtdeterministische endliche Automaten (Abschnitt 2.1) und deterministische endliche Automaten (Abschnitt 2.2) an.
Die Definitionen sind dabei aus dem Buch „Theoretische Informatik - kurz gefasst“ [9] entnommen. Dazu betrachten wir zunächst die Begriffe Alphabet, Wort und Sprache, wobei wir für das Alphabet nicht die herkömmliche Notation Σ verwenden. Dies liegt daran, dass wir im weiteren Verlauf dieser Arbeit das Alphabet in einigen Formeln in Kombination mit dem Summenzeichen $\sum$ verwenden und diese so nur schwer zu unterscheiden wären.

- Ein **Alphabet** Γ ist eine endliche, nicht leere Menge von Zeichen.
- Ein **Wort** w ist eine endliche Folge von Zeichen aus dem Alphabet.
  - $|w|$ beschreibt die Wortlänge des Wortes w.
  - ϵ beschreibt, dass leere Wort, welches als einziges die Länge 0 besitzt.
  - Γ^n ist die Menge aller Wörter über dem Alphabet Γ für die gilt $|w| = n$.
  - Γ^* ist die Menge aller Wörter über dem Alphabet Γ.
  - $w = uv$ ist die Konkatenation des Wortes w aus den beiden Wörtern u und v.
- Eine **(formale) Sprache** L auf dem Alphabet Γ ist eine Menge $L \subseteq \Gamma^*$ von Wörtern.

2.1 Nichtdeterministische endliche Automaten

Ein nichtdeterministischer endlicher Automat oder in englischer Sprache nondeterministic finite Automation (daher kurz NFA), ist eine Charakterisierung der regulären Sprachen. Um die reguläre Sprache zu bestimmen, die durch einen NFA repräsentiert werden, schauen wir uns im Folgenden die Syntax sowie die Semantik auf einem NFA an.

Syntax. Ein NFA ist ein 5-Tupel $A = (Q, \Gamma, \delta, Q_0, Q_e)$, wobei gilt:

- Q ist eine endliche, nicht leere Menge an Zuständen.
- Γ ist das Alphabet des Automaten.
- $\delta \subseteq Q \times \Gamma \times Q$ ist die Transitionsrelation.
- $Q_0 \subseteq Q$ ist die Menge der Startzustände.
- $Q_e \subseteq Q$ ist die Menge der Endzustände.

Semantik. Die Semantik eines NFA A der durch das 5-Tupel $A = (Q, \Gamma, \delta, Q_0, Q_e)$ gegeben ist, ist folgendermaßen definiert.

- Eine **Konfiguration** C ist ein Wort qu mit $q \in Q$ und $u \in \Gamma^*$. Dies beschreibt das Teilwort u des Eingabewortes $w \in \Gamma^*$, welches noch von dem Automaten gelesen werden muss und den aktuellen Zustand q in dem sich der Automat befindet.
  - Eine **Startkonfiguration** $q_0 w$ mit $q_0 \in Q_0$ ist eine Konfiguration, bei der sich der aktuelle Zustand der Konfiguration in der Menge der Startzustände befindet und bei der w das zu lesende Wort ist.
  - Eine **akzeptierende Konfiguration** $q\epsilon$ mit $q \in Q_e$ ist eine Konfiguration, bei der der aktuelle Zustand sich in der Menge der Endzustände befindet und kein Teilwort mehr von dem Automaten gelesen werden muss.
- Die **Schrittrelation** $\vdash_A$ ist eine binäre Relation auf der Menge aller Konfigurationen. Dabei ist eine Schrittrelation $pav \vdash_A qv$ mit $a \in \Gamma, v \in \Gamma^*$ und $p, q \in Q$ genau dann gegeben, wenn für die Transitionsrelation $(p, a, q) \in \delta$ gilt. Damit können wir durch Lesen des Buchstaben a von Zustand p zum Zustand q gelangen, sodass danach nur noch das Wort v von dem Automaten gelesen werden muss.

- Eine **Rechnung** auf dem Eingabewort $w \in \Gamma^*$ ist eine endliche Folge von Konfigurationen $C_0 \vdash_A C_1 \vdash_A \cdots \vdash_A C_n$. Dabei ist C_0 eine Startkonfiguration. Durch $C_i \vdash_A^* C_j$ kann man verkürzend ausdrücken, dass eine Rechnung von C_i nach C_j existiert. Durch $C_i \vdash_A^n C_j$ lässt sich zusätzlich noch angeben, dass die Rechnung die Länge n aufweist.
  - Eine Rechnung heißt **akzeptierende Rechnung**, wenn C_n **eine** akzeptierende Konfiguration ist.
  - Eine Rechnung heißt **verwerfende Rechnung**, wenn C_n **keine** akzeptierende Konfiguration ist und das Wort w zu Ende gelesen wurde.
  - Eine **Teilrechnung** ist eine Rechnung, bei der C_0 nicht zwingend eine Startkonfiguration sein muss.
  - Die **Länge** einer Rechnung definieren wir als die Länge des Wortes, welches von dem Automaten gelesen wird.

Die akzeptierte Sprache. Eine Sprache heißt regulär, wenn es einen NFA gibt, der die Menge der Wörter, die in dieser Sprache liegt, beschreibt. Dabei liegt ein Wort genau dann in der Sprache, wenn für das Wort auf dem NFA mindestens eine akzeptierende Rechnung existiert.

$$L(A) = \{w \mid \text{ es gibt eine Rechnung } pw \vdash_A^* q\epsilon \text{ mit } p \in Q_0 \text{ und } q \in Q_e\}$$

Beispiel 1. Zum Schluss dieses Abschnitts möchten wir anhand eines Beispiels betrachten, wie eine reguläre Sprache durch einen NFA beschrieben wird. Dazu betrachten wir den in Abbildung 1 angegebenen NFA über dem Alphabet $\Gamma = \{a, b\}$. Für diesen erkennen wir, dass wir ausgehen von dem einzigen Startzustand q_0 nur den Buchstaben a lesen können und uns nach einem Schritt auf dem Automaten immer im Zustand q_1 befinden. Damit muss jedes Wort, welches von dem Automaten akzeptiert wird, mit dem Buchstaben a beginnen. Von q_1 aus können wir anschließend durch Lesen eines beliebigen Buchstabens des Alphabets in den Zustand q_2 gehen als auch weiterhin in q_1 bleiben. Da es sich nur bei dem Zustand q_2 um einem akzeptierenden Zustand handelt, müssen alle akzeptierenden Rechnungen in dem Zustand q_2 enden. Da wir den Zyklus mit dem wir in dem Zustand q_1 bleiben, beliebig oft wiederholen können und dann durch Lesen des letzten Buchstabens in den Zustand q_2 gehen können, lässt sich erkennen, dass die Sprache alle Wörter akzeptiert, die mindestens die Länge 2 besitzen und mit dem Buchstaben a beginnen.

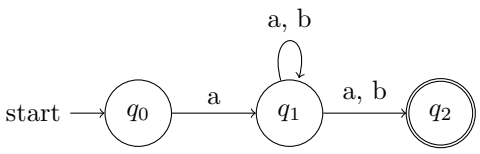

Abbildung 1: $L(A) = \{w \in \Gamma^* \mid |w| \geq 2 \text{ und } w \text{ beginnt mit a}\}$.

2.2 Deterministische endliche Automaten

Ein deterministischer endlicher Automat oder in englischer Sprache deterministic finite automaton (daher kurz DFA), ist eine weitere Charakterisierung der regulären Sprachen. DFAs besitzen im Gegensatz zu NFAs die Eigenschaft des Determinismus, was bedeutet das der Lauf auf dem Automaten für jedes Eingabewort eindeutig ist. Im Folgenden schauen wir uns die Syntax sowie die semantischen Unterschiede von DFAs zu NFAs an.

Syntax. Ein DFA ist ein 5-Tupel $A = (Q, \Gamma, \delta, q_0, Q_e)$, wobei gilt:

- Q ist eine endliche, nicht leere Menge an Zuständen.
- Γ ist das Alphabet des Automaten.
- $Q \times \Gamma \to Q$ ist die Transitionsfunktion.
- $q_0 \in Q$ ist der Anfangzustand.
- $Q_e \subseteq Q$ ist die Menge der Endzustände.

Semantik. Die Semantik eines DFA A der durch das 5-Tupel $A = (Q, \Gamma, \delta, q_0, Q_e)$ gegeben ist, lässt sich größtenteils von der Definition der Semantik eines NFAs übertragen. Daher gehen wir im Folgenden nur auf die Unterschiede der Semantik ein.

- Die **Schrittrelation** ist weiterhin eine binäre Relation auf der Menge aller Konfigurationen. Allerdings mit $pu \vdash_A qv$ genau dann, wenn $u = av$ und $\delta(p, a) = q$ für $a \in \Gamma$ gilt. Somit ist eine Schrittrelation zwischen zwei Konfigurationen C_1 und C_2 für einen DFA genau dann gegeben, wenn die Transitionsfunktion von dem aktuellen Zustand aus C_1 mit einem Buchstaben a aus dem Alphabet auf C_2 abbildet und das Wort, welches noch gelesen werden muss in C_1 gleich, dem Wort aus C_2 dem noch der Buchstabe a angehängt wird, ist.

- Ansonsten lässt sich die Definition der Semantik von der Semantik eines NFAs übertragen.

Die akzeptierte Sprache. Durch das Verwenden einer Transitionsfunktion anstatt einer Transitionsrelation und der Eigenschaft, dass nur genau ein Startzustand existiert, führt dies dazu, dass für jedes Wort auf dem Automaten genau eine Rechnung existiert, sodass die reguläre Sprache, die durch einen DFA akzeptiert wird, die Menge der Wörter ist, für die die Rechnung auf dem jeweiligen Wort in einer akzeptierenden Konfiguration endet.

$$L(A) = \{w \mid q_0 w \vdash_A^* q\epsilon \text{ mit } q \in Q_e\}$$

2.2.1 Die Potenzmengenkonstruktion

Nachdem wir NFAs und DFAs vorgestellt haben, möchten wir uns im Folgenden anschauen, wie wir einen NFA mithilfe der Potenzmengenkonstruktion in einen äquivalenten DFA umwandeln können. Dafür gehen wir davon aus, dass der NFA $A = (Q_A, \Gamma, \delta_A, Q_0^A, Q_e^A)$ gegeben ist. Diesen überführen wir in den DFA $B = (\mathcal{P}(Q_A), \Gamma, \delta_B, Q_0^A, Q_e^B)$, dessen Zustände die Potenzmenge der Zustände des NFA A sind. Somit werden unsere Zustände des DFA durch Mengen repräsentiert. Die Transitionsfunktion δ_B bildet für eine Menge $P \in \mathcal{P}(Q_A)$ in Kombination mit dem Buchstaben $a \in \Gamma$ auf die größte Menge $M \in \mathcal{P}(Q_A)$ ab, für die gilt, dass auf jedes Element der Menge M durch die Transitionsrelation des NFA A ausgehend von einem Zustand aus P mit dem Buchstaben a abgebildet wird. Formal definiert bedeutet dies für δ_B Folgendes:

$$\delta_B(P, a) \coloneqq \{q \mid \text{es gibt } p \in P \text{ und } (p, a, q) \in \delta_A \text{ mit } q \in Q_A\}$$

Die Menge der Endzustände ist die Mengen der Elemente der Potenzmenge, die mindestens einen Endzustand aus Q_e^A enthalten.

$$Q_e^B \coloneqq \{P \in \mathcal{P}(Q_A) \mid P \cap Q_e^A \neq \emptyset\}$$

Für den Startzustand können wir die Menge an Startzuständen des NFA verwenden, da diese Menge dadurch, dass unsere Zustände des DFA durch die Potenzmenge repräsentiert werden, genau einen Zustand darstellt.

Beispiel 2. Nachdem wir die Potenzmengenkonstruktion vorgestellt haben, möchten wir diese noch anhand eines Beispiels verdeutlichen. Dazu betrachten wir den NFA $A = (Q_A, \Gamma, \delta_A, Q_0^A, Q_e^A)$ (Abbildung 1), für den wir in Beispiel 1 bereits die Sprache bestimmt haben. Den NFA A überführen wir in den DFA $B = (\mathcal{P}(Q_A), \Gamma, \delta_B, Q_0^A, Q_e^B)$, indem wir zuerst die Potenzmenge der Zustände aus A bilden $\mathcal{P}(Q_A) = \{\{q_0\}, \{q_1\}, \{q_2\}, \{q_0, q_1\}, \{q_1, q_2\}, \{q_0, q_2\}, \{q_0, q_1, q_2\}, \emptyset\}$. Anschließend bestimmen wir die Transitionsfunktion δ_B nach obiger Formel. Die Tabelle 1 drückt die resultierende Transitionsfunktion aus. Danach bestimmen wir die Menge der Endzustände, dies sind alle Zustände, deren Mengenrepräsentation den einzigen Endzustand aus A q_2 enthalten. Daher gilt $Q_e^B = \{\{q_2\}, \{q_1, q_2\}, \{q_0, q_2\}, \{q_0, q_1, q_2\}\}$. Zum Schluss übernehmen wir noch die Menge der Startzustände $Q_0^A = \{q_0\}$ aus A, die unseren Startzustand beschreibt.
Eine grafische Darstellung des Konstruierten DFA B findet sich unter der Abbildung 2. Dabei sind die schwarzen Knoten die Konten, die wir von dem Startzustand aus erreichbar sind. Für alle anderen Knoten existiert keine Rechnung auf dem DFA, sodass diese Knoten nicht besucht werden können. Selbiges gilt auch für die Kanten, hier können wir ebenfalls nur die schwarzen Kanten bei einer Rechnung benutzen.

Zustand Q	$\delta_B(Q, a)$	$\delta_B(Q, b)$
$\{q_0\}$	$\{q_1\}$	$\emptyset$
$\{q_1\}$	$\{q_1, q_2\}$	$\{q_1, q_2\}$
$\{q_2\}$	$\emptyset$	$\emptyset$
$\{q_0, q_1\}$	$\{q_1, q_2\}$	$\{q_1, q_2\}$
$\{q_1, q_2\}$	$\{q_1, q_2\}$	$\{q_1, q_2\}$
$\{q_0, q_2\}$	$\{q_1\}$	$\emptyset$
$\{q_0, q_1, q_2\}$	$\{q_1, q_2\}$	$\{q_1, q_2\}$
$\emptyset$	$\emptyset$	$\emptyset$

Tabelle 1: Transitionsfunktion δ_B.

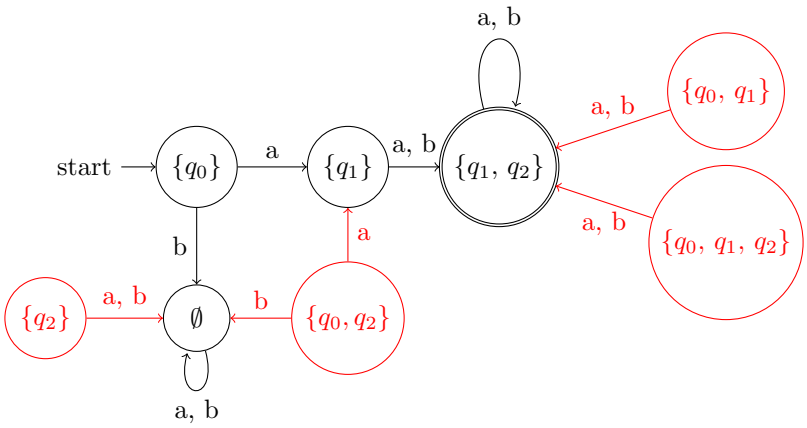

Abbildung 2: $L(B) = \{w \in \Gamma^* \mid |w| \geq 2 \text{ und } w \text{ beginnt mit a}\}$.

2.2.2 Komplementbildung regulärer Sprachen

Als Nächstes betrachten wir, dass Komplement einer Sprache. Das Komplement einer Sprache, der durch den DFA A gegeben ist, ist die Menge an Wörtern, für die die Rechnung auf dem DFA verwerfend ist. Dies sind alle Wörter, die keine akzeptierende Rechnung besitzen.

$$\overline{L(A)} = \overline{\{w \mid q_0 w \vdash_A^* q\epsilon \text{ mit } q \in Q_e\}} = \{w \mid q_0 w \vdash_A^* q\epsilon \text{ mit } q \notin Q_e\}$$

Aus der Definition des Komplements wird auch ersichtlich, wie wir aus einem DFA $A = (Q_A, \Gamma, \delta_A, q_0, Q_e^A)$ den Automaten des Komplements bilden können. Dafür müssen wir alle bisherigen Endzustände in nicht Endzuständen überführen und andersherum alle nicht Endzuständen in Endzustände überführen.

$$\overline{A} = (Q_A, \Gamma, \delta_A, \ q_0, Q_A \setminus Q_e^A)$$

Wie wir erkennen können, handelt es sich bei dem Komplementautomaten eines DFAs ebenfalls um einen DFA. Außerdem ist leicht zu begründen, warum durch Vertauschung der Endzustandsmenge das Komplement gebildet werden kann. Dies liegt daran, dass alle akzeptierenden Rechnungen ursprünglich in einem Endzustand enden, allerdings sollen diese Rechnungen nicht mehr akzeptierend sein, somit dürfen sie auch nicht mehr in einem Endzustand enden. Umgekehrt gilt, dass selbe für die verwerfenden Rechnungen.

Beispiel 3. Nachdem wir vorgestellt haben, wie wir für einen DFA, das Komplement bilden können, möchten wir für den Beispiel DFA B (Abbildung 2), den wir bereits in Beispiel 2 in einen DFA überführt haben, das Komplement angeben. Die grafische Angabe befindet sich unter der Abbildung 3, wobei zur Übersichtlichkeit nur die erreichbaren Zustände und Kanten eingezeichnet wurden. Des Weiteren haben wir die Zustände unbenannt, sodass es für uns einfacher ist, im nächsten Abschnitt dieses Beispiel fortzuführen.

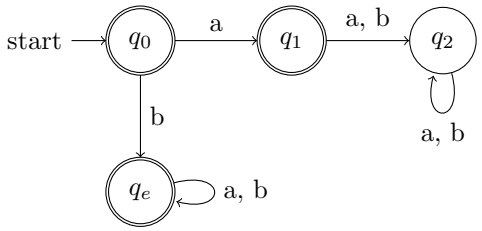

Abbildung 3: $L(C) = \overline{L(B)}$.

2.2.3 Die Produktkonstruktion

Um die Menge an Wörtern, die im Schnitt oder der Vereinigung von zwei regulären Sprachen liegen bestimmen zu können, benötigen wir den Produktautomaten der beiden Automaten. Den Produktautomat C für den DFA $A = (Q_A, \Gamma, \delta_A, q_0, Q_e^A)$ und den DFA $B = (Q_B, \Gamma, \delta_B, p_0, Q_e^B)$ ist definiert als der Automat $C = ((Q_A \times Q_B), \Gamma, \delta_C, (q_0, p_0), Q_e^C)$, dessen Zustände durch das kartesische Produkt der Zustände aus A und der Zustände aus B beschreiben werden. Dabei wird der Startzustand des Produktautomaten durch das geordnete Paar aus dem kartesischen Produkt beschrieben, welches sich aus dem Startzustand des Automaten A und dem Startzustand des Automaten B zusammensetzt. Die Transitionsfunktion δ_C ist für ein geordnetes Paar (q, p) mit $q \in Q_A$ und $p \in Q_B$ zusammen mit dem Buchstaben $a \in \Gamma$ definiert als das geordnete Paar (u, v), für dass δ_A mit q und a auf u und δ_B mit p und a auf v abbildet.

$$\delta_C((q, p), a) = (\delta_A(q, a), \delta_B(p, a))$$

Ob wir mithilfe des Produktautomaten den Schnitt oder die Vereinigung der beiden DFAs bestimmen, hängt von der Definition der Menge der Endzustände Q_e^C des Produktautomaten ab. Möchten wir die Vereinigung bestimmen, ist die Endzustandsmenge die Menge an geordneten Paaren, die unsere Zustände repräsentieren, für die mindestens einer der beiden Zuständen in dem jeweils ursprünglichen DFA als Endzustand deklariert ist.

$$\textit{Vereinigung}\text{: } Q_e^C = (Q_e^A \times Q_B) \cup (Q_A \times Q_e^B)$$

Soll hingegen durch den Produktautomaten der Schnitt der beiden Automaten bestimmt werden. So muss die Endzustandsmenge Q_e^C als die Menge definiert werden, die die geordneten Paare enthält, für die beide Zustände aus den ursprünglichen Automaten als Endzustände deklariert sind.

$$\textit{Schnitt}\text{: } Q_e^C = Q_e^A \times Q_e^B$$

Beispiel 4. Auch die Konstruktion eines Produktautomaten möchten wir anhand eines Beispiels verdeutlichen. Dafür nehmen wir den Komplementautomat C (Abbildung 3) und definieren einen zweiten DFA D, den wir in Abbildung 4 dargestellt haben. Für diese beiden Automaten möchten wir im Folgenden den Produktautomaten E bestimmen. Wie man leicht erkennen kann, ist der Startzustand des Produktautomaten $E = ((Q_C \times Q_D), \Gamma, \delta_E, (q_0, p_0), Q_e^C \times Q_e^D)$, der Zustand (q_0, p_0). Die Transitionsfunktion bestimmen wir nach obiger Formel. Die Tabelle 2 drückt dabei die resultierende Transitionsfunktion aus. Außerdem möchten wir, wie man bereits bei der Angabe des Produktautomaten E erkennen kann, in diesem Fall den Schnittautomaten der Automaten C und D bestimmen, sodass unsere Endzustandsmenge, die Menge $Q_e^C \times Q_e^D = \{(q_0, p_2), (q_1, p_2), (q_e, p_2)\}$ ist.

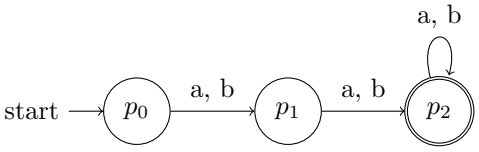

Abbildung 4: $L(D) = \{w \in \Gamma^* \mid |w| \geq 2\}$.

In Abbildung 5 befindet sich die grafische Darstellung des Produktautomaten E. Dabei sind die schwarzen Knoten die Knoten, die durch eine beliebige Rechnung erreicht werden können, Selbiges gilt auch für die Kanten.

Zustand Q	$\delta_E(Q,a)$	$\delta_E(Q,b)$
(q_0,p_0)	(q_1,p_1)	(q_e,p_1)
(q_0,p_1)	(q_1,p_2)	(q_e,p_2)
(q_0,p_2)	(q_1,p_2)	(q_e,p_2)
(q_1,p_0)	(q_2,p_1)	(q_2,p_1)
(q_1,p_1)	(q_2,p_2)	(q_2,p_2)
(q_1,p_2)	(q_2,p_2)	(q_2,p_2)

Zustand Q	$\delta_E(Q,a)$	$\delta_E(Q,b)$
(q_2,p_0)	(q_2,p_1)	(q_2,p_1)
(q_2,p_1)	(q_2,p_2)	(q_2,p_2)
(q_2,p_2)	(q_2,p_2)	(q_2,p_2)
(q_e,p_0)	(q_e,p_1)	(q_e,p_1)
(q_e,p_1)	(q_e,p_2)	(q_e,p_2)
(q_e,p_2)	(q_e,p_2)	(q_e,p_2)

Tabelle 2: Transitionsfunktion δ_E.

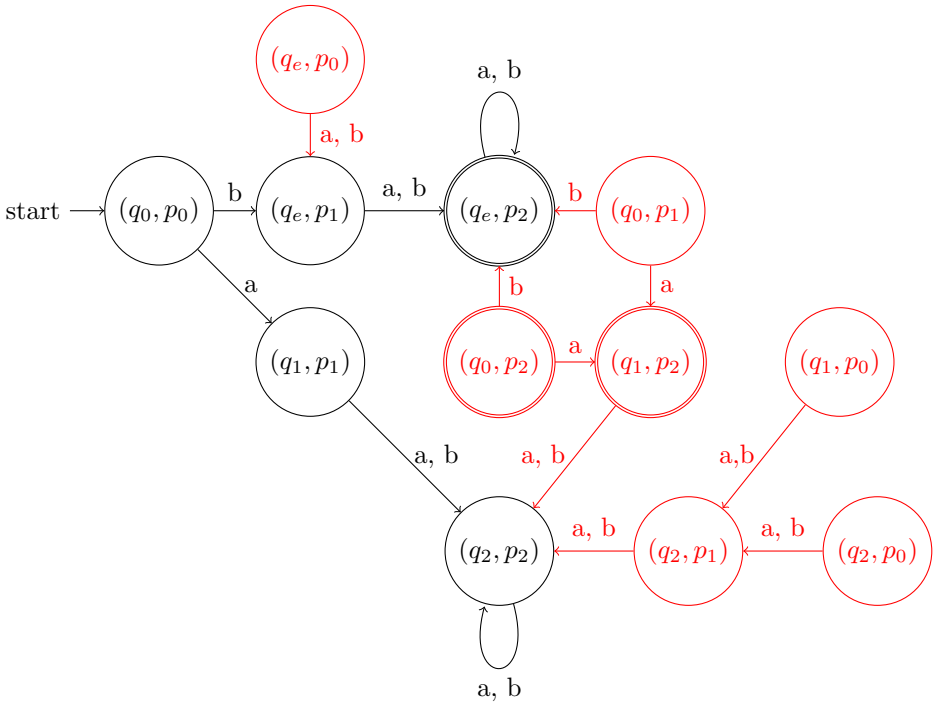

Abbildung 5: $L(E) = L(C) \cap L(D) = \{w \in \Gamma^* \mid |w| \geq 2 \text{ und } w \text{ beginnt mit b}\}$.

2.2.4 Die symmetrische Differenz regulärer Sprachen

Die symmetrische Differenz beinhaltet bei dem Vergleich von zwei DFAs, die die Sprachen L_1 und L_2 repräsentieren, die Menge an Wörtern, die nur von genau einem der beiden Automaten akzeptiert werden und von dem jeweils anderen nicht akzeptiert werden.

$$L_1 \triangle L_2 = (L_1 \cap \overline{L_2}) \cup (\overline{L_1} \cap L_2)$$

Um die symmetrische Differenz zu bestimmen, müssen wir zweimal, wie in der Formel angegeben, den Schnittautomat bilden. Dabei muss für beide Sprachen einmal das Komplement verwendet werden. Aus den Ergebnissen der Schnittautomaten bilden wir dann wiederum die Vereinigung.
Mithilfe der symmetrische Differenz lässt sich zum Beispiel das Äquivalenzproblem für DFAs lösen, da die Automaten genau denn äquivalent zueinander sind, wenn ihre symmetrische Differenz die leere Menge beschreibt. Auch wenn die Antiketten-Methode [13] wie bereits erwähnt eine praktisch deutlich schnelle Laufzeit zum Lösen des Äquivalenzproblems besitzt, verwenden wir die symmetrische Differenz, um damit die Differenz zwischen einer studentischen Abgabe und einer Musterlösung zu bestimmen, indem wir das Gewicht auf der Sprache der symmetrischen Differenz bestimmen (Abschnitt 5.1).
Die höhere Laufzeit ist dadurch zu erklären, dass wir auf DFAs durch das bilden eines Produktautomaten quadratisch viele Zustände für beide Schnittautomaten bekommen. Auch durch das Bilden der Vereinigung bekommen wir ebenfalls quadratisch viele Zustände bezüglich der Schnittautomaten. Für NFAs kommt noch erschwerend hinzu, dass wir diese mithilfe der Potenzmengenkonstruktion zunächst in einen äquivalenten DFA überführen müssen. Allerdings hat die symmetrische Differenz den Vorteil, dass sie als einzige Methode genau die Menge an Wörtern beschreibt, die nur von genau einem der beiden Automaten akzeptiert werden.

Beispiel 5. Zur Verdeutlichung der symmetrische Differenz gehen wir im Folgenden auf die symmetrische Differenz der beiden Automaten B (Abbildung 2) und D (Abbildung 4) ein, für die wir in Beispiel 4 bereits einen der beiden benötigten Produktautomaten gebildet haben. Den Produktautomat für $L(B) \cap \overline{L(D)}$ erhalten wir auf die gleiche Weise, dieser besitzt als Ergebnis allerdings die leere Menge, sodass die Vereinigung der beiden Schnittautomaten wieder den Automaten E (Abbildung 5) ergibt. Zur Übersichtlichkeit haben wir in Abbildung 6 noch einmal den Automaten für die symmetrische Differenz dargestellt, da wir ihn in weiteren Beispielen benötigen. Dabei haben wir nur die erreichbaren Zustände eingezeichnet und die beiden unproduktiven Zustände zusammengefasst, da dies, wie man offensichtlich erkennt, nichts an der durch den DFA charakterisierten Sprache ändert.

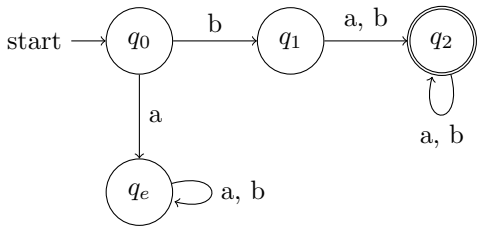

Abbildung 6: $L(F) = (\overline{L(B)} \cap L(D)) \cup (L(B) \cap \overline{L(D)}) = \{w \in \Gamma^* \mid |w| \geq 2 \text{ und } w \text{ beginnt mit b}\}$.

2.3 Das Pumping-Lemma

Die Pumping-Eigenschaft ist eine Abschlusseigenschaft der regulären Sprachen. Dabei wird ein Teil eines Wortes aus der Sprache so aufgepumpt, dass die dabei entstandenen neuen Wörter ebenfalls in der jeweiligen Sprache liegen. Dies entspricht auf einem DFA, dass eine Teilrechnung, einer akzeptierenden Rechnung beliebig oft in der akzeptierenden Rechnung vorkommen kann und die Rechnung trotzdem akzeptierend bleibt. Vereinfacht ausgedrückt bedeutet dies, dass ein Zyklus von Zuständen existiert, der beliebig oft durchlaufen werden kann.

Lemma 1. Sei L eine reguläre Sprache, dann gibt es ein $p \in \mathbb{N}$, sodass für alle $w \in L$ mit $|w| \geq p$ es eine Zerlegung in $w = xyz$ gibt und Folgendes gilt:

- $|xy| \leq p$
- $|y| \geq 1$
- Für alle $i \in \mathbb{N}_0$ gilt: $xy^iz \in L$

Das Pumping-Lemma wird hauptsächlich dazu eingesetzt, um zu zeigen, dass eine gegebene Sprache nicht in der Menge der regulären Sprachen liegt. Wir interessieren uns aber im weiteren Verlauf dieser Arbeit für die **Pumping-Zahl**, dies ist die kleinste Zahl p, sodass für die Sprache alle oben genannten Eigenschaften erfüllt werden. Die Pumping-Zahl lässt sich für reguläre Sprachen die durch einen DFA oder NFA charakterisiert werden, auch dadurch bestimmen, dass wir die längste Rechnung betrachten, die jeden Zustand maximal einmal besucht. Diese Zahl um 1 inkrementiert, ergibt dann die Pumping-Zahl p der Sprache des Automaten.

Beispiel 6. Daher wollen wir im Folgenden für den Beispielautomat $F = (Q_F, \Gamma, \delta_F, q_0, Q_e^F)$ der in Abbildung 6 dargestellt ist, nicht beweisen, dass er die Pumping-Eigenschaft besitzt. Dies wissen wir bereits, da es sich bei diesem Automaten offensichtlich um einen DFA handelt und DFAs die regulären Sprachen charakterisieren, die die Pumping-Eigenschaft besitzen. Stattdessen sind wir an der Pumping-Zahl für die Sprache $L(F)$ interessiert.
Dazu wählen wir zu Beginn p als 3 und betrachten beispielhaft, dass Wort $w \coloneqq baa$. Dieses liegt, wie man erkennen kann, in der Sprache $L(F)$ und $|baa| \geq 3$ gilt ebenfalls. Außerdem gibt es eine Zerlegung von w in xyz mit $x = ba, \quad y = a$ und $z = \epsilon$, sodass alle obigen Eigenschaften erfüllt werden. Auch alle anderen Wörter die eine Länge größer gleich 3 besitzen lassen sich auf dieselbe Art zerlegen, sodass sie das Pumping Lemma ebenfalls erfüllen.
Damit wissen wir bereits, dass die Pumping-Zahl kleiner gleich 3 ist. Daher betrachten wir als Nächstes $p = 2$ und das Wort $w \coloneqq ba$ aus der Sprache. Für das Wort gibt es die folgenden drei Zerlegungen, für die wie wir erkennen können, dass xy^0z nicht in der der Sprache $L(F)$ liegt.

- $x = \epsilon, \quad y = ab, \quad z = \epsilon$ für $i = 0$ gilt: $(ab)^0 = \epsilon \notin L(B)$.
- $x = b, \quad y = a, \quad z = \epsilon$ für $i = 0$ gilt: $b(a)^0 = b \notin L(B)$.
- $x = \epsilon, \quad y = b, \quad z = a$ für $i = 0$ gilt: $(b)^0 a = a \notin L(B)$.

Damit gilt für $p = 2$ das Pumping Lemma nicht, somit kann 2 nicht die Pumping-Zahl dieser Sprache sein. Damit wissen wir, dass die Pumping-Zahl p für die Sprache $L(F)$ den Wert 3 besitzt.

3 Das Gewicht einer Sprache

Im folgenden Kapitel möchten wir jeder formalen Sprache L ein Gewicht $\omega(L)$ zuordnen. Dabei soll jedes Wort, welches in der Sprache L liegt, mit einem Wert, der durch die Wortlänge des Wortes bestimmt ist, zum Gesamtgewicht der Sprache beitragen. Außerdem soll das Gewicht der leeren Sprache den Wert 0 besitzen und das Gewicht der vollen Sprache den Wert 1 annehmen. Die Gewichte aller anderen Sprachen müssen demnach zwischen 0 und 1 liegen.
Um dies zu erreichen, geben wir eine exponentiell zerfallende Funktion an, die beschreibt, mit welchen Anteil die Wortlänge n zum Gesamtgewicht der Sprache beiträgt (Abschnitt 3.1). Wir verwenden hier eine exponentiell zerfallende Funktion, da die Wortlänge bei formalen Sprachen nicht begrenzt ist und wir somit für eine konstante Funktion nicht in der Lage wären, die Anteile auf unendlich viele Wortlängen zu verteilen. Da für jede Wortlänge n genau $|\Gamma|^n$ Wörter existieren und alle gleich gewichtet werden sollen, ist somit auch das Gewicht jedes einzelnen Wortes bestimmt. Wir werden sehen, dass wir für die exponentiell zerfallende Funktion bezüglich der Wortlänge in der Lage sind, das Gewicht für reguläre Sprache, die durch einen DFA charakterisiert werden, mittels eines Gleichungssystems explizit bestimmen zu können (Abschnitt 3.2). Für alle anderen Darstellungsformen von regulären Sprachen als DFAs sowie für allgemeine Grammatiken, kontextsensitive Sprachen und kontextfreie Sprachen existiert bisher keine explizite Lösung, da sie nicht die Eigenschaft besitzen, dass es für jedes Wort genau eine Rechnung auf dem Automaten existiert.
Darüber hinaus werden wir uns anschauen, wie wir die explizite Formel anpassen müssen, wenn wir das Gewicht für die ersten η Wortlängen so umverteilen, dass alle Wörter, die eine Wortlänge kleiner gleich η aufweisen, mit dem gleichen Wert in das Gewicht der Sprache einfließen (Abschnitt 3.3).

3.1 Gewicht einer Wortlänge

Im Folgenden möchten wir eine Funktion $\omega_n(L)$ bestimmen, die für eine formale Sprache L angibt, wie hoch die Summe der Gewichte aller Wörter der Länge n ist, die in der Sprache von L liegen. Dazu benötigen wir die Funktion $\psi(n)$, die jeder Wortlänge n den Anteil zuordnet, mit dem diese Wortlänge zum Gesamtgewicht der Sprache beiträgt. Wie einführend bereits erwähnt, soll das Gewicht mit größer werdender Wortlänge exponentiell abnehmen, sodass es sich bei der Funktion $\psi(n)$ um eine exponentiell zerfallende Funktion bezüglich der Wortlänge n handeln muss. Dafür benötigen wir eine Zerfallsrate $\lambda \in (0,1)$, diese beschreibt wie stark der Anteil des Gewichts für größer werdende Wortlängen abnimmt. Des Weiteren benötigen wir noch einen Normierungsfaktor, damit wir für alle Zerfallsraten als maximales Gewicht einer Sprache so wie wir es einführend gefordert haben, den Wert 1 herausbekommen. Als Normierungsfaktor benutzen wir dabei den Kehrwert der Konvergenz der Summe über alle λ^{n+1}, diesen können wir mithilfe der geometrischen Reihe für Werte kleiner 1 berechnen. Damit sieht unsere Funktion $\psi(n)$ wie folgt aus.

$$\psi(n) = \lambda^{n+1} \cdot \frac{1-\lambda}{\lambda}$$

Damit kennen wir für jede Wortlänge n das maximale Gewicht, welches eine Sprache L für die Wortlänge n annehmen kann, dieses ist zugleich das Gewicht der vollen Sprache bezüglich dieser Wortlänge. Dies ist damit zu erklären, dass die volle Sprache alle Wörter einer bestimmten Länge n enthält und somit das maximale Gewicht dieser Wortlänge ebenfalls beschreibt. Daher gilt $\psi(n) = \omega_n(\Gamma^*)$.
Um das Gewicht $\omega_n(L)$ der Länge n einer Sprache L bestimmen zu können, benötigen wir noch den Anteil $f_n(L)$ der von der Sprache akzeptierten Wörter der Länge n im Verhältnis zu allen Wörtern der Länge n über dem Alphabet der Sprache.

$$f_n(L) = \frac{|\{w \mid w \in L \text{ und } |w| = n\}|}{|\Gamma^n|}$$

Wie wir den Anteil $f_n(L)$ für reguläre Sprachen, die als DFA vorliegen, effizient bestimmen können, behandeln wir in Kapitel 4. Für kontextsensitive Sprachen und kontextfreie Sprachen existiert bisher leider kein Verfahren, um den Anteil effizienter zu bestimmen, als alle Wortprobleme für eine bestimmte Wortlänge zu lösen. Da allerdings der Zeitbedarf für das Lösen eines einzelnen Wortproblems für diese Sprachklassen bereits exponentielle beziehungsweise kubische Zeit benötigt und wir dieses Problem für die Wortlänge n genau $|\Gamma|^n$ mal lösen müssen, führt dies zu sehr langen Laufzeiten. Für allgemeine Grammatiken sind wir auch durch diese triviale Methode nicht in der Lage, den Anteil zu bestimmen, da das Wortproblem für allgemeine Grammatiken nicht entscheidbar ist.
Da jedes Wort einer Wortlänge mit demselben Gewicht zum Gesamtgewicht der Sprache sowie zum

Gesamtgewicht der Wortlänge beitragen soll, ergibt sich das Gewicht einer Wortlänge für die Sprache L, indem wir das maximale Gewicht $\psi(n)$ für die Wortlänge n mit dem Anteil $f_n(L)$ der akzeptierenden Wörter multiplizieren.

$$\omega_n(L) = \psi(n) \cdot f_n(L) = \lambda^{n+1} \cdot \frac{1-\lambda}{\lambda} \cdot f_n(L)$$

Damit sind wir in der Lage, das Gewicht $\omega(L)$ einer Sprache L zu bestimmen, dazu bilden wir die Summe über alle Wortlängen mit der jeweiligen Gewichtung der Wortlänge.

Definition 1. $\omega(L) = \sum_{n=0}^{\infty} \omega_n(L) = \sum_{n=0}^{\infty} \psi(n) \cdot f_n(L) = \sum_{n=0}^{\infty} \lambda^{n+1} \cdot \frac{1-\lambda}{\lambda} \cdot f_n(L) = \frac{1-\lambda}{\lambda} \sum_{n=0}^{\infty} \lambda^{n+1} \cdot f_n(L)$

Eine Möglichkeit ist, das Gewicht einer Sprache näherungsweise zu bestimmen. Dazu betrachten wir nur die ersten k Wortlängen und lassen die restlichen Wortlängen als Ungenauigkeit der Gewichtung der Sprache einfließen. Damit können wir das Gewicht bis auf eine Ungenauigkeit $\leq \lambda^{k+1}$ bestimmen (die Ungenauigkeit wird aus der Berechnung in Abschnitt 3.3 zum Anteil des konstanten Teils ersichtlich). Auch wenn diese Annäherung für große k in der Praxis ein sehr gutes und brauchbares Ergebnis liefert, schauen wir uns im nachfolgenden Kapitel eine explizite Variante zur Bestimmung des Gewichtes einer Sprache an, welche auf die in diesem Kapitel gemachten Vorarbeiten aufbaut, allerdings nur für DFAs anwendbar ist.
Im Folgenden möchten wir zeigen, dass der Wertebereich der Gewichte für jede formale Sprache im Intervall von 0 bis 1 liegt. Dazu zeigen wir zunächst, dass für alle gewählten λ die Summe über die Wortlängen n von 0 bis ∞ für $\psi(n)$ gleich 1 ist, was gleichbedeutend damit ist, dass $\omega(\Gamma^*) = 1$ für alle λ gilt. Außerdem wissen wir, das die Leere Sprache für keine Wortlänge ein Wort enthält. Daraus folgt das $f_n(\emptyset)$ für alle n gleich 0 ist. Damit ist das Gewicht der leeren Sprache immer 0. Darauf aufbauen zeigen wir, dass alle anderen Sprachen ein Gewicht zwischen 0 und 1 annehmen.
Für die folgenden Beweise benötigen wir noch die geometrische Reihe, daher schauen wir uns deren Definition an.

Lemma 2. *Geometrische Reihe*: für $|r| < 1$ gilt: $\sum_{i=0}^{\infty} r^i = \frac{1}{1-r}$ [3].

Lemma 3. Für alle $\lambda \in (0,1)$ gilt: $\sum_{n=0}^{\infty} \psi(n) = 1$.

Beweis.
Durch den Wertebereich von λ wird auch ersichtlich, dass die Voraussetzung $|\lambda| < 1$ gilt, sodass wir die Geometrische Reihe in Schritt 5 des Beweises verwenden können.

$$\sum_{n=0}^{\infty} \psi(n) = 1 \tag{1}$$

$$\Leftrightarrow \quad \sum_{n=0}^{\infty} \lambda^{n+1} \cdot \frac{1-\lambda}{\lambda} = 1 \tag{2}$$

$$\Leftrightarrow \quad \sum_{n=0}^{\infty} \lambda^{n+1} = \frac{\lambda}{1-\lambda} \tag{3}$$

$$\Leftrightarrow \quad \lambda \sum_{n=0}^{\infty} \lambda^{n} = \frac{\lambda}{1-\lambda} \tag{4}$$

$$\Leftrightarrow \quad \lambda \cdot \frac{1}{1-\lambda} = \frac{\lambda}{1-\lambda} \tag{5}$$

$$\Leftrightarrow \quad \frac{\lambda}{1-\lambda} = \frac{\lambda}{1-\lambda} \tag{6}$$

Damit haben wir gezeigt, dass beide Seiten unserer Gleichung gleich sind und somit das Lemma bewiesen ist. □

Theorem 1. Für alle $L \subseteq \Gamma^*$ gilt: $\omega(L) \in [0,1]$.

Beweis.
Wir zeigen zunächst die obere Schranke, die besagt, dass keine Sprache ein höheres Gewicht als $L(\Gamma^*)$ besitzt. Für die Sprache $L(\Gamma^*)$ wissen wir durch Lemma 3 bereits, dass sie das Gewicht 1 besitzt. Anschließend zeigen wir noch die unter Schranke, die besagt, dass alle Sprachen ein Gewicht größer gleich 0 besitzen. Auch hier wissen wir, dass das Gewicht der leeren Sprachen den Wert 0 besitzt.

Obere Schranke: für alle $L \subseteq \Gamma^*$gilt: $\omega(L) \leq 1$.

Um die obere Schranke zu zeigen, nutzen wir die Tatsache aus, dass der Anteil der akzeptierten Wörter f_n für Teilmengen von Γ^n einen Wert kleiner gleich 1 besitzt, um so zu zeigen, das jede Sprache maximal das Gewicht von Γ^* besitzt.

$$
\begin{aligned}
& \text{Für alle } L \text{ gilt: } L \subseteq \Gamma^* && (1)\\
\Rightarrow\ & \text{Für alle } n \in \mathbb{N} \text{ und } L \subseteq \Gamma^* \text{ gilt: } \{w \mid w \in L \text{ und } |w| = n\} \subseteq \Gamma^n && (2)\\
\Rightarrow\ & \text{Für alle } n \in \mathbb{N} \text{ und } L \subseteq \Gamma^* \text{ gilt: } f_n(L) \leq f_n(\Gamma^n) = 1 && (3)\\
\Rightarrow\ & \text{Für alle } n \in \mathbb{N} \text{ und } L \subseteq \Gamma^* \text{ gilt: } \psi(n) \cdot f_n(L) \leq \psi(n) = \omega_n(\Gamma^n) && (4)\\
\Rightarrow\ & \text{Für alle } L \text{ gilt: } \sum_{n=0}^{\infty} \psi(n) \cdot f_n(L) \leq \omega(\Gamma^*) = 1 && (5)\\
\Rightarrow\ & \text{Für alle } L \text{ gilt: } \omega(L) \leq 1 && (6)
\end{aligned}
$$

Damit haben wir bereits die obere Schranke bewiesen und müssen im Folgenden noch die untere Schranke beweisen. Dabei nutzen wir aus, dass der Anteil der akzeptierten Wörter für eine Wortlänge niemals negativ sein kann. Um somit zu zeigen, dass alle Sprachen mindestens das Gewicht der leeren Sprache besitzen.

Untere Schranke: für alle $L \subseteq \Gamma^*$gilt: $\omega(L) \geq 0$.

$$
\begin{aligned}
& \text{Für alle } L \subseteq \Gamma^* \text{ gilt: } L \supseteq \emptyset && (1)\\
\Rightarrow\ & \text{Für alle } n \in \mathbb{N} \text{ und } L \subseteq \Gamma^* \text{ gilt: } f_n(L) \geq f_n(\emptyset) = 0 && (2)\\
\Rightarrow\ & \text{Für alle } L \text{ gilt: } \sum_{n=0}^{\infty} \psi(n) \cdot f_n(L) \geq 0 && (3)\\
\Rightarrow\ & \omega(L) \geq 0 && (4)
\end{aligned}
$$

Damit ist auch die untere Schranke bewiesen und es ist gezeigt, dass der Wertebereich von $\omega(L)$ für alle Sprachen im Intervall [0,1] liegt. □

3.2 Explizite Lösung für reguläre Sprachen

Nachdem wir gesehen haben, wie wir das Gewicht einer Sprache über die unendliche Summe der Wortlängen und den dazugehörigen Anteilen der akzeptierten Wörter definiert haben, möchten wir im Folgenden ein Gleichungssystem vorstellen, mit dem wir in der Lage sind, das Gewicht einer Sprache explizit für DFAs berechnen zu können.
Dafür bestimmen wir für jeden Zustand $q \in Q$ des gegebenen Automaten ein Gewicht ω_q, welches sich aus den Gewichten der Zustände $p \in Q$ zusammensetzt, für die die Transitionsfunktion für ein $a \in \Gamma$ von q auf p abbildet. Somit erhalten wir ein Gleichungssystem, welches, wie wir im Folgenden sehen werden, immer genau eine eindeutige Lösung besitzt. Damit besitzt jeder Zustand $q \in Q$ ein eindeutiges Gewicht, welches beschreibt, wie das Gewicht der Sprache wäre, wenn q der Startzustand des gegebenen Automaten wäre. Damit wird auch ersichtlich, dass wir das Gewicht der Sprache des DFAs in ω_{q_0} ablesen können.
Um das Gewicht eines Zustands bestimmen zu können, benötigen wir noch den Anteil der akzeptierenden Wörter der Länge n, wenn wir annehmen das der Zustand q der Startzustand des gegeben DFA ist, damit starten alle Rechnungen von L_q im Zustand q. Den Anteil der akzeptierenden Rechnungen beziehungsweise Wörter der Länge n, lassen sich dann folgendermaßen bestimmen.

$$f_n(L_q) = \frac{|\{qw \vdash^n p\epsilon \mid \text{mit } p \in Q_e \text{ und } w \in \Gamma^n\}|}{|\Gamma^n|}$$

Des Weiteren wir wissen, dass das leere Wort das einzige Wort der Länge 0 ist, daraus folgt, dass der Anteil $f_0(L_q)$ entweder 1 oder 0. Ist der Wert 1 dann liegt der Zustand q in der Menge der Endzustände im Fall das der Anteil 0 ist, so liegt der Zustand nicht in der Menge der Endzustände.

$$f_0(L_q) = \begin{cases} 1, & \text{wenn } q \in Q_e \\ 0, & \text{sonst} \end{cases}$$

Damit können wir das Gewicht des Zustandes q für die Wortlänge 0 durch die Formel $e(q)$ bestimmen. Die Formel e berechnen wir, indem wir überprüfen, ob der Zustand in der Menge der Endzustände liegt und mit der Gewichtung $\psi(0)$ der Wortlänge 0 multiplizieren. Die Gewichtung $\psi(0)$ lässt sich folgendermaßen bestimmen.

$$\psi(0) = \lambda^{0+1} \cdot \frac{1-\lambda}{\lambda} = 1 - \lambda$$

Damit ergibt sich der Wert der Formel e für alle Zustände $q \in Q$ wie folgt.

$$e(q) = \psi(0) \cdot f_0(L_q) = (1-\lambda) \cdot f_0(L_q) = \begin{cases} 1-\lambda, & \text{wenn } q \in Q_e \\ 0, & \text{sonst} \end{cases}$$

Das Gewicht ω_q für den Zustand q für Wörter der Länge größer gleich 1 setzt sich durch die Gewichte aller anderen Zustände p zusammen, die jeweils mit dem Anteil der Buchstaben des Alphabets $a \in \Gamma$, für die die Transitionsfunktion δ mit q und a auf p abbildet, gewichtet werden. Diese Gewichtung der anderen Zustände beschreiben wir für ein Paar von Zuständen q und p mit der Variable $a_{q,p}$, indem $a_{q,p}$ angibt, wie hoch der Anteil der ausgehenden Kanten von q ist die nach p gehen.

$$a_{q,p} = \frac{|\{\delta(q,a) = p \mid a \in \Gamma\}|}{|\Gamma|} \text{ mit } q, p \in Q$$

Da es sich durch das lesen einer Transition um Wörter der Länge +1 handelt, muss das Resultat noch mit λ multipliziert werden, sodass sich alle Rechnungen ausgehen vom Zustand q auf dem Automaten rekursiv durch Verwendung der Gewichte der anderen Zustände im Gewicht ω_q niederschlagen.
Damit lässt sich für alle q die folgende Gleichung erstellen, die das Gewicht des Zustands beschreibt.

$$\omega_q = e(q) + \lambda \sum_{p \in Q} a_{q,p} \cdot \omega_q$$

Wenn wir diese Gleichung für alle Zustände des gegebenen Automaten erstellen, erhalten wir das folgende Gleichungssystem.

$$\begin{array}{ccccc} \lambda \cdot a_{q_0,q_0} \cdot \omega_{q_0} + \cdots + \lambda \cdot a_{q_0,q_n} \cdot \omega_{q_n} + e(q_0) = \omega_{q_0} \\ \vdots & \ddots & \vdots & \vdots & \vdots \\ \lambda \cdot a_{q_n,q_0} \cdot \omega_{q_0} + \cdots + \lambda \cdot a_{q_n,q_n} \cdot \omega_{q_n} + e(q_n) = \omega_{q_n} \end{array}$$

Dieses Gleichungssystem können wir so umstellen, dass wir auf der rechten Seite aller Gleichungen einen festen Wert haben, der einzig davon abhängt, welchen wert die Funktion e bezüglich der Variable, nach der die jeweilige Zeile des Gleichungssystem entwickelt wurden ist, besitzt.

$$\begin{array}{ccccc} \lambda \cdot a_{q_0,q_0} \cdot \omega_{q_0} - \omega_{q_0} + \cdots + \lambda \cdot a_{q_0,q_n} \cdot \omega_{q_n} & & & & = -e(q_0) \\ \vdots & & \ddots & \vdots & \vdots \\ \lambda \cdot a_{q_n,q_0} \cdot \omega_{q_0} & & + \cdots + \lambda \cdot a_{q_n,q_n} \cdot \omega_{q_n} - \omega_{q_n} & & = -e(q_n) \end{array}$$

Damit haben wir ein Gleichungssystem mit genau $|Q|$ Gleichungen, die jeweils genau $|Q|$ Variablen besitzen. Somit lässt sich dieses Gleichungssystem durch bekannte Lösungsverfahren wie das Gaußsche Eliminationsverfahren lösen.

Theorem 2. Das Gleichungssystem lässt sich durch das Gaußsche Eliminationsverfahren lösen, falls es nicht unterbestimmt ist[8].

Da wir allerdings in Theorem 3 sehen werden, dass das jede Gleichung des Gleichungssystems eindeutig bestimmt wird. Folgt daraus, dass das Gleichungssystem immer vollen Rang besitzt und wir somit eine eindeutige Lösung durch das Gaußsche Eliminationsverfahren erhalten.

Beispiel 7. Nachdem wir ein Gleichungssystem vorgestellt haben, mit dem wir das Gewicht einer Sprache explizit bestimmen können, möchten wir auf diese Weise das Gewicht des DFA F (Abbildung 6) explizit bestimmen. Da die Alphabetsgröße der Sprache $L(F)$ 2 beträgt, existieren von jedem Zustand des Automatens genau 2 ausgehende Kanten, was mit der verwendeten Zerfallsrate von $\lambda = 0{,}8$ dazu führt, dass das Gewicht des Zustandes auf den abgebildet wird, mit der Gewichtung von $a = 0{,}4$ einbezogen wird. Bilden hingegen beide Buchstaben auf denselben Zustand ab, dann ist entsprechend 0,8 der Wert von a, sodass wir für den Automaten das folgende Gleichungssystem erhalten, welches sich so umstellen lässt, dass wir durch Einsetzen der offensichtlichen Variablenwerte alle anderen Werte erhalten und somit eine eindeutige Lösung für dieses Gleichungssystem bestimmen können.

$$
\begin{aligned}
0 \cdot \omega_{q_0} \quad &+0{,}4 \cdot \omega_{q_1} \quad +0 \cdot \omega_{q_2} \quad +0{,}4 \cdot \omega_{q_e} \quad +0 \quad &= \quad \omega_{q_0} \quad &(1)\\
0 \cdot \omega_{q_0} \quad &+0 \cdot \omega_{q_1} \quad +0{,}8 \cdot \omega_{q_2} \quad +0 \cdot \omega_{q_e} \quad +0 \quad &= \quad \omega_{q_1} \quad &(2)\\
0 \cdot \omega_{q_0} \quad &+0 \cdot \omega_{q_1} \quad +0{,}8 \cdot \omega_{q_2} \quad +0 \cdot \omega_{q_e} \quad +0{,}2 \quad &= \quad \omega_{q_2} \quad &(3)\\
0 \cdot \omega_{q_0} \quad &+0 \cdot \omega_{q_1} \quad +0 \cdot \omega_{q_2} \quad +0{,}8 \cdot \omega_{q_e} \quad +0 \quad &= \quad \omega_{q_e} \quad &(4)
\end{aligned}
$$

$$
\begin{aligned}
-\omega_{q_0} \quad &+0{,}4 \cdot \omega_{q_1} \quad +0 \cdot \omega_{q_2} \quad +0{,}4 \cdot \omega_{q_e} \quad &= \quad 0 \quad &(1)\\
0 \cdot \omega_{q_0} \quad &-\omega_{q_1} \quad +0{,}8 \cdot \omega_{q_2} \quad +0 \cdot \omega_{q_e} \quad &= \quad 0 \quad &(2)\\
0 \cdot \omega_{q_0} \quad &+0 \cdot \omega_{q_1} \quad -0{,}2 \cdot \omega_{q_2} \quad +0 \cdot \omega_{q_e} \quad &= \quad -0{,}2 \quad &(3)\\
0 \cdot \omega_{q_0} \quad &+0 \cdot \omega_{q_1} \quad +0 \cdot \omega_{q_2} \quad -0{,}2 \cdot \omega_{q_e} \quad &= \quad 0 \quad &(4)
\end{aligned}
$$

In Zeile 3 und 4 erkennen wir sofort, dass der Wert von ω_{q_2} gleich 1 beträgt und der Wert von $\omega_{q_e} = 0$ ist. Den Wert von ω_{q_2} können wir benutzen, um diesen in Zeile 2 einzusetzen, damit erhalten wir für $\omega_{q_1} = 0{,}8$. Dieses Ergebnis können wir wiederum in Zeile 1 einsetzen, somit erhalten wir für $\omega_{q_0} = 0{,}32$. Da q_0 den Startzustand des Automaten F darstellt, beschreibt ω_{q_0} auch gleichzeitig das Gewicht der Sprache $\omega(L(F)) = 0{,}32$.

Beweis der Korrektheit. Im Folgenden möchten wir zeigen, dass sich durch Verwendung des vorgestellten Gleichungssystems das Gewicht einer Sprache, die durch einen DFA charakterisiert wird, sich explizit bestimmen lässt.
Dazu zeigen wir zuerst, dass wir mithilfe des vorgestellten Gleichungssystems ein Gewicht für die Sprache bestimmen können, um anschließend zu zeigen, dass dieses Gewicht eindeutig ist, da nur genau eine Lösung für dieses Gleichungssystem existiert.
Bevor wir uns dem Beweis widmen, müssen wir allerdings noch einige Vorarbeiten leisten, die wir für den anschließenden Beweis benötigen.
Dazu betrachten wir zunächst die folgende Nebenbedingung des Gleichungssystems, die besagt, dass die Summe der Anteile a für jede Zeile 1 ergibt.

$$\textit{Nebenbedingung:}\ \text{für alle } q \in Q \text{ gilt: } \sum_{p \in Q} a_{q,p} = \sum_{p \in Q} \frac{|\{\delta(q,a) = p \mid a \in \Gamma\}|}{|\Gamma|} = 1$$

Dass diese Nebenbedingung gilt, ist damit zu erklären, dass es sich bei DFAs um eine Transitionsfunktion handelt, die für jedes Paar von einem Zustand und einem Buchstaben des Alphabets auf genau einen anderen Zustand abbildet.
Des Weiteren betrachten wir, wie wir den Anteil der akzeptierenden Rechnungen $f_n(L_q)$ für einen Zustand der Sprache rekursiv bestimmen können. Für Wortlängen n größer gleich 1, lässt sich der Anteil dadurch bestimmen, indem wir betrachten, wie hoch der Anteil $f_{n-1}(L_p)$ der Zustände p für die Länge $n-1$ ist auf die ausgehen von q mit mindesten einem Buchstaben des Alphabets abgebildet wird und diese jeweils mit dem Anteil $a_{q,p}$ gewichten. Dies ist dadurch zu erklären, dass wir für alle mögliche Wörter einen Schritt auf dem DFA machen und dann schauen, wie hoch jeweils der Anteil der akzeptierenden Läufe nach diesem Schritt ist mit denen wir die Rechnung weiterführen können. Das Rekursions-Abbruchkriterium ist dabei die Wortlänge 0. Für diese können wir den Anteil wie bereits erwähnt auf triviale Weise bestimmen,

indem wir überprüfen, ob es sich bei dem Zustand indem wir uns befinden, um einen akzeptierenden Zustand handelt.

$$f_n(L_q) = \begin{cases} \sum\limits_{p \in Q} a_{q,p} \cdot f_{n-1}(L_p), & \text{wenn } n \geq 1 \\ 1, & \text{wenn } n = 0 \text{ und } q \in Q_e \\ 0, & \text{sonst} \end{cases}$$

Damit lässt sich der Anteil einer bestimmten Wortlänge n durch die zuvor definierte Rekursionsformel folgendermaßen bestimmen.

$$f_n(L_q) = \underbrace{\sum_{p_1 \in Q} a_{q,p_1} \sum_{p_2 \in Q} \cdots \sum_{p_n \in Q} a_{p_{n-1},p_n} \cdot f_0(L_{p_n})}_{n-\text{mal}}$$

Um die Gewichtung der Wortlänge mit einzubeziehen, können wir die obige Formel so umschreiben, dass wir für jeden Rekursionschritt das Gewicht der nächsten Länge mit der Zerfallsrate multiplizieren und die Länge 0 wieder als unser Abbruchkriterium der Rekursion betrachten. Für das Gewicht der Länge 0 haben wir bereits die Formel $e(q)$ für alle $q \in Q$ bestimmt, sodass wir die Formel in dieser Rekursionsformel verwenden können.

$$\omega_n(L_q) = \begin{cases} \lambda \sum\limits_{p \in Q} a_{q,p} \cdot \omega_{n-1}(L_p), & \text{wenn } n \geq 1 \\ e(q), & \text{sonst} \end{cases}$$

Auch der Normierungsfaktor wird bereits durch die Formel $e(q)$ mit in die Rekursionsgleichung eingebracht. Dies zeigen wir beispielhaft daran, dass das Gewicht der vollen Sprache für eine beliebige Wortlänge n den Wert von $\psi(n)$ ergibt. Dabei nutzen wir neben der Nebenbedingung für das Gleichungssystem aus, das alle erreichbaren Zustände der vollen Sprache akzeptierend sind, sodass all diese Zustände dasselbe Gewicht aufweisen. Damit gilt $\sum\limits_{p \in Q} a_{q,p} \cdot \omega_{n-1}(L_p) = 1 \cdot \omega_{n-1}(L_q)$, somit lässt sich das Gewicht eines Zustandes für eine Wortlänge folgendermaßen bestimmen.

$$\omega_n(\Gamma_q^*) = \underbrace{\lambda \cdot \cdots \cdot \lambda}_{n-\text{mal}} \cdot e(q) = \lambda^n \cdot (1-\lambda) = \lambda^n \cdot \lambda \cdot \frac{1-\lambda}{\lambda} = \lambda^{n+1} \cdot \frac{1-\lambda}{\lambda} = \psi(n)$$

Da es sich bei der Wortlänge 0 um das Abbruchkriterium der Rekursion handelt und wir diesen Wert durch die Formel e bestimmen können, lässt sich die Wortlänge 0 aus der Summe zur Bestimmung des Gesamtgewichts ω_q heraus ziehen.

$$\omega_q = \sum_{n=0}^{\infty} \omega_n(L_q) = e(q) + \sum_{n=1}^{\infty} \omega_n(L_q)$$

Dies ist eine wichtige Eigenschaft, um nachfolgend zu zeigen, dass wir mithilfe des vorgestellten Gleichungssystems das Gewicht einer durch einen DFA charakterisierten Sprache eindeutig bestimmen können. Damit haben wir alle Vorarbeiten geleistet, um im Folgenden zu zeigen, dass sich das Gewicht einer Sprache durch das Gleichungssystem bestimmen lässt.

Theorem 3. $\omega_{q_0} = \omega(L)$

Beweis. Um das Theorem zu beweisen, möchten wir zeigen, dass das Gewicht eines Zustands genau das Gewicht der Sprache beschreibt, wenn dieser Zustand der Startzustand der Sprache wäre. Dies impliziert, dass das Gewicht des Startzustands das tatsächliche Gewicht der Sprache beschreibt.
Dazu zeigen wir, wie wir die Gleichung für ω_q für alle $q \in$ Q so umformen können, dass wir die Gleichung $\omega(L_q)$, die das Gewicht einer Sprache über die Definition der Wortlänge mit ihrer jeweiligen Gewichtung sowie dem Anteil der akzeptierenden Wörter ausgehen, von dem Zustand q definiert erhalten.

$$\text{zu zeigen: } \omega_q = \frac{1-\lambda}{\lambda} \sum_{n=0}^{\infty} \lambda^{n+1} \cdot f_n(L_q) = \omega(L_q)$$

Dabei nutzen wir in Schritt 2 und 3 der folgenden Umformung aus, dass sich das Gewicht rekursiv durch die Gewichte der Zustände auf die abgebildet wird, darstellen lässt und wir für jedes dieser Gewichte das Gewicht der Länge 0 herausziehen können. Dieses Gewicht stellt dann das Gewicht für die Wortlänge

dar, die gleich unserer aktuellen Rekursionstiefe ist. Damit können wir nach einem Rekursionschritt das Gewicht der Wörter, die eine Länge kleiner gleich 1 aufweisen, explizit berechnen beziehungsweise nach zwei Schritten alle Wörter mit einer Länge kleiner gleich 2. Diesen Rekursionsschritt können wir beliebig oft wiederholen, wobei wir im Folgenden sehen werden, dass das Gewicht für Wortlängen, die gegen unendlich gehen, den Wert 0 annehmen. Damit erhalten wir in Schritt 4 eine Formel, die für jede Wortlänge einen Term besitzt, der das Gewicht für diese Wortlängen explizit angibt. Diese Gleichung formen wir dann schrittweise so um, dass wir die Gleichung $\omega(L_q)$ erhalten. Dabei nutzen wir die bereits berechneten Umformungen für die Formeln e und f_n aus. Des Weiteren haben wir zur Übersichtlichkeit noch unter jeden Term die Wortlänge n geschrieben, dessen Gewichtung der jeweilige Term bezüglich ω_q interpretiert.

$$\omega_q = \underbrace{e(q)}_{n=0} + \underbrace{\lambda \sum_{p_1 \in Q} a_{q,p_1} \cdot \omega_{p_1}}_{n \geq 1} \tag{1}$$

$$= \underbrace{e(q)}_{n=0} + \underbrace{\lambda \sum_{p_1 \in Q} a_{q,p_1} \cdot e(p_1)}_{n=1} + \underbrace{\lambda^2 \sum_{p_1 \in Q} a_{q,p_1} \sum_{p_2 \in Q} a_{p_1,p_2} \cdot \omega_{p_2}}_{n \geq 2} \tag{2}$$

$$= \underbrace{e(q)}_{n=0} + \underbrace{\lambda \sum_{p_1 \in Q} a_{q,p_1} \cdot e(p_1)}_{n=1} + \underbrace{\lambda^2 \sum_{p_1 \in Q} a_{q,p_1} \sum_{p_2 \in Q} a_{p_1,p_2} \cdot e(p_2)}_{n=2} + \underbrace{\lambda^3 \sum_{p_1 \in Q} a_{q,p_1} \cdots \sum_{p_3 \in Q} a_{p_2,p_3} \cdot \omega_{p_3}}_{n \geq 3} \tag{3}$$

Damit haben wir bereits für alle Wortlängen ≤ 2 einen Term, der das Gewicht der jeweiligen Wortlänge beschreibt. Diese Aufdeckungen der einzelnen Wortlängen können wir auf dieselbe Weise für alle Wortlängen bis unendlich durchführen. Daher bestimmen wir den Grenzwert der Gewichtung für Wortlängen gegen unendlich.

$$\lim_{n \to \infty} \lambda^n \sum_{p_1 \in Q} a_{q,p_1} \sum_{p_2 \in Q} \cdots \sum_{p_n \in Q} a_{p_{n-1},p_n} \cdot e(p_n) = \lim_{n \to \infty} \lambda^n \cdot (1-\lambda) \cdot f_n(L_q) = \lim_{n \to \infty} \lambda^n = 0$$

Der Grenzwert von 0 ist damit zu erklären, dass wie wir erkennen können, dass λ^n den dominierenden Faktor der Gleichung darstellt und da λ im Intervall von (0,1) liegt, geht der Grenzwert gegen 0. Damit können wir diesen Wert für die weiter Umformung von ω_q benutzen.

$$\omega_q = \underbrace{e(q)}_{n=0} + \underbrace{\lambda \sum_{p_1 \in Q} a_{q,p_1} \cdot e(p_1)}_{n=1} + \underbrace{\lambda^2 \sum_{p_1 \in Q} a_{q,p_1} \sum_{p_2 \in Q} a_{p_1,p_2} \cdot e(p_2)}_{n=2} + \cdots + \underbrace{0}_{n=\infty} \tag{4}$$

$$= \underbrace{(1-\lambda) \cdot f_0(L_q)}_{n=0} + \underbrace{\lambda \cdot (1-\lambda) \cdot f_1(L_q)}_{n=1} + \underbrace{\lambda^2 \cdot (1-\lambda) \cdot f_2(L_q)}_{n=2} + \cdots + \underbrace{0}_{n=\infty} \tag{5}$$

$$= \sum_{n=0}^{\infty} \lambda^n \cdot (1-\lambda) \cdot f_n(L_q) \tag{6}$$

$$= \sum_{n=0}^{\infty} \lambda^n \cdot \lambda \cdot \frac{1-\lambda}{\lambda} \cdot f_n(L_q) \tag{7}$$

$$= \frac{1-\lambda}{\lambda} \sum_{n=0}^{\infty} \lambda^{n+1} \cdot f_n(L_q) = \omega(L_q) \tag{8}$$

Damit haben wir gezeigt, dass das Gewicht ω_q des Zustands q genau den Wert $\omega(L_q)$ annimmt, den die Sprache L annehmen würde, wenn q der Startzustand dieser Sprache wäre. Daher besitzt ω_q für alle $q \in Q$ einen eindeutigen Wert, woraus folgt, dass das Gleichungssystem immer vollen Rang besitzt und alle Gleichungen linear unabhängig sind. Daher bekommen wir durch das Gaußsche Eliminationsverfahren eine eindeutige Lösung für das Gleichungssystem.
Damit ist gezeigt, dass Theorem 3 korrekt ist und wir für ω_{q_0} das Gewicht der Sprache (Definition 1) erhalten.

□

Laufzeitenanalyse. Im Folgenden möchten wir erklären, warum ein Vergleich der im vorherigen Kapitel vorgestellten näherungsweise Bestimmung einer Gewichtung der Sprache und der expliziten Berechnung der Gewichtung einer Sprache nicht sinnvoll ist. Dies liegt daran, dass wir bei der Annäherung an das exakte Gewicht einen Parameter k besitzen, der bestimmt, bis zu welcher Länge wir das Gewicht bestimmen. Da dieser Wert frei wählbar ist und nur einen Einfluss auf die Ungenauigkeit besitzt, könnten wir k ziemlich klein wählen, sodass die praktische Laufzeit der Annäherung deutlich schneller ist als die explizite Bestimmung, auf der andern Seite könnte man für k den Wert unendlich wählen, sodass diese Methode nicht terminiert und trotzdem nur eine Annäherung an das exakte Gewicht liefert. Damit hat die vorgestellte Methode den Vorteil, dass wir durch Lösen des Gleichungssystems das exakte Gewicht der Sprache bekommen und der Lösungalgorithmus des Gleichungssystems auch immer terminiert.

3.3 Umverteilung der initialen Gewichte

Das der Anteil des Gewichtes einer Wortlänge n zum Gesamtgewicht der Sprache für größer werdende Wortlängen immer weiter abnimmt, ist so gewollt. Allerdings sind im Beispiel 7 aus dem vorherigen Kapitel mit der Zerfallsrate von $\lambda = 0{,}8$ bereits $\frac{1}{5}$ des Gesamtgewichts durch das leere Wort bestimmt wurden und ein einziges Wort der Länge 3 hat nur noch mit einem Anteil von $\frac{4}{625}$ zum Gesamtgewicht der Sprache beigetragen. Für kleinere Zerfallsraten als 0,8 ist die Differenz der Anteile noch wesentlich drastischer. Auch wenn die Wahl eines λ welches gegen 1 geht, dazu führt, dass die Differenz der Anteile kleiner wird, ist der Anteil eines Wortes der Länge 3 immer noch deutlich geringer als der Anteil des leeren Wortes. Dies liegt daran, dass neben dem Zerfall der Gewichtung der einzelnen Wortlängen die Anzahl der Wörter pro Wortlänge für jede weitere Wortlänge um den Faktor $|\Gamma|$ zunimmt, sodass selbst bei einer konstanten Gewichtung das Gewicht eines Wortes für längere Wörter abnimmt.
Da es keinen ersichtlichen Grund gibt, Wörter der Länge 0 oder 1 bereits ein deutlich höheres Gewicht als Wörtern der Länge 2 oder 3 zuzuordnen, möchten wir im Folgenden einen Schwellenwert η bestimmen, um alle Wörter mit einer Wortlänge $n \leq \eta$ mit dem gleichen Gewicht bewerten zu können. Diesen Teil werden wir im weiteren Verlauf auch den konstanten Teil des Gewichts nennen. In Kapitel 5.2 werden wir sehen, dass die Pumping-Zahl ein guter Wert für η darstellt. Da wir um die Wörter konstant bewerten zu können die Gewichte der Wortlängen umverteilen müssen, können wir zur Bestimmung des Gewichtes das Ergebnis des Gleichungssystems, welches wir im Folgenden als $\omega^{exp}(L)$ bezeichnen, für eine Sprache L nicht mehr so wie bisher Verwenden. Stattdessen können wir dieses Ergebnis nur noch dazu verwenden, das Gewicht für den Teil größer η explizit zu bestimmen. Dazu müssen wir von $\omega^{exp}(L)$ den Wert $\omega^{exp}_{\leq\eta}(L)$ abziehen, dieser beschreibt das Gewicht, welches die ersten η Wortlängen zusammen besitzen, wenn wir sie exponentiell zerfallend bewerten würden.

$$\omega^{exp}_{\leq\eta}(L) \coloneqq \frac{1-\lambda}{\lambda} \cdot \sum_{n=0}^{\eta} \lambda^{n+1} \cdot f_n(L)$$

Somit beschreibt $\omega^{exp}(L) - \omega^{exp}_{\leq\eta}(L)$ den Teil des Gewichtes für Wortlängen größer als η. Diesen Teil wollen wir weiter exponentiell zerfallend bewerten, daher nennen wir diesen Teil auch den exponentiell zerfallenden Teil des Gewichts einer Sprache. Um das Gesamtgewicht der Sprache zu erhalten, müssen wir noch das Gewicht des Konstanten Teil $\omega^{cons}_{\leq\eta}(L)$ der Wortlängen bis einschließlich η hinzuaddieren, sodass sich das Gewicht folgendermaßen ergibt.

$$\omega(L) = \underbrace{\omega^{cons}_{\leq\eta}(L)}_{\text{konstanter Teil}} + \underbrace{\omega^{exp}(L) - \omega^{exp}_{\leq\eta}(L)}_{\text{exponentieller Teil}}$$

Um das Gewicht des konstanten Teils $\omega^{cons}_{\leq\eta}(L)$ bestimmen zu können, benötigen wir zunächst einmal den Anteil, den diese Wortlängen zum Gesamtgewicht einer Sprache beitragen. Diesen bestimmen wir, indem wir die Summe über den Gewichtszuordnungen für die jeweiligen Wortlängen $\psi(n)$ bilden. Dafür benutzen wir in Schritt 3 der Umformung die endliche Partialsumme der geometrischen Reihe.

Lemma 4. *Partialsumme der Geometrischen Reihe*: für alle $r \neq 1$ gilt: $\sum_{n=0}^{k} r^n = \frac{1-r^{k+1}}{1-r}$[3].

$$\sum_{n=0}^{\eta} \psi(n) = \sum_{n=0}^{\eta} \lambda^{n+1} \cdot \frac{1-\lambda}{\lambda} \tag{1}$$

$$= \lambda \cdot \frac{1-\lambda}{\lambda} \cdot \sum_{i=0}^{\eta} \lambda^{n} \tag{2}$$

$$= (1-\lambda) \cdot \frac{1-\lambda^{\eta+1}}{1-\lambda} \tag{3}$$

$$= 1-\lambda^{\eta+1} \tag{4}$$

Diesen Anteil [2] müssen wir anschließend gleichmäßig auf alle möglichen Wörter $w \in \Gamma^*$ mit $|w| \leq \eta$ aufteilen. Da wir wissen, dass pro Wortlänge n genau $|\Gamma|^n$ Wörter existieren, können wir für Alphabete mit mehr als einen Buchstaben die Anzahl ebenfalls durch Verwendung der endlichen Partialsumme der geometrischen Reihe bestimmen. Für Alphabet mit nur einem Buchstaben ist dies nicht möglich, da die Voraussetzung zur Verwendung der endlichen Partialsumme nicht erfüllt werden. Da aber für unäre Alphabet pro Wortlänge nur genau ein Wort existiert, ist ersichtlich, dass im Bereich von 0 bis η genau $\eta+1$ Wörter liegen.

$$\textit{Anzahl der Wörter im konstanten Teil} = \sum_{n=0}^{\eta} |\Gamma|^n = \begin{cases} \frac{1-|\Gamma|^{\eta+1}}{1-|\Gamma|}, & \text{wenn } |\Gamma| \geq 1 \\ \eta+1, & \text{sonst} \end{cases}$$

Dadurch sind wir in der Lage, jedem Wort aus dem konstanten Teil ein festes Gewicht zuordnen zu können, indem wir das zuvor bestimmte Gesamtgewicht dieses Bereichs durch die Anzahl an Wörter in diesem Bereich teilen. Des Weiteren können wir dieses Ergebnis mit der Anzahl der Wörter für eine bestimmte Wortlänge multiplizieren, sodass wir den Anteil $\psi^{cons}(n)$, der die Gewichtung einer Wortlänge im Konstanten Teil nach der Umverteilung beschreibt, ebenfalls bestimmen können.

$$\textit{Gewicht eines Wortes im konstanten Teil} = \frac{1-\lambda^{\eta+1}}{\sum_{n=0}^{\eta} |\Gamma|^n}$$

$$\psi^{cons}(n) = \frac{1-\lambda^{\eta+1}}{\sum_{n=0}^{\eta} |\Gamma|^n} \cdot |\Gamma|^n$$

Damit erhalten wir das Gewicht des konstanten Teils einer Sprache durch die folgende Formel.

$$\omega_{\leq \eta}(L) = \sum_{n=0}^{\eta} \psi^{cons}(n) \cdot f_n(L)$$

Da wir nichts anderes gemacht haben, als die Gewichte der einzelnen Wortlängen umzuverteilen und nichts am Gesamtgewicht geändert haben, bleibt der Wertebereich von $\omega(L)$ für alle $L \subseteq \Gamma^*$ weiterhin bestehen. Allerdings haben die Sprachen durch die Umverteilung nicht mehr dasselbe Gewicht wie vor der Umverteilung. Dies gilt nur für einige Ausnahmen, wie zum Beispiel die leere oder die volle Sprache.

Beispiel 8. Nachdem wir beschrieben haben, warum es zur Bestimmung des Gewichtes einer Sprache Sinn macht, einen konstanten Teil für die Wortlängen bis η einzuführen, möchten wir das Beispiel 7 aus dem vorherigen Abschnitt so abändern, dass alle Wörter bis einschließlich der Länge 3 konstant bewertet werden. Dafür benötigen wir noch die Anteile der akzeptierenden Wörter für diese Wortlängen, wie wir diese bestimmen, zeigen wir in Kapitel 4. Hier geben wir vorgreifend schon die Anteile, die wir in Beispiel 9 und Beispiel 10 für diesen Automaten bestimmen werden, für die benötigend Wortlängen in Tabelle 3 an. Diese benötigen wir, um im Folgenden zunächst $\omega_{\leq 3}^{exp}(L(F))$ und anschließend $\omega_{\leq 3}^{cons}(L(F))$

[2] Durch Umstellen dieser Formel zu $1 - \sum_{n=0}^{k} \psi(n) = \lambda^{k+1}$ wird die Ungenauigkeit der annäherungsweisen Berechnung der Gewichtung einer Sprache aus Abschnitt 3.1 ersichtlich.

Wortlänge n	$f_n(L(F))$
0	0
1	0
2	$\frac{1}{2}$
3	$\frac{1}{2}$

Tabelle 3: $f_n(L(F))$.

zu bestimmen. $\omega^{exp}_{\leq 3}(L(F))$ bestimmen wir folgendermaßen.

$$\omega^{exp}_{\leq 3}(L(F)) = \cdot\frac{1-\lambda}{\lambda}\sum_{n=0}^{3}\lambda^{n+1}\cdot f_n(L(F)) \tag{1}$$

$$= \frac{1}{4}\cdot\left(0{,}8^1\cdot 0 + 0{,}8^2\cdot 0 + 0{,}8^3\cdot\frac{1}{2} + 0{,}8^4\cdot\frac{1}{2}\right) \tag{2}$$

$$= \frac{1}{4}\cdot\left(0 + 0 + \frac{32}{125} + \frac{128}{625}\right) \tag{3}$$

$$= \frac{1}{4}\cdot\frac{288}{625} = \frac{72}{125} = 0{,}1152 \tag{4}$$

Um $\omega^{cons}_{\leq 3}(L(F))$ bestimmen zu können, benötigen wir zunächst den Anteil, den jedes einzelne Wort zum Gesamtgewicht des konstanten Teils beiträgt.

$$\omega^{cons}_{\leq 3}(w) = \frac{(1-0{,}8^4)\cdot(1-2)}{1-2^4} = \frac{0{,}5904\cdot -1}{-15} = \frac{123}{3125} = 0{,}03936 \quad \text{für alle } w \text{ mit } |w| \leq 3$$

Durch Tabelle 3 wissen wir, dass für die Wortlängen 0 und 1 kein Wort von dem DFA F akzeptiert wird und es für die Längen 2 und 3 genau 2 beziehungsweise 4 Wörter sind, die akzeptiert werden. Daraus folgt, dass insgesamt 6 Wörter im konstanten Teil von der Sprache $L(F)$ akzeptiert werden. Somit können wir das Gewicht des konstanten Teils der Sprache des Automaten F folgendermaßen verkürzend bestimmen.

$$\omega^{cons}_{\leq 3}(L(F)) = 6\cdot\frac{123}{3125} = \frac{738}{3125} = 0{,}23616$$

Damit haben wir alle Teilgewichte bestimmt, um im abschließenden Schritt das Gewicht der Sprache für die Werte $\eta = 3$ und $\lambda = 0{,}8$ zu bestimmen.

$$\omega(L(F)) = \omega^{cons}_{\leq\eta}(L(F)) + \omega^{exp}(L(F)) - \omega^{exp}_{\leq\eta}(L(F)) = 0{,}23616 + 0{,}32 - 0{,}1152 = 0{,}44096$$

Wir erkennen das unser neues Gewicht mit dem eingeführten konstanten Teil um 0,12096 von dem alten Gewicht ohne den konstanten Teil abweicht. Dies ist dadurch zu erklären, dass für die Sprache $L(F)$ kein Wort der Länge 0 und 1 akzeptiert wird und diesen Wortlängen durch die Umverteilung ein deutlich geringeres Gewicht zugeordnet wird als zuvor. Dadurch erkennen wir auch anhand dieses Beispiels, dass es durchaus sinnvoll ist zur Bestimmung des Gewichts einer Sprache einen konstanten Teil einzuführen.

4 Der Anteil akzeptierter Wörter für endliche Wortlängen

Im vorherigen Kapitel haben wir gesehen, dass wir den Anteil der akzeptierten Wörter für eine bestimmte Wortlänge benötigen, wenn wir bei der Bestimmung des Gewichts einen konstanten Teil einführen möchten oder das Gewicht durch die näherungsweise Bestimmung, dass wir nur die ersten k Wortlängen betrachten, bestimmen möchten. Daher beschäftigen wir uns in diesem Kapitel mit dem Problem, wie wir den Anteil der Wörter der Länge n, die durch einen gegebenen DFA $A = (Q, \Gamma, \delta, Q_0, Q_e)$ akzeptiert werden, im Verhältnis zu allen möglichen Wörtern der Länge n über dem Alphabet des Automaten effizient berechnen können. Damit möchten wir das Problem lösen, dass uns eine Wortlänge n sowie ein DFA gegeben ist und wir den Anteil der akzeptierenden Wörter $f_n(L(A))$ für die Sprache $L(A)$ bezüglich der Wortlänge n bestimmen wollen.

$$f_n(L(A)) = \frac{|\{w \mid w \in L \text{ und } |w| = n\}|}{|\Gamma^n|}$$

Würden wir dieses Problem auf herkömmliche Weise lösen, würde dies das Lösen von $|\Gamma|^n$ vielen Wortproblemen erfordern. Da dies, wie wir in Abschnitt 4.3.1 sehen werden, keine effektive Methode darstellt, stellen wir im folgenden zwei effektivere Verfahren zum Lösen dieses Problems vor, die beide im Vergleich zum Wortproblem allerdings den Nachteil besitzen, dass jegliche Information darüber verloren geht, welche Wörter akzeptiert werden, sondern wir nur den Anteil für eine Wortlänge bestimmen können. Für das erste Verfahren schauen wir uns an, wie wir das Gleichungssystem aus Abschnitt 3.2 abändern können, sodass wir damit den Anteil einer bestimmten Wortlänge bestimmen können (Abschnitt 4.1). Des Weiteren stellen wir ein Verfahren vor, bei dem wir die Matrizen, die bei dem Wortproblem für unäre Alphabete [11] verwendet werden, so abändern, dass sie auf unsere Problemstellung anwendbar sind (Abschnitt 4.2.2). Abschließend vergleichen wir die beiden vorgestellten Verfahren bezüglich ihrer Laufzeit (Abschnitt 4.3), dabei werden wir sehen, dass in dem meisten Fällen das Matrizen-Verfahren das effektiver Verfahren darstellt.

4.1 Ansatz mittels eines Gleichungssystems

Um das Gleichungssystem aus Kapitel 3.2, so abzuändern, dass wir nicht mehr das Gewicht einer Sprache bestimmen, sondern den Anteil der akzeptierenden Wörter einer bestimmten Wortlänge, geben wir für jeden Zustand $q \in Q$ für jede Wortlänge n eine Variable X_q^n an, diese beschreibt, wie hoch der Anteil an Teilrechnungen der Länge n, die im Zustand q starten und nach genau n Schritten sich in einem akzeptierenden Zustand befinden im Verhältnis zu allen Teilrechnungen der Länge n, die in q starten, ist. Dabei berechnen sich die Werte für die Länge n iterativ aus den Werten für die Länge $n-1$. Dies ist damit zu erklären, dass wir uns vorstellen müssen, unsere Wörter rückwärts zu konstruieren beziehungsweise den Lauf des Automaten auf dem gegebenen Wort rückwärts ausgehend vom letzten Zustand zu durchlaufen. Dies bedeutet, wenn ein Teilwort aw mit $a \in \Gamma$ und $w \in \Gamma^n$ existiert, welches im Zustand q startet, dann endet die Rechnung auf dem Teilwort im selben Zustand wie das Teilwort w welches im Zustand p starte und es gilt $\delta(q, a) = p$. Somit betrachten wir für jede ausgehende Transition, um den Anteil für die Länge n zu bestimmen, wie hoch der Anteil des Zustandes, auf den die Transition abbildet, für die Länge $n-1$ ist. Damit wird ersichtlich warum wir dieses Verfahren nur auf DFAs anwenden können. Formal ausgedrückt bedeutet dies Folgendes.

$$X_q^n = \frac{1}{|\Gamma|} \sum_{p \in Q} X_p^{n-1} \cdot |\{\delta(q, a) = p \mid a \in \Gamma\}|$$

In anderen Worten beschreibt die Formel, mit wie vielen Buchstaben des Alphabets wir durch Lesen eines Buchstabens von q nach p kommen und verwenden jeweils den zuvor bestimmten Wert für X_p^{n-1}. Um den Anteil und nicht die Anzahl an Wörtern zu bekommen, müssen wir den so bestimmten Wert noch mit $\frac{1}{|\Gamma|}$ multiplizieren. Außerdem benötigen wir noch den Basisfall für die Länge 0. Dieser bringt zum Ausdruck, ob die Teilrechnung der Länge 0 ausgehend von einem Zustand q in einem akzeptierenden Zustand endet, ist dies der Fall, so können wir der Variable X_q^0 den Wert 1 zuordnen. Da nur genau eine Teilrechnung der Länge 0 existiert und zwar die Rechnung, die keinen Schritt ausgehen von dem Zustand q macht, können wir für den Basisfall nachschauen, ob der Zustand q in der Menge der Endzustände enthalten ist um den Basisfall für die Wortlänge 0 zu bestimmen.

$$X_q^0 = \begin{cases} 1, & \text{wenn } q \in Q_e \\ 0, & \text{sonst} \end{cases}$$

Der Anteil der Wörter der Länge n die durch den DFA A akzeptiert werden, kann in der Variable $X^n_{q_0}$ des Startzustands q_0 für die Länge n abgelesen werden, da es sich bei einer Teilrechnung, die in einem akzeptierenden Zustand endet, nur um eine akzeptierende Rechnung handelt, wenn sie auch in einem Startzustand startet, da es sich hier allerdings nur um den Anteil der akzeptierten Wörter einer bestimmten Wortlänge handelt, ist die Zerfallsrate λ nicht nötig, da wir diese nur benötigen, wenn wir zusätzlich noch die Gewichtung der Wortlänge bestimmen möchten.

$$f_n(L(A)) = X^n_{q_0}$$

Beispiel 9. Im Folgenden möchten wir durch das vorgestellte Gleichungssystem die Anteile der akzeptierenden Wörter für die Wortlängen 0 bis 3 der Sprache des Automaten $F = (Q_F, \Gamma, \delta_F, q_0, Q^F_e)$ aus Abbildung 6 bestimmen. Für diesen Automaten erkennen wir, dass nur der Zustand q_2 ein Endzustand des Automaten darstellt. Daher können wir allen Variablen der Länge 0 außer $X^0_{q_2}$, die den Wert 1 besitzt, den Wert 0 zuweisen. Die Werte für X^0 sowie für die Wortlängen 1 bis 3 sind noch einmal in der Tabelle 4 zusammengefasst. In der Tabelle gibt die Zeile den Zustand q der Variable und die jeweilige Spalte die Wortlänge i an, sodass die jeweilige Zelle den Anteil der Läufe angibt, die in q starten und sich nach genau i Schritten in einem akzeptierenden Zustand befinden. Da der Wert der Variable $X^0_{q_0} = 0$ ist, wissen wir auch, das kein Wort der Länge 0 von unseren Automaten akzeptiert wird.

Nachdem wir die Werte für X^0 bestimmt haben, können wir darauf aufbauend auch die Werte für X^1 bestimmen. Für $X^1_{q_0}$ und $X^1_{q_e}$ erkennen wir, dass für alle Zustände $p \in Q$ entweder $X^0_p = 0$ oder $|\{\delta(q, a) = p \mid a \in \Gamma\}| = 0$ gilt. Damit ist der Wert der Variablen $X^1_{q_0}$ und $X^1_{q_e}$ gleich 0, wodurch wir auch wissen, dass ebenfalls kein Wort der Länge 1 von dem Automaten F akzeptiert wird. Für die Zustände q_1 und q_2 können wir $|\{\delta(q_1, a) = q_2 \mid a \in \Gamma\}| = 2$ und $|\{\delta(q, a) = q_2 \mid a \in \Gamma\}| = 2$ bestimmen für alle anderen Zustände ergibt sich als Betrag 0, sodass wir durch Verwendung von $X^0_{q_1}$ für $X^1_{q_0}$ und $X^1_{q_e}$ den Wert 1 bestimmen können.

X^i_q	0	1	2	3
q_0	0	0	$\frac{1}{2}$	$\frac{1}{2}$
q_1	0	1	1	1
q_2	1	1	1	1
q_e	0	0	0	0

Tabelle 4: f_n für verschiedene X^i_q.

Die Werte für X^2 und X^3 sowie für alle anderen X^n ergeben sich auf dieselbe weise wie für X^1. Die Ergebnisse befinden sich ebenfalls in der Tabelle 4. Hier erkennen wir durch Ablesen der Variable $X^n_{q_0}$, dass für 2 und 3 jeweils $\frac{1}{2}$ aller Wörter dieser Länge von der Sprache $L(F)$ akzeptiert werden. Da wir wissen, dass pro Wortlänge $|\Gamma|^n$ Wörter existieren, kommen wir so umgerechnet auf 2 beziehungsweise 4 Wörter, die von dem Automaten für diese Längen akzeptiert werden. Würden wir für dieses Beispiel die weiteren Werte für Wortlängen größer 3 berechnen, so würden wir als Wert der Variable eines Zustandes denselben Wert herausbekommen, wie wir ihn schon für die Längen 2 und 3 erhalten haben. Dies ist allerdings nicht immer der Fall, sondern nur genau dann, wenn ab einer bestimmten Länge alle Rechnungen in Zyklen der Länge 1 enden, dies bedeutet, dass alle erreichbaren Zustände ab einer bestimmten Länge nur noch auf sich selber abbilden. In diesem Beispiel ist dies bereits für alle Wörter, die mindestens die Länge 2 besitzen geben, des Weiteren hängt für diese Wörter die Akzeptanz nur vom ersten Buchstaben des Wortes ab.

Beweis der Korrektheit. Nachdem wir das Verfahren vorgestellt haben und anhand eines Beispiels verdeutlicht haben, möchten wir im Folgenden die Korrektheit dieses Verfahrens beweisen und zeigen, dass sich der Anteil der Wörter, die durch einen DFA für eine beliebige Wortlänge n akzeptiert werden, durch das vorgestellte Gleichungssystem berechnen lässt. Dazu zeigen wir, dass die Variable X^n_q genau den Anteil an Teilrechnungen der Länge n beschreibt, die ausgehend von q sich nach n Schritten in einem Endzustand befinden. Da bei DFAs nur genau ein Startzustand q_0 existiert, gilt nur für die Läufe, die in diesem Zustand Starten und in einem akzeptierenden Zustand Enden, dass es sich um eine akzeptierende Rechnung handelt. Somit beinhaltet die Variable $X^n_{q_0}$ für alle Wortlängen n den Anteil der akzeptierten Wörter dieser Wortlänge.

Theorem 4. Für alle $q \in Q$ und $n \in \mathbb{N}_0$ gilt: $X^n_q = \frac{|\{qw \vdash^n p\epsilon | p \in Q_e, w \in \Gamma^n\}|}{|\Gamma|^n}$

Beweis.
Induktionsanfang: $n = 0$
Zu zeigen: Für alle $q \in Q$ gilt: $X^0_q = \frac{|\{qw \vdash^0 p\epsilon | p \in Q_e, w \in \Gamma^0\}|}{|\Gamma|^0}$
Eine Rechnung der Länge 0 besitzt keine Schrittrelation, damit ist die Startkonfiguration auch die ak-

zeptierende Konfiguration. Dies bedeutet für uns, dass wir Folgendes zeigen müssen.

$$\textit{Zu zeigen}: \text{Für alle } q \in Q \text{ gilt:} X_q^0 = \frac{|\{q\epsilon \mid q \in Q_e\}|}{1}$$

Da jeder Zustand einzigartig ist und es für das leere Wort für jeden Zustand nur genau eine Rechnung existiert, folgt daraus, dass X_q^0 genau dann den Wert 1 besitzt, wenn $q \in Q_e$ und ansonsten den Wert 0. Da unsere Formel für X_q^0 genau dies beschreibt, haben wir den Induktionsanfang für die Länge 0 bereits bewiesen.

Induktionsschritt: $n \to n+1$
Zu zeigen: Wenn die Behauptung für alle X_q^n gilt, dann gilt sie auch für alle X_q^{n+1}.
Induktionsannahme: Für alle $q \in Q$ gilt: $X_q^n = \frac{|\{qw \vdash^n p\epsilon | p \in Q_e, w \in \Gamma^n\}|}{|\Gamma|^n}$
Induktionsbehauptung: (zu zeigen) Für alle $q \in Q$ gilt: $X_q^{n+1} = \frac{|\{qw \vdash^{n+1} p\epsilon | p \in Q_e, w \in \Gamma^{n+1}\}|}{|\Gamma|^{n+1}}$

Beweis des Schrittes:
Jede Teilrechnung der Länge $n+1$ von q nach p mit $q \in Q$ und $p \in Q_e$ besitzt die Form $qaw \vdash q'w \vdash^n p\epsilon$. Damit besteht sie aus den Teilrechnungen $qa \vdash q'\epsilon$ und $q'w \vdash^n p\epsilon$ für $q' \in Q, a \in \Gamma$ und $w \in \Gamma^n$. Somit lässt sich der Anteil für X_q^{n+1} folgendermaßen bestimmen.

$$X_q^{n+1} = \frac{|\{qaw \vdash^{n+1} p\epsilon \mid p \in Q_e, w \in \Gamma^{n+1}\}|}{|\Gamma|^{n+1}} \tag{1}$$

$$= \sum_{q' \in Q} X_{q'}^n \cdot \frac{|\{qa \vdash q'\epsilon \mid a \in \Gamma\}|}{|\Gamma|} \tag{2}$$

$$= \frac{1}{|\Gamma|} \sum_{q' \in Q} X_p^n \cdot |\{\delta(q,a) = p \mid a \in \Gamma\}| \tag{3}$$

Bei der Umformung haben wir ausgenutzt, dass Teilrechnungen der Länge 1 durch die Transitionsfunktion eines DFA dargestellt werden, wodurch eine Teilrechnung der Form $qa \vdash q'\epsilon$ mit $a \in \Gamma$ und $q, q' \in Q$ genau dann existiert, wenn $\delta(q,a) = p$ gilt. Diese Tatsache nutzen wir in Schritt 3 der Umformung aus. Außerdem wissen wir, dass die Transitionsfunktion für alle $a \in \Gamma$ mit q definiert ist. Daraus folgt, dass insgesamt $|\Gamma|$ Rechnungen der Länge 1 existieren, was wir in Schritt 2 der Umformung benutzen.
Da wir so die Formel zu Bestimmung einer Variable bekommen, ist auch der Induktionsschritt bewiesen und das Theorem als korrekt bewiesen ist. □

4.2 Ansatz mittels Matrizen

Eine weitere Variante ist, dass das Verhalten eines DFA $A = (Q, \Gamma, \delta, Q_0, Q_e)$ durch Matrizen zu interpretieren. Dabei nutzen wir die Eigenschaft aus, dass es für jedes Wort nur genau eine Rechnung auf einem DFA existiert und bauen damit auf dem Wortproblem für unäre Alphabete [11] auf.
Dafür erstellen wir für jede Wortlänge n eine Matrix $M_{Q\times Q}^n$. Dabei besteht die Matrix $M_{Q\times Q}^n$ für die Wortlänge n aus genau $|Q|$ Zeilen und $|Q|$ Spalten, die jeweils genau einen Zustand repräsentieren, sodass die Zelle $M_{q,p}^n$ mit $q, p \in Q$ den Anteil der $|\Gamma|^n$ Teilrechnungen, die im Zustand q starten und nach n Schritten sich im Zustand p befindet, zum Ausdruck bringt.

4.2.1 Das Wortproblem für unäre Alphabete

Für unäre Alphabete gibt es für jede Wortlänge genau ein Wort. Damit muss nur entschieden werden, ob dieses Wort von der Sprache akzeptiert wird oder nicht. Dies wurde sich beim Wortproblem für unäre Alphabet zunutze gemacht[11], welches im Gegensatz zu unserem Verfahren im nachfolgenden Kapitel auch auf NFAs angewendet werden kann. Dabei wurde die Frage gestellt, ob für die Wortlänge n ein Wort von der Sprache akzeptiert wird, um das Wortproblem für unäre Alphabete in polynomialer Laufzeit für exponentiell lange Wörter zu lösen. Das wir das Wortproblem für den Spezialfall, dass wir nur ein unäres Alphabet besitzen, im Gegensatz zu dem herkömmlichen Wortproblem in polynomialer Laufzeit lösen können, liegt daran, dass unsere Eingabe kein Wort, sondern nur dessen Länge enthalten muss und wir beim Lösen des Problems die Abkürzung über die 2-adische Entwicklung der Wortlänge nehmen können. Damit müssen für eine Wortlänge n nur maximal logarithmisch viele Wortlängen, die jeweils durch eine Matrix interpretiert werden, zur Berechnung betrachtet werden und es muss nicht wie üblich die gesamte

Rechnung des Wortes auf dem Automaten durchlaufen werden, sodass wir alle Wortlängen bis n betrachten müssen. Daher schauen wir uns, bevor wir die verwendeten Erreichbarkeitsmatrizen vorstellen, an, wie wir eine Wortlänge n in ihre 2-adische Entwicklung zerlegen können, dabei beschreibt $m = \lfloor \log_2(n) \rfloor$ den größten benötigten Exponenten.

$$n = \sum_{i=0}^{m} a_i \cdot 2^i = a_0 + a_1 2 + a_2 2^2 + \cdot + a_m 2^m \text{ mit } a_i \in \{0, 1\}$$

Die Erreichbarkeitsmatrix für die Wortlänge 1 ist so definiert, dass eine Zelle der Matrix genau dann den Wert 1 besitzen, wenn das Paar, was die Zelle repräsentiert, auch für den einzigen Buchstaben in der Transitionsrelation des Automaten vorkommt, ist dies nicht der Fall, so enthält die Zelle den Wert 0.

$$M^1_{q,p} = \begin{cases} 1, & \text{wenn } q \times a \times p \in \delta \text{ für } a \in \Gamma \\ 0, & \text{sonst} \end{cases} \tag{4}$$

Die Matrix M^{2^i} mit $i > 1$ für die Wortlänge 2^i erhalten wir dann durch i-faches Quadrieren der Matrix M^1, sodass wir alle benötigten Matrizen durch maximal m-faches Quadrieren der Ursprungsmatrix bestimmen können. Die Matrix für die Länge n erhalten wir anschließend, indem wir alle Zweierpotenz Matrizen miteinander multiplizieren die, die 2-adische Entwicklung von n darstellen und für die a_i den Wert 1 annimmt.
Das Wort der Länge n wird genau dann von dem Automaten akzeptiert, wenn in der Erreichbarkeitsmatrix für die Länge n ein Paar existiert, bei dem $q \in Q_0$ und $p \in Q_e$ und die entsprechende Zelle $M^n_{q,p}$ den Wert 1 besitzt. Ist dies nicht der Fall, so liegt das Wort nicht in der Sprache, die durch den Automaten repräsentiert wird.
Damit ist auch die ursprüngliche Frage beantwortet wie viele Wörter für die Wortlänge n von dem gegebene Automaten akzeptiert werden, da wie bereits erwähnt pro Wortlänge nur genau ein Wort existieren kann.

4.2.2 Größere Alphabete

Nachdem wir gezeigt haben, wie wir das Problem für unäre Alphabete lösen, betrachten wir im folgenden Alphabete, die aus mehr als einem Buchstaben bestehen, allerdings mit der Einschränkung, dass unser Verfahren nur noch auf DFAs angewendet werden kann. Dabei nutzen wir die Eigenschaft von DFAs aus, dass es für jedes Wort nur genau eine Rechnung auf dem Automaten existiert. Da NFAs diese Eigenschaft nicht besitzen, können wir das nachfolgende Verfahren nur dann auf einem NFA anwenden, wenn wir ihn mittels der Potenzmengenkonstruktion, die wir in Kapitel 2.2.1 vorgestellt haben, in einen äquivalenten DFA überführen. Auch andere Ansätze, das Verfahren so abzuändern, um es auf NFAs anwenden zu können, haben bisher nicht zum Erfolg geführt.
Im Folgenden repräsentiert der Wert der Zelle $M^n_{q,p}$ genau den Anteil an Teilrechnungen die nach n Schritten ausgehend vom Zustand q sich im Zustand p befinden im Verhältnis zu allen Teilrechnungen der Länge n die im Zustand q starten. Möchten wir daraus die Anzahl der Teilrechnungen bestimmen die nach n Schritten ausgehend von q sich in p befinden, müssen wir den Wert der Zelle $M^n_{q,p}$ mit der maximalen Anzahl an Wörtern für diese Wortlänge $|\Gamma|^n$ multiplizieren.
Zu Beginn betrachten wir, wie wir die Erreichbarkeitsmatrix M^1 für die Länge 1 bestimmen können. Dazu bestimmen wir eine Funktion g die jedem Tripel (q, p, a) mit $q, p \in Q$ und $a \in \Gamma$ den Wert 0 oder $\frac{1}{|\Gamma|}$ zuordnet. Das Tripel besitzt genau dann den Wert $\frac{1}{|\Gamma|}$, wenn die Transitionsfunktion δ von q mit a auf p abbildet.

$$g(q, a, p) = \begin{cases} \frac{1}{|\Gamma|}, & \text{wenn } \delta(q, a) = p \\ 0, & \text{sonst} \end{cases}$$

Darauf aufbauend bestimmen wir den Wert der Zelle $M^1_{q,p}$ indem wir die Summe der Funktion g über alle Buchstaben des Alphabets, für das geordnete Paar welches die Zelle repräsentiert, bilden.

$$M^1_{q,p} = \sum_{a \in \Gamma} g(q, a, p)$$

Anders ausgedrückt ist die Matrix M^1 eine andere Darstellung der Transitionsfunktion, bei der die Informationen, welcher Buchstabe durch die Benutzung einer Transition gelesen wird, verloren geht. Allerdings beschreibt die Matrix auf wie vielen unterschiedlichen Wegen wir von dem Zustand q zum Zustand p mit

nur einem Schritt gelangen. An der Anzahl der unterschiedlichen Wege sind wir im folgenden auch für Wörter, die eine Länge größer 1 besitzen, interessiert, diese Anzahl können wir beispielhaft für die Länge 2 durch $M^1 \cdot M^1$ bestimmen. Dies funktioniert, da wenn wir uns vorstellen, dass es auf einem DFA 2 Wege von p nach q und ebenfalls 2 Wege von q nach r der Länge 1 existieren, so existieren genau 4 verschiedene Wege von p nach r der Länge 2.
Die Matrix für die Länge n erhalten wir durch $n-$faches multiplizieren der Ursprungsmatrix M^1 mit sich selber. Auch die Zerlegung der Wortlänge n in ihre 2-adische Entwicklung und das Multiplizieren der entsprechenden 2-er Potenz Matrizen wie beim Wortproblem für unäre Alphabete (Abschnitt 4.2.1) ist möglich, allerdings für den Anwendungsfall dieser Arbeit nicht wirklich sinnvoll, da wir alle Zwischenschritte zum Bestimmung des Gewichts einer Sprache ohnehin benötigen.
Da wir nicht an der Anzahl, sondern den Anteil der Teilrechnungen interessiert sind, wird hier auch erkenntlich, warum wir bei der Funktion jede vorkommende Transition noch durch die Größe des Alphabets teilen müssen.
Wenn wir die Matrix M^n bestimmt haben, können wir auch den Anteil der Wörter der Länge n bestimmen, die von dem DFA A akzeptiert werden, dazu bilden wir die Summe über alle Zellen der Zeile des Startzustands, für die der Zustand, der angibt, in welchem Zustand die Rechnungen enden, sich in der Menge der Endzustände befindet.

$$f_n(L(A)) = \sum_{p \in Q_e} M^n_{q_0,p}$$

Beispiel 10. Wir möchten anhand des DFA $F = (Q_F, \Gamma, \delta_F, q_0, Q_e^F)$ der in Abschnitt 2.2.4 die symmetrische Differenz beschreibt und unter Abbildung 6 dargestellt ist, dass Verfahren verdeutlichen und so den Anteil der akzeptierenden Wörter sowie die Anzahl der Wörter für die Wortlängen 1 bis 4 mithilfe des vorgestellten Matrizen-Verfahren bestimmen.
Dafür erstellen wir durch die gegebene Transitionsfunktion δ_F und für das Alpabet $\Gamma = \{a, b\}$ mit der Alphabetsgröße 2, die initiale Matrix M^1 für die Länge 1 die in der unten stehenden Matrix dargestellt ist. Der DFA F besitzt den Startzustand q_0 und als einzigen Endzustand q_2, sodass uns der Wert der Zelle $M^1_{q_0,q_2} = 0$ den Anteil der akzeptierenden Wörter der Länge 1 angibt. Da der Anteil 0 ist, gibt es auch kein Wort der Länge 1, welches von dem Automaten akzeptiert wird.
Durch Quadrierung der Matrix M^1 erhalten wir die Matrix M^2 für die Wortlänge 2. Auch hier können wir leicht den Anteil der akzeptierenden Wörter ablesen. Dieser Anteil beträgt $\frac{1}{2}$, daraus können wir wiederum die Anzahl der akzeptierten Wörter für die Wortlänge 2 mittels $\frac{1}{2}|\Gamma|^2 = 2$ bestimmen.

$$M^1 = \begin{array}{c} \\ q_0 \\ q_1 \\ q_2 \\ q_e \end{array} \begin{array}{c} \begin{array}{cccc} q_0 & q_1 & q_2 & q_e \end{array} \\ \begin{pmatrix} 0 & \frac{1}{2} & 0 & \frac{1}{2} \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \end{array} \qquad M^2 = \begin{array}{c} \\ q_0 \\ q_1 \\ q_2 \\ q_e \end{array} \begin{array}{c} \begin{array}{cccc} q_0 & q_1 & q_2 & q_e \end{array} \\ \begin{pmatrix} 0 & 0 & \frac{1}{2} & \frac{1}{2} \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \end{array}$$

Würden wir für dieses Beispiel die weiteren Matrizen für die Längen $n \geq 3$ berechnen, würden wir für alle Wortlängen dieselbe Matrix wie für die Länge 2 als Ergebnis heraus bekommen, sodass wir auch für diese Wortlängen schnell die Anzahl der Wörter berechnen können. Damit kommen wir zum Beispiel für die Wortlängen 3 und 4 auf 4 beziehungsweise 8 Wörter, die von dem Automaten F akzeptiert werden. Dass die Matrizen ab einer bestimmten Wortlänge gleich bleiben, ist aus denselben Gründen wie beim Beispiel 9 für das Gleichungssystem nicht für alle Automaten gegeben.

Beweis der Korrektheit. Nachdem wir das Verfahren vorgestellt und anhand eines Beispiels durchgerechnet haben, wollen wir im Folgenden zeigen, dass wir mithilfe von Matrizen den korrekten Anteil der Wörter, die durch einen DFA akzeptiert werden, für eine Wortlänge n erhalten. Dazu zeigen wir zunächst, dass die Summe jeder Zeile den Wert 1 ergibt und das, wenn m Wörter der Länge n existieren, die in q starten und in p enden, die Zelle $M^n_{q,p}$ den Wert $\frac{m}{|\Gamma|^n}$ annimmt. Da wir wissen das für ein DFA nur einen Startzustand q_0 existiert, repräsentieren die Zellen der Matrix $M^n_{q_0,p}$ mit $p \in Q_e$ die akzeptierenden Rechnungen der Länge n, sodass die Summe über diese Zellen den Anteil f_n ergeben.

Lemma 5. Für alle $q \in Q$ und $n \in \mathbb{N}_{\geq 1}$ gilt: $\sum_{p \in Q} M^n_{q,p} = 1$.

Beweis.
Induktionsanfang: $n = 1$
Zu zeigen: Für alle $q \in Q$ gilt: $\sum_{p \in Q} M^1_{q,p} = 1$.
Dazu betrachten wir den Definitionsbereich der Transitionsfunktion δ: $D_\delta = \{(q, a) \mid q \in Q \text{ und } a \in \Gamma\}$. Hier ist zu erkennen, dass es für jedes $q \in Q$ genau $|\Gamma|$ viele $a \in \Gamma$ gibt. Die jeweils auf genau einen Zustand abbilden. Da wir aus der Definition der Funktion g wissen, dass jede Vorkommende Transition den Wert $\frac{1}{|\Gamma|}$ besitzt, folgt daraus:

$$\Rightarrow \text{ für alle } q \in Q \text{ gilt: } \sum_{a \in \Gamma} \sum_{p \in Q} g(q, a, p) = |\Gamma| \frac{1}{|\Gamma|} = 1$$

Damit ist die Aussage für die Wortlänge 1 gezeigt.

Induktionsschritt: $n \to n + 1$
Zu zeigen: Wenn für alle $q \in Q$ $\sum_{p \in Q} M^n_{q,p} = 1$ gilt, so gilt auch für alle $q \in Q$ $\sum_{p \in Q} M^{n+1}_{q,p} = 1$.
Induktionsannahme: für alle $q \in Q$ gilt: $\sum_{p \in Q} M^n_{q,p} = 1$.
Induktionsbehauptung: (zu zeigen) für alle $q \in Q$ gilt: $\sum_{p \in Q} M^{n+1}_{q,p} = 1$.

Beweis des Schrittes:
Um die Matrix M^{n+1} zu erhalten, müssen wir die Matrizen M^n und M^1 miteinander multiplizieren, sodass sich der Wert einer Zelle folgendermaßen berechnen lässt.

$$M^{n+1}_{q,p} = \sum_{q' \in Q} M^n_{q,q'} \cdot M^1_{q',p}$$

Um den Wert der Zeile q zu erhalten, müssen wir die Summe über alle Zellen dieser Zeile bilden. Dabei nutzen wir aus, dass wir durch den Induktionsanfang bereits wissen, dass für die Matrix M^1 die Summe jeder Zeile den Wert 1 ergibt. Des Weiteren nehmen wir an, dass unsere Induktionsannahme korrekt ist. Diese beide Tatsachen nutzen wir in Schritt 3 der folgenden Umformung aus, um damit zu Zeigen, dass auch der Induktionsschritt von Lemma 5 korrekt ist.

$$\begin{aligned} \sum_{p \in Q} M^{n+1}_{q,p} &= \sum_{q' \in Q} \sum_{p \in Q} M^n_{q,q'} \cdot M^1_{q',p} && (1) \\ &= \sum_{q' \in Q} M^n_{q,q'} \sum_{p \in Q} M^1_{q',p} && (2) \\ &= 1 \cdot 1 = 1 && (3) \end{aligned}$$

Damit ist gezeigt das für alle $q \in Q$ $\sum_{p \in Q} M^n_{q,p} = 1$ gilt. □

Theorem 5. Der wert von $M^n_{p,q}$ ist genau $\frac{m}{|\Gamma|^n}$, falls es m verschiedene Wörter w existieren, sodass $qw \vdash^n p\epsilon$ gilt.

Beweis.
Dazu zeigen wir im Folgenden, dass jedes Wort w, welches die Bedienung erfüllt, dass $qw \vdash^n p\epsilon$ gilt, mit $\frac{1}{|\Gamma|^n}$ zum Gesamtwert der Zelle $M^n_{p,q}$ beiträgt, sodass wenn m verschiedene Wörter die Bedingung erfüllen der Wert der Zelle $M^n_{p,q}$ gleich $\frac{m}{|\Gamma|^n}$ ist.

Induktionsanfang: $n = 1$
Zu zeigen: Jedes Wort $w \in \Gamma$ mit $qw \vdash^1 p\epsilon$ und $q, p \in Q$ trägt mit $\frac{1}{|\Gamma|}$ zum Gesamtwert der Zelle $M^1_{q,p}$ bei.
Die Wörter der Länge 1 werden durch die Transitionsfunktion repräsentiert. Das heißt, wenn eine Teilrechnung ausgehend von q nach p existiert und dabei das Wort $w = a$ mit $a \in \Gamma$ gelesen wird, bedeutet dies auch, dass $\delta(q, a) = p$ gilt, wodurch die Funktion g für das Tripel (q, a, p) den Wert $\frac{1}{|\Gamma|}$ annimmt. Dies wiederum bedeutet, dass dieses Tripel mit $\frac{1}{|\Gamma|}$ zum Gesamtwert der Zelle $M^1_{q,p}$ beiträgt, da sich der Wert über die Summe aller Buchstaben des Alphabets ergibt und jeder Buchstabe mit dem Zustand q nur auf genau einen Zustand abbildet, existiert für jeden Buchstaben auch nur genau eine Rechnung der

Länge 1, wodurch wir den Induktionsanfang bereits zeigen konnten.

Induktionsschritt: $n \to n+1$
Zu zeigen: Wenn die Behauptung für Wörter der Länge n gilt, so gilt sie auch für Wörter der Länge $n+1$.
Induktionsannahme: Jedes Wort $w \in \Gamma^n$ mit $qw \vdash^n p\epsilon$ und $q, p \in Q$ trägt mit $\frac{1}{|\Gamma|^n}$ zum Gesamtwert der Zelle $M^n_{q,p}$ bei.
Induktionsbehauptung: (zu zeigen) Jedes Wort $w \in \Gamma^{n+1}$ mit $qw \vdash^{n+1} p\epsilon$ und $q, p \in Q$ trägt mit $\frac{1}{|\Gamma|^{n+1}}$ zum Gesamtwert der Zelle $M^{n+1}_{q,p}$ bei.

Beweis des Schrittes:
Wenn eine Teilrechnung $qwa \vdash^{n+1} p\epsilon$ mit $w \in \Gamma^n$ und $a \in \Gamma$ existiert, dann muss auch eine Teilrechnung $qwa \vdash^n q'a$ mit $q' \in Q$ existieren und zusätzlich muss für δ gelten, dass: $\delta(q', a) = p$ gilt.
Dies bedeutet der Wert der Zelle $M^{n+1}_{q,p}$ für das Wort wa setzt sich zusammen aus dem Wert für das Wort w, welches nach Induktionsannahme sich mit $\frac{1}{|\Gamma|^n}$ in der Zelle $M^n_{q,q'}$ niederschlägt und dem Wert für das Wort a, welches sich nach Induktionsanfang sich mit $\frac{1}{|\Gamma|}$ in der Zelle $M^1_{q',p}$ niederschlägt. Dies ist damit zu erklären, dass das Matrixprodukt zweier Matrizen die Konkatenation der Wörter, die durch die beiden Matrizen interpretiert werden, darstellt. Damit ergibt sich der Wert, der für ein Wort w, welches im Zustand q starte und nach $n+1$ Schritten im Zustand p endet folgendermaßen:

$$\frac{1}{|\Gamma|^n} \cdot \frac{1}{|\Gamma|} = \frac{1}{|\Gamma|^{n+1}}$$

Damit ist gezeigt, dass jedes Wort der Länge n welches die Bedingung $qw \vdash^n p\epsilon$ erfüllt mit $\frac{1}{|\Gamma|^n}$ zum Wert der Zelle $M^n_{q,p}$ beiträgt. Damit ist gezeigt, dass wenn m verschieden Wörter dies erfüllen, der Wert der Zelle $M^n_{q,p} = \frac{m}{|\Gamma|^n}$ beträgt, womit Theorem 5 bewiesen ist. □

Somit konnten wir zeigen, dass sich der Anteil der akzeptierenden Wörter auch mithilfe der vorgestellten Matrizen bestimmen lässt.

Korollar 1. Damit ist gezeigt, dass $f_n(L(A)) = \sum_{p \in Q_e} M^n_{q_0,p}$ korrekt ist.

Dies ist damit zu erklären, dass eine akzeptierende Rechnung in dem Startzustand q_0 beginnt und in einem akzeptierenden Zustand endet. Damit müssen wir nur die Rechnungen betrachten, die dies erfüllen, für alle diese Rechnungen wissen wir durch Theorem 5, dass sie sich in den Zellen q_0, p mit $p \in Q_e$ mit dem jeweiligen Wert niederschlagen. Somit müssen wir abschließend nur die Summe über alle diese Zellen bilden, um den Anteil zu bekommen, da wir bereits in Lemma 5, dass die Summe jeder Zeile 1 ergibt.

4.3 Vergleich der Varianten

Zum Abschluss dieses Kapitels möchten wir die beiden vorgestellten Verfahren zur effektiven Bestimmung des Anteils der akzeptierenden Wörter für eine Wortlänge bezüglich ihrer Laufzeiten miteinander vergleichen. Dazu betrachten wir zunächst die asymptotischen Laufzeiten (Abschnitt 4.3.1) und vergleichen sie mit der Laufzeit des Wortproblems, welches wir ebenfalls zum lösen des Problems verwenden können. Anschließend betrachten wir die praktischen Laufzeiten (Abschnitt 4.3.2) der Verfahren für die Aufgabenstellung der Bestimmung der Gewichtung einer Sprache.

4.3.1 Asymptotische Laufzeit

Da wir für unsere konkrete Aufgabenstellung immer die Anteile aller Wortlängen bis zu einer bestimmten Wortlänge benötigen, können wir für eine Wortlänge n auch davon ausgehen, dass der Anteile der Wortlänge $n-1$ bereits bekannt ist. Daher bestimmen wir im Folgenden die Laufzeit, die wir zur Bestimmung des Anteils der Wortlänge n benötigen, wenn der Anteil für die Wortlänge $n-1$ bereits bekannt ist. Hier besteht auch schon der erste Unterschied der beiden Verfahren für das Gleichungssystem müssen wir ohnehin alle Zwischenwerte berechnen, während wir für die Matrizen den Anteil der akzeptierenden Wörter auch über die Binärdarstellung bestimmen können, was bezüglich der Laufzeit deutlich effektiver ist. Auch wenn wir dies für unsere Aufgabenstellung nicht benötigen, könnte es für andere Anwendungsfälle durchaus von Interesse sein. Außerdem geben wir die Laufzeit für die erste Initialisierung des jeweiligen Verfahrens an, dies entspricht der Bestimmung der Wortlänge 0 beziehungsweise bei dem

Verfahren mittels Matrizen für die Wortlänge 1. Für das Matrizen-Verfahren könnten wir ebenfalls auch eine Matrix für die Länge 0 angeben, dies wäre die Einheitsmatrix. Doch dies ist nicht wirklich sinnvoll, da wir mit dieser Matrix nicht weiter Rechnen können, weil sie nicht die Transitionsfunktion eines DFAs interpretiert. Daher ist es praktisch deutlich sinnvoller für das Matrizen-Verfahren bei der Wortlänge 1 zu beginnen und falls benötigt die Länge 0 dadurch zu bestimmen, dass wir überprüfen ob, das leere Wort in der Sprache des Automaten liegt.
Bevor wie die asymptotischen Laufzeiten für die beiden Varianten Gleichungssystem (Abschnitt 4.1) und Matrizen (Abschnitt 4.2.2) bestimmen und miteinander vergleichen, möchten wir zunächst uns die Laufzeit für das Lösen des Problems, dass wir den Anteil der akzeptierenden Wörter für eine Wortlänge bestimmen wollen, durch die Variante, dass wir für alle möglichen Wörter dieser Wortlänge das Wortproblem lösen um so den Anteil zu bestimmen, anschauen.

Lösen aller Wortprobleme. Das Wortproblem für DFAs lässt sich in linearer Zeit bezüglich der Wortlänge des Wortes, für das wir bestimmen wollen ob es von dem DFA akzeptiert wird, lösen [9]. Dies ist damit zu erklären, dass für jedes Wort auf einem DFA genau eine Rechnung existiert, wir müssen also nur Schritt für Schritt die von der Rechnung verwendeten Transitionen durchgehen und nach dem letzten Schritt entscheiden, ob wir uns in einem Endzustand befinden oder nicht, um das Wortproblem zu lösen. Unser Problem bezüglich der Laufzeit ist nur, dass wir für jede Wortlänge n nicht nur ein Wortproblem lösen müssen, sondern $|\Gamma|^n$ viele, eben für jedes mögliche Wort der Länge n das entsprechende Wortproblem, sodass wir auf eine asymptotische Laufzeit von $\mathcal{O}\left(|\Gamma|^n \cdot n\right)$ kommen. Diese Laufzeit können wir noch leicht verbessern, indem wir uns die gelösten Wortprobleme für die Länge $n-1$ zwischenspeichern. Somit müssen wir nur noch den letzten Schritt der Rechnung durchführen, was uns zu einer konstanten Laufzeit für das Lösen der weiteren Wortprobleme führt. Allerdings müssen wir dies weiterhin $|\Gamma|^n$ oft machen, sodass wir eine Laufzeit von $\mathcal{O}\left(|\Gamma|^n \cdot 1\right) = \mathcal{O}\left(|\Gamma|^n\right)$ erreichen. Dafür ist die Initialisierung dieses Verfahrens für die Wortlänge 0 in konstanter Zeit durchführbar, dies liegt daran, dass nur genau ein Wort für die Länge 0 existiert und wir keinen Schritt auf dem Automaten machen müssen. Somit müssen wir nur überprüfen, ob sich der Startzustand auch in der Menge der Endzustände befindet.

Ansatz mittels eines Gleichungssystems. Als Nächstes betrachten wir die asymptotische Laufzeit, wenn wir das Problems der Bestimmung des Anteils mithilfe eines Gleichungssystems lösen. Hier benötigen wir für die Initialisierung der Wortlänge eine Laufzeit von $\mathcal{O}\left(|Q|\right)$. Dies ist damit zu erklären, dass wir einmal über alle Zustände laufen müssen und jeweils überprüfen müssen, ob sich der jeweilige Zustand in der Menge der Endzustände befindet. Um anschließend den Anteil für die weiteren Wortlängen zu bestimmen, müssen wir für jede Wortlänge zunächst für jedes geordnete Paar von zwei Zuständen die Anzahl der vorkommenden Transitionen zwischen ihnen bestimmen, um darauf aufbauend dieses Ergebnis mit dem Wert der Variable für die vorangegangenen Wortlänge zu multiplizieren. Dies bedeutet wir müssen für jedes mögliche Tripel (p, a, q) mit $a \in \Gamma$ und $p, q \in Q$ bestimmen ob die entsprechende Transition vorkommt oder nicht vorkommt. Die anschließende Multiplikation mit dem Wert der vorherigen Wert der Variable lässt sich in konstanter Zeit durchführen, sodass wir durch die Tatsache das wir den Anteil f_n anhand der Variable $X^n_{q_0}$ ablesen können auf eine asymptotische Laufzeit von $\mathcal{O}\left(|Q|^2 \cdot |\Gamma|\right)$ kommen.
Diese Laufzeit können wir dadurch verbessern, dass wir die Anzahl der Transitionen zwischen zwei Zuständen nicht für jede Wortlänge neu bestimmen müssen, weil diese sich nicht verändert. Stattdessen können wir diese Anzahl einmal initial bestimmen und abspeichern, was uns in die Lage versetzt, für alle Wortlängen darauf zuzugreifen, ohne die Anzahl neu bestimmen zu müssen. Damit müssen wir nur für jedes Paar von Zuständen die Anzahl der Transitionen zwischen ihnen abrufen und dann mit dem zuvor bestimmten Gewicht des Zustands, auf den abgebildet wird, multiplizieren, was zu folge hat, dass sich unsere Laufzeit auf $\mathcal{O}\left(|Q|^2\right)$ verbessert. Allerdings kommt in diesem Fall bei der Initialisierung noch die Laufzeit für die einmalige Bestimmung der Anzahl der Transitionen hinzu, diese Bestimmung hat eine Laufzeit wie bereits erwähnt von $\mathcal{O}\left(|Q|^2 \cdot |\Gamma|\right)$, da dies größer ist als die Laufzeit der bisherigen Initialisierung ergibt, dies auch unsere asymptotische Laufzeit für die Initialisierung. $\mathcal{O}\left(|Q|^2 \cdot |\Gamma|\right) + \mathcal{O}\left(|Q|\right) = \mathcal{O}\left(|Q|^2 \cdot |\Gamma|\right)$.

Ansatz mittels Matrizen. Die Zeitkomplexität für die Multiplikation von zwei $n \times n$ Matrizen beträgt, wenn man die Multiplikation auf herkömmliche Weise durchführt $\mathcal{O}\left(n^3\right)$. Allerdings ist die Zeitkomplexität für die Multiplikation von Matrizen ein aktuelles Forschungsthema, sodass die aktuell besten Verfahren zur Multiplikation von von zwei $n \times n$ Matrizen eine Laufzeit von $\mathcal{O}\left(n^{2,37286}\right)$ benötigen [1, 6], sodass wir eine Laufzeit von $\mathcal{O}\left(|Q|^{2,37286}\right)$ benötigen um die Matrix der Wortlänge n zu bestimmen,

wenn die Matrix der Wortlänge $n-1$ bereits bekannt ist. Da wir um den Anteil f_n zu bestimmen nur die Summe über gewisse Zellen bilden müssen, ist die Laufzeit zur Multiplikation von Matrizen auch die asymptotische Laufzeit dieses Verfahrens.
Damit benötigen wir nur noch die Laufzeit zu Erstellung der initiale Matrix für die Länge 1. Dazu müssen wir einmal über alle Buchstaben des Alphabets laufen und nachschauen, ob das Tripel bestehend aus den beiden Zuständen, die die Zelle repräsentieren und den aktuellen Buchstaben in den Transitionen vorkommt. Da genau $|Q| \cdot |Q|$ Zellen existieren, kommen wir so auf eine asymptotische Laufzeit von $\mathcal{O}\left(|Q|^2 \cdot |\Gamma|\right)$.

Zusammenfassung. Damit haben wir für alle drei Ansätze die asymptotische Laufzeit bestimmt, wenn wir den Anteil bestimmen und der Anteil der vorherigen Wortlänge bereits bekannt ist. Für unseren Anwendungsfall müssen wir die Anteile allerdings für alle Wortlängen bis zu einer bestimmen Länge η bestimmen, daher bestimmen wir noch die dafür benötigte Laufzeit. Diese Laufzeit bestimmen wir, indem wir über alle Wortlängen iterieren und die jeweils benötigte Laufzeit aufsummieren. Für die Ansätze mittels Gleichungssystems und Matrizen müssen dafür einfach nur η mal den Schritt zur Bestimmung der nächsten Länge anwenden und die Laufzeit zur Initialisierung hinzuaddieren. Dies ist möglich, da die Laufzeiten dieser Verfahren nicht von der Wortlänge abhängen. Für den Ansatz mittels des Wortproblems ist dies nicht so, sodass wir auf eine asymptotische Laufzeit von $\mathcal{O}\left(\sum_{n=0}^{\eta} |\Gamma|^n\right)$ kommen, den Wert dieser Summe haben wir bereits in Kapitel 3.3 für die Anzahl der Wörter im konstanten Teil der Gewichtung mittels der Partialsumme der geometrischen Reihe bestimmt.
In Tabelle 5 haben wir die herausgearbeiteten Laufzeiten noch einmal zusammen gefasst. Hier erkennen wir, dass das Verwenden des Wortproblems zeittechnisch nicht wirklich sinnvoll ist, weil wir für jede Bestimmung des Anteils einer Wortlänge n eine exponentielle Laufzeit bezüglich der Wortlänge benötigen. Daher betrachten wir im folgenden Abschnitt bei den praktischen Laufzeiten diese Variante nicht.
Die Ansätze mittels des Gleichungssystems und der Matrizen besitzen im Vergleich zum Wortproblem eine deutlich effektivere Laufzeit zur Bestimmung eines Anteils, so benötigt das Gleichungssystem nur quadratische Laufzeit und die Matrizen etwas mehr als quadratische Laufzeit. Trotzdem werden wir im folgenden Abschnitt sehen, dass sich in der Praxis herausgestellt hat, dass Matrizen in den meisten Fällen das effektivere Verfahren darstellt. Für die Initialisierung benötigen beide Verfahren die gleiche Laufzeit, die zwar deutlich schlechter ist als die des Wortproblems. Aber durch die Tatsache, dass die Initialisierung nur einmal durchgeführt werden muss, ist dieser Vorteil des Wortproblems zu vernachlässigen.

	Wortproblem	Gleichungsystem	Matrizen
Initialisierung	$\mathcal{O}(1)$	$\mathcal{O}\left(\lvert Q\rvert^2 \cdot \lvert\Gamma\rvert\right)$	$\mathcal{O}\left(\lvert Q\rvert^2 \cdot \lvert\Gamma\rvert\right)$
Bestimmung f_n, wenn f_{n-1} bekannt.	$\mathcal{O}\left(\lvert\Gamma\rvert^n\right)$	$\mathcal{O}\left(\lvert Q\rvert^2\right)$	$\mathcal{O}\left(\lvert Q\rvert^{2,37286}\right)$
Bestimmung aller f_n von 0 bis η.	$\mathcal{O}\left(\sum_{n=0}^{\eta} \lvert\Gamma\rvert^n\right)$	$\mathcal{O}\left(\eta \cdot \lvert Q\rvert^2 + \lvert Q\rvert^2 \cdot \lvert\Gamma\rvert\right)$	$\mathcal{O}\left(\eta \cdot \lvert Q\rvert^{2,37286} + \lvert Q\rvert^2 \cdot \lvert\Gamma\rvert\right)$

Tabelle 5: Asymptotische Laufzeiten der verschiedene Verfahren zur Bestimmung des Anteils akzeptierender Wörter einer regulären Sprache, die durch eine DFA charakterisiert wird.

4.3.2 Praktische Laufzeit

Abschließend möchten wir uns die praktischen Laufzeiten für die Varianten zu Bestimmung des Anteils der akzeptierenden Rechnungen anschauen. Dabei erfolgte die Implementierung in Python 3 und die Laufzeiten für die verschiedenen Verfahren wurden auf einem Rechner mit 16 GB Arbeitsspeicher und einer Prozessorleistung von 1,8 GHz durchgeführt. Für den Vergleich betrachten wir studentische Abgaben von verschiedenen Aufgabenstellungen. Die Aufgabenstellungen, die wir betrachten, wurden alle samt in der Vorlesung „Formale Sprachen und Logik“ an der Universität Kassel im Sommersemester 2021 beziehungsweise im Sommersemester 2020 als bewertete Hausaufgaben gestellt. Dabei sollte jeweils ein NFA angegeben werden, der die Sprache einer der folgenden Aufgabenstellungen beschreibt.

a) $\Gamma = \{a, 0, 1\}$ und $L = \{x \cdot a^n \in \Gamma^* \mid x = 0, \text{ falls } n \text{ gerade und } x = 1 \text{ sonst}\}$.

b) $L = \{w \in \{a, b\}^* \mid w \text{ enthält das Wort } abba \textbf{ nicht}\}$.

c) $L = \{w \in \{a, b, c, d\}^* \mid |w|_a \geq 1 \text{ und } |w|_b \geq 1 \text{ und } |w|_c \geq 1\}$.

d) Typ-3 Grammatik $G = (\{S, A, B, C\}, \{a, b\}, P, S)$ mit
$$\begin{aligned} P = \{&S \to \epsilon \mid aS \mid bB \mid B \\ &B \to bB \mid aA \mid A \\ &A \to bS \mid aC \\ &C \to aC \mid bC \mid a \mid b\}, \end{aligned}$$
somit gilt: $L(G) = \Gamma^*$.

e) Typ-3 Grammatik $G = (\{S, A, B, C\}, \{a, b\}, P, S)$ mit
$$\begin{aligned} P = \{&S \to \epsilon \mid aA \\ &A \to aA \mid bB \\ &B \to bB \mid aC \\ &C \to bA \mid aB \mid b\}. \end{aligned}$$

f) $\Gamma = \{a, b\}$ und $L = \{w \in \Gamma^* \mid \text{es existieren } u_1, u_2, u_3 \in \Gamma^* \text{ mit } |u_2| = 3, \text{ sodass } w = u_1 a u_2 b u_3\}$.

Bei dem Vergleich wurde, dass Gewicht auf die in Kapitel 3.1 beschriebenen Annäherung an das exakte Gewicht bestimmt, indem wir für die ersten k Wortlängen den Anteil der akzeptierenden Wörter durch Verwendung eines der beiden zu vergleichenden Verfahren bestimmt haben. Wir haben uns für diese näherungsweise Bestimmung des Gewichts bei diesem Vergleich entschieden, da so die Laufzeitunterschiede zwischen den beiden Verfahren deutlich besser zum Ausdruck kommt, als wie wenn wir nur die Anteile für die Wortlängen bis zur Pumping-Zahl der Musterlösung bestimmen würden. In diesem Fall haben wir alle Wortlängen n betrachtet für die $\lambda^{n+1} \cdot \frac{1}{|\Gamma|^n}$ größer ist als der kleinste Wert, der durch einen herkömmlichen Datentyp dargestellt werden kann, da es sonst bei den weiteren Wortlängen zu Rundungsfehlern bei der Bestimmung des Gewichts der Wortlänge aufgrund des gewählten Datentyps kommen würde.
Außerdem wurde das Gewicht nicht auf den Abgaben, sondern auf den beiden disjunkten Teilmengen, die zusammen die symmetrische Differenz bilden, bestimmt. Warum dies für die Bewertung von studentischen Abgaben sinnvoller ist, diskutieren wir in Abschnitt 5.1. Des Weiteren wurden bei dem Vergleich ausschließlich Abgaben betrachtete, die nicht äquivalent zur Musterlösung sind, da bei äquivalenten Abgaben die symmetrische Differenz, durch die leere Sprache dargestellt wird und die Bestimmung des Gewichtes der symmetrischen Differenz daher immer zu dem Wert 0 führt. Daher wurde in diesem Fall dem Automaten gleich das Gewicht 0 zugeordnet, ohne das Gewicht mittels einer der beiden vorgestellten Verfahren zu bestimmen. Abschließend ist noch zu erwähnen, dass wir aus Gründen der Vergleichbarkeit für alle Automaten die Zerfallsrate $\lambda = 0{,}8$ verwendet haben. Die Ergebnisse sind für die verschiedenen Aufgabenstellungen so wie für die Gesamtheit in der Tabelle 6 angegeben. Außerdem stellt die Abbildung 7 die zugehörigen Funktionen der Verfahren da, die jedem Automaten eine Laufzeit zuordnet. Zur Übersichtlichkeit wurden die Automaten bei dieser Abbildung auf der $X-$Achse nach ihrer Laufzeit geordnet.

Aufgabe	Anzahl falscher Abgaben	Durchschnittliche Laufzeit Gleichungssystem	Durchschnittliche Laufzeit Matrizen	Durchschnittliche Laufzeit Kombination
a	29	0,085s	0,099s	0,081
b	92	0,322s	0,228s	0,227s
c	71	0,286s	0,201s	0,201s
d	29	0,067s	0,073s	0,062s
e	33	0,122s	0,094s	0,088s
f	47	0,791s	0,754s	0,755s
Gesamt	301	0,467s	0,338s	0,337s

Tabelle 6: Laufzeitenvergleich verschiedener Verfahren zur Bestimmung des Anteils der akzeptierenden Wörter einer Wortlänge gerundet auf 3 Nachkommastellen.

Wir erkennen, dass in der Gesamtheit die Berechnung mittels Matrizen die effizientere Variante darstellt. Allerdings erkennen wir auch, dass für einige Automaten und Aufgabenstellungen das Gleichungssystem die effizientere Variante darstellt. Dies ist der Fall, wenn die Schnittautomaten wenige Zustände besitzen.

Bei unserer kleinen Anzahl von Testfällen war die Grenze, bis zu der die Berechnung durch das Gleichungssystem die effizientere Variante darstellt 3 Zustände, sodass wir durch die Kombination der beiden Verfahren eine geringe Laufzeitverbesserung erreichen können. Dies bedeutet für Automaten die mehr als 3 Zustände besitzen, verwenden wir die Berechnung mittels Matrizen und für alle anderen Automaten die Berechnung mittels des Gleichungssystems.
Ein Erklärungsansatz dafür ist, dass wir für die Implementierung durch Matrizen das Python „NumPy“ Paket [7] verwenden konnten. Das Paket ermöglicht neben vielen weiteren Optimierungen, durch die „Vektorisierung“ der Daten eine deutlich schnellere Berechnung als durch eine üblichen Multiplikation von Matrizen durch for-Schleifen [12], sodass es für Automaten mit vielen Zustände die effizienter Variante darstellt.
Des Weiteren ist in Abbildung 7 zu erkennen, dass für ein Großteil der Abgaben die Laufzeit nur wenige Millisekunden beträgt, während die Berechnung für einige Automaten einige Sekunden in Anspruch nimmt. Dies ist damit zu erklären, dass für studentische Abgaben mit vielen Zuständen, die zusätzlich noch sehr weit von der Musterlösungssprache entfernt sind, Schnittautomaten entstehen, die exponentiell viele Zustände besitzen und damit eine höhere Rechenleistung zur Bestimmung des Gewichtes in Anspruch nehmen. Dies ist besonders bei Abgaben der Aufgabenstellung f zu beobachten.

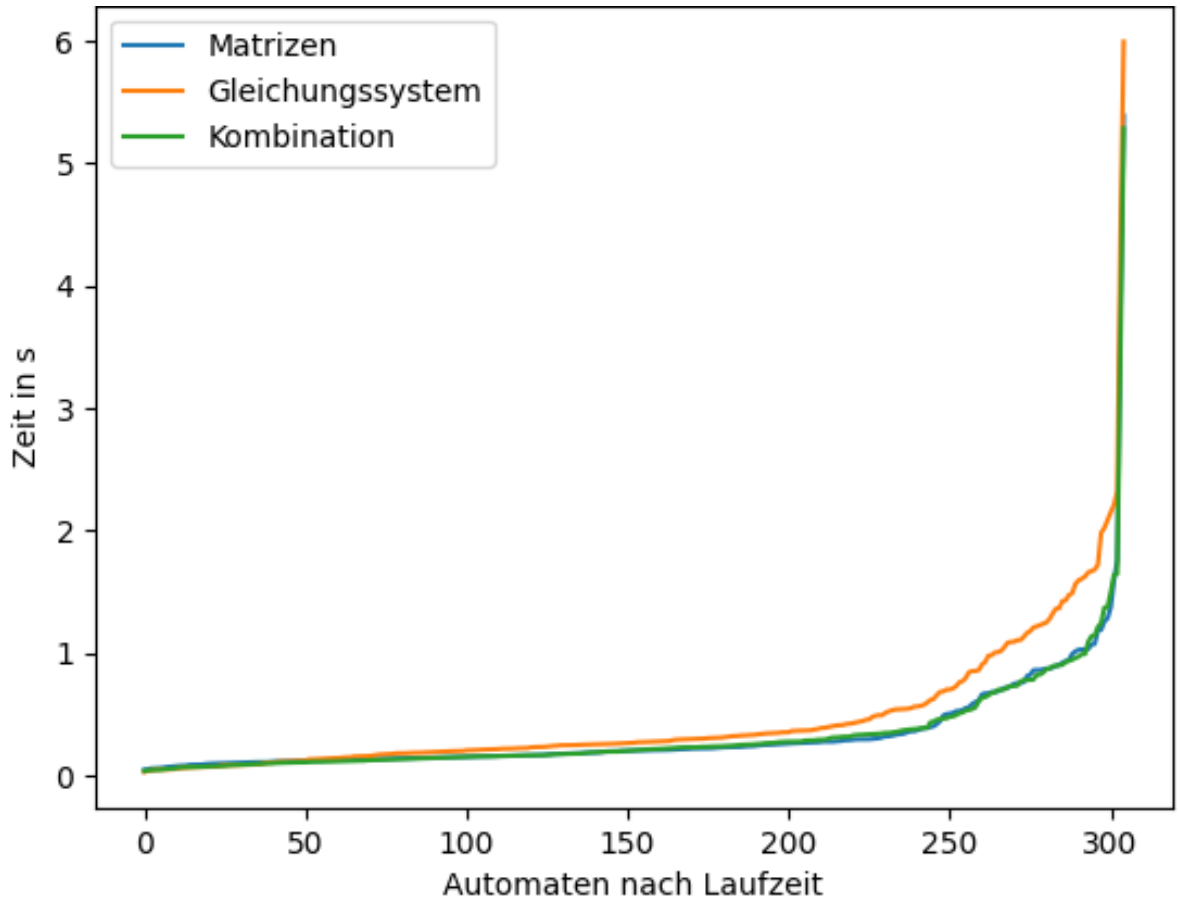

Abbildung 7: Laufzeitenvergleich verschiedener Verfahren zur Bestimmung des Anteils der akzeptierenden Wörter einer Wortlänge.

5 Praktische Anwendung der Gewichtung

Im Folgenden Kapitel möchten wir auf die praktische Anwendung der in Kapitel 3 vorgestellten Gewichtung einer (regulären) Sprache eingehen, um damit studentische Abgaben bewerten zu können (Abschnitt 5.1). Außerdem diskutieren wir darüber, was für unsere Anwendung geeignete Werte für η und λ darstellen (Abschnitt 5.2). Des Weiteren ordnen wir experimentell gewonnene Gewicht für studentische Abgaben ein (Abschnitt 5.3).

5.1 Differenz zweier Sprachen

In diesem Abschnitt widmen wir uns dem ursprünglichen Ziel dieser Arbeit, das Gewicht der Differenz zweier regulärer Sprachen zu bestimmen. Der triviale Ansatz, die Differenz zwischen den beiden Gewichten, die den Sprachen zugeordnet werden, zu bestimmen, funktioniert leider nicht, weil zwei Sprachen, dass selbe Gewicht haben können, obwohl sie sehr verschiedenen sind. So besitzen zum Beispiel über dem Alphabet $\Gamma = \{a, b\}$ die beiden Sprachen $L_1 = a^*$ und $L_2 = b^*$ das selbe Gewicht. Somit wäre die Differenz zwischen diesen beiden Sprachen 0, was nicht wirklich sinnvoll ist.
Daher definieren wir das Gewicht der Differenz $\triangle$ zwischen zwei regulären Sprachen L_1 und L_2, indem wir das Gewicht der symmetrischen Differenz (Abschnitt 2.2.4) dieser beiden Sprachen bestimmen.

$$\omega(L_1 \triangle L_2) = \omega((L_1 \cap \overline{L_2}) \cup (\overline{L_1} \cap L_2))$$

Damit können wir für DFAs das Gewicht der Differenz zwischen zwei regulären Sprachen bestimmen. Für andere Charakterisierung von regulären Sprachen als DFAs ist es leider unausweichlich, diese in eine äquivalenten DFA zu überführen um der Differenz zwischen zwei regulären Sprachen ein Gewicht zu geben.
Auch für kontextsensitive Sprachen und kontextfreie Sprachen lässt sich das Gewicht der Differenz zu mindesten näherungsweise bestimmen, indem wir alle Wortproblem bis zu einer bestimmen Wortlänge k auf beiden Automaten lösen, um so die Menge an Wörtern zu bestimmen, die nur in einer der beiden Sprachen liegen. Anschließend bestimmen wir das Gewicht auf dieser Menge, indem wir alle Wörter der Menge mit der jeweiligen Gewichtung aufsummieren, um das Gewicht zwischen zwei Sprachen zu erhalten. Auch wenn wir so theoretisch für kontextsensitive Sprachen und kontextfreie Sprachen eine Annäherung von λ^{k+1} an das exakte Gewicht bekommen, ist dieses Verfahren praktisch nicht wirklich sinnvoll, weil wir $2 \cdot \sum_{n=1}^{k} |\Gamma|^n$ Wortprobleme lösen müssen, die jeweils exponentielle beziehungsweise kubische Laufzeit benötigen.
Der Korrigierende hat mit der Gewichtung der symmetrischen Differenz ein gutes Hilfsmittel zur Hand, um damit die Abgabe von studentischen Abgaben bewerten zu können. Dieses Gewicht können wir noch aufteilen in den Teil des Gewichtes der symmetrischen Differenz, der aus den Wörtern besteht, die nur in der Sprache der Abgabe des Studierenden liegen und nicht in der Sprache der Musterlösung liegen, sowie den Teil, der die Wörter enthält, die nur in der Sprache der Musterlösung liegen. Damit bekommt der Korrigierende weitere Informationen über die Sprache der studentischen Abgabe. Damit lässt sich das Gewicht der symmetrischen Differenz auch darüber bestimmen, dass wir das Gewicht auf den beiden Schnittautomaten, die vereinigt die symmetrische Differenz ergeben, bestimmen und anschließend die Gewichte der beiden Schnittautomaten addieren. Dies hat neben der zusätzlichen Information auch den Vorteil, dass sich das Gewicht der symmetrischen Differenz auf diese Weise deutlich effektiver bezüglich der Laufzeit bestimmen lässt. Die Erklärung für die Effizienzsteigerung ist, dass wir durch das Bilden des Produktautomaten eine quadratische Anzahl an Zuständen erhalten, was zufolge hat, dass die Laufzeit zur Bestimmung des Gewichts auch quadratisch zunimmt. Im Gegensatz dazu nimmt die Laufzeit nur um den Faktor 2 zu, wenn wir das Verfahren auf beiden Schnittautomaten anwenden.
Abschließend zeigen wir, dass wir auch wenn wir das Gewicht auf den disjunkten Sprachen der beiden Schnittautomaten bilden und die Gewichte addieren, das Gewicht der symmetrischen Differenz erhalten. Dies liegt daran, dass die Gewichtsfunktion ω bezüglich zweier disjunkter Mengen A und B kommutiert.

$$\omega(A) + \omega(B) = \omega(A \cup B) \text{ wenn } A \cap B = \emptyset$$

Die Korrektheit der Gleichung ist damit zu begründen, dass kein Wort existiert, welches in beiden Mengen auftaucht. Außerdem liegen alle Wörter der Vereinigung $A \cup B$ in genau einer der beiden Teilmengen. Da wir in Kapitel 3 das Gewicht einer Sprache dadurch bestimmt haben, dass wir jedem Wort ein Gewicht zugeordnet haben und anschließend die Gewichte aller Wörter der Sprache aufsummiert haben, ist damit zu erkennen, dass kein Wort in A und B gewichtet wird und somit doppelt zum Gewicht von $A \cup B$ beiträgt. Außerdem existiert auch kein Wort, welches in A oder B gewichtet wird, aber nicht in $A \cup B$.

5.2 Wahl der Parameter eta und lambda

Im Folgenden möchten wir darauf eingehen, was für unsere Aufgabenstellung, die Bewertung von studentischen Abgaben, geeignete Werte für die in Kapitel 3 vorgestellten Parameter η und λ darstellen. Zur Erinnerung: η ist der Schwellenwert, bis zu dem alle Wörter mit einer geringeren Wortlänge konstant in die Gewichtung einer Sprache einfließen und λ ist die Zerfallsrate, die beschreibt, wie stark die Gewichtung für größere werdende Wortlängen abnimmt.
Dabei besitzen diese Parameter keinen unerheblichen Einfluss auf das resultierende Gewicht einer Sprache. So haben wir bereits in Beispiel 8, bei der Bestimmung des Gewichts der Sprache des Automaten F (Abbildung 6) gesehen, dass durch die Modifizierung von $\eta = 0$ auf den Wert $\eta = 3$ eine Veränderung des Gewichts der Sprache $L(F)$ von 0,12096 mit sich führte. Für verschiedene Werte von λ sind die Veränderungen der Gewichtung, vor allem durch die Einführung eines zu Beginn konstanten Teils, nicht ganz so drastisch, trotzdem hat auch die Wahl von λ einen großen Einfluss auf das resultierende Gewicht einer Sprache.
Doch die gute Nachricht ist das die Parameter Wahl, bei dem Vergleich mehrerer Automaten gegen die selbe Musterlösung, nicht so einen großen Einfluss spielt wie man zunächst annehmen würde. Dies liegt daran, dass es oftmals zu Teilmengenbeziehungen zwischen den Automaten kommt. Dazu betrachten wir beispielhaft als Musterlösung den Automaten B (Abbildung 2) und als zwei mögliche studentische Abgaben die Automaten G (Abbildung 8) und H (Abbildung 9). Hier erkennen wir das die Sprache von G eine echte Teilmenge der Sprache von B ist und die Sprache von H wiederum eine echte Teilmenge von G ist. Somit erhalten wir folgende Teilmengenbeziehungen: $L(B) \supseteq L(G) \supseteq L(H)$. Woraus folgt, dass für alle Parameter η und λ das Gewicht der Differenz von $L(H)$ zu $L(B)$ größer ist als das Gewicht der Differenz von $L(G)$ zu $L(B)$, vorausgesetzt die Parameter sind bei beiden Gewichtsbestimmungen gleich.

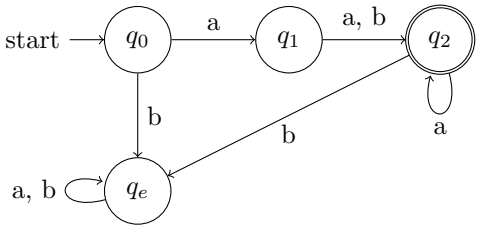

Abbildung 8: $L(G) = a(a + b)a^*$.

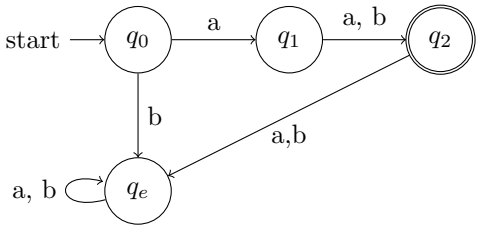

Abbildung 9: $L(H) = a(a + b)$.

Auch wenn es sich bei den Beispielen um fiktive Beispiele handelt, sind diese Teilmengen Beziehungen ebenfalls bei regulären studentischen Abgaben zu beobachten, sodass durch die Parameterwahl nur sichergestellt werden muss, dass Automaten, die keine Teilmengenbeziehung zueinander haben, allerdings einen ähnlich schwerwiegenden Fehler im Vergleich zur Musterlösung aufweisen, ein ähnliches Gewicht zugeordnet wird.
Hier hat sich für η durch experimentelle Untersuchungen herausgestellt, dass die Pumping-Zahl p der Musterlösung nach Meinung des Autors die besten Ergebnisse liefert. Dies ist darauf zurückzuführen, dass es ab der Länge p nur noch zu Wiederholungen kommt, die durch Zyklen auf dem DFA erzeugt werden. Damit sind bis zu diesem Zeitpunkt alle Fehler bereits mindesten einmal begannen worden. Außerdem wird die Sprache bereits durch die Wörter bis zu dieser Länge relativ gut charakterisiert. Dies bedeutet, dass wenn eine studentische Abgabe in diesem Teil keine oder nur geringfügige Fehler enthält, der Studierende den Ansatz zur Konstruktion der entsprechenden Sprache relativ gut verstanden hat.
Daraus folgt auch, dass wir λ so wählen sollten, dass diesem Teil bereits ein relativ großer Anteil des Gesamtgewichts der Sprache zugeordnet werden sollte. Hier hat sich herausgestellt, dass, wenn 50 bis 70 Prozent des Gewichtes auf den konstanten Teil entfallen, es nach Meinung des Autors zu den besten Ergebnissen führt. Allerdings kann der Anwender λ auch so wählen, dass der Anteil des konstanten Teils mehr oder weniger gewichtet wird, wenn er der Meinung ist, dass es bei einer Sprache besonders auf das Verstehen der Konzeption des konstanten Teils oder des exponentiell zerfallenden Teils ankommt.
In Tabelle 7 ist für einige λ und η der Anteil des konstanten Teils zur Übersicht dargestellt. Hier ist zu erkennen, dass für kleine Zerfallsraten wie 0,5 bereits fast das gesamte Gewicht auf die ersten Wortlängen entfällt und längere Wortlängen nur verschwindend gering in die Gewichtung einfließen. Da bei allen Zerfallsraten kleiner 0,7 mehr als die Hälfte des Gewichts auf das leere Wort und Wörter der Länge 1 entfallen, ist es nicht sinnvoll λ auf einen Wert kleiner als 0,7 zu setzen um brauchbare Ergebnisse zu bekommen. Auch ist es nur dann sinnvoll, die Zerfallsrate auf einen Wert, der gegen 1 geht, zu setzen, wenn

λ \ η	1	2	3	4	5	10	15	25	50	75	100	250	500
0,5	0,75	0,875	0,938	0,969	0,984	1	1	1	1	1	1	1	1
0,6	0,64	0,784	0,87	0,922	0,953	0,996	1	1	1	1	1	1	1
0,7	0,51	0,657	0,76	0,832	0,882	0,98	0,997	1	1	1	1	1	1
0,8	0,36	0,488	0,59	0,672	0,738	0,914	0,972	0,997	1	1	1	1	1
0,9	0,19	0,271	0,344	0,41	0,469	0,686	0,8146	0,935	0,995	1	1	1	1
0,95	0,098	0,143	0,185	0,226	0,265	0,431	0,56	0,36	0,927	0,98	0,995	1	1
0,98	0,04	0,059	0,078	0,096	0,114	0,199	0,276	0,409	0,643	0,785	0,87	0,994	1
0,99	0,02	0,03	0,039	0,049	0,059	0,105	0,149	0,23	0,401	0,534	0,638	0,92	0,993

Tabelle 7: Anteil des Konstanten Teils x für verschiedene λ und η gerundet auf 3 Nachkommastellen.

der Automat der Musterlösung eine sehr hohe Pumping-Zahl besitzt. Dies ist allerdings für studentische Aufgabenstellungen eher unüblich und didaktisch wahrscheinlich nur begrenzt zielführend. Stattdessen ist es sinnvoller den Schwellenwert η durch Bestimmung der Pumping-Zahl der Musterlösung zu bestimmen und sich dann die Gewichtung des konstanten Teils zu wählen um daraus die Zerfallsrate zu bestimmen. Wie wir den Anteil des konstanten Teils x berechnen können, haben wir bereits in Kapitel 3.3 vorgestellt. Durch umstellen dortiger Formel können wir, wenn wir uns für einen festen Anteil x entschieden haben λ bestimmen. Die Formel lautet dann folgendermaßen.

$$\lambda = \sqrt[\eta+1]{1-x}$$

5.3 Die aus der Gewichtung resultierende Ordnung

Das wir für jede studentische Abgabe die symmetrische Differenz bilden können und für die symmetrische Differenz in der Lage sind, anschließend ein Gewicht zu bestimmen, bringt uns in die Lage, dass wir jeder Abgabe ein Gewicht zuordnen können, um so in der Lage zu sein, alle Abgaben miteinander vergleichen zu können. Damit können wir auf der Menge der studentischen Abgaben eine Ordnung erstellt, die die Abgaben nach der Schwere der gemachten Fehler ordnet. Dadurch sind wir in der Lage, Abgaben miteinander zu vergleichen, für die vorher keine Teilmengenbeziehung existiert hat immer vorausgesetzt, wir verwenden bei allen Gewichts Bestimmungen dieselben Parameter η und λ. Daher ist es empfehlenswert, wenn wir η als die Pumping-Zahl der Musterlösung wählen.
Durch Erfahrungen des Autors als Korrektor ist dieser der Meinung, dass durch das vorgestellte Verfahren eine brauchbare Ordnung auf der Menge der Abgaben entsteht, die dem Korrigierende es deutlich vereinfacht, eine angemessene Bepunktung für die Abgaben vornehmen zu können. Dies ist damit zu erklären, dass sich beobachten lässt, dass schwerwiegende Fehler zu einem größeren Gewicht der symmetrischen Differenz führen, als dies kleiner Fehler tun. Allerdings sind wir nicht in der Lage, automatisiert jedem Gewicht eine mögliche Punktzahl zuordnen zu können, sodass wir kein voll automatisiertes Bewertungstool besitzen. Dies liegt daran, dass für unterschiedliche Aufgabenstellungen stark abweichende Gewichte als Ergebnisse herauskommen. So kann es sein, dass bei einer Aufgabenstellung das Gewicht 0,1 ein Gewicht darstellt, welches relativ nah an der Musterlösung liegt und bei einer anderen Aufgabenstellung das Gewicht 0,1 einem Automaten zugeordnet wird, der relativ weit weg von der Musterlösung liegt, sodass diesen beiden Automaten niemals dieselbe Punktzahl zugeordnet werden sollte.
Daher muss der Anwender für jede konkrete Aufgabenstellung eine neue Bewertungsskala definieren, die jedem Gewicht eine konkrete Punktzahl zuordnet. Somit sind wir leider nicht in der Lage studentische Abgaben voll automatisiert bewerten zu können. Allerdings erleichtert uns das vorgestellte Verfahren durch die resultierende Ordnung auf den Abgaben die Korrekturarbeit deutlich.

6 Fazit und Ausblick

Zum Abschluss möchten wir noch ein Fazit dieser Arbeit ziehen, in dem wir darauf eingehen, was wir in dieser Arbeit erreichen konnten (Abschnitt 6.1) und warum wir das ursprüngliche Ziel nicht im vollen Umfang erreicht haben, aber trotzdem ein gutes Hilfsmittel zur Korrektur erarbeitet konnten (Abschnitt 6.2). Des Weiteren gehen wir auf einige Erweiterungen des vorgestellen Verfahren und seiner möglichen Anwendung ein (Abschnitt 6.3).

6.1 Resultate

In dieser Arbeit haben wir eine Gewichtsfunktion ω vorgestellt, die jeder Sprache ein Gewicht zuordnet, welches sich für reguläre Sprachen, die durch einen DFA charakterisiert werden, explizit lösen lässt. Auch für kontextsensitive und kontextfreie Sprachen haben wir erwähnt, wie wir ein Gewicht der Sprache näherungsweise bestimmen können.
Das Gewicht einer Sprache hängt dabei davon ab, wie viele Wörter für die einzelnen Wortlängen in der gegebenen Sprache liegen. Die Wortlängen werden dabei mit steigender Länge immer weniger gewichtet. Dies ist allerdings für die initialen Wortlängen nur bedingt sinnvoll, sodass wir gezeigt haben, wie wir die Gewichte dieser Wortlängen so umverteilen können, dass alle Wörter bis zu einem bestimmten Schwellenwert konstant gewichtet werden. Durch die Umverteilung müssen wir die explizite Lösung für DFAs noch anpassen, wodurch wir den Anteil der akzeptierenden Wörter für eine bestimmte Wortlänge benötigen. Um den Anteil effizient bestimmen zu können, geben wir sowohl ein Verfahren mithilfe eines Gleichungssystems wie auch ein Verfahren mittels Matrizen an, wobei wir durch experimentell gewonnene Laufzeiten heraus arbeiten konnten, dass in den meisten Fällen die Matrizen, dass effektivere Verfahren bezüglich der Laufzeit darstellen.
Des Weiteren haben wir gezeigt, wie sich die Gewichtung einer Sprache dazu einsetzen lässt, um studentische Abgaben gegen eine Musterlösung zu bewerten. Dafür bestimmen wir das Gewicht auf der symmetrischen Differenz der beiden Sprachen, sodass wir für mehrere studentische Abgaben eine nach der Meinung des Autors vielversprechende Ordnung erhalten, die angibt, wie stark die abgegebene Sprache von der Sprache der Musterlösung abweicht.

6.2 Fazit

Durch das in dieser Arbeit vorgestellten Verfahren sind wir in der Lage, studentische Abgaben gegen eine Musterlösung zu vergleichen und der Differenz zwischen den beiden Sprache ein Gewicht zuzuordnen, sodass der Korrektur ein gutes Hilfsmittel bei der Bewertung von studentischen Aufgaben zu Hand hat, was diese Aufgabe deutlich vereinfacht. Allerdings ist es uns aufgrund des unter Abschnitt 5.3 beschriebenen Problems nicht gelungen, eine voll automatisierte Bewertungsmethode zu erstellen. Ein weiteres Problem dieses Verfahrens besteht darin, dass bereits eine falsch eingezeichnete Kante des Automaten zu einem relativ hohen Gewicht führen kann, während eine andere falsch eingezeichnete Kante vergleichsweise zu nur einem sehr geringen Gewicht führen kann. Wie stark eine falsch eingezeichnete Kante sich im Gesamtgewicht der Sprache niederschlägt, hängt davon ab, wo sie im Automaten vorkommt. So tragen Kanten, die bereits bei den ersten Schritten einer Rechnung benutzt werden können, zu einem höheren Gewicht bei als Kanten, die erst bei späteren Schritten benutzt werden können.
Das vorgestellte Gewicht einer Sprache lässt sich nicht nur für reguläre Sprachen einsetzen, auch für allgemeine Grammatiken, kontextsensitive Sprachen und kontextfreie Sprachen lässt sich das Gewicht einer Sprache auf die vorgestellte Art definieren, wobei wir für kontextsensitive und kontextfreie Sprachen nur eine Annäherung an das exakte Gewicht bekommen, die Genauigkeit hängt dabei davon ab, wie viele Wortlängen wir betrachten. Allerdings ist dies in der Praxis aufgrund der hohen Laufzeit Komplexität nur begrenzt nutzbar. Für allgemeine Grammatiken hingegen ist auch die näherungsweise Bestimmung dieser Arbeit nicht empfehlenswert, da das Wortproblem für diese Sprachen unentscheidbar ist.

6.3 Ausblick

Eine mögliche Erweiterung der Gewichtung wäre zu betrachten, ob die Laufzeit der näherungsweisen Bestimmung für kontextsensitive Sprachen und kontextfreie Sprachen deutlich zu verbessern wäre, sodass man für diese Sprachklassen eine in der Praxis vertretbare Laufzeit erreicht oder im optimalen Fall eine explizite Lösungsmethode zu finden. Darüber hinaus wäre es wünschenswert, auch für allgemeine Grammatiken das Gewicht bestimmen zu können, damit wir die Gewichtung von studentischen Abgaben als Hilfsmittel bei der Bewertung auch auf für diese Sprachklassen verwenden können.

Darüber hinaus wäre es beachtenswert, ob es eventuell doch möglich ist, auch für andere Charakterisierungen von regulären Sprachen als DFAs das Gewicht direkt bestimmen zu können ohne sie in einen äquivalenten DFA umwandeln zu müssen.
Sehr interessant wäre auch, ob man durch einen größeren Datensatz an studentischen Abgaben und einem damit verbunden größeren Wissen über die resultierende Gewichte es doch schafft, in die zu Lage kommen, eine voll automatisierte Bewertung vornehmen zu können. Hierbei wäre auch vorstellbar, dass die Gewichtung der symmetrischen Differenz durch weitere Faktoren ergänzt wird, sodass durch Kombination der Einzelergebnisse eine gute und faire, automatisierte Bewertung entsteht. Dabei wären neben den Faktoren des „Grading Algorithmus“[2] auch andere Faktoren denkbar. Mögliche Faktoren wäre dabei, die Abgaben mithilfe von Neuronalen Netzen zu bewerten[10] oder zu bestimmen, ob es sich bei der Abgabe um den kleinsten Automaten bezüglich der Zustände handelt, der die Sprache beschreibt.

Literaturverzeichnis

[1] Josh Alman und Virginia Vassilevska Williams. „A Refined Laser Method and Faster Matrix Multiplication“. In: *CoRR* abs/2010.05846 (2020). DOI: 10.1137/1.9781611976465.32. arXiv: 2010.05846.

[2] Rajeev Alur, Loris D'Antoni, Sumit Gulwani, Dileep Kini und Mahesh Viswanathan. „Automated Grading of DFA Constructions“. In: Hrsg. von Francesca Rossi. IJCAI/AAAI, 2013, S. 1976–1982.

[3] George E. Andrews. „The Geometric Series in Calculus“. In: *The American Mathematical Monthly* 105.1 (1998), S. 36–40. ISSN: 00029890, 19300972. DOI: 10.1080/00029890.1998.12004846.

[4] Loris D'Antoni, Martin Helfrich, Jan Kretınský, Emanuel Ramneantu und Maximilian Weininger. „Automata Tutor v3“. In: Hrsg. von Shuvendu K. Lahiri und Chao Wang. Bd. 12225. Lecture Notes in Computer Science. Springer, 2020, S. 3–14. DOI: 10.1007/978-3-030-53291-8_1.

[5] Oliver Friedmann und Martin Lange. „Ramsey-Based Analysis of Parity Automata“. In: Hrsg. von Cormac Flanagan und Barbara König. Bd. 7214. Lecture Notes in Computer Science. Springer, 2012, S. 64–78. DOI: 10.1007/978-3-642-28756-5_6.

[6] François Le Gall. „Powers of tensors and fast matrix multiplication.“ In: Hrsg. von Katsusuke Nabeshima, Kosaku Nagasaka, Franz Winkler und Ágnes Szántó. ACM, 2014, S. 296–303. ISBN: 978-1-4503-2501-1. DOI: 10.1145/2608628.2608664.

[7] Charles R. Harris, K. Jarrod Millman, Stéfan van der Walt, Ralf Gommers, Pauli Virtanen, David Cournapeau, Eric Wieser, Julian Taylor, Sebastian Berg, Nathaniel J. Smith, Robert Kern, Matti Picus, Stephan Hoyer, Marten H. van Kerkwijk, Matthew Brett, Allan Haldane, Jaime Fernández del Rıéo, Mark Wiebe, Pearu Peterson, Pierre Gérard-Marchant, Kevin Sheppard, Tyler Reddy, Warren Weckesser, Hameer Abbasi, Christoph Gohlke und Travis E. Oliphant. „Array programming with NumPy“. In: *Nat.* 585 (2020), S. 357–362. DOI: 10.1038/s41586-020-2649-2.

[8] Christian Karpfinger und Hellmuth Stachel. *Lineare Algebra*. Springer Spektrum, Berlin, Heidelberg, 2020. ISBN: 9783662613405. DOI: 10.1007/978-3-662-61340-5.

[9] U. Schöning. *Theoretische Informatik - kurz gefasst*. Spektrum Akademischer Verlag, 2008. ISBN: 9783827418241.

[10] Georg Siebert. *Bachelorarbeit: Auf dem Weg zum kunstlichen Lehrassistenten: Das Lernen von Bewertungsschemata fur endliche Automaten mit GNNs*. Universität Kassel, 2021.

[11] Larry J. Stockmeyer und Albert R. Meyer. „Word Problems Requiring Exponential Time: Preliminary Report“. In: Hrsg. von Alfred V. Aho, Allan Borodin, Robert L. Constable, Robert W. Floyd, Michael A. Harrison, Richard M. Karp und H. Raymond Strong. ACM, 1973, S. 1–9. DOI: 10.1145/800125.804029.

[12] Stéfan van der Walt, S. Chris Colbert und Gaël Varoquaux. „The NumPy Array: A Structure for Efficient Numerical Computation“. In: *Comput. Sci. Eng.* 13.2 (2011), S. 22–30. DOI: 10.1109/MCSE.2011.37.

[13] Martin De Wulf, Laurent Doyen, Thomas A. Henzinger und Jean-François Raskin. „Antichains: A New Algorithm for Checking Universality of Finite Automata“. In: Hrsg. von Thomas Ball und Robert B. Jones. Bd. 4144. Lecture Notes in Computer Science. Springer, 2006, S. 17–30. DOI: 10.1007/11817963_5.