Duale Hochschule Baden-Württemberg Mannheim

Seminararbeit

Endliche Zustandautomaten

Studiengang Wirtschaftsinformatik

Studienrichtung Data Science

Verfasser(in):	Aref Hasan, Nik Yakovlev, Katharina Thiel
Matrikelnummer:	1335610, 4144212, 5910231
Kurs:	WWI-21-DSA
Studiengangsleiter:	Prof. Dr.-Ing. habil. Dennis Pfisterer
Modul:	Integrationsseminar
Dozent:	Prof. Dr. Maximilian Scherer
Eingereicht:	26.02.2024

Inhaltsverzeichnis

Abkürzungsverzeichnis

EZA	endlicher Zustandsautomaten
VFX	Visual Effects
CGI	Computer Generated Imagery
FSM	Finite State Machine
KI	Künstlichen Intelligenz
DEAs	Deterministische endliche Automaten
NEAs	Nichtdeterministische endliche Automaten
XFAs	Erweiterten finiten Automaten

Kurzfassung

Diese Seminararbeit erkundet die wesentlichen Aspekte endlicher Automaten (Finite State Machines, FSMs) und deren Anwendung in der theoretischen Informatik sowie in praktischen Bereichen wie KI, Robotik und Netzwerksicherheit. Im Mittelpunkt stehen deterministische endliche Automaten (DEAs), nichtdeterministische endliche Automaten (NEAs) und erweiterte finite Automaten (XFAs), einschließlich ihrer Anwendung in der Verarbeitung von Regular Expressions (Regex). Ein besonderes Augenmerk liegt auf der Visualisierung dieser Konzepte mittels der Animationsbibliothek Manim, die eine intuitive Darstellung ermöglicht. Die Arbeit vergleicht endliche Automaten auch mit anderen Maschinenmodellen wie Kellerautomaten und Turingmaschinen, um ein breiteres Verständnis der Computationsmodelle zu bieten. Ziel ist es, die Relevanz endlicher Automaten in der Informatik zu unterstreichen und einen praktischen Ansatz für deren Implementierung und Nutzung zu bieten, was sowohl für Studierende als auch für Fachleute von Interesse ist.

1 Einleitung

Die Seminararbeit „Grundlagen der Endlichen Automaten“ bietet einen umfassenden Einblick in die Theorie und Praxis endlicher Automaten, ein zentrales Konzept der theoretischen Informatik, das weitreichende Anwendungen in der Computerwissenschaft, Künstlichen Intelligenz (KI), Robotik und darüber hinaus findet. Neben der detaillierten Untersuchung deterministischer endlicher Automaten (DEAs) und nichtdeterministischer endlicher Automaten (NEAs), ihrer Typen und Anwendungsbereiche, von der Sprachverarbeitung bis hin zur Netzwerksicherheit, erweitert diese Arbeit ihren Fokus um die Darstellung erweiterter Konzepte wie erweiterter finiter Automaten (XFAs) und deren Einsatz in der Verarbeitung von Regular Expressions (Regex).

Ein besonderer Schwerpunkt der Arbeit liegt auf den Grundlagen der Visualisierung dieser Automaten. Die Visualisierungstechniken und -werkzeuge, insbesondere die Nutzung der Animationsbibliothek Manim, werden eingehend erläutert. Manim, eine leistungsstarke Bibliothek für die Erstellung präziser und didaktisch wertvoller mathematischer Animationen, ermöglicht es, die abstrakten Konzepte endlicher Automaten und deren Dynamik in einer visuell ansprechenden und intuitiv verständlichen Form darzustellen. Diese Visualisierungskomponente ist entscheidend für ein tieferes Verständnis der Materie und erleichtert die Analyse und das Design von Automaten erheblich.

Darüber hinaus wird in der Seminararbeit ein Vergleich mit anderen theoretischen Maschinenmodellen wie dem Kellerautomat und der Turingmaschine gezogen, um ein breiteres Verständnis für die Vielfalt und die Grenzen computergestützter Informationsverarbeitung zu schaffen. Die Einbeziehung der Visualisierungsgrundlagen und der praktischen Anwendung von Manim unterstreicht die Bedeutung effektiver Darstellungsmethoden in der theoretischen Informatik und bietet einen praktischen Leitfaden für die Implementierung und Nutzung dieser Konzepte in realen Anwendungsszenarien.

In einer Zeit, in der Algorithmen zunehmend komplexere Probleme lösen müssen, betont diese Seminararbeit die anhaltende Relevanz endlicher Automaten als fundamentales Werkzeug in der theoretischen Informatik und Technologieentwicklung. Die Arbeit zielt darauf ab, ein umfassendes Verständnis der Theorie und Praxis endlicher Automaten zu fördern und zeigt auf, wie grundlegende Konzepte und Visualisierungstechniken in der Praxis angewendet werden können, um effiziente und effektive Lösungen zu entwerfen.

2 Visualisierungstheorie

Die Theorie der Visualisierung stellt eine fundamentale Basis dar, um zu verstehen, wie Menschen visuelle Informationen aufnehmen und verarbeiten. Mayer hebt in seiner kognitiven Theorie des multimedialen Lernens hervor, dass die Kombination von visuellen und verbalen Informationen das Lernen signifikant verbessern kann, indem sie die kognitive Verarbeitung unterstützt und so das Verstehen komplexer Konzepte erleichtert (Vgl. Mayer, 2009). Diese Theorie verdeutlicht die Notwendigkeit einer ausgewogenen Präsentation von Text und Bildmaterial, um die Lernprozesse zu optimieren.

Visuelle Hilfsmittel spielen eine entscheidende Rolle in der Lehre und Forschung, indem sie dazu beitragen, abstrakte Konzepte zu veranschaulichen und das Behalten von Informationen zu fördern. Clark & Lyons zeigen auf, dass insbesondere in mathematischen und technischen Bereichen, wo Konzepte oft abstrakt und komplex sind, visuelle Darstellungen essentiell für das Verständnis und die praktische Anwendung dieser Konzepte sein können (Vgl. Clark et al., 2004).

Die Darstellung von endlichen Zustandsautomaten bringt spezifische Herausforderungen mit sich. Diese beinhalten die angemessene Repräsentation von Zuständen, Übergängen und Eingaben, die sowohl präzise als auch intuitiv verständlich für Lernende sein müssen. Naps et al. weisen darauf hin, dass eine wirksame Visualisierung von algorithmenbasierten Konzepten, wie sie bei endlichen Zustandsautomaten vorkommen, eine durchdachte Planung und Umsetzung erfordert, um Missverständnisse zu minimieren und ein tieferes Verständnis der Konzepte zu fördern (Vgl. Naps et al., 2002).

Ein weiterer wichtiger Aspekt ist die Anpassung der Visualisierungsansätze an verschiedene Lernstile. Fleming & Mills unterstreichen die Bedeutung, Visualisierungstools so zu gestalten, dass sie unterschiedliche Lernpräferenzen berücksichtigen. Dies ist insbesondere bei der Visualisierung von endlichen Zustandsautomaten relevant, da nicht alle Lernenden durch dieselbe Darstellungsform optimal erreicht werden (Vgl. Fleming & Mills, 1992).

2.1 Grundlagen der visuellen Technologien

Die Entwicklung und Anwendung visueller Technologien haben einen tiefgreifenden Einfluss auf die Art und Weise, wie wir Informationen verarbeiten und verstehen. Im Folgenden wird ein erweiterter Überblick über die zentralen Konzepte und Technologien gegeben.

2.1.1 Visuelle Perzeption

Die visuelle Perzeption bildet die Grundlage für das Design und die Umsetzung effektiver Visualisierungen. Das Verständnis, wie das menschliche Gehirn visuelle Reize verarbeitet, ist entscheidend, um Daten und Konzepte visuell so darzustellen, dass sie schnell und effektiv erfasst werden können. Die Forschung in diesem Bereich untersucht, wie Farben, Formen, Muster und Bewegung die Aufmerksamkeit und das Verständnis beeinflussen können. Goldstein (2010) bietet einen tiefen Einblick in die kognitiven Prozesse, die der visuellen Wahrnehmung zugrunde liegen, und unterstreicht die Bedeutung eines designorientierten Ansatzes, der auf diesen Erkenntnissen aufbaut (Vgl. Goldstein, 2014).

2.1.2 Computergrafik

Computergrafik befasst sich mit der Erzeugung und Manipulation von Bildern und Grafiken durch Computer. Dieses Feld hat die Visualisierung komplexer Datenstrukturen und Modelle revolutioniert, indem es Tools und Techniken bereitstellt, um realistische und abstrakte visuelle Darstellungen zu erstellen. Die algorithmischen Grundlagen der Computergrafik, wie sie von Hughes et al. (2014) beschrieben werden, umfassen Rendering-Verfahren, Modellierungstechniken und die Simulation von Lichteffekten, die alle dazu beitragen, verständliche und ästhetisch ansprechende Visualisierungen zu erzeugen (Vgl. Foley, 1996).

2.1.3 Animation und Motion Graphics

Animation und Motion Graphics beziehen sich auf die Technik, statischen Bildern Bewegung zu verleihen, um Narrationen zu erstellen oder komplexe Ideen zu vermitteln. Während Animation oft in der Unterhaltungsindustrie verwendet wird, hat sie auch in der wissenschaftlichen Visualisierung eine wichtige Rolle gespielt, insbesondere bei der Darstellung zeitabhängiger Prozesse und dynamischer Systeme. Thomas und Johnston (1981) haben mit ihren Prinzipien der Animation einen Meilenstein gesetzt, der auch heute noch in der Erstellung von Lehrmaterialien und Erklärungsvideos Anwendung findet (Vgl. Thomas & Johnston, 1981).

2.1.4 Visual Effects und Computer Generated Imagery

Visual Effects (VFX) und Computer Generated Imagery (CGI) sind Techniken, die oft in der Filmindustrie eingesetzt werden, um realistische oder imaginäre Szenen zu schaffen, die mit traditionellen Filmtechniken schwer oder unmöglich zu realisieren wären. Diese Technologien

haben jedoch auch Anwendungen in der wissenschaftlichen Visualisierung gefunden, insbesondere bei der Darstellung von Phänomenen, die nicht direkt beobachtet oder leicht veranschaulicht werden können. Wright diskutiert die Rolle von CGI in der Schaffung komplexer visueller Inhalte und betont dessen Potenzial, das Verständnis von wissenschaftlichen Konzepten durch hochdetaillierte und präzise Visualisierungen zu verbessern (Vgl. Wright, 2006).

2.2 Visualisierungstechniken

Die Visualisierung endlicher Zustandsautomaten (EZA) ist ein zentrales Thema in der Informatik und der computergestützten Bildung, das darauf abzielt, das Verständnis dieser komplexen Strukturen zu erleichtern. Die Darstellung von Zuständen, Übergängen und der Dynamik von EZAs kann durch diverse Techniken erfolgen, deren Einsatz von den spezifischen Lernzielen und dem jeweiligen Kontext abhängig ist.

2.2.1 Diagramme und graphische Darstellungen

Diagramme stellen traditionelle und effektive Mittel dar, um die Struktur und Funktionsweise von EZAs zu visualisieren. Zustandsdiagramme, auch bekannt als Zustandsübergangsdiagramme, bieten eine visuelle Repräsentation der verschiedenen Zustände eines Automaten sowie der Übergänge zwischen diesen Zuständen auf Basis von Eingabesymbolen. Hopcroft, Motwani und Ullman betonen die Bedeutung solcher Diagramme für das grundlegende Verständnis der Automatentheorie und Algorithmen (Vgl. Hopcroft et al., 2001).

2.2.2 Interaktive Modelle

Durch den technologischen Fortschritt haben sich interaktive Modelle als innovative Methode zur Visualisierung und Exploration von EZAs etabliert. Plattformen wie JFLAP ermöglichen es Nutzern, durch direkte Manipulation von Zuständen und Übergängen die Funktionsweise von Automaten experimentell zu erkunden. Rodger und Finley illustrieren, wie solche interaktiven Werkzeuge nicht nur das Verständnis fördern, sondern auch die Entdeckung komplexer Konzepte durch eigenständiges Lernen unterstützen (Vgl. Rodger & Finley, 2006).

2.2.3 Simulationen und Animationen

Simulationen und Animationen bieten erweiterte Möglichkeiten zur Visualisierung, indem sie die dynamischen Aspekte von EZAs in Echtzeit darstellen. Softwares wie Manim, die speziell

für die Erstellung mathematischer Visualisierungen konzipiert wurden, erlauben es, abstrakte Konzepte in einer anschaulichen und greifbaren Weise zu präsentieren. Die Studie von Leiserson et al. über den Einsatz von Animationen in der Informatiklehre unterstreicht die Effektivität dieses Ansatzes, um bei Studierenden ein tieferes Verständnis und eine bessere Behaltensleistung zu erzielen (Vgl. *SIGCSE '20: Proceedings of the 51st ACM Technical Symposium on Computer Science Education*, 2020).

2.2.4 Herausforderungen und Lösungsansätze

Trotz der vielfältigen Visualisierungsmöglichkeiten stellen die Auswahl und Implementierung geeigneter Techniken Herausforderungen dar. Aspekte wie die Komplexität der darzustellenden Informationen, die Notwendigkeit der Anpassung an unterschiedliche Lernstile und die technischen Anforderungen müssen berücksichtigt werden. Eine effektive Visualisierung erfordert demnach nicht nur technisches Know-how, sondern auch ein profundes Verständnis der didaktischen Ziele und der Zielgruppe.

3 Erklärungsvideos in der Bildung

Erklärungsvideos spielen eine zunehmend wichtige Rolle im Bildungsbereich. Sie bieten eine attraktive Alternative zu traditionellen Lehrmethoden, indem sie komplexe Konzepte auf einfache und verständliche Weise vermitteln. Die Forschung zeigt, dass Erklärungsvideos die Aufmerksamkeit steigern, das Engagement fördern und zum langfristigen Behalten von Informationen beitragen können.

3.1 Bedeutung in der Bildung

Steigerung der Aufmerksamkeit

Erklärungsvideos können durch visuelle und auditive Reize die Aufmerksamkeit der Lernenden besser fesseln als traditionelle Texte oder Vorlesungen. Eine Studie fand heraus, dass Videos das Potenzial haben, Lernende effektiv zu engagieren und deren Verständnis zu verbessern (Krämer & Böhrs, 2017).

Förderung des Engagements

Der Einsatz von Erklärungsvideos in Lehrplänen kann die Motivation der Studierenden erhöhen und ein aktiveres Lernumfeld schaffen. Sie ermöglichen eine interaktive Erfahrung, die über die passive Aufnahme von Informationen hinausgeht (Ivanova, 2017).

Verbesserung der Langzeitretention

Das regelmäßige Ansehen von Erklärungsvideos kann die Erinnerungsfähigkeit und das Verständnis von Konzepten über längere Zeiträume verbessern. Forschungsergebnisse zeigen, dass visuelle und verbale Hinweise, die in Videos enthalten sind, die Merkfähigkeit der Lernenden steigern können (Cohen et al., 2018).

3.2 Gestaltungselemente effektiver Erklärungsvideos

- **Storytelling:** Eine starke narrative Struktur hilft, die Zuschauer zu fesseln und komplexe Informationen leichter verdaulich zu machen.
- **Klare visuelle Darstellungen:** Einfache und intuitive Grafiken oder Animationen erleichtern das Verständnis und die Erinnerung an die dargestellten Informationen.
- **Verständliche Sprache:** Eine einfache und klare Sprache macht die Inhalte zugänglicher für ein breiteres Publikum.

- **Interaktionsmöglichkeiten:** Durch Einbindung von Quizzen oder interaktiven Elementen können Lernende das Gesehene sofort anwenden und festigen.

3.3 Best Practices für die Erstellung von Erklärungsvideos

1. **Skriptentwicklung:** Ein gut durchdachtes Skript ist die Grundlage für ein erfolgreiches Video. Es sollte die Kernbotschaften klar definieren und eine logische Struktur aufweisen.
2. **Storyboarding:** Die visuelle Planung des Videos durch Storyboards hilft, die Erzählung und die visuellen Elemente effektiv aufeinander abzustimmen.
3. **Auswahl der richtigen Visualisierungssoftware:** Die Wahl der Software sollte auf den spezifischen Bedürfnissen des Projekts basieren, einschließlich der Komplexität der zu visualisierenden Konzepte.
4. **Techniken zur Aufrechterhaltung der Zuschauerbindung:** Methoden wie das Einbauen von Fragen, interaktiven Elementen oder Überraschungsmomenten halten die Aufmerksamkeit und fördern ein aktives Mitdenken (Brame, 2016).

4 Manim als Werkzeug für Erklärungsvideos

4.1 Über Manim

Manim, kurz für Mathematical Animation Library, ist ein leistungsstarkes Open-Source Visualisierungswerkzeug, das ursprünglich von Grant Sanderson, dem Schöpfer des YouTube-Kanals 3Blue1Brown, entwickelt wurde. Sandersons Ziel war es, ein Werkzeug zu schaffen, mit dem komplexe mathematische Konzepte auf intuitive und visuell ansprechende Weise dargestellt werden können. Seit seiner Einführung hat sich Manim zu einem von der Community getriebenen Projekt entwickelt, das nicht nur in der Mathematik, sondern auch in technischen und wissenschaftlichen Disziplinen breite Anwendung findet (Manim Community, 2024).

Die Hauptfunktionen von Manim umfassen die Erstellung präziser und hochwertiger Animationen, die Nutzung von Python zur Steuerung der Visualisierungsdetails und die Fähigkeit, mathematische Ausdrücke direkt in die Animationen zu integrieren. Diese Funktionen machen Manim besonders geeignet für die Visualisierung mathematischer und technischer Konzepte, da es die Darstellung von Formeln, Graphen, geometrischen Figuren und vielem mehr mit hoher Präzision ermöglicht (Manim Community, 2024).

Ein wesentlicher Vorteil von Manim ist seine Flexibilität. Benutzer können mit Manim sowohl einfache Diagramme als auch komplexe Animationen erstellen, die für Erklärungsvideos benötigt werden. Da es von einer aktiven Community unterstützt wird, stehen ständig neue Erweiterungen und Verbesserungen zur Verfügung, die die Funktionalität von Manim erweitern.

4.2 Anwendungsbeispiel mit Manim

Ein typisches Projekt mit Manim beginnt mit der Konzeption und dem Skript, in dem die zu vermittelnden Konzepte und die Struktur des Videos festgelegt werden. Sobald das Skript steht, kann mit der Erstellung der Animationen begonnen werden.

Manim verwendet Python-Skripte, um Animationen zu definieren. Dies bedeutet, dass Benutzer Python-Code schreiben, um die gewünschten visuellen Elemente und ihre Animationen zu spezifizieren. Zum Beispiel kann ein Zustandsdiagramm mit Manim durch Definition der

Zustände, Übergänge und entsprechenden Beschriftungen in Python erstellt werden. Anschließend können Animationen hinzugefügt werden, um die Bewegung von einem Zustand zum anderen zu demonstrieren (Manim Docs, 2024).

Ein einfaches Beispiel könnte die Animation eines Kreises sein, der sich in eine Ellipse transformiert. Mit Manim's Syntax könnten Sie zuerst den Kreis definieren, dann die Transformation spezifizieren und schließlich die Animation ausführen, die diesen Übergang zeigt (Manim Docs, 2024).

4.3 Vorteile und Herausforderungen

Die Flexibilität und Präzision von Manim stellen bedeutende Vorteile gegenüber anderen Visualisierungswerkzeugen dar. Die Fähigkeit, komplexe Ideen genau zu visualisieren und mathematische Ausdrücke nahtlos in Animationen zu integrieren, macht Manim zu einem unschätzbaren Werkzeug für die Erstellung von Bildungsinhalten.

Zu den Herausforderungen bei der Verwendung von Manim zählt die steile Lernkurve, insbesondere für Benutzer ohne Programmiererfahrung. Die Notwendigkeit, Python zu beherrschen und sich mit den spezifischen Funktionen von Manim vertraut zu machen, kann anfangs entmutigend sein. Um diese Herausforderungen zu überwinden, ist es hilfreich, mit der umfangreichen Dokumentation und den Tutorials, die von der Manim-Community bereitgestellt werden, zu beginnen. Darüber hinaus fördert die aktive und unterstützende Manim-Community den Austausch von Wissen und Erfahrungen, was neuen Benutzern den Einstieg erleichtert (Manim Community, 2024).

Zusammenfassend bietet Manim eine einzigartige Kombination aus Flexibilität, Präzision und Community-Unterstützung, die es zu einem idealen Werkzeug für die Erstellung von Erklärungsvideos macht, besonders wenn es um die Darstellung mathematischer und technischer Konzepte geht. Die Überwindung der Einstiegshürden lohnt sich angesichts der qualitativ hochwertigen und informativen Visualisierungen, die mit Manim möglich sind.

5 Grundlagen der Endlichen Automaten

5.1 Geschichte der Endlichen Automaten

Die Ursprünge der endlichen Automaten (Finite State Machines, FSM) lassen sich bis zu den bahnbrechenden Arbeiten von Warren S. McCulloch und Walter Pitts im Jahr 1943 zurückverfolgen. Ihr innovativer Ansatz zur Modellierung neuronaler Netzwerke legte den Grundstein für die Entwicklung von Automatentheorien, die sich mit Zustandsübergängen und -verhalten in komplexen Systemen befassten. Diese Forschung inspirierte viele Wissenschaftler, die FSMs als ein Mittel sahen, um die Dynamik von Systemen zu verstehen und vorherzusagen (vgl. Koshy, 2004).

5.1.1 Weiterentwicklung und Verfeinerung

In den Jahrzehnten nach McCulloch und Pitts erlebten FSMs zahlreiche Weiterentwicklungen. Forscher wie Grant Dick und X. Yao haben intensiv die Flexibilität und Komplexität von Mealy- und Moore-Automaten, zwei grundlegende Typen von FSMs, untersucht. Sie stellten fest, dass diese Automaten zwar eine hohe Anpassungsfähigkeit aufweisen, jedoch mit einer steigenden Anzahl von Zuständen und Ausgabealphabeten auch eine erhöhte Komplexität und damit verbundene Herausforderungen mit sich bringen (vgl. Dick & Yao, 2014).

5.1.2 Anwendungen über verschiedene Disziplinen hinweg

Die Vielseitigkeit und Anwendbarkeit von FSMs wurden in einer Vielzahl von Disziplinen deutlich. In der Kommunikationstechnik, beispielsweise, wurden FSMs für die Fehleridentifikation und -analyse eingesetzt, wie die Arbeit von Bouloutas, Hart und Schwartz zeigt

Darüber hinaus fanden FSMs Anwendung in der Evolutionären Berechnung, wo sie zur Simulation von evolutionären Prozessen und natürlicher Auslese verwendet wurden, wie die Studie von Fogel, Owens und Walsh belegt (vgl. Fogel et al., 1965).

5.1.3 In der modernen Informatik und Künstlichen Intelligenz

In der modernen Programmierung und Softwareentwicklung spielen FSMs eine zentrale Rolle. Sie bieten eine visuelle Darstellung von Algorithmen und ermöglichen es, den Zustand eines Algorithmus in einer Maschine mit einer endlichen Anzahl von Zuständen abzubilden. Dies bietet einen abstrakten Weg zur Algorithmenentwicklung und ist besonders nützlich für die Gestaltung deterministischer Maschinen (vgl. Hardy & Steeb, 2001).

In der Künstlichen Intelligenz (KI) und Robotik werden FSMs eingesetzt, um Verhaltensmodelle und Entscheidungsfindungsprozesse zu gestalten. FSMs ermöglichen es Robotern und KI-Systemen, eine Reihe von Zuständen zu durchlaufen und auf Eingaben oder Umgebungsveränderungen entsprechend zu reagieren (vgl. Deng, 2018).

5.2 Deterministische endliche Automaten

Deterministische endliche Automaten (DEAs) bilden ein grundlegendes Konzept in der theoretischen Informatik und dienen als robustes Modell für die Verarbeitung und Erkennung regulärer Sprachen. Ein DEA wird durch eine endliche Menge von Zuständen, einen Anfangszustand, eine Menge von Endzuständen und eine Übergangsfunktion definiert, die für jedes Eingabesymbol genau einen Übergang zu einem neuen Zustand vorgibt. Diese Eindeutigkeit der Zustandsübergänge unterscheidet DEAs von NEAs, in denen mehrere Zustandsübergänge für dasselbe Eingabesymbol möglich sind (vgl. Kolan et al., 2020).

DEAs zeichnen sich durch ihre Effizienz in der Verarbeitung regulärer Ausdrücke und lineare Zeitkomplexität bei der Mustererkennung aus, was sie insbesondere für Anwendungen wie die Tiefenpaketinspektion unentbehrlich macht (vgl. Becchi & Crowley, 2007). Sie sind außerdem zentral für das Verständnis und die Entwicklung von Software, die auf Sprachverarbeitung basiert.

In jüngster Zeit wurden Fortschritte in der DFA-Forschung erzielt, wie etwa die Einführung von erweiterten finiten Automaten (XFAs), die eine 20-fach höhere Geschwindigkeit bei der Mustererkennung in Netzwerksicherheitssystemen ermöglichen, während sie zehnmal weniger Speicher als traditionelle DFA-basierte Lösungen verwenden (vgl. Smith et al., 2008).
Ebenso hat sich der neue inkrementelle Minimierungsalgorithmus für DFAs als effizient erwiesen und kann jederzeit angehalten werden, um teilweise minimierte Automaten zu liefern (vgl. Watson & Daciuk, 2003).

Diese Entwicklungen zeigen, dass DEAs weiterhin ein aktives und wichtiges Forschungsfeld in der theoretischen Informatik darstellen. Ihre Fähigkeit zur effizienten Darstellung und Verarbeitung regulärer Sprachen macht sie zu einem wesentlichen Instrument in der Sprachmodellierung und -analyse.

5.3 Nichtdeterministische endliche Automaten

Im Gegensatz zu DEAs, die wir im vorherigen Abschnitt diskutiert haben, bieten nichtdeterministische endliche Automaten (NEAs) eine erweiterte Flexibilität in der Darstellung und Verarbeitung regulärer Sprachen. Ein NEA ist durch eine endliche Menge von Zuständen, einen Anfangszustand, eine Menge von Endzuständen und eine Übergangsfunktion charakterisiert, die Zuordnungen von Zuständen und Eingabesymbolen zu Mengen von Zuständen definiert. Diese Struktur ermöglicht es NEAs, mehrere Zustandsübergänge gleichzeitig zu berücksichtigen, was sie besonders effizient in der Darstellung bestimmter Sprachen macht. Im Vergleich zu DEAs können NEAs reguläre Sprachen in einigen Fä/len exponentiell konziser darstellen (vgl. Kintala et al., 1993).

Ein wesentlicher Unterschied zwischen NEAs und DEAs liegt in der Komplexität von Zustandsoperationen. Während Operationen wie Umkehrung und Verkettung bei NEAs eine lineare Komplexität aufweisen, haben sie bei DEAs eine exponentielle Komplexität (vgl. Holzer & Kutrib, 2003).
Trotz des inhärenten Nichtdeterminismus der NEAs besteht eine Äquivalenz zu DEAs hinsichtlich der Sprachen, die sie erkennen können. Für jede Sprache, die von einem NEA akzeptiert wird, existiert ein äquivalenter DEA, der dieselbe Sprache akzeptiert und umgekehrt. In der Praxis werden NEAs oft in DEAs umgewandelt, um die Vorteile deterministischer Berechnungen, wie zum Beispiel die einfachere Implementierung und das effizientere Verhalten in Bezug auf Zeit und Speicher, zu nutzen (vgl. Afat, 2019; Holub, 2007)

Neuere Entwicklungen in der NEA-Forschung konzentrieren sich unter anderem auf die Simulation, Minimierung, Größenabschätzung und Zustandskomplexität von Sprachoperationen. Ein bemerkenswerter Fortschritt ist die Einführung von NEAs mit markierten Übergängen, die effizient in der Verarbeitung von Zeichenketten und in Anwendungen bei regulären Ausdrücken sind (vgl. Laurikari, 2000). Die Einführung der Baumweite als Funktion zur Bestimmung der maximalen Anzahl von Blättern im Berechnungsbaum eines NEA auf einem Eingabewert sowie die exponentiell wachsende Spurenfunktion eines NEA sind ebenfalls wichtige Beiträge (vgl. Palioudakis et al., 2013).

Zusammenfassend bilden sowohl DEAs als auch NEAs ein dynamisches Forschungsfeld in der theoretischen Informatik. Während DEAs durch ihre Eindeutigkeit und Effizienz in der Sprachverarbeitung bestechen, bieten NEAs eine größere Flexibilität und Effizienz bei der Darstellung komplexer Sprachstrukturen. Beide Automatentypen sind unverzichtbar für das Verständnis und die Modellierung regulärer Sprachen.

6 Weitere Automaten und Maschinen

Neben dem Endlichen Automaten, auch Akzeptor genannt, gibt es auch weitere Maschinen. Diese sind ebenfalls theoretische Konstrukte, welche verschiedene Bestandteile haben. Beispielhaft werden hier der Akzeptor, Transduktor, Kellerautomat und Turingmaschine vorgestellt und anschließend miteinander verglichen.

6.1 Akzeptor

Ein Akzeptor besteht aus einem Eingabealphabet (X), einer Zustandsmenge (Z), einer Überführungsfunktion (δ), Anfangszustand(z_0) und einer Endzustandsmenge (Z_E).
Das Eingabealphabet ist eine Menge an Symbolen, die der Automat annimmt. Die Zustandsmenge ist die Menge aller Zustände des Automatens. Der Anfangszustand ist in dieser Zustandsmenge enthalten. Jeder Automat hat exakt einen Anfangszustand. Das ist der initiale Zustand, den der Automat annimmt. Die Endzustandsmenge ist ebenfalls eine Teilmenge der Zustandsmenge. Befindet sich der Automat in diesem Zustand, dann geht er in das Akzeptanzverhalten. Das bedeutet, dass er die Eingabe akzeptiert. In allen anderen Zuständen ist der Automat abweisend (vgl. S.64 Wagenknecht und Hielscher, 2009).
Probleme, für die ein Akzeptor gebraucht wird, sind z.B.: das Entsperren eines Handys, welches bei der Eingabe der richtigen PIN das Handy entsperrt. Auch kann er als Prüfbitgenerator verwendet werden, der prüft, ob die Bits in der richtigen Reihenfolge und vollständig ankommen.

6.2 Transduktor

Ein Transduktor hat anders als ein Akzeptor keine Endzustandsmenge. Stattdessen hat er eine Ausgabefunktion und ein Ausgabealphabet. Die Aufgabe eines Transduktores ist es aus Eingaben des Eingabealphabetes in Kombination des aktuellen Zustandes des Automaten Ausgaben mit dem Ausgabealphabet zu generieren.
Die Elemente eines Transduktores sind: Eingabealphabet (X), Ausgabealphabet (Y), Zustandsmenge (Z) Überführungsfunktion (δ), Ausgabefunktion (λ) und einem Anfangszustand (z_0). Die Überführungsfunktion kann ebenso wie bei einem Akzeptor als Graph dargestellt werden.

Die Ausgabefunktion kann entweder in den Zuständen oder an die Übergänge geschrieben werden. Je nachdem wo sie steht, ist es eine Moore-Maschine oder eine Mealy Maschine.
Wenn es eine Moore-Maschine ist, dann hängt diese Ausgabe „ausschließlich von dem jeweils Maschine aktuellen Zustand der Maschine [ab], d.h. der zugehörige Übergang [spielt] keine Rolle"(S. 98 Wagenknecht und Hielscher, 2022. Bei einer Mealy-Maschine hängt die Ausgabe von dem Übergang der Maschine ab. Es ist immer möglich aus einer Moor-Maschine in eine Mealy Maschine umzuformen.

6.3 Kellerautomat

Kellerautomaten sind 7-Tupel. Sie bestehen aus einem einem Eingabealphabet (X), einer Zustandsmenge (Z), einer Überführungsfunktion (δ), Anfangszustand(z_0), einer Endzustandsmenge (Z_E), einem Kelleralphabet(γ) und einem Kellerstartsymbol() (vgl. S. 208 Vossen und Witt, 2016).
Der Unterschied zu Transduktoren und Akzeptoren besteht darin, dass durch das Kelleralphabet Dinge speichern kann. Dafür hat er ein so genanntes Kellerband, auf das Elemente des Kelleralphabets geschrieben werden können. Anders als die beiden vorher beschriebenen Automaten hängt der nächste Schritt eines Kellerautomaten nicht nur von dem aktuellen Zustand und der Eingabe ab, sondern auch von der letzten Eingabe. Diese ist auf dem Kellerband gespeichert (vgl. S. 207 Vossen und Witt, 2016). Dies ermöglicht es dem Automaten, komplexere Probleme zu lösen.

6.4 Turing Maschine

Die Turingmaschine ist ein Theoretisches Modell von Alan Turing. Sie ist ein 8-Tupel und besteht einer Zustandsmenge (Z), einem Eingabealphabet (X), Bandalphabet(γ), einer Überführungsfunktion (δ), einem Anfangszustand (z_0), einem leeren Feld, einem akzeptierenden Zustand (f_a) und einem verwerfenden Zustand (f_r) (vgl. S.269 Vossen und Witt, 2016). Ähnlich wie die Kellerautomaten hat auch die Turingmaschine ein Band, auf das sie mit dem Bandalphabet schreiben kann. Anders als ein Kellerautoamt hat sie jedoch potenziell Zugriff auf alle Elemente des Bandes. Dies wird dadurch ermöglicht, dass der lesende Zeiger auf dem Band nach links und rechts bewegt werden kann.

6.5 Vergleich der Automaten und Maschinen

Hier eine Übersicht der verschiedenen Bestandteile der Automaten und Maschinen:

	Akzeptor	Transduktor	Kellerautomat	Turing Maschine
Eingabealphabet	ja	ja	ja	ja
Zustandsmenge	ja	ja	ja	ja
Überführungsfunktion	ja	ja	ja	ja
Anfangszustand	ja	ja	ja	ja
Endzustand	ja	nein	ja	nein
Ausgabealphabet	nein	ja	nein	nein
Ausgabefunktion	nein	ja	nein	nein
Kelleralphabet	nein	nein	ja	nein
Kellerstartsymbol	nein	nein	ja	nein
Bandalphabet	nein	nein	nein	ja
leeres Feld	nein	nein	nein	ja
akzeptierender Zustand	nein	nein	nein	ja
verwerfender Zustand	nein	nein	nein	ja

Die größte Schwäche von endlichen Automaten ist, dass deren Ausgabe von dem Zustand, in dem sie sich befinden, und der aktuellen Eingabe abhängen. Die vergangenen Eingaben sind für die Überführungsfunktion nicht relevant. Der Automat kennt immer nur seinen aktuellen Zustand und seine aktuelle Eingabe- er kann auf die vergangenen Eingaben nicht zugreifen. Transduktoren, 6-Tupel, können dies ebenfalls nicht.
Ein so genannter Kellerautomat kann dies. Dieser besitzt zusätzlich zu den Bestandteilen der endlichen Automaten ein Kelleralphabet. In diesem kann er vergangene Eingaben speichern und immer auf die letzte Eingabe zugreifen. Die Vorherigen kann er nicht lesen. Seine Überführungsfunktion kann also nicht nur von dem aktuellen Zustand und der aktuellen Eingabe abhängen, sondern auch auf die letzte Eingabe zugreifen.
Die Turingmaschine ist wiederum dem Kellerautomaten überlegen. Anders als ein Kellerautomat kann die Turingmaschine auf alle vorherigen Eingaben zugreifen. Dies wird ihr durch die Menge der Kopfbewegungen ermöglicht, welche der Kellerautomat nicht besitzt. Jede Erweiterung des theoretischen Modells ermöglicht also immer komplexere Zusammenhänge zu simulieren.

7 Endliche Automaten und Regular Expressions (Regex)

7.1 Einführung in Regular Expressions (Regex)

Regular Expressions, kurz Regex, sind eine mächtige Methode zur Suche und Manipulation von Text durch Beschreibung von Mustern. Sie basieren auf einer formalen Sprache und ermöglichen es, komplexe Textmuster zu definieren und zu erkennen. Regex wird in vielen Programmiersprachen und Tools zur Textverarbeitung eingesetzt, um Textsuchen, Textersetzungen, Datenvalidierungen und -extraktionen durchzuführen. Ein Regex-Muster besteht aus einer Reihe von Zeichen, wobei jedes Zeichen eine spezifische Bedeutung hat und zur Beschreibung des gewünschten Textmusters beiträgt (Arcaini et al., 2019).

7.2 Endliche Automaten in der Anwendung von Regex

Endliche Automaten sind ein fundamentales Konzept in der Informatik und spielen eine entscheidende Rolle bei der Implementierung und Effizienz von Regular Expressions. Ein endlicher Automat ist ein abstraktes Modell eines Rechensystems, das eine begrenzte Anzahl von Zuständen besitzt und in Abhängigkeit von der Eingabe von einem Zustand in einen anderen übergehen kann. Für die Anwendung von Regex sind insbesondere NEAs von Bedeutung, da sie eine natürliche Repräsentation von Regex-Mustern ermöglichen.

7.2.1 Von Regex zu Endlichen Automaten

Die Umwandlung eines Regex in einen endlichen Automaten erfolgt in zwei Schritten:

1. **Konstruktion eines NEA**: Zuerst wird das Regex-Muster in einen Nichtdeterministischen Endlichen Automaten umgewandelt. Dieser Schritt nutzt die inhärente Nichtdeterminiertheit von NEAs, um die verschiedenen möglichen Pfade, die ein Regex-Muster darstellen kann, zu modellieren (Salomaa, 1966).
2. **Transformation in einen DEA**: Obwohl NEAs eine direktere Modellierung von Regex erlauben, sind DEAs in der Praxis effizienter zu evaluieren. Daher wird der NEA in

einen äquivalenten DEA umgewandelt, welcher dann zur schnellen Mustererkennung verwendet wird (Yu et al., 2014).

7.3 Vorteile und Herausforderungen

Die Verwendung endlicher Automaten für die Verarbeitung von Regular Expressions bietet mehrere Vorteile:

- **Effizienz**: DEAs ermöglichen eine schnelle und effiziente Mustererkennung, da für jede Eingabe genau ein Zustandsübergang möglich ist. Die Entwicklung eines automatabasierten Prozessors demonstriert die Überlegenheit dieser Architektur in der Mustererkennung, insbesondere bei der Verarbeitung von Regular Expressions (Dlugosch et al., 2014).
- **Robustheit**: Die formale Grundlage endlicher Automaten gewährleistet, dass Regex korrekt und eindeutig interpretiert werden.
- **Flexibilität**: Durch die Umwandlung von Regex in endliche Automaten können komplexe Suchmuster in Texten erkannt werden.

7.4 Grenzen von Regular Expressions

Trotz ihrer Mächtigkeit und Flexibilität stoßen Regular Expressions in bestimmten Anwendungsfällen an ihre Grenzen. Diese Einschränkungen sind wichtig zu verstehen, um die Einsatzmöglichkeiten und die Effektivität von Regex in verschiedenen Kontexten realistisch einschätzen zu können.

- **Komplexität und Lesbarkeit**: Mit zunehmender Komplexität der Muster können Regex-Ausdrücke schnell unleserlich und schwer zu warten werden. Dies erschwert die Fehlersuche und die Weitergabe des Codes an andere Entwickler.
- **Verarbeitung verschachtelter Strukturen**: Regex ist nicht optimal geeignet, um verschachtelte Strukturen oder rekursive Muster, wie sie in XML- oder HTML-Dokumenten vorkommen, zu verarbeiten. Dies liegt daran, dass Regex grundsätzlich nicht ßählen"kann und daher nicht effektiv zwischen korrespondierenden öffnenden und schließenden Tags unterscheiden kann.

- **Performance**: Obwohl endliche Automaten eine effiziente Verarbeitung von Regex ermöglichen, kann die Performance bei sehr komplexen Ausdrücken oder bei der Verarbeitung großer Textmengen leiden. Insbesondere Backtracking, das bei manchen Regex-Implementierungen zum Einsatz kommt, kann zu erheblichen Leistungseinbußen führen (Patel et al., 2014).
- **Eingeschränkte Ausdrucksstärke**: Regex eignet sich hervorragend für Mustererkennung auf der Basis von Zeichenfolgen, stößt jedoch bei Aufgaben, die ein tieferes Verständnis der Datenstruktur oder des Inhalts erfordern, an seine Grenzen. Zum Beispiel ist Regex nicht in der Lage, semantische Analysen durchzuführen oder den Kontext zu berücksichtigen, in dem ein Muster auftritt.

Diese Grenzen bedeuten nicht, dass Regex nicht wertvoll oder nützlich ist, sondern unterstreichen die Bedeutung der Auswahl des richtigen Werkzeugs für die jeweilige Aufgabe. In vielen Fällen, insbesondere bei der Verarbeitung einfacher oder gut definierter Textmuster, bieten Regular Expressions eine unübertroffene Flexibilität und Effizienz. Für komplexere Textverarbeitungsaufgaben oder die Analyse verschachtelter Strukturen können jedoch andere Methoden oder Werkzeuge besser geeignet sein.

Literatur

Afat, A. M. (2019). The Different Ways to Describe Regular Languages by Using Finite Automata and the Changing Algorithm Implementation. *International Journal of Aerospace and Mechanical Engineering*, *8.0*(6). https://doi.org/10.5281/zenodo.2672609

Arcaini, P., Gargantini, A., & Riccobene, E. (2019). Fault-based test generation for regular expressions by mutation. *Software Testing*, *29*. https://doi.org/10.1002/stvr.1664

Becchi, M., & Crowley, P. (2007). A Hybrid Finite Automaton for Practical Deep Packet Inspection. *Proceedings of the 2007 ACM CoNEXT Conference*. https://doi.org/10.1145/1364654.1364656

Brame, C. (2016). Effective Educational Videos: Principles and Guidelines for Maximizing Student Learning from Video Content. *CBE Life Sciences Education*, *15*. https://doi.org/10.1187/cbe.16-03-0125

Clark, R., Lyons, C., & Hoover, L. (2004). Graphics for learning: Proven guidelines for planning, designing, and evaluating visuals in training materials. *Performance Improvement*, *43*. https://doi.org/10.1002/pfi.4140431011

Cohen, S. S., Madsen, J., Touchan, G., Robles, D., Lima, S. F. A., Henin, S., & Parra, L. (2018). Neural engagement with online educational videos predicts learning performance for individual students. *Neurobiology of Learning and Memory*, *155*, 60–64. https://doi.org/10.1016/j.nlm.2018.06.011

Deng, L. (2018). Artificial Intelligence in the Rising Wave of Deep Learning: The Historical Path and Future Outlook [Perspectives]. *IEEE Signal Processing Magazine*, *35*(1), 180–177. https://doi.org/10.1109/MSP.2017.2762725

Dick, G., & Yao, X. (2014). Model representation and cooperative coevolution for finite-state machine evolution. *2014 IEEE Congress on Evolutionary Computation (CEC)*, 2700–2707. https://doi.org/10.1109/CEC.2014.6900622

Dlugosch, P., Brown, D., Glendenning, P., Leventhal, M., & Noyes, H. (2014). An Efficient and Scalable Semiconductor Architecture for Parallel Automata Processing. *IEEE Transactions on Parallel and Distributed Systems*, *25*, 3088–3098. https://doi.org/10.1109/TPDS.2014.8

Fleming, N., & Mills, C. (1992). Not Another Inventory, Rather a Catalyst for Reflection. *To improve the academy*, *11*. https://doi.org/10.1002/j.2334-4822.1992.tb00213.x

Fogel, L., Owens, A., & Walsh, M. (1965). Intelligent decision-making through a simulation of evolution. *SIMULATION*, *5*(4), 267–279. https://doi.org/10.1177/003754976500500413

Foley, J. (1996). *Computer Graphics: Principles and Practice*. Addison-Wesley. https://books.google.de/books?id=-4ngT05gmAQC

Goldstein, E. (2014). *Cognitive Psychology: Connecting Mind, Research and Everyday Experience*. Cengage Learning. https://books.google.de/books?id=4LI8AwAAQBAJ

Hardy, Y., & Steeb, W.-H. (2001). Finite State Machines. In *Classical and Quantum Computing: with C++ and Java Simulations* (S. 229–250). Birkhäuser Basel. https://doi.org/10.1007/978-3-0348-8366-5_12

Holub, J. (2007). Finite Automata Implementations Considering CPU Cache. *Acta Polytechnica*, *47*(6). https://doi.org/10.14311/1008

Holzer, M., & Kutrib, M. (2003). State Complexity of Basic Operations on Nondeterministic Finite Automata. In J.-M. Champarnaud & D. Maurel (Hrsg.), *Implementation and Application of Automata* (S. 148–157). Springer Berlin Heidelberg.

Hopcroft, J., Motwani, R., & Ullman, J. (2001). *Introduction to Automata Theory, Languages, and Computation*. Addison-Wesley. https://books.google.de/books?id=omIPAQAAMAAJ

Ivanova, S. (2017). USING EXPLAINER VIDEOS TO TEACH WEB DESIGN CONCEPTS. *eLearning and Software for Education*. https://doi.org/10.12753/2066-026x-17-063

Kintala, C. M. R., Pun, K., & Wotschke, D. (1993). Concise Representations of Regular Languages by Degree and Probabilistic Finite Automata. *Math. Syst. Theory*, *26*(4), 379–395. https://doi.org/10.1007/BF01189856

Kolan, A., Sreevani, K. S. S., & Jayasree, H. (2020). Comparative Study of Performance of Tabulation and Partition Method for Minimization of DFA. In K. S. Raju, A. Govardhan, B. P. Rani, R. Sridevi & M. R. Murty (Hrsg.), *Proceedings of the Third International Conference on Computational Intelligence and Informatics* (S. 17–27). Springer Singapore.

Koshy, T. (2004). A Word to the Student. In T. Koshy (Hrsg.), *Discrete Mathematics with Applications* (S. xxi–xxiv). Academic Press. https://doi.org/https://doi.org/10.1016/B978-012421180-3/50001-2

Krämer, A., & Böhrs, S. (2017). Erklärvideos als effektives und ef zientes Marketing-Instrument. *Marketing Review St Gallen*, *2*, 54–61.

Laurikari, V. (2000). NFAs with tagged transitions, their conversion to deterministic automata and application to regular expressions. *Proceedings Seventh International Symposium on String Processing and Information Retrieval. SPIRE 2000*, 181–187. https://doi.org/10.1109/SPIRE.2000.878194

Manim Community. (2024). Manim - Mathematical Animation Framework [Zugriff am: 19.02.2024].

Manim Docs. (2024). Manim - Mathematical Animation Framework-Documentation [Zugriff am: 19.02.2024].

Mayer, R. E. (2009). *Multimedia Learning* (2. Aufl.). Cambridge University Press.

Naps, T. L., Rößling, G., Almstrum, V., Dann, W., Fleischer, R., Hundhausen, C., Korhonen, A., Malmi, L., McNally, M., Rodger, S., & Velázquez-Iturbide, J. Á. (2002). Exploring the role of visualization and engagement in computer science education. *Working Group Reports from ITiCSE on Innovation and Technology in Computer Science Education*, 131–152. https://doi.org/10.1145/960568.782998

Palioudakis, A., Salomaa, K., & Akl, S. G. (2013). Comparisons between Measures of Nondeterminism on Finite Automata. In H. Jurgensen & R. Reis (Hrsg.), *Descriptional Complexity of Formal Systems* (S. 217–228). Springer Berlin Heidelberg.

Patel, J., Liu, A., & Torng, E. (2014). Bypassing Space Explosion in High-Speed Regular Expression Matching. *IEEE/ACM Transactions on Networking*, *22*, 1701–1714. https://doi.org/10.1109/TNET.2014.2309014

Rodger, S., & Finley, T. (2006). *JFLAP: An Interactive Formal Languages and Automata Package*. Jones; Bartlett. https://books.google.de/books?id=494hTkZ_gu4C

Salomaa, A. (1966). Two Complete Axiom Systems for the Algebra of Regular Events. *J. ACM*, *13*, 158–169. https://doi.org/10.1145/321312.321326

SIGCSE '20: Proceedings of the 51st ACM Technical Symposium on Computer Science Education. (2020). Association for Computing Machinery.

Smith, R., Estan, C., & Jha, S. (2008). XFA: Faster Signature Matching with Extended Automata. *2008 IEEE Symposium on Security and Privacy (sp 2008)*, 187–201. https://doi.org/10.1109/SP.2008.14

Thomas, F., & Johnston, O. (1981). *Disney Animation: The Illusion of Life* [112 reviews]. Abbeville Press.

Vossen, G., & Witt, K.-U. (2016). *Grundkurs Theoretische Informatik: Eine anwendungsbezogene Einführung - Für Studierende in allen Informatik-Studiengängen* (6., erw. u. überarb. Aufl. 2016). Springer Fachmedien Wiesbaden. https://doi.org/10.1007/978-3-8348-2202-4

Wagenknecht, C., & Hielscher, M. (2009). *Formale Sprachen, abstrakte Automaten und Compiler: Lehr- und Arbeitsbuch für Grundstudium und Fortbildung*. Vieweg+Teubner Verlag / GWV Fachverlage GmbH, Wiesbaden. https://doi.org/10.1007/978-3-8348-9972-9

Wagenknecht, C., & Hielscher, M. (2022). *Formale Sprachen, abstrakte Automaten und Compiler: Lehr- und Arbeitsbuch mit FLACI für Grundstudium und Fortbildung* (3., überarbeitete und ergänzte Auflage). Springer Vieweg. https://doi.org/10.1007/978-3-658-36853-1

Watson, B., & Daciuk, J. (2003). An efficient incremental DFA minimization algorithm. *Natural Language Engineering*, *9*, 49–64. https://doi.org/10.1017/S1351324903003127

Wright, S. (2006). *Digital Compositing for Film and Video*. Focal Press. https://books.google.de/books?id=DDveY2hHLGQC

Yu, X., Lin, B., & Becchi, M. (2014). Revisiting State Blow-Up: Automatically Building Augmented-FA While Preserving Functional Equivalence. *IEEE Journal on Selected Areas in Communications*, *32*, 1822–1833. https://doi.org/10.1109/JSAC.2014.2358840