Explorative Datenanalyse der Studierendenperformance in der Theoretischen Informatik

Christiane Frede, Maria Knobelsdorf
Universität Hamburg, Fachbereich Informatik
Vogt-Kölln-Straße 30
22527 Hamburg
frede@informatik.uni-hamburg.de

Universität Wien, Zentrum für LehrerInnenbildung/Fakultät für Informatik
Währinger Straße 29
1090 Wien
maria.knobelsdorf@univie.ac.at

Abstract: In diesem Artikel werden die Ergebnisse einer explorativen Datenanalyse über die Studierendenperformance in Klausur- und Hausaufgaben eines Einführungskurses der Theoretischen Informatik vorgestellt. Da bisher empirisch wenig untersucht ist, welche Probleme Studierenden in den Einführungskursen haben und die Durchfallquoten in diesen Kursen sehr hoch sind, soll auf diesem Weg ein Überblick gegeben werden. Die Ergebnisse zeigen, dass alle Studierenden unabhängig von ihrer Klausurnote die niedrigste Performance in den Klausur- und Hausaufgaben aufweisen, in denen formale Beweise gefordert sind. Dieses Ergebnis stärkt die Vermutung, dass didaktische Ansätze und Maßnahmen sich insbesondere auf das Erlernen formaler Beweismethoden fokussieren sollten, um Informatik-Studierende nachhaltiger dabei zu unterstützen, in Theoretischer Informatik erfolgreich zu sein.

Keywords: Theoretische Informatik, Explorative Datenanalyse, Formale Sprachen und Automaten, Studierendenperformance, Beweisaufgaben.

1 Einleitung und Verwandte Arbeiten

Die mathematischen und theoretischen Fachanforderungen in Einführungsveranstaltungen der Informatik-Bachelorstudiengänge stellen in Deutsch-

land nach wie vor eine große Herausforderung dar. Im Jahrgang 2010/2011 brachen 45 % aller Studierenden an deutschen Hochschulen ihr Studium u. a. aufgrund von Leistungsproblemen ab [He17]. Dies zeichnet sich vor allem in der Studieneingangsphase ab, in der die Abbruch- und Durchfallquoten in entsprechenden Einführungsmodulen sehr hoch sind. An vielen Standorten gilt dies insbesondere für Einführungskurse der Theoretischen Informatik. Es ist nicht ungewöhnlich, dass gerade hier die überwiegende Mehrheit der Informatik-StudienanfängerInnen mittelmäßige bis schlechte Leistungen erzielt und nur eine kleine Teilgruppe der Gesamtkohorte bei der Abschlussklausur sehr gute Ergebnisse erreicht. Es stellt sich daher die Frage, was die hohen Abbruch- und Durchfallquoten in Theorie-Kursen bedingt. Aus Sicht der Lehrenden werden oftmals folgende Vermutungen geäußert: Die Studierenden investieren nicht genug Zeit, sich mit den Themen und den Übungsaufgaben auseinanderzusetzen, sie sind nicht genügend motiviert, interessiert oder intelligent genug, um die Themen und Fachkonzepte zu durchdringen und zu verstehen (vgl. [CGM04][Pi10][SM11]). Diese Annahmen basieren in der Regel auf Beobachtungen der Lehrpersonen und Einzelgesprächen mit beteiligten Studierenden und TutorInnen, jedoch selten auf gesicherten Daten (vgl. [CGM04][Sc13). Aufbauend auf diesen Annahmen wurden verschiedene fachdidaktische Ansätze vorgestellt, um die Eingangslehre zu verbessern, z. B. [CGM04], [Ko07], [Sc13]. Diese Ansätze arbeiten mit teils sehr unterschiedlichen didaktischen Konzepten und Methoden, sind jedoch sehr erfolgreich in der Gestaltung einer konstruktivistischen Lehr-Lern-Umgebung. Zusammenfassend versuchen diese Ansätze, die Lehre der Theoretischen Informatik praktischer und anwendungsorientierter zu gestalten, um die Motivation der Studierenden für die mathematischen und abstrakten Themen zu erhöhen. Weitere existierende Einzelfallstudien [KF16][PL14] bieten hingegen Einblicke in die tatsächlichen Schwierigkeiten der Studierenden, ohne sich dabei auf ihre Fähigkeiten zur Selbstauskunft zu verlassen. Knobelsdorf und Frede (2016) haben Studierendengruppen dabei beobachtet, wie sie einen Reduktionsbeweis zur NP-Vollständigkeit bearbeitet haben [KF16]. Während dieser Studie hatten die Studierenden beispielsweise das Konzept der NP-Vollständigkeit durchaus verstanden, aber waren nicht in der Lage, den für die Aufgabe benötigten formalen Reduktionsbeweis zu entwickeln. Auch Armoni (2009) hat eine Studie zu Reduktionsbeweisen durchgeführt. Durch eine Dokumentenanalyse über Studierendenergebnisse hat er herausgefunden, dass es für Studierende schwierig war, Reduktion als eine Problemlösestrategie zu fassen, die vielseitig anwendbar ist [Ar09]. In einer anderen Studie analysierte Pillay (2010) drei Tests und wöchentliche Übungen

von dreizehn Studierenden [Pi10]. Da die Studierenden auch hier die größten Schwierigkeiten während des Problemlöseprozesses hatten, schlägt sie als didaktische Maßnahme vor, Studierende während der verschiedenen Schritte dieses Prozesses mittels Experten-Feedback dezidierter zu unterstützen. Die genannten Studien haben den Nachteil, dass sie sich nur auf einzelne Themen der zugehörigen Kurse beschränken. Eine umfassendere Analyse darüber, mit welchen Themen Studierende in einem Kurs über Algorithmen und Datenstrukturen besondere Probleme haben, hat Enström (2014) vorgelegt [En14]. Hierzu hat sie ein automatisiertes Beurteilungssystem für Studien verwendet und weitere Informationen aus Fragebögen, mündlichem Feedback der Studierenden und ihren Noten gewonnen. Als Ergebnis hält sie fest, dass die Studierenden Schwierigkeiten mit vielen Beweisaufgaben hatten, wobei explizit die Komplexitätsbeweise eine große Herausforderung darstellten. Bis auf die Arbeit von Enström beschränken sich alle erwähnten Studien auf ein oder wenige ausgewählte Themen, Konzepte und fachspezifische Kompetenzen aus dem entsprechenden Einführungskurs. Aus diesem Grund können diese Arbeiten keinen detaillierten und differenzierten Überblick darüber geben, welche Themen von allen Themen des entsprechenden Einführungskurses den Studierenden die meisten Probleme bereiten.

Dieser Artikel soll einen dezidierten Überblick darüber geben, welche Themen der Theoretischen Informatik den Studierenden die meisten Schwierigkeiten bereiten. Dafür wurde eine Studie durchgeführt, welche die Gesamtleistung der Studierenden untersucht. Als Datenquellen wurden die Ergebnisse aus den Hausaufgaben und der Abschlussklausur in einem Einführungskurs der Theoretischen Informatik von ca. 500 Informatik-Studierenden an der Technischen Universität Berlin aus dem Wintersemester 2016/2017 verwendet. Diese Daten wurden mittels explorativer Datenanalyse untersucht, die Einblicke in die Studierendenperformance liefert. Im folgenden Abschnitt wird das Forschungsdesign vorgestellt, sowie notwendige Informationen über den Aufbau des Kurses und die Datenquellen erläutert (Abschnitt 2). Anschließend werden die Ergebnisse präsentiert und interessante Erkenntnisse diskutiert (Abschnitt 3), bevor der Artikel in Abschnitt 4 mit einem Fazit schließt.

2 Forschungsdesign

In diesem Abschnitt wird der Forschungsansatz vorgestellt, die dafür verwendeten Datenquellen, sowie die Besonderheiten über den Aufbau der Lehrveranstaltung, aus der unsere Daten hervorgingen.

2.1 Forschungsmethode

Der Begriff der Explorativen Datenanalyse (EDA) wurde in den 1970er Jahren von Tukey vorgestellt [Tu77][TMH00]. Im Zuge dieses Forschungsansatzes werden oftmals leicht zugängliche Stichproben von Daten analysiert, um dadurch offene Fragestellungen zu beantworten. Über das betrachtete Forschungsfeld liegen dabei bisher keine Hypothesen, Modelle oder detaillierte Einblicke vor. Die EDA fasst hierbei verschiedene Techniken und Vorschläge zusammen, um die vorhandenen Daten zu strukturieren und zu untersuchen. Im Gegensatz zur Hypothesenüberprüfung, bei der Daten speziell auf die für das Testen der Hypothesen notwendigen Daten beschränkt werden, wird bei der EDA versucht, einen möglichst vollständigen und detaillierten Überblick des Datensatzes zu geben [DB16]. Um sich einem Datensatz zu nähern, können u. a. zusammenfassende Statistiken [CH86][Mo09] verwendet werden. Diese ermöglichen einen Überblick über die allgemeine Datenstruktur, wodurch erste Auffälligkeiten sichtbar werden können. Darauf aufbauend können anschließend weitere Analysen durchgeführt werden. Auch Diagramme können je nach Datensatz und Fragestellungen ein nützliches Werkzeug sein, um einen Überblick über die zusammengefassten Daten zu geben und Besonderheiten visuell darzustellen. Die EDA kann als Kombination aus verschiedenen Erkundungstechniken verwendet werden, um sich einem wenig erforschten Forschungsfeld zu nähern und die Grundlage für Hypothesen zu bilden, auf denen weiterführende Analysen durchgeführt werden können.

Für die statistische Analyse dieser Studie wurde die Software SPSS[1] verwendet. Während der explorativen Datenanalyse wurden die Daten zusammengefasst, um einen Überblick über die Datenstruktur zu bekommen und Auffälligkeiten zu erkennen. Diese wurden mit Hilfe von Tabellen und Diagrammen visualisiert. Zusätzlich dazu wurde Qualitative Inhaltsanalyse [Ma10] verwendet, um die Hausaufgaben nach Aufgabentypen zu kategorisieren.

2.2 Aufbau des Kurses und Datenquellen

Die für die Studie verwendeten Daten stammen aus einem Einführungskurs zur Theoretischen Informatik für Bachelorstudierende der Informatik der Technischen Universität Berlin, an dem ca. 500 Studierende teilgenommen

1 https://www.ibm.com/products/spss-statistics, zugegriffen am 27. 06. 2018

haben und der im Wintersemester 2016/2017 stattgefunden hat. Es wurden nicht reaktiv erhobene Daten verwendet, um eine Studierendenperformance zu untersuchen, wie sie in vergleichbaren Kursen an deutschen Universitäten auftritt. Als Indikatoren für die Performance wurden die bearbeiteten Hausaufgaben und die Ergebnisse der Abschlussklausur verwendet. Die Hausaufgaben geben einen Überblick über die Lernleistung während des Kurses, während die Klausurergebnisse ein möglicher Indikator für die bis dahin entwickelten bzw. erreichten Fachkompetenzen darstellt.

Der betrachtete Einführungskurs zur Theoretischen Informatik behandelt die Themen Formale Sprachen und Automaten und ist für alle Informatik-Studierenden verpflichtend, während die Anwesenheit in Vorlesung und Übungsgruppen nicht verpflichtend ist. Parallel dazu nahmen die Studierenden an einem verpflichtenden Mathematikkurs über Lineare Algebra und Analysis teil. Um den Kurs zu bestehen, mussten die Studierenden sog. Portfolio-Punkte sammeln. Dafür konnten sie verteilt über das Semester vier Blöcke von Hausaufgaben (insgesamt 31 Aufgaben) in Kleingruppen bearbeiten und zur Prüfung abgeben. Jeder Hausaufgabenblock konnte maximal fünf Punkte zum Portfolio beitragen, so dass insgesamt 20 Punkte durch Hausaufgaben zu erreichen waren. Weitere 30 Punkte konnten über einen online Multiple-Choice Test in der Mitte des Semesters gesammelt werden. Die sechs Aufgaben der Abschlussklausur am Ende des Semesters konnten bis zu 50 Punkte zum Portfolio beitragen. Das Modul galt als bestanden, wenn am Ende insgesamt mehr als 49 Portfolio-Punkte erreicht wurden.

Der erste Hausaufgabenblock wurde von 571 Studierenden abgegeben, während sich die Zahl der Abgaben für den vierten und damit letzten Hausaufgabenblock auf 491 verringert hatte. Es wurden auch nicht vollständige Hausaufgabenblöcke in der Analyse berücksichtigt, da fast alle Studierendengruppen zwischendurch immer wieder einzelne Aufgaben nicht bearbeitet haben. Die Abgaben wurden dennoch einbezogen, da keine Daten über die Gründe erhoben wurden, warum die Studierenden bestimmte Aufgaben nicht bearbeitet haben. Mangelnde Zeit, mangelndes Verständnis oder mangelndes Interesse sind nur einige der möglichen Gründe.

Unabhängig von den Hausaufgaben haben 419 Studierende an der Abschlussklausur teilgenommen. Die Wiederholungsklausur vier Wochen später wurde von sechzehn Studierenden bearbeitet, wobei dies für acht Studierende der erste Versuch war. Damit haben 427 Studierende der ursprünglichen 571 Studierenden an einer der beiden Klausuren teilgenommen (75 %). Von diesen 427 Studierenden, haben 295 (69 %) mit mehr als 49 % der Punkte bestanden. Nach den endgültigen Portfoliopunkten gerechnet, haben 339 (59 %)

von den ursprünglichen 571 Studierende den Kurs bestanden und 232 Studierende (41 %) nicht.

Die Daten wurden nach Absprache mit der Datenschutzbeauftragten der Technischen Universität Berlin nur anonymisiert verwendet, so dass kein Rückschluss auf einzelne Personen möglich war und ist. Außerdem waren die Autorinnen in keiner Form in die Lehrveranstaltung involviert.

3 Ergebnisse und Diskussion

Im Folgenden werden zuerst die Ergebnisse der Abschlussklausur vorgestellt und diskutiert, um mögliche Auffälligkeiten in der Studierendenperformance zu erkennen. Die Analyse der Performance wird anschließend mit den Hausaufgaben weitergeführt. Während dieser explorativen Datenanalyse werden die einzelnen Schritte der Analyse einzeln diskutiert, bevor anschließend die nachfolgenden Schritte präsentiert und diskutiert werden.

3.1 Ergebnisse der Abschlussklausur

Zu Beginn der Analyse wurden die Klausurergebnisse der Studierenden betrachtet. Abbildung 1 zeigt die Verteilung der erreichten Punkte auf die 419 Studierenden, die an der Abschlussklausur teilgenommen haben.

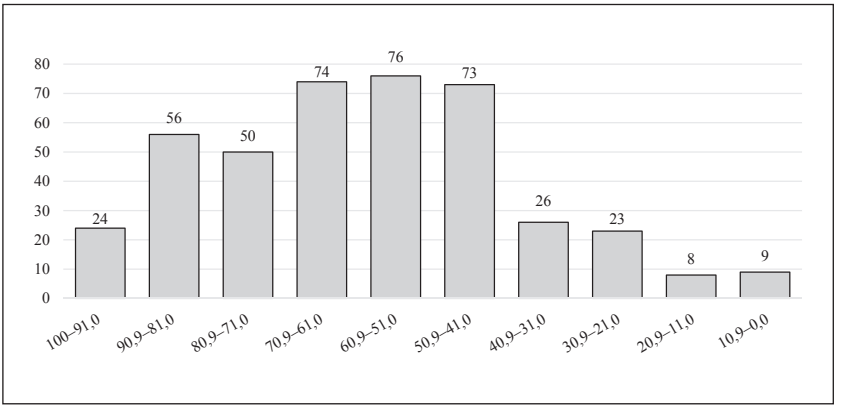

Abb. 1: Verteilung der erreichten Punkte in der Abschlussklausur: Die x-Achse zeigt den Punktebereich, die y-Achse die Anzahl der Studierenden.

Die mehr oder weniger typische Gaußkurve der Klausurergebnisse lässt annehmen, dass die Schwierigkeit der Abschlussklausur angemessen war. Um die Notenverteilung darzustellen, wird der allgemeine Notenschlüssel des Fachbereichs Informatik der Technischen Universität Berlin verwendet:

- Note 1: Notwendige Punkte 100–81. Erreicht von 80 Studierenden.
- Note 2: Notwendige Punkte 80–70. Erreicht von 67 Studierenden.
- Note 3: Notwendige Punkte 69–59. Erreicht von 88 Studierenden.
- Note 4: Notwendige Punkte 58–50. Erreicht von 61 Studierenden.
- Note 5: Unter 50 Punkten. Erreicht von 123 Studierenden.

Von den insgesamt 419 Studierenden haben 80 Studierende eine sehr gute Note erzielt, während 216 Studierende eine gute bis ausreichende Leistung erreicht haben. Um die Frage zu beantworten, ob sich diese Gesamtleistung in den Abschlussklausuren auch in den Einzelaufgaben wiederspiegelt, wurden die sechs Klausuraufgaben detaillierter betrachtet. Hierfür wurden die Studierenden abhängig von ihrer Note in der Abschlussklausur in die Gruppen 1 bis 5 aufgeteilt. Da für jede Aufgabe eine unterschiedliche maximale Punktzahl erreicht werden konnte, wurde diese Punktzahl in Prozentwerte umgerechnet. Anschließend konnte die durchschnittliche Prozentzahl der erreichten Punkte pro Notengruppe und Aufgabe berechnet werden, so dass die Performance in den einzelnen Aufgaben miteinander verglichen werden kann. Tab. 1 zeigt die Werte pro Notengruppe und Aufgabennummer und K1 bis K6 stellen die sechs Klausuraufgaben dar.

Tab. 1: Durchschnittlich erreichte Punkte in Prozent für jede Klausuraufgabe nach Notengruppe

Gruppe / Aufgabe	1	2	3	4	5
K1	91 %	86 %	78 %	74 %	43 %
K2	95 %	90 %	82 %	79 %	44 %
K3	95 %	89 %	81 %	78 %	44 %
K4	74 %	52 %	29 %	16 %	5 %
K5	89 %	64 %	43 %	27 %	10 %
K6	76 %	54 %	41 %	32 %	14 %

In Tab. 1 ist wie erwartet auf den ersten Blick zu erkennen, dass Gruppe 1 in jeder der sechs Klausuraufgaben die höchsten Werte hat. Die Gruppen 2 und 3 haben durchschnittliche Werte, während die Gruppen 4 und 5 die niedrigsten Werte aufweisen. Auf den zweiten Blick ist jedoch zu erkennen, dass die Gruppen 1 bis 4 in den ersten drei Aufgaben gut bis sehr gut abgeschnitten haben, während in den Aufgaben K4, K5 und K6 ein z. T. deutlicher Punktabfall für alle Notengruppen sichtbar ist. Um genauer zu verstehen, was die Aufgaben K4, K5 und K6 von den ersten drei Aufgaben unterscheidet und was der Grund für den Punktabfall sein könnte, müssen die Themen und Aufgabenstellungen der Klausuraufgaben betrachtet werden:

- K1: Das Thema war reguläre Sprachen. Angeben eines nicht-deterministischen Automaten (NFA), einer Grammatik für eine gegebene Sprache und Ableitungen von Wörtern.
- K2: Das Thema war Automaten. Es war notwendig, einen deterministischen Automaten (DFA) von einem gegebenen NFA zu konstruieren.
- K3: Minimierung eines DFA. Es war notwendig den Table-filling-Algorithmus anzuwenden, um einen DFA zu minimieren. Außerdem sollten Äuqivalenzklassen angegeben und der DFA visualisiert werden.
- K4: Reguläre Sprachen. Einen Beweis mit Hilfe des Pumping Lemmas entwickeln. Myhill-Nerode Äquivalenzklassen für eine Sprache angegeben.
- K5: Kontextfreie Sprachen. Es musste eine Typ-2-Grammatik und ein Kellerautomat (PDA) angegeben werden.
- K6: Kontextfreie Sprachen. Es mussten Ableitungen eines PDA angegeben werden und ein Beweis geführt werden, dass eine Sprache nicht regulär ist.

Werden nun die letzten drei mit den ersten drei Aufgaben verglichen, ist festzustellen, dass es in K4 und K6 u. a. notwendig war einen formalen Beweis zu entwickeln. In K1, K2 und K3 hingegen mussten Grammatiken und Wörter angegeben, Automaten konstruiert sowie spezifische Algorithmen angewendet werden. Hierbei wurde mit vorgegebenen Sprachen und Grammatiken und oftmals an einem konkreten Beispiel gearbeitet. Für alle Notengruppen ist K4 die Aufgabe mit dem höchsten Punktabfall. Hier mussten die Studierenden das Pumping Lemma im Rahmen eines formalen Beweises anwenden. Daraus lässt sich insgesamt schließen, dass alle Studiereden unabhängig von ihrer Klausurnote Schwierigkeiten mit dieser Art der Aufgabenstellung hatten. Dieses Ergebnis unterstützt die Ergebnisse anderer Studien, dass Studierende besondere Schwierigkeiten mit Beweisaufgaben haben (vgl. Abschnitt 2).

3.2 Hausaufgabenperformance

Ausgehend von den Ergebnissen aus Abschnitt 3.1 stellte sich die Frage, ob die Studierenden schon vor der Abschlussklausur in den 31 Hausaufgaben Schwierigkeiten hatten, formale Beweis zu entwickeln. Wie im vorherigen Abschnitt wird auch hier bei den Hausaufgabenpunkten mit Prozentwerten gearbeitet und die Verteilung der Punkte bezogen auf die Notengruppe in der Klausur berechnet (Tab. 2).

Auch wenn der Unterschied zwischen den einzelnen Notengruppen dieses Mal geringer ausfällt, ist auch in Tab. 2 zu erkennen, dass Gruppe 1 in jeder Aufgabe die beste Performance aufweist. Insgesamt sind die auffälligsten Punktabfälle für die Aufgaben A2, A5, A6, A7, A8, A10, A15, A20, A21, A24 und A29 zu sehen (die entsprechenden Zeilen sind in Tab. 2 fett markiert). Bei diesen elf Aufgaben liegt der Punktabfall für jede Notengruppe bis zu 30 % unter der sonstigen durchschnittlichen Performance pro Gruppe. Um auch zwischen diesen Aufgaben Unterschiede und Gemeinsamkeiten erkennen zu können, wurden die Themen und Aufgabenstellungen dieser elf einzelnen Aufgaben analysiert und zusammengefasst dargestellt:

- A2: Mengen. Beweisen von Eigenschaften.
- A5: Aussagenlogik. Beweisen von Variablenbelegungen.
- A6: Prädikatenlogik. Eine gegebene Aussage für zwei Prädikate beweisen.
- A7: Prädikatenlogik. Die ersten Schritte für einen Widerspruch angeben.
- A8: Mengen. Einen Induktionsbeweis über eine Zahlenmenge führen.
- A10: Relationen und Funktionen. Ordnungen beweisen.
- A15: Äquivalenzklassen. Beweisen, dass zwei Variablen äquivalent sind.
- A20: Wörter und Sprachen. Induktionsbeweis über ein gegebenes Alphabet und enthaltende Wörter.
- A21: Grammatiken. Ableitungen für Wörter und Sprachen angeben.
- A24: Reguläre Sprachen: Einen Beweis mit dem Pumping Lemma entwickeln.
- A29: Reguläre Sprachen: Alle Myhill Nerode-Äquivalenzklassen angeben.

Bei einem ersten Vergleich der Aufgaben wird deutlich, dass ein formaler Beweis in acht von elf Aufgaben gefordert ist. Nur in den Aufgaben A7, A21 und A29 musste kein formaler Beweis entwickelt werden. Thematisch hingegen werden verschiedene Themen abgedeckt. In Aufgabe A24 war wie in Klausuraufgabe K4 ein Beweis mit Hilfe des Pumping Lemmas gefordert.

Tab. 2: Durchschnittlich erreichte Prozente für jede Hausaufgabe aufgeteilt nach Notengruppe

Gruppe / Aufgabe	1	2	3	4	5
A1	95 %	94 %	91 %	90 %	86 %
A2	**77 %**	**69 %**	**69 %**	**57 %**	**70 %**
A3	92 %	86 %	87 %	83 %	90 %
A4	85 %	85 %	83 %	82 %	79 %
A5	**80 %**	**69 %**	**63 %**	**51 %**	**56 %**
A6	**50 %**	**46 %**	**46 %**	**40 %**	**41 %**
A7	**69 %**	**59 %**	**64 %**	**53 %**	**56 %**
A8	**79 %**	**79 %**	**72 %**	**65 %**	**68 %**
A9	92 %	92 %	84 %	83 %	85 %
A10	**77 %**	**72 %**	**69 %**	**65 %**	**68 %**
A11	86 %	80 %	74 %	72 %	71 %
A12	87 %	82 %	69 %	70 %	63 %
A13	97 %	96 %	85 %	88 %	87 %
A14	84 %	74 %	64 %	58 %	63 %
A15	**66 %**	**46 %**	**45 %**	**43 %**	**35 %**
A16	83 %	70 %	67 %	64 %	59 %
A17	98 %	94 %	89 %	89 %	91 %
A18	90 %	77 %	75 %	66 %	68 %
A19	99 %	91 %	90 %	84 %	87 %
A20	**72 %**	**62 %**	**60 %**	**49 %**	**50 %**
A21	**77 %**	**64 %**	**64 %**	**54 %**	**58 %**
A22	92 %	82 %	79 %	71 %	71 %
A23	82 %	78 %	73 %	63 %	66 %
A24	**82 %**	**70 %**	**65 %**	**52 %**	**63 %**
A25	97 %	86 %	86 %	80 %	80 %
A26	89 %	78 %	77 %	72 %	76 %
A27	99 %	91 %	88 %	79 %	85 %
A28	99 %	93 %	92 %	88 %	50 %
A29	**76 %**	**62 %**	**57 %**	**49 %**	**30 %**
A30	86 %	75 %	67 %	47 %	35 %
A31	99 %	93 %	88 %	82 %	50 %

Bei A24 konnte zwar ein starker Punktabfall registriert werden, doch dieser war nicht so deutlich wie bei K4. Auch hier kann es verschiedene Gründe für den Unterschied geben, für die in dieser Analyse keine Daten erhoben wurden, z. B. Hausaufgabenabgabe in Gruppen, weniger Zeitdruck als bei der Abschlussklausur.

Zusätzlich stellt sich nun die Frage, was die Aufgaben mit Punktabfall von denen unterscheidet, die eine höhere Performance aufweisen. Aus Platzgründen werden die Informationen über Themen und Aufgabentyp für alle Aufgaben zusammengefasst dargestellt (Tab. 3). Die Aufgaben mit auffälligem Punktabfall sind erneut fett markiert. Um die Aufgaben einem Thema zuzuordnen, wurden die korrespondieren Themen aus der Formelsammlung des Kurses verwendet. Durch Anwendung einer zusammenfassenden qualitativen Inhaltsanalyse angelehnt an Mayring [Ma10] wurden die Aufgaben außerdem einem der folgenden Aufgabentypen zugeordnet: *Beweisen* (Entwickeln eines Beweises), *Angeben* (von z. B. Ableitungen, Sprachen, Relationen, Grammatiken, etc.), *Konstruieren* (von Automaten), *Berechnen* (von z. B. Mengen). Diese ersten Kategorien bilden eine abstrakte Zusammenfassung von möglichen Gemeinsamkeiten der Aufgabenstellungen.

Da auch hier drei Beweisaufgaben (A3, A4, A12) zu finden sind, stellt sich die Frage, was diese Beweisaufgaben von denen unterscheidet, bei denen ein auffälliger Punktabfall zu beobachten war. In A3 und A4 war die Entwicklung eines Beweises gefordert, der als schematisch und weniger formal bewertet werden kann (Wahrheitstabellen, Äquivalenzumformungen). In Aufgabe A12 sollte Kardinalität bewiesen werden. Die hier betrachteten Daten können allerdings nicht verwendet werden, um eine Aussage darüber zu treffen, was A12 von den anderen Beweisaufgaben unterscheidet. Der einzige ersichtliche Unterschied besteht darin, dass A12 der einzige Beweis zu dem Thema „Funktionen/Abbildungen“ war. Hier ist eine genauere Analyse der verschiedenen Beweisarten nötig.

Insgesamt wurden elf Aufgaben mit auffälligem Punktabfall erkannt. Von diesen elf Aufgaben war in acht Aufgaben eine formale Beweisentwicklung gefordert. Beweisart und Thema unterschieden sich zum Großteil für die einzelnen Aufgaben. In zwei der drei Beweisaufgaben, in denen die Performance der Studierenden höher war, musste ein schematischer und weniger formaler Beweis entwickelt werden. Zusammenfassend lässt sich sagen, dass sich die geringe Performance in den Beweisaufgaben der Klausur auch in den Hausaufgaben wiederfinden lässt.

Tab. 3: Kategorisierung der Hausaufgaben ohne auffälligen Punktabfall

Aufgabe	Thema (Formelsammlung)	Aufgabentyp
A1	Mengen	Berechnen
A2	**Mengen**	**Beweisen**
A3	Aussagenlogik	Beweisen
A4	Aussagenlogik	Beweisen
A5	**Aussagenlogik**	**Beweisen**
A6	**Prädikatenlogik**	**Beweisen**
A7	**Mengen**	**Angeben**
A8	**Mengen**	**Beweisen**
A9	Relationen/Ordnungen	Angeben
A10	**Relationen/Ordnungen**	**Beweisen**
A11	Funktionen/Abbildungen	Angeben
A12	Funktionen/Abbildungen	Beweisen
A13	Relationen/Ordnungen	Angeben
A14	Relationen/Ordnungen	Angeben
A15	**Funktionen/Abbildungen**	**Beweisen**
A16	Funktionen/Abbildungen	Angeben
A17	Wörter/Sprachen	Angeben
A18	Wörter/Sprachen	Angeben
A19	Wörter/Sprachen	Angeben
A20	**Wörter/Sprachen**	**Beweisen**
A21	**Grammatiken**	**Angeben**
A22	Grammatiken	Angeben
A23	Grammatiken	Angeben
A24	**Reguläre Sprachen**	**Beweisen**
A25	Automaten	Angeben
A26	Automaten	Konstruieren
A27	Minimierung von Automaten	Angeben
A28	Minimierung von Automaten	Konstruieren
A29	**Reguläre Sprachen**	**Angeben**
A30	Reguläre Sprachen	Angeben
A31	Minimierung von Aut.	Angeben

4 Fazit und Zusammenfassung

In diesem Artikel wurde explorative Datenanalyse verwendet, um die Leistung von Informatik-Studierenden in einem Einführungskurs der Theoretischen Informatik differenzierter zu bewerten. Für die Analyse wurden die Studierenden abhängig von ihrer Note in der Abschlussklausur in die Gruppen 1 bis 5 aufgeteilt. Dadurch wurden interessante Muster und Auffälligkeiten in den Klausurergebnissen und Hausaufgaben der Studierenden entdeckt. Zusammenfassend gab es für alle Gruppen unabhängig von ihrer Note in der Abschlussklausur eine schlechtere Performance in denselben Klausur- und Hausaufgaben. Dies waren zum Großteil Aufgaben, in denen ein formaler Beweis gefordert war, während die Aufgaben sich thematisch unterschieden. Damit zeigt die Analyse der vorliegenden Daten, dass in einem Einführungskurs der Theoretischen Informatik formale Beweise unabhängig vom Thema oder zugrundeliegendem Fachkonzept für alle Informatik-Studierenden die größten Herausforderungen darstellen. Die Leistung der Studierenden ist dabei umso schlechter, je formaler der Beweis ist. Dieses Ergebnis unterstützt die diskutierten Ergebnisse der Einzelfallstudien aus Abschnitt 2. Es deutet außerdem darauf hin, dass die Studierenden nicht per se Schwierigkeiten mit Themen der Theoretischen Informatik als solcher haben, sondern mit formalen Beweistechniken. Fachdidaktische Ansätze und Maßnahmen zur Senkung von Durchfallquoten in entsprechenden Lehrveranstaltungen, sollten sich daher explizit auf die Vermittlung und intensive Einübung formaler Beweismethoden fokussieren. Dies steht im Gegensatz zu den bisherigen Ansätzen, die vorschlagen, dass die theoretischen Konzepte praktischer und anwendungsorientierter gelehrt werden sollten.

Danksagung

Wir möchten uns bei Prof. Dr. Uwe Nestmann und seiner Arbeitsgruppe „Modelle und Theorie Verteilter Systeme“ am Institut für Softwaretechnik und Theoretische Informatik der Technischen Universität Berlin für die Unterstützung unserer Studie bedanken.

Literaturverzeichnis

[Ar09] Armoni, M.: Reduction in CS: A (mostly) quantitative analysis of reductive solutions to algorithmic problems. Journal on Educational Resources in Computing (JERIC) 8.4: 11, 2009.

[CGM04] Chesñevar, C.; González, M.; Maguitman, A.: Didactic strategies for promoting significant learning in formal languages and automata theory. In: Proceedings of the 9th annual SIGCSE conference on Innovation and technology in computer science education, ITiCSE '04, S. 7–11, 2004.

[Ch86] Chatfield, C.: Exploratory data analysis. European journal of operational research 23.1: 5–13, 1986.

[DB16] Döring, N.; Bortz, J.: Forschungsmethoden und Evaluation, 5. Auflage, Springer, 2016.

[En14] Enström, E.: On difficult topics in theoretical computer science education. Diss. KTH Royal Institute of Technology. 2014.

[He17] Heublein, U. et al.: Zwischen Studienerwartungen und Studienwirklichkeit. DZHW, Hannover, 2017.

[HMU01] Hopcroft, J. E.; Motwani, R.; Ullman J. D.: Introduction to Automata Theory, Languages and Computation. Addison-Wesley, 2. Auflage, 2001.

[KF16] Knobelsdorf, M.; Frede, C.: Analyzing Student Practices in Theory of Computation in Light of Distributed Cognition Theory. In: Proceedings of the Conference on International Computing Education Research (ICER '16). S. 73–81, ACM, 2016.

[Ko07] Korte, L. et al.: Learning by Game-building: A Novel Approach to Theoretical Computer Science Education. In: SIGCSE Bull. 03/07, S. 53–57, 2007.

[Ma10] Mayring, P.: Qualitative Content Analysis. Forum: Qualitative Social Research 2/2010, http://www.qualitative-research.net/index.php/fqs/article/view/1089, Abrufdatum: 19. 05. 2018.

[Mo09] Morgenthaler, S.: Exploratory data analysis. Wiley Interdisciplinary Reviews: Computational Statistics 1.1: 33–44, 2009.

[Pi10] Pillay, N.: Learning difficulties experienced by students in a course on formal languages and automata theory. SIGCSE Bulletin 41.4: 48–52, 2010.

[PL14] Parker, M.; Lewis, C.: What Makes big-O Analysis Difficult: Understanding How Students Understand Runtime Analysis. J. Comput. Sci. Coll., 04/14, S. 164–174, 2014.

[Sc13] Schäfer, A. et al.: From boring to scoring – a collaborative serious game for learning and practicing mathematical logic for computer science education. Computer Science Education 23: 87–111, 2013.

[SM11] Schulmeister, R.; Metzger, Ch.: Der Workload im Bachelor: Zeitbudget und Studierverhalten. Eine empirische Studie. Waxmann. Münster, 2011.

[Tu77] Tukey, J. W.: Exploratory data analysis. Vol. 2, 1977.

[TMH00] Tukey, J. W.; Mosteller, F.; Hoaglin, D. C.: Understanding robust and exploratory data analysis. Wiley Classics Library ed New York, 2000.