Raphael Zender et al. (Hrsg.): Die 18. Fachtagung Bildungstechnologien (DELFI),
Lecture Notes in Informatics (LNI), Gesellschaft für Informatik, Bonn 2020 133

Automatisierte Generierung von Automaten und automatenbasierten Aufgaben

Sven Judel[1], Timo Bergerbusch[2], Ulrik Schroeder[1]

Abstract: Dieser Beitrag stellt das Tool *Automata Task Random Generator* (*AuTaRG*) vor, das für die Bachelorvorlesung *Formale Systeme, Automaten, Prozesse* der RWTH Aachen automatenbasierte Aufgaben generiert. Es wird präsentiert, welche automatenbasierte Aufgabentypen behandelt werden, was das jeweilige Lernziel des Aufgabentyps ist, wie Eigenschaften und Zahlenwerte, die ein Automat besitzen muss um für die einzelnen Aufgabentypen geeignet zu sein, ermittelt wurden und wie solche Automaten generiert werden können. In einer Evaluation wurde erwiesen, dass durch die Nutzung des Tools Zeit bei der Generierung der Aufgaben und der Überführung in LaTeX-Code gespart wird und eine, im Vergleich zur händischen Erstellung, größere Aufgabenvariation und –menge generiert werden kann. Dies führt dazu, dass Assistenten weniger Aufwand im Übungsbetrieb betreiben müssen und Studierende mehr Aufgaben zum Üben erhalten.

Keywords: Automatisierte Generierung, Automaten, Automatentheorie

1 Einleitung

Die händische Erstellung von Übungsaufgaben und ihrer Lösungen ist zeitintensiv und birgt die Gefahr von Fehlern. Daher werden oft alte Aufgaben übernommen oder angepasst. Die automatisierte Generierung von Aufgaben und deren Lösungen soll die Erstellung beschleunigen, auf Korrektheit prüfen und eine größere Aufgabenvariation und –menge ermöglichen. [Ku20] Für die Umsetzung einer automatisierten Aufgabengenerierung wurden automatenbasierte Aufgaben der Vorlesung *Formale Systeme, Automaten, Prozesse* (*FoSAP*) an der RWTH Aachen betrachtet. Diese Vorlesung ist ein Pflichtmodul des Informatik-Bachelorstudiums. Die Ausgabe der wöchentlichen Aufgaben erfolgt durch den Upload eines PDFs in den Moodle-Lernraum des Kurses.

Dieser Beitrag gibt eine Antwort auf die Frage: *Wie können geeignete automatenbasierte Aufgaben für die FoSAP Vorlesung generiert werden?* Zur Beantwortung dieser Forschungsfrage wurden vier Teilforschungsfragen vorangestellt:

1. Welche automatenbasierten Aufgabentypen gibt es?
2. Wann ist ein Automat für einen Aufgabentypen verwendbar?

[1] RWTH Aachen, Lehr- und Forschungsgebiet Informatik 9, Ahornstr. 55, 52074 Aachen, {judel, schroeder}@informatik.rwth-aachen.de
[2] RWTH Aachen, Ahornstr. 55, 52074 Aachen, timo.bergerbusch@rwth-aachen.de

3. Was definiert eine Aufgabe als geeignet?
4. Wie können verwendbare Automaten für einen Aufgabentypen generiert werden?

Hier wird zwischen *Aufgabentyp*, einem Konzept (z. B. „Addition“), und *Aufgabe*, einer konkreten Instanz eines Aufgabentyps (z. B. „Berechne 2 + 2“) unterschieden.

Die Fragen 1 bis 3 werden in Kapitel 2 beantwortet, während Kapitel 3 das Konzept des erarbeiteten Generators als Antwort für Frage 4 präsentiert. In Kapitel 4 wird die Umsetzung des Generators als Onlinetool vorgestellt und in Kapitel 5 die Evaluation der Performance und Bedienbarkeit. Kapitel 6 fasst den Beitrag zusammen und stellt mögliche Weiterentwicklung für die Generierung von Aufgaben sowie der Nutzung dieser für automatisierte Bewertungen vor.

2 Aufgabenanalyse

Die Beantwortung der ersten drei Teilforschungsfragen beruht auf der Analyse der verfügbaren Übungsmaterialien. Die Aufgaben der Jahre 2015 bis 2019 wurden gesichtet und acht automatenbasierte Aufgabentypen identifiziert. Die verwendeten Automaten wurden auf ihre Eigenschaften und Zahlenwerte hin untersucht, um eine Eignungsklassifikation von Aufgaben für Aufgabentypen zu erstellen.

2.1 Gruppierung der existenten Aufgaben

Tab. 1 gibt die acht identifizierten Aufgabentypen, als Antwort auf Teilforschungsfrage 1, wieder. Die Gruppierung basiert auf den automatenbasierten Konzepten der Formale Systeme, Automaten, Prozesse (FoSAP) Vorlesung. Beispielhaft werden drei dieser Aufgabentypen zusammen mit ihrem Lernziel beschrieben.

Einführung	Zustandsäquivalenz	Produktautomat	NFA-Pfade
Spracherkennung	Blockverfeinerung	Potenzmengenkonstruktion	ϵ-NFA

Tab. 1: Die acht identifizierten Aufgabentypen

Als Einstieg in das Thema *Automaten*, mit dem Ziel die Studierenden mit den Grundlagen vertraut zu machen, wird der Aufgabentyp *Einführung* ausgegeben. Zu einem gegebenen Deterministischen Endlichen Automaten (*deterministic finite automata - DFA*) $\mathcal{A}$, soll das äquivalente 5-Tupel aufgeschrieben werden. Anschließend soll für die Wörter einer bestimmten Länge angeben werden, ob $\mathcal{A}$ sie akzeptiert. Abschließend soll eine umgangssprachliche Beschreibung aller von $\mathcal{A}$ akzeptierten Wörter angegeben werden. Ziel dieses Typs ist die Studierenden mit den Grundlagen vertraut zu machen.

In Aufgaben vom Typ *Produktautomat* soll das Produkt $\mathcal{P}$ von zwei Automaten $\mathcal{A}$ und $\mathcal{B}$ konstruiert werden, sodass $\mathcal{P}$ die Vereinigung der von $\mathcal{A}$ und $\mathcal{B}$ akzeptierten Wörter selbst

akzeptiert ($\mathcal{L}_{\mathcal{P}} = \mathcal{L}_{\mathcal{A}} \cup \mathcal{L}_{\mathcal{B}}$). Dadurch soll die Übertragbarkeit von Mengenoperationen über Sprachen auf Automaten verdeutlicht werden.

Mit dem Aufgabentyp *NFA-Pfade* wird das Thema Nichtdeterminismus behandelt. Zu einem gegebenen Nichtdeterministischen Endlichen Automaten (*non-deterministic finite automata - NFA*) $\mathcal{A}$ und einem Wort ω sollen alle möglichen Pfade für ω in $\mathcal{A}$, die Menge der mit ω in $\mathcal{A}$ erreichbaren Zustände und ob $\mathcal{A}$ ω akzeptiert angegeben werden. Ziel ist die Auseinandersetzung mit und das Verständnis von Nichtdeterminismus.

2.2 Analyse der Automateneigenschaften

Für die Klassifizierung eines Automaten hinsichtlich der Eignung für einen Aufgabentyp und deren Aufgaben, wurden die Automaten der vorhandenen Aufgaben auf charakterisierende Eigenschaften (für die Aufgabentypen) und Zahlenwerte (für die Aufgaben) hin untersucht.

Die betrachteten Eigenschaften waren *zugänglich*, *co-zugänglich*, *akzeptierend*, *vollständig* und *deterministisch.* Diese fünf Eigenschaften reichten aus, um die Automaten im gegebenen Kontext zu klassifizieren.

In Tab. 2 ist die Analyse der Eigenschaften der drei verfügbaren Automaten zum Aufgabentyp *Potenzmengenkonstruktion* gegeben. Zur besseren Übersicht wurde zusätzlich zur Deterministisch-Spalte eine Nichtdeterministisch-Spalte hinzugefügt. Die Eigenschaften der drei Automaten sind in je einer Zeile gegeben. Das Vorhandensein einer Eigenschaft wird für jede Spalte aufsummiert und der relative Wert (Prozentwert) für die Gesamtmenge der Automaten berechnet. Basierend auf diesem Prozentwert wird eine Eigenschaft als *unzulässig* (0%), *vernachlässigbar* ($0 < x \leq 50\%$), *gewünscht* ($50\% < x < 100\%$) oder *notwendig* (100%) klassifiziert.

	Zugänglich	**Co-zugänglich**	**Akzeptier-end**	**Vollständig**	**Deterministisch**	**Nicht-deterministisch**
1	✓	✓	✓	×	×	✓
2	×	✓	✓	×	×	✓
3	×	✓	✓	×	×	✓
Summe	1	3	3	0	0	3
Prozentwert	33%	100%	100%	0%	0%	100%
Ergebnis	Vernach-lässigbar	Notwen-dig	Notwendig	Unzuläs-sig	Unzulässig	Notwendig

Tab. 2: Eigenschaften der verfügbaren Automaten zum Aufgabentyp Potenzmengenkonstruktion

Zusätzlich zu den Eigenschaften, die ein Automat hat oder nicht, wurden die Zahlenwerte *Anzahl Zustände (n)*, *Länge des Alphabets (k)*, *Übergangsdichte* und *absolute* und *relative Anzahl Finalzustände* für jeden Automaten bestimmt. Tab. 3 gibt die Ergebnisse der Zahlenwertanalyse für die drei Automaten der Aufgaben zur Potenzmengenkonstruktion

wieder. Die Werte der einzelnen Automaten sind in je einer Zeile gegeben. Zu jedem Zahlenwert wurden das Minimum, das Maximum und der Durchschnitt berechnet.

	n	**k**	**Übergangsdichte**	**Finalzustände (absolut)**	**Finalzustände (relativ)**
1	4	3	0,1667	1	25%
2	5	3	0,1333	3	60%
3	5	3	0,16	1	20%
Minimum	4	3	0,1333	1	20%
Durchschnitt	4,6667	3	0,1533	1,6667	35%
Maximum	5	3	0,1667	3	60%

Tab. 3: Zahlenwerte der verfügbaren Automaten zum Aufgabentyp Potenzmengenkonstruktion

Mit diesen Ergebnissen leiten sich die notwendigen Eigenschaften (die zu erfüllen sind) und den unzulässigen Eigenschaften (die nicht gegeben sein dürfen) ab, welche die Eignung eines Automaten für einen Aufgabentypen festlegen. Die initialen Eignungsherleitungen für alle acht Aufgabentypen sind die Antwort auf Teilforschungsfrage 2. Diese sollen mit dem Feedback aus der Nutzung des Tools im Lehrbetrieb kontinuierlich angepasst werden.

2.3 Aufgabenkriterien

Eine Aufgabe ist für einen Aufgabentyp geeignet, wenn durch die Bearbeitung das Lernziel erreicht wird. Dies stellt einige Bedingungen (*Kriterien*) an die Aufgabenstellung und Lösungen. So müssen z. B. Aufgaben vom Typ *NFA-Pfad* über einen Automaten verfügen, der für das gegebene Wort mindestens zwei verschiedene Pfade besitzt. Die Lösung einer Aufgabe zum Produktautomat soll weniger als 12 Zustände haben, um noch gut per Hand gelöst und gezeichnet werden zu können, und gleichzeitig mehr Zustände als der Automat der Aufgabenstellung. Der Rahmen von Automaten, deren Größe für den Übungsbetrieb geeignet sind, wird weiter durch die erhobenen Intervalle (Minimum bis Maximum) für jeden Zahlenwert definiert.

Für jeden Aufgabentyp gibt die Kombination der Zahlenwertintervalle und Kriterien die Definition einer geeigneten Aufgabe an und somit die Antwort auf Teilforschungsfrage 3.

3 Automatengenerator

Nach der Analyse der verfügbaren Aufgaben und der Beantwortung der ersten drei Fragen werden nun die Ergebnisse genutzt um einen Generator für Automaten zu konzipieren. Dabei stellen sich die Fragen nach einem geeigneten Generierungsverfahren und einer geeigneten Generierungsstrategie.

3.1 Generierungsverfahren

Zunächst wurde nach vorhandenen Generatoren gesucht, die zufällige Automaten generieren. Die Forderung nach der zufälligen Generierung verhindert die Notwendigkeit einen Automaten vorzugeben, was zu der oben beschriebenen Wiederverwendung oder Anpassung alter Automaten führen kann.

Die drei vielversprechendsten der gefundenen Generatoren von NFAs (ein DFA ist per Definition auch ein NFA), *Bit Streaming*, *Leslie Generator* und *FIFA Generator* wurden näher betrachtet. Beim *Bit Streaming* wird eine Bit-Sequenz der Länge $n^2 \times k$ generiert, in welcher eine *1* als Transition zwischen zwei Zuständen interpretiert wird [Ch04]. Ein Beispiel ist in Abb. 1 gegeben.

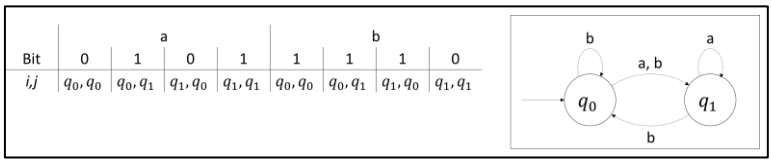

Abb. 1: Darstellung eines Automaten als Bitstream und als Transitionsdiagramm

Ein Nachteil vom Bit Streaming ist die Möglichkeit unzusammenhängende Automaten zu generieren. Dies wird vom *Leslie Generator* durch die Nutzung von Bit Streaming in Kombination mit einer zusätzlichen Bedingung und dem Parameter *Übergangsdichte* umgangen. Zunächst wird ein Automat als zufällig verbundene Struktur generiert, sodass es keine isolierten Zustände gibt, und anschließend Transitionen hinzugefügt, um die gewünschte Dichte zu erreichen. [Le95]

Die Notwendigkeit einen weiteren Parameter angeben zu müssen wirkt aber der zufälligen Generation entgegen. Dem *FIFA Generator* wiederum reichen die Parameter *n* und *k*. Ein *FIFA* (*forward injective finite automata*) ist ein NFA, der u. a. initial verbunden ist und durch einen kanonischen String repräsentiert werden kann. [Fe18]

Beginnend mit dem Startzustand wird angegeben, mit welchen Buchstaben des Alphabets auf bisher gefundene und auf neu entdeckte Zustände verwiesen wird. Die Buchstaben des Alphabets werden durch Nummern als Indizes eines Arrays aller Buchstaben-kombinationen inklusive *keine Verbindung* dargestellt. Für das Alphabet $\Sigma = \{a, b\}$ wäre dies das Array [„*keine Verbindung*“, „*a*“, „*b*“, „*a, b*“]. Abb. 2 gibt einen Automaten mit diesem Alphabet und *n = 5* zusammen mit seiner Darstellung durch den kanonischen String an. Der Startzustand ist *A*. Er besitzt keine Transition auf sich selbst, dargestellt durch die 0. Von *A* aus erreicht man die Zustände *B* und *C* mit den Buchstaben *a* (Index 1) bzw. *b* (Index 2). Die Suche nach Folgezuständen erfolgt durch eine Breitensuche nach der Reihenfolge der Buchstaben im Array. So werden zuerst die Folgezustände für *a*, dann für *b* und abschließend für *a, b* gesucht. Der Zustand *B* hat keine Transition zurück auf *A*, aber für die Buchstaben *a* und *b* (Index 3) eine Transition auf sich selbst. Er hat ebenfalls keine Transition zu *C* und auch keine Transitionen auf neue Zustände. Die Zustände *C*, *D*

und *E* werden analog notiert. Die letzte Liste gibt binär an, welcher der gefundenen Zustände ein Endzustand ist. In dem gegebenen Beispiel ist dies Zustand *D*, der als viertes gefunden wurde, sodass nur an der vierten Stelle des Arrays eine *1* steht.

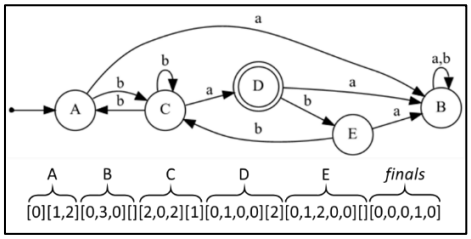

Abb. 2: Ein Automat als Transitionsdiagramm und als kanonischer String

Der FIFA Generator erstellt einen Automaten mit *n* Zuständen durch Breitensuche mit der zufälligen Auswahl von Buchstaben für jede Zustandskombination, sodass Verbundenheit garantiert wird. So lassen sich für beispielweise *n=k=2* ohne Berücksichtigung von Endzuständen 192 Automaten generieren, mit der Berücksichtigung von Endzuständen 768. Für *n=5* und *k=2* (wofür in Abb. 2 ein Beispiel gegeben ist) lassen sich unter Berücksichtigung der Endzustände $3{,}32224 \times 10^{14}$ Automaten generieren.

Leider war es nicht möglich den veröffentlichten Pseudocode zu implementieren, da es einige logische Fehler, die z. B. in Endlosschleifen resultierten, gab. Emails an die Autoren blieben unbeantwortet. [Be20] Daher wurde das Konzept des FIFA Generators übernommen und an den gegebenen Kontext angepasst.

3.2 Generierungsstrategie

Zwei Strategien der Generierung waren möglich: *Proaktiv* und *auf Abruf*. Die proaktive Strategie trennt die Generierung der Automaten von der Generierung der Aufgaben (Abb. 3). Es werden periodisch Automaten erzeugt und in einer Datenbank gespeichert, sodass bei einer Aufgabengenerierung in der Datenbank nach einem geeigneten Automaten gesucht werden kann. In dem Fall, dass kein geeigneter Automat gefunden wird, wird auf einen Fallback-Automaten zurückgegriffen. Dies sind die Automaten, die in den analysierten Aufgaben genutzt werden. Die „auf Abruf"-Strategie generiert nur dann einen Automaten, wenn auch eine Aufgabe generiert wird. Hier könnte durch einen internen Counter nach *x* Generierungen ein Abbruch und Zurückgreifen auf einen Fallback-Automaten realisiert werden. Die Herausforderung liegt dabei im Setzen des Schwellwerts *x*. Die Wartezeit pro Aufgabe ist davon abhängig, wie schnell ein geeigneter Automat gefunden wird, was bei einer zufälligen Generierung ohne Abbruch im Worst Case nie der Fall ist. Der Schwellwert für einen Abbruch darf aber nicht zu niedrig gesetzt werden, um seltener auf Fallback-Automaten zurückgreifen zu müssen.

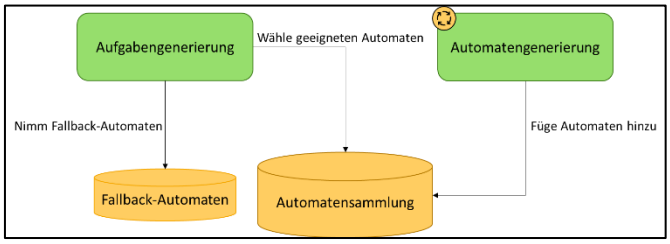

Abb. 3: Hinzufügen und Abrufen von Automaten bei der proaktiven Generierung

Durch das immer neue Generieren von Automaten in der „auf Abruf"-Strategie wird mit jedem generierten aber verworfenen Automaten Rechenleistung und Zeit verschwendet. Ein generierter Automat kann für die gewollte Aufgabe ungeeignet, für eine andere jedoch verwendbar sein.

Daraus folgend wurde sich für die proaktive Generierung entschieden. Ein kontinuierlich laufender Prozess erzeugt Automaten, bestimmt deren Eigenschaften und Zahlenwerte und speichert sie in einer Datenbank. Bei der Aufgabengenerierung werden die vorhandenen Automaten nach ihren Attributen gemäß der Eignungsklassifizierung des Aufgabentyps selektiert und zufällig einer zurückgegeben.

3.3 Filterung ungeeigneter Automaten

Um keine Automaten zu speichern, die für keinen der implementierten Aufgabentypen geeignet sind, wird eine Heuristik genutzt, die Automaten aussortiert, bei denen mindestens ein Zahlenwert außerhalb des Intervalls über die Zahlenwerte aller Aufgabentypen liegt. Tab. 4 gibt die Intervalle aller Zahlenwerte an, wobei für die Grenzen von Übergangsdichte und relativer Finalzustände zwischen deterministischen und nichtdeterministischen Automaten unterschieden wird.

Zahlenwert	Deterministisch	Minimum	Maximum
n	Keine Unterscheidung	2	8
k	Keine Unterscheidung	2	3
Übergangsdichte	Deterministisch	0,142	0,67
Übergangsdichte	Nichtdeterministisch	0,125	0,67
Finalzustände (relativ)	Deterministisch	12,5%	50%
Finalzustände (relativ)	Nichtdeterministisch	10,9%	26%

Tab. 4: Minimum und Maximum der Zahlenwerte über alle Aufgabentypen

Diese simple Filterung sortiert nicht alle ungeeigneten Automaten aus, doch würde die Überprüfung komplexerer Kriterien eine höhere Rechenzeit pro Automat bedeuten. Mit der beschriebenen Heuristik wurde ein guter Mittelwert zwischen Filterung und Einfachheit gefunden. In der Implementierung wird die Heuristik als eigenständiges Modul eingebunden, sodass sie durch eine andere ersetzt werden kann.

4 Der Webservice

Der *Automata Task Random Generator* (*AuTaRG*) wurde als Webtool umgesetzt. Diese Umsetzung erlaubt einen schnellen, orts- und betriebssystemunabhängigen Zugriff ohne die Notwendigkeit einer Installation oder dem Zugang zu einem bestimmten PC. Die Architektur ist in Abb. 4 gegeben und wird im Folgenden näher erläutert.

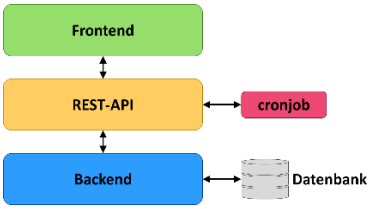

Abb. 4: Die Architektur des Webtools

Das Backend ist in Python geschrieben und setzt sich aus mehreren Modulen zusammen. Einzelne Module sind für die Generierung von Automaten oder Aufgaben zuständig, erstellen Export-Dateien oder dienen als Kommunikationsschnittstelle mit der Datenbank. Dieser Aufbau erlaubt einfache Anpassungen, Wartungen und Erweiterungen.

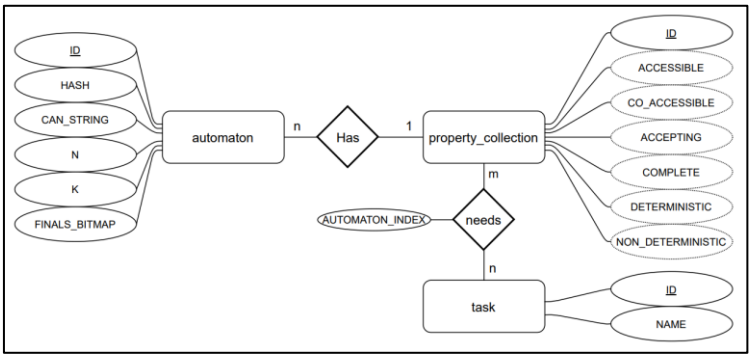

Abb. 5: Das Datenbankmodell

Abb. 5 zeigt das genutzte Datenbankmodell. Die Aufgabentypen und jede Kombination von Eigenschaften, die für einen Aufgabentyp benötigt werden, sind hinterlegt. Der `AUTOMATON_INDEX` wird zur Identifizierung der Automaten pro Aufgabe genutzt, da manche Aufgabentypen mehr als einen Automaten, ggf. mit verschiedenen Eigenschaften, benötigen. Nach der Generierung werden die Automaten auf ihre Eigenschaften untersucht und bei der Speicherung in der Datenbankbank mit der ID der dazugehörigen Eigenschaftenkombination verknüpft. Wird eine Aufgabe generiert, werden die entsprechenden Automaten mit den geforderten Eigenschaften selektiert und zufällig ein Automat zurückgegeben. Die so erstellte Aufgabe wird anschließend auf ihre Eignung für den Aufgabentyp geprüft.

Das Frontend ist als Single-Page-Application umgesetzt, welche Anfragen zur Generierung von Aufgaben oder Automaten durch die REST-API an das Backend sendet und die empfangenen Daten für den Nutzer aufbereitet und darstellt. Diese REST-API wird auch vom cronjob genutzt, der durch diese Schnittstelle einmal stündlich die Generierung von 100 kanonischen Strings mit zufälligen n und k im Backend anstößt. Diese werden zunächst ohne Endzustände erstellt und anschließend alle Endzustandskombinationen auf ihre Eignung getestet. Es resultierten durchschnittlich 34 geeignete Automaten aus einem kanonischen String. Die 100 Generierungen wurden durch Kalibrierung ermittelt, da sie eine gute Balance zwischen stündlichem Rechenaufwand und generierten, geeigneten Automaten bieten (durchschnittlich 3361,27 Automaten in 5 Sekunden).

Zusätzlich kann die Automatengenerierung durch das Frontend manuell, mit der Möglichkeit konkrete Parameter für die Zahlenwerte vorzugeben, angestoßen werden. Dies kann genutzt werden, wenn AuTaRG neu aufgesetzt wird oder viele Automaten für einen bestimmten Aufgabentyp gebraucht werden. Auch das Generieren eines einzelnen Automaten ist möglich. Dazu werden im Frontend, neben dem Transitionsdiagramm und dem kanonischen String, die Eigenschaften des Automaten aufgelistet. Weiterhin kann die Akzeptanz von Worten getestet werden.

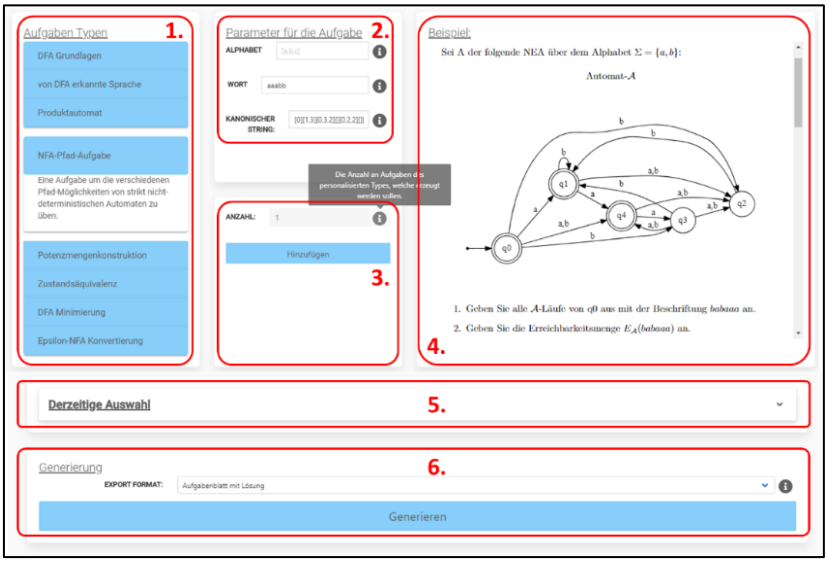

Abb. 6: Das Interface zur Aufgabengenerierung

Das sechsteilige Interface zur Aufgabengenerierung ist in Abb. 6 dargestellt. Zu Beginn wird im ersten Teil ein Aufgabentyp ausgewählt, für den Aufgaben generiert werden sollen. Soll eine Aufgabe mit einem bestimmten Automaten generiert werden, können im zweiten Teil der kanonische String eingefügt und ein Alphabet und weitere Parameter für die Aufgabenkonfiguration angegeben werden. Sollen stattdessen eine oder mehrere

Aufgaben mit zufälligen Automaten generiert werden, wird die gewünschte Anzahl im dritten Teil angegeben. Mit einem Klick auf den *Hinzufügen*-Button werden die angegebenen Parameter validiert, ehe die gewünschten Aufgaben der Auswahl angehangen werden.

Der vierte Teil zeigt eine statische Beispielaufgabe des ausgewählten Typs, während im fünften Teil die Auswahl eingesehen und einzelne Aufgaben wieder entfernt werden können. Abschließend wird im sechsten Teil das gewünschte Exportformat (PDF, LaTeX-Code oder Moodle-XML) ausgewählt und mit dem Klicken des *Generieren*-Buttons eine Anfrage an das Backend geschickt, die als Antwort eine Datei im gewünschten Format mit den geforderten Aufgaben zurückgibt, deren Download vom Frontend angestoßen wird.

Für den Export als PDF kann angegeben werden, ob die Lösung mitgegeben werden soll oder nicht, während im LaTeX-Code eine Variable eingesetzt wird, die festlegt, ob die Musterlösung beim Kompilieren generiert wird. Der Export als Moodle-XML ist gegeben, um die Aufgaben in Moodle-Quizze zu importieren. Zur Anpassung an verschiedene Fragetypen kann das genaue Format der XML im Backend definiert werden.

5 Performance und Nutzerevaluation

Die Performance der Automatengenerierung wurde durch 100 Generierungen von 100 Automaten bei einer leeren Datenbank mit und ohne Nutzung der Heuristik zur Filterung getestet. Abb. 7 gibt die durchschnittlich benötigte Zeit zur Generierung eines Automaten in Millisekunden für beide Testläufe an.

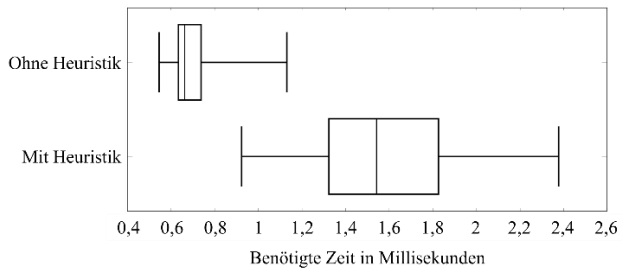

Abb. 7: Durchschnittlich benötigte Zeit zur Generierung eines Automaten mit und ohne Heuristik

Die Vorteile der Filterung ungeeigneter Automaten durch die Heuristik (weniger Suchdurchläufe und geringere Speicherkomplexität) rechtfertigen die geringfügig längere Laufzeit bei der Generierung.

Die Aufgabengenerierung wurde für je 500 Aufgaben zu jedem Aufgabetyp in drei Einzelschritten gemessen: Die Generierung der Aufgaben, das Rendern der Automaten als

Transitionsdiagramm und die Exporte als *LaTeX-Code*, *ein PDF pro Aufgabe* und *ein PDF für alle Aufgaben*. Der Export als Moodle-XML wurde nicht gemessen, da dieser nur für eine Aufgabe definiert ist (mehr dazu im Ausblick). Tab. 5 gibt die Ergebnisse wieder.

	Benötigte Zeit in Sekunden					
			Exporte			
Aufgabentyp	*Aufgaben-generierung*	*Rendering*	*LaTeX-Code*	*Einzelnes PDF*	*Separate PDFs*	∑
Einführung	19,98	28,41	0,70	3,89	6,33	59,31
Spracherkenn.	7,44	28,71	0,28	2,22	4,26	42,91
Produktauto.	36,57	78,43	0,80	4,97	8,14	128,91
NFA-Pfade	183,78	30,81	0,44	3,01	5,12	223,16
Potenzmenge.	392,86	60,48	0,61	3,80	5,84	463,59
Zustandsäqui.	12,33	18,02	1,25	3,19	5,10	39,89
Blockverfein.	10,33	92,94	1,28	6,38	8,81	119,74
ϵ-NFA	173,38	56,97	0,48	3,81	6,12	240,76

Tab. 5: Benötigte Zeit für 500 Aufgaben pro Aufgabentyp in den Schritten Generierung, Rendering und den drei Exporten

Die Komplexität der Aufgabenkriterien (siehe 2.3) und die automatische Berechnung der Lösung hat einen starken Einfluss auf die benötigte Zeit der Aufgabengenerierung. Die längste Generierung (500 Aufgaben zur *Potenzmengenkonstruktion*) dauert etwas weniger als acht Minuten. Das händische Erstellen, Lösen und Übersetzen in LaTeX-Code einer einzigen Aufgabe dieses Aufgabentyps dauert erfahrungsgemäß länger. Die Nutzung von AuTaRG erweist sich somit als große Zeiteinsparung.

Das Frontend von AuTaRG wurde von sechs ehemaligen und zukünftigen FoSAP-Assistenten mit der Thinking-Aloud-Methode und einem abschließenden fragebogengestützten Interview evaluiert. Das Frontend wurde als verständlich und gut nutzbar bewertet. Dabei wurde die Anzeige einer Beispielaufgabe, obwohl sie statisch ist, als besonders hilfreich hervorgehoben. Anpassungen zur Beseitigung von Darstellungsfehlern wurden vorgenommen und als Reaktion auf weitere Wünsche der Assistenten an das Tool die Funktionalität zur grafischen Erstellung eines einzelnen Automaten hinzugefügt. Weiterhin wurde die Korrektheit der generierten Aufgaben samt Musterlösung geprüft und bestätigt.

AuTaRG wurde als sehr nützlich für den Übungsbetrieb empfunden. Die schnelle Generierung von zufälligen Aufgaben entlastet die Assistenten darin, sich keine neuen Aufgaben ausdenken zu müssen. Das Einsparen von Zeit durch die schnelle Generierung wurde bestätigt. Die implementierten Exportformate sind gut gewählt und umgesetzt. Für Sprechstunden können schnell nach Bedarf Aufgaben und als Anschauungsbeispiele für z. B. die Vorlesung einzelne Automaten generiert werden. Ein Teilnehmer würde beim Export von LaTeX-Code zwei Dateien, eine mit Lösung, eine ohne, bevorzugen, da sich dies besser in das Template des Lehrstuhls integrieren lässt. Dies kann aktuell durch das Kopieren des Lösungscodes in eine neue Datei umgesetzt werden.

6 Fazit und Ausblick

Dieser Beitrag zeigt, wie Automaten und automatenbasierte Aufgaben zufällig automatisiert generiert werden können. Am Beispiel der Aufgaben der Vorlesung Formale Systeme, Automaten, Prozesse (FoSAP) wurden Aufgabentypen und deren Lernziele identifiziert. Anhand von Eigenschaften und Zahlenwerten wurde präsentiert, wie für jeden Aufgabentyp Eignungsklassifikationen für Automaten erstellt wurden. Zur zufälligen Generierung von Automaten wurden verschiedene Generatoren aufgearbeitet und angelehnt an den FIFA Generator ein eigenes Verfahren entwickelt. Dieses Verfahren wurde im Webtool *Automata Task Random Generator* (*AuTaRG)* implementiert und eine proaktive Generierung der Automaten umgesetzt, sodass bei der Aufgabengenerierung aus einem großen Pool zufällig ein Automat ausgewählt wird. AuTaRG wurde in der Nutzerevaluation mit FoSAP-Assistenten als sehr nützlich für den Übungsbetrieb und darüber hinaus empfunden. Die schnelle Generierung von zufälligen Aufgaben entlastet sie darin, sich keine neuen Aufgaben ausdenken zu müssen und spart Zeit ein.

Es ist zunächst nicht geplant, dass Tool Studierenden zur Verfügung zu stellen. Stattdessen werden Questiontypes für Moodle entwickelt, welche die Aufgabentypen als Fragen für Moodle-Quizze mit einer automatisierten Bewertung umsetzen. Während Fragen zur Wortakzeptanz als Multiple-Choice Frage umgesetzt werden, bedarf es bei anderen Aufgaben einer anderen Eingabe z. B. einer grafischen, was in Moodle aktuell nicht möglich ist. Auch das Hinzufügen weiterer Exportformate für die Verwendung in anderen Lernmanagement Systemen ist möglich.

Ein weiteres interessantes Kriterium für die generierten Aufgaben ist die Schwierigkeit, basierend auf den verwendeten Automaten. Dazu können z. B. die Antworten auf die Aufgaben ausgewertet werden. Eine erste Verwendung von AuTaRG im Übungsbetrieb der FoSAP Vorlesung findet im Sommersemester 2020 statt.

Literaturverzeichnis

[Be20] Bergerbusch, T.: Generating Computer Science Exercises with Suitable Automata, Master Thesis, 2020.

[Ch04] Champarnaud, J. et al.: Random Generation Models for NFAs. Journal of Automata, Languages and Combinatorics. 9, 203-216, 2004.

[Fe18] Ferreira, M.; Moreira, N.; Reis, R.: Forward Injective Finite Automata: Exact and Random Generation of Nonisomorphic NFAs. In: Konstantinidis S., Pighizzini G. (eds) Descriptional Complexity of Formal Systems. DCFS 2018. Lecture Notes in Computer Science, vol 10952. Springer, Cham, 2018.

[Ku20] Kurdi, G. et al.: A Systematic Review of Automatic Question Generation for Educational Purposes. International Journal of Artificial Intelligence in Education 30, 121–204, 2020.

[Le95] Leslie, T.: Efficient Approaches to Subset Construction, 1995.