Diproche – Entwurf, Umsetzung und Erprobung eines automatischen Systems zur Unterstützung des Beweisenlernens bei StudienanfängerInnen

Merlin Carl

Inhaltsverzeichnis

1 Vorwort **5**

1.1 Danksagungen 7

2 Problemstellung **9**

2.1 Zur Rolle und Bedeutung von Beweisen in der Mathematik 9

2.1.1 Zur Relevanz des mathematischen Beweisens in der Lehramtsausbildung 14

2.2 Überlegungen zum Übungsbetrieb zu universitären mathematischen Lehrveranstaltungen 17

2.3 Vorläufige Zielsetzung 20

2.4 Formale Mathematik und automatische Beweisprüfer 21

2.5 Automatische Beweisprüfer in didaktischen Kontexten 22

3 Abgrenzung zu einigen bestehenden oder in Entwicklung befindlichen Systemen **25**

3.1 Diskussion einiger Systeme und Projekte zum natürlichsprachlichen Beweisprüfen und zum computergestützten Erwerb von Beweiskompetenzen 27

3.1.1 VIP 27

3.1.2 Naproche 28

3.1.3 SAD 32

3.1.4 E-Proofs 33

3.1.5 ETPS 36

3.1.6 Edukera 36

3.1.7 Concludio 38

3.1.8 GEOLOG und GEOBEWEIS 39

3.1.9 QED-Tutrix 40

3.1.10 Advanced Geometry Tutor 43

3.1.11 Terence Taos QED 43

3.1.12 Lehrprojekte unter direkter Verwendung professioneller Beweisprüfer bzw. Beweisassistenten 44

3.1.13 Vellemans Proof Designer 47
3.1.14 Lurch 48
3.1.15 Elfe 51
3.2 Der angezielte didaktische Mehrwert von Diproche 52
3.2.1 Erwartete didaktische Vorteile des automatischen Beweisprüfens im Allgemeinen 52
3.2.2 Erwartete didaktische Vorteile durch den Einsatz von Diproche gegenüber anderen Systemen 54

4 Vorüberlegungen zur Machbarkeit: Möglichkeiten und Grenzen 57
4.1 Die naive Vorstellung automatischer Beweisverifikation 58
4.2 Die (vermeintliche) Nutzlosigkeit automatischer Beweisprüfung: Das Argument von Rav 59
4.3 Die Fragwürdigkeit eines einheitlichen Verständnisses von "korrekter Beweistext" 60
4.3.1 Empirische Ergebnisse: Die Studie von Inglis et al. 61
4.4 Die Kontextabhängigkeit des Verständnisses von "korrekter Beweis" 63
4.4.1 Beweisbewertung in Lehrkontexten: Moores Lehren 66
4.5 "Inhaltliche" und "formale" Mathematik 67
4.6 Exkurs: Zur Hermeneutik mathematischer Texte 74
4.6.1 Andere Aspekte des Verstehens 78
4.7 Nachteile eines Beharrens auf einem rein formalen Verständnisses von "korrekter Beweis" 81
4.8 Präzise Zielsetzung 86

5 Funktionen und Aufbau des Systems 89
5.1 Prüfen 89
5.1.1 Sprachliche Prüfung 89
5.1.2 Zielverfolgung 90
5.1.3 Type-Checking 94
5.1.4 Logische Prüfung 95
5.2 Hilfestellung 100
5.3 Diagnose von Fehlschlüssen 103
5.3.1 Feststellbare Fehlschlüsse 105
5.4 Erzeugung von Gegenbeispielen 108
5.5 Ausblick: Beweisanalyse 110
5.6 Integration von Kontexten: Die "Spielwiesen" 113
5.7 Weitere Funktionen: Formalisierungsübungen, "Beweispuzzles" und mehr 115
5.7.1 Beweispuzzles 124
5.7.2 "Markup" 128

5.7.3 Das “Korrekturspiel” 128
5.7.4 “Diagnose” 129
5.7.5 “Vereinfachen” 129
5.7.6 “Reformulate” 131
5.7.7 “DefCheck” 132

6 Das System Diproche 133
6.1 Das Interface 137
6.2 Vorverarbeitung 138
6.3 Parsing und Formalisierung 139
6.3.1 Definite Clause Grammars 139
6.3.2 Der Formelparser 143
6.3.3 Der Textparser 145
6.3.4 Formalisierung 149
6.4 Die Annotation 153
6.5 Ermittlung der Textstruktur 154
6.6 Erzeugung von Beweiserabfragen 155
6.7 Der automatische Beweiser 157
6.7.1 Aussagenlogische ATP-Regeln 158
6.7.2 Quantorenlogische Regeln 159
6.7.3 Gebietsspezifische ATP-Regeln 160
6.7.4 Verifikation von Term- und Gleichungsumformungen 161
6.8 Hinweise für BenutzerInnen 166
6.9 Rückmeldungen 167
6.10 Automatische Erzeugung von Beweisaufgaben 169
6.10.1 Automatische Erzeugung von Beweisaufgaben zur Aussagenlogik 169
6.10.2 Automatische Erzeugung von Beweisaufgaben zur Booleschen Mengenlehre 171
6.10.3 Automatische Erzeugung von Beweisaufgaben zur vollständigen Induktion 172
6.11 Einige “Spielwiesen” 174
6.11.1 Zur Methodik der Implementierung von Spielwiesen und ihrer Validierung 174
6.11.2 Aussagenlogik 176
6.11.3 Boolesche Mengenlehre 177
6.11.4 Gruppentheorie 181
182
6.11.6 Die Implementierung der Zahlentheorie-Spielwiese 183
188
6.12.1 Didaktische Vorzüge der axiomatischen Geometrie 188

7 Die (Eingabe)sprache von Diproche 199
7.1 Vorüberlegungen zur Konstruktion einer geeigneten CNL für Diproche 199
7.2 Die Diproche-CNL . . . 203
7.2.1 Aussagen . . . 203
7.2.2 Annahmen . . . 206
7.2.3 Deklarationen . . . 207
7.2.4 Definitionen . . . 212
7.2.5 Axiome . . . 213
7.2.6 Behauptungen . . . 213
7.2.7 Multiple Behauptungen . . . 214
7.2.8 Multiple bedingte Behauptungen . . . 215
7.2.9 Begründete Behauptungen . . . 216
7.2.10 Annotationen . . . 218
7.3 Formelsyntax . . . 219
7.3.1 Sonstiges . . . 220
7.4 Die Logik von Diproche . . . 220
7.4.1 Implizite Geltungsbereiche von Deklarationen mit inhaltlicher Annahme . . . 224
7.5 Beispieltexte . . . 227
7.5.1 Beispieltexte zur Aussagenlogik . . . 227
7.5.2 Boolesche Mengenlehre . . . 228
7.5.3 Gerade und ungerade Zahlen . . . 229
7.5.4 Teilbarkeit . . . 230

8 Die Einbindung des Systems in die Lehre 233
8.1 Beschreibung des Einsatzes von Diproche im Rahmen der Algebra-Vorlesung an der Europa-Universität Flensburg im Herbstsemester 2020/2021 . . . 234
8.1.1 Einbindung in den Übungsbetrieb . . . 238
8.1.2 Erfahrungen mit Diproche-Übungsaufgaben . . . 239
8.2 Zur Förderung von Kernkompetenzen . . . 262
8.3 Rückmeldungen von Seiten der Studierenden . . . 266
8.3.1 Studentische Rückmeldungen zur Prüfkomponente . . . 266
8.3.2 Studentische Rückmeldungen zu den Mathediktaten . . . 281
8.3.3 Studentische Rückmeldungen zum “Game of Def” . . . 288
8.4 Bearbeitung von Diproche-Aufgaben und Bearbeitungserfolg bei Beweisaufgaben . . . 296
8.4.1 Operationalisierung . . . 297
8.4.2 Hypothesen . . . 300
8.4.3 Durchführung . . . 301
8.4.4 Auswertung . . . 302

8.4.5 Fazit 310

9 Ausblick: Entwicklungsmöglichkeiten 313
9.1 Zur Perspektive des Einsatzes in der Schule 313
9.2 Bewertung von Beweisschritten 313
9.3 Erkennung unnötiger Beweisschritte 314
9.4 Erweiterung der Ausdrucksmittel auf diagrammatische Beweise . . . 315
9.5 Systematisierung der Fehlerdiagnose 315
9.6 Automatisiertes Erkennen neuer Fehlermuster 316
9.7 Interaktive Aspekte der Fehlerdiagnose 318
9.8 Weitere "Spielwiesen" 318
9.9 Andere Sprachen 320
9.10 Verfeinerung und Ergänzung der Eingabesprache 320
9.11 Interaktive Skripte 321
9.12 Verfeinerungen des Tippgebers 321
9.13 Integration von Beweisframes 322
9.14 Einsatz als Untersuchungsmittel für didaktische Fragestellungen . . 324

10 Anhang A: Die Formelsyntax von Diproche 325
10.1 Terme 325
10.1.1 Aussagenlogische Terme 325
10.1.2 Prädikatenlogische Ausdrücke 326
10.1.3 Boolesche Mengenterme 327
10.1.4 Arithmetische Terme 327
10.2 Umformungsketten 328
10.2.1 Folgerungsketten in der Aussagenlogik 329
10.2.2 Ungleichungsketten in der Booleschen Mengenlehre 330
10.2.3 Ungleichungsketten in der Arithmetik 330
10.2.4 Implikationsketten in der Booleschen Mengenlehre 331
10.2.5 Äquivalenzketten in der Arithmetik 332

11 Anhang B: "Game of Def"-Aufgaben auf den Übungsblättern 7 und 8 333
11.1 Blatt 7 333
11.2 Blatt 8 334

12 Anhang C: Fragebögen zur Evaluation der Diproche-Komponenten 337
12.1 Fragebogen zur Prüfkomponente 337
12.2 Fragebogen zu den "Mathediktaten" 340
12.3 Fragebogen zum "Game of Def" 342

Bibliographie **345**

"Ich erkläre hiermit an Eides Statt, dass ich die vorliegende Arbeit selbstständig verfasst und andere als in der Dissertation angegebene Hilfsmittel nicht benutzt habe; die aus fremden Quellen (einschließlich elektronischer Quellen, dem Internet und mündlicher Kommunikation) direkt oder indirekt übernommenen Gedanken sind ausnahmslos unter genauer Quellenangabe als solche kenntlich gemacht. Zentrale Inhalte der Dissertation sind nicht schon zuvor für eine andere Qualifikationsarbeit verwendet worden. Insbesondere habe ich nicht die Hilfe sogenannter Promotionsberaterinnen bzw. Promotionsberater in Anspruch genommen. Dritte haben von mir weder unmittelbar noch mittelbar Geld oder geldwerte Leistungen für Arbeiten erhalten, die im Zusammenhang mit dem Inhalt der vorgelegten Dissertation stehen. Die Arbeit wurde bisher weder im Inland noch im Ausland in gleicher oder ähnlicher Form einer anderen Prüfungsbehörde vorgelegt. Auf die Bedeutung einer eidesstattlichen Versicherung und die strafrechtlichen Folgen einer, auch fahrlässigen, falschen oder unvollständigen eidesstattlichen Versicherung und die Bestimmungen der §§156, 161 StGB bin ich hingewiesen worden."[1]

[1]Promotionsordnung der Europa-Universität Flensburg vom 30. Januar 2017, S. 8.

Kapitel 1

Vorwort

Diese Arbeit wäre nicht entstanden ohne das Zusammenwirken zweier vordergründig recht unterschiedlicher Einflussfaktoren: Einerseits meine aus meiner Arbeit in Peter Koepkes Forschungsgruppe zur mathematischen Logik an der rheinischen Friedrich-Wilhelms-Universität Bonn hervorgegangene Vertrautheit mit dem Naproche-Projekt und -System, andererseits meine 9 Jahre später gesammelten Erfahrungen im Umgang mit AnfängerInnen im Lehramtsstudium an der Abteilung für Mathematik und ihre Didaktik am Institut für mathematische, naturwissenschaftliche und technische Bildung an der Europa-Universität in Flensburg. Mathematik-Didaktik einerseits, mathematische Logik mit Einschlägen aus der Informatik, der formalen Linguistik und der Philosophie der Mathematik andererseits: Diese beiden Bereiche kommen in der vorliegenden Arbeit zusammen. Salopp gesagt: Diese Arbeit ist das, was passiert, wenn ein Logiker beginnt, sich mit Didaktik zu befassen. Ich möchte daher an dieser Stelle all jenen danken, die daran beteiligt waren, das Naproche-Projekt zu initiieren und zu entwickeln: Prof. Dr. Peter Koepke als Initiator von Seiten der Mathematik, Prof. Dr. Bernhard Schröder und Dr. Bernhard Fisseni als Mitwirkende von Seiten der Linguistik sowie Dr. Marcos Cramer, der das System im Rahmen seiner Dissertation wie auch danach wesentlich konzeptionell weiterentwickelte und einen wesentlichen Teil der Implementierung übernommen hat.

Obwohl die vorliegende Arbeit sich wesentlich auf die Ergebnisse und Möglichkeiten der mathematischen Logik stützt, ist sie ein Beitrag zur Didaktik der Mathematik, nicht zur mathematischen Logik. Die Techniken, die in Diproche bei der Übersetzung natürlicher Sprache in formale Repräsentationen, bei der Übersetzung der formalen Repräsentationen in Ziele für einen automatischen Theorembeweiser und bei der logischen Verifikation der Beweisschritte zum Einsatz kommen, sind vereinfachte Varianten der Techniken, die etwa von Klaus Zinns VIP (siehe [217]) oder dem zuletzt hauptsächlich von Marcos Cramer implementierten Naproche-System (siehe [53]) hinlänglich bekannt

sind.[1] Neu ist also nicht die Möglichkeit einer automatischen Verifikation natürlichsprachlicher Texte oder die dazu benutzten Techniken – Verwendung einer kontrollierten natürlichen Sprache, Übersetzung in ein semiformales Zwischenformat, Ermittlung der logischen Abhängigkeitsbeziehungen zwischen den einzelnen Sätzen, Erzeugung formaler Beweisziele und Einsatz eines automatischen Theorembeweisers für deren Überprüfung –, sondern die spezifische Anpassung an einen (hochschul)didaktischen Kontext, genauer die Förderung von Anfängerstudierenden mit wenig mathematischer Vorerfahrung. Dazu gehört insbesondere die Ausarbeitung einer kontrollierten natürlichen Sprache, die für die Bearbeitung von Übungsaufgaben geeignet ist, aber auch eine differenzierte Rückmeldung zu verschiedenen Fehlertypen (u.a. eine Typenverifikation), eine Fehlerdiagnose ("Anti-ATP"), die Erzeugung von Hilfestellungen ("Tippgeber"), die Orientierung des ATP an einer qualifizierten und begründeten Vorstellung dessen, was im Rahmen einer Übungsaufgabe als "elementarer Schritt" gelten kann (und eine dazugehörige Ausdifferenzierung in "Schwierigkeitsgrade"), die Ausarbeitung und Implementierung von thematischen und strategischen Kontexten ("Spielwiesen") sowie des Datentyps "Aufgabe", ferner die Erarbeitung von passendem Übungsmaterial, das mit dem System zusammen verwendet werden kann, sowie dessen praktische Erprobung.

Darüber hinaus stellt die Arbeit einen Beitrag zur Erprobung der Möglichkeiten dar, digitale Medien in der mathematischen Universitätsbildung – und potenziell auch in der Schulbildung – einzusetzen. So ein Einsatz hat neben einer technischen auch immer eine menschliche Seite. Für ein System wie das hier vorgestellte ist neben Entwicklung und Implementierung auch die Erstellung von passendem Einführungs- und Arbeitsmaterial für die Einsetzbarkeit wesentlich: AnwenderInnen müssen lernen, mit der zwar der natürlichen Sprache nachempfundenen, aber eben doch normierten Sprache umzugehen, ebenso wie mit anderen vom System akzeptierten Darstellungsformen wie Annotationen. Sie müssen darüber hinaus lernen, die vom System ausgegebenen Rückmeldungen richtig zu interpretieren und daraus ggf. Ansätze zur Verbesserung ihrer Eingabe zu ziehen. Soll das System studienbegleitend eingesetzt werden, muss daher beides in den Verlauf der Vorlesung und der Übungen integriert werden. Diese Integration ist letztlich eine Gemeinschaftsleistung der Abteilung für Mathematik und ihre Didaktik, in der das System im Herbstsemester 2020/2021 erstmals eingesetzt wurde.

Wir hoffen, dass die vorliegende Arbeit deutlich macht, dass die (mathematische) Logik, auch und gerade in ihrem Bezug zur Philosophie der Mathematik, kein künstliches Konstrukt ist, das nur im formallogischen Zugang zum Beweisen präsent ist, sondern einen engen Bezug zur "natürlichen"

[1] An letzterem Projekt war der Autor, wenn auch eher am Rande, beteiligt.

Mathematik hat. Die weiter unten dargestellten Überlegungen zur Möglichkeit und Abgrenzung des Machbaren vom "im Prinzip Möglichen" gehen aus einer Beschäftigung mit dem Verhältnis zwischen natürlichem und formalen Beweis hervor, der in der Philosophie der Mathematik thematisiert wird (siehe z.B. Rav [173] oder Azzouni [14], [15]) und waren für die Konzipierung des Diproche-Projektes richtungsweisend. Der "Anti-ATP" geht aus der Befassung mit der Argumentationstheorie hervor, einem Zweig der philosophischen Logik, dem wir u.a. eine umfangreiche Klassifikation (formaler) Fehlschlüsse verdanken. So hatten Diskurse zur Philosophie der Mathematik und Beiträge zur philosophischen Logik einen wesentlichen Einfluss auf das, was in der konkreten Implementierung letztlich nur seinen sichtbaren Ausdruck findet. Wenn diese Arbeit einen Beitrag dazu leistet, dass die (mathematische) Logik und die Philosophie der Mathematik nicht mehr, wie so häufig, als mathematische Randgebiete und Orchideenfächer angesehen werden, hätte sie eines ihrer Ziele erreicht.

1.1 Danksagungen

An erster Stelle danke ich Hinrich Lorenzen für seine Bereitschaft, dieses Projekt zu betreuen und auf verschiedene Arten zu unterstützen, nicht zuletzt durch seine Bereitschaft, den hier entwickelten experimentellen Ansatz in seiner Lehrveranstaltung auszuprobieren, sowie dafür, dass er mir, zusammen mit Michael Schmitz, zahlreiche hilfreiche Anregungen und Hinweise für die Entwicklung des Diproche-Systems gegeben hat; beiden danke ich ebenso für die angenehme Zusammenarbeit bei der Erstellung und Auswahl von der Übungsaufgaben, die mit dem System behandelt werden können. Mein Dank geht ebenso an Peter Koepke, Marcos Cramer, Bernhard Fisseni, Bernhard Schröder stellvertretend für die zahlreichen Menschen, die an der Konzeption, Organisation und Implementierung des Naproche-Projektes beteiligt waren, ohne welches das Diproche nicht möglich gewesen wäre. Ich danke weiterhin den MitarbeiterInnen der Abteilung für Mathematik und ihre Didaktik der Europa-Universität Flensburg, die die Übungsgruppen und vorbereitenden Übungen geleitet haben, in denen die Diproche-Aufgaben besprochen wurden: Inga Stolz, Michael Schmitz, Selma Stronzik, Stefan Virchow, Philipp Lampe und Christian Hercher.

Ein besonderer Dank gilt schließlich meinem Freund und Mitwanderer Dr. Thomas Bliem, der durch die Implementierung des Webinterface den praktischen Einsatz von Diproche in der Hochschullehre überhaupt erst möglich gemacht hat und entscheidend mitgeholfen hat, den Webserver einzurichten und das System so zugänglich zu machen. Inzwischen ist auch unser physikalischer Webserver online und erlaubt einen komfortablen Zugriff auf das System; dafür geht mein Dank vor

allem an Mats Lorenzen für die Einrichtung des Webservers und an Herrn Karsten Veers für die Integration in das Netz der EUF, sowie erneut an Michael Schmitz und Hinrich Lorenzen für ihre Unterstützung.

Kapitel 2

Problemstellung

2.1 Zur Rolle und Bedeutung von Beweisen in der Mathematik

Beweise sind ein essentieller Bestandteil der Mathematik. Weit davon entfernt, lediglich der Absicherung vermuteter mathematischer Wahrheit zu dienen, bilden sie den zentralen Gegenstand des Interesses und Bemühens arbeitender Mathematiker. Man ist nicht um Beispiele verlegen, in denen der Grund für das Interesse an einer mathematischen Aussage primär in deren Beweis zu finden ist: Man denke etwa an den 4-Farben-Satz oder den großen Satz von Fermat. Beide wären wohl kaum Teil der mathematischen Allgemeinbildung, hätten sie einfache Beweise statt der aufsehenerregenden, die bisher gefunden wurden. Einer noch radikaleren Ansicht zufolge sind Beweise sogar der eigentliche Gegenstand mathematischen Wissens und Theoreme etc. dienen nur als "Schlagzeilen", um sich im Geflecht des in Beweisen niedergelegten Denkens zurecht zu finden; diese Ansicht wird etwa vom Logiker und Mathematik-Philosophen Y. Rav in [172] auf S. 20 vertreten, wenn er schreibt:

> "(...) *proofs rather than the statement-forms of theorems are the bearers of mathematical knowledge.* Theorems are in a sense just tags, labels for proofs, summaries of information, headlines of news, editorial devices."

Und, weiter unten ([172], S. 20):

> "Think of proofs as a network of roads in a public transportation system, and regard statements of theorems as bus stops; the site of the stops is just a matter of convenience."

An der gleichen Stelle ([172], S. 20, Fußnote 21) weist Rav darauf hin, dass mathematische Texte aus früheren Zeiten oft ausschließlich aus Beweisen bestehen:

> "It is noteworthy that words like theorem, proposition, or the like appear nowhere in Descartes's *La Geometrie*; his books consist of a continual unfolding of methods. (...) Such examples are quite typical of pre-nineteenth century texts."

Ob man sich dieser Haltung nun anschließen und eine historische Gestalt der Mathematik als verbindliche Informationsquelle über die Mathematik anerkennen will oder nicht: Das exakte Begründen, das konzeptionelle Erkunden, das strategische Suchen, die im Beweisen zutage treten, schließlich die Möglichkeit, Behauptungen auf präzise Begriffsdefinitionen und wenige Grundannahmen zurückführen zu können, sind jedenfalls ein wesentliches Merkmal der Mathematik. Zur Mathematik gehört eine gewisse Art zu denken, zu "sehen" und "an Dinge heranzugehen" so sehr, dass, wer diese Fähigkeiten einmal erworben hat, auch dann noch jederzeit in der Lage sein dürfte, Mathematik hervorzubringen, wenn alles mathematische Faktenwissen aus irgend einem Grund verloren gegangen sein sollte.[1] Um mit L. Brouwer zu sprechen: "Die Mathematik ist mehr ein Tun als eine Lehre."[2]

Was heißt das für die Didaktik der Mathematik? Schließlich muss die Didaktik nicht unbedingt das Fach als wissenschaftliche Disziplin abbilden. Zentrale mathematische Alltagskompetenzen, etwa das Rechnen mit Uhrzeiten oder Geld, lassen sich sicherlich erwerben, ohne näher auf die spezifisch mathematische Form des Begründens einzugehen. Hier kommt es unter anderem darauf an, was man als Ziel einer mathematischen Ausbildung ansehen will. Mögliche Ziele des mathematischen Unterrichts wären etwa folgende, von denen viele sich im Sinne der Fachanforderungen im Bereich Mathematik des Ministeriums für Schule und Berufsbildung in Schleswig-Holstein [78] unter den Aspekt der "Selbstkompetenz" einordnen lassen:

- Alltagskompetenz
- Beherrschung der Mathematik als Hintergrund des naturwissenschaftlichen Weltbildes

[1] So beschreibt etwa A. Schoenfeld in seiner Studie zum mathematischen Problemlösen [178] auf S. 121f das eindrucksvolle Beispiel eines Problemlöseprozesses, in dem ein Mathematiker, der seit langer Zeit keine Elementargeometrie mehr betrieben und entsprechend in diesem Bereich kaum noch spezifische Ressourcen oder Heuristiken kannte, allein durch seine Kontrollstrategien zur Lösung gelangte, die also genügten, um einen Mangel an Ressourcen und spezifischen Heuristiken auszugleichen.

[2] So gängig dieses (vermeintliche?) Brouwer-Zitat auch ist, so haben wir doch trotz intensiver Recherche keinen Text von Brouwer finden können, in dem es enthalten wäre.

- Beherrschung der Mathematik als Grundlage für technisches Verständnis, besonders im Kontext der Digitalisierung
- Mathematik als Vorbereitung auf den Berufseinstieg
- Mathematik als Grundlage der Fähigkeit zur Urteilsbildung aufgrund von numerisch aufbereiteten Datensätzen wie Statistiken, Tabellen, Diagrammen etc.
- Ausbildung einer kritischen Aufmerksamkeit in der logischen Analyse und Beurteilung von Argumenten
- Schulung des präzisen Denkens und Argumentierens

Sicherlich kann von einer Kenntnis der Mathematik nicht schon dann gesprochen werden, wenn jemand eine Vielzahl wahrer mathematischer Sätze kennt und aufsagen kann.

Wir erwähnen hier erneut Rav ([172], S. 20):

> "(...) the entire mathematical know-how is embedded in proofs. (...) Proofs are for the mathematician what experimental procedures are for the experimental scientist: in studying them one learns new ideas, new concepts, new strategies (...)."

Eine Wirkung, die eine fundierte und auf Begründungen bzw. Beweise hin orientierte mathematische Ausbildung entfalten kann und die von ihr auch erwartet wird[3] ist die Übung in und die Gewöhnung an **kritisches**, **selbstständiges**, **präzises** und **radikal begründendes** Denken. Das beweisende Denken ist **kritisch**, insofern es Aussagen nicht ungeprüft hinnimmt und für jeden Schluss eine Rechtfertigung verlangt und auch liefern kann; ferner, weil das eigene Beweisen wie auch das Verstehen der Beweisführungen anderer stets die kritische Aufmerksamkeit auf den dargelegten Gedankengang erfordert. Es ist **selbstständig**, insofern beim mathematischen Denken keine Autorität, kein Bericht anderer, ja – im Prinzip – nicht einmal die eigene Sinneswahrnehmung äußerer Gegenstände erforderlich sind, um mathematische Sätze einzusehen und zu begründen und mathematische Theorien aufzubauen; jede Behauptung, die man für wahr, jedes Verfahren, das man für korrekt halten soll, läßt sich vollständig in Verständnis auflösen (wobei freilich für die Kommunikation

[3] So schreibt etwa Polya in Band 2 von "Vom Lösen mathematischer Aufgaben", [163], S. 139: "Von einem gewissen Gesichtspunkt liegt der Hauptwert des Erlernens mathematischer Beweisführung darin, daß es uns dem idealen, vernünftigen Verhalten näherbringt, das dem Menschen, dem "Vernunftwesen", zukommt."

dieses Verständnisses gewisse Bezeichnungskonventionen einzuhalten sind). Es ist **präzise**, insofern sich im mathematischen Beweisen jeder Versuch sofort rächt, einen Begriff lediglich als vage Vorstellung etwa anhand von Beispielen erfassen zu wollen: Mathematische Begriffe funktionieren erst dann im Rahmen von Beweisen, wenn ihr Begriffsumfang durch eine Definition begrifflich erfasst worden ist. Um etwa in der Geometrie mit dem Begriff des Kreises umgehen zu können, genügt es nicht, Kreise unter anderen Figuren erkennen oder selbst welche zeichnen zu können, selbst wenn das ausreichen mag, um dieselbe Kompetenz anderen zu vermitteln: Erst mit einer Erklärung wie "geometrischer Ort aller Punkte, die von einem gegebenen Punkt den gleichen Abstand haben" wird man in der Lage sein, Aussagen über Kreise zu verstehen und zu begründen, die den Bereich des unmittelbar anschaulich einsichtigen übersteigen. Wo eine anschauliche Vorstellung mathematisch wirksam gemacht werden soll, bedarf sie daher zuvor immer erst einer **Präzisierung**.[4] Das mathematische Denken ist präzise weiterhin in dem Sinn, dass beim mathematischen Schließen und der Anwendung von Regeln nicht nach vagen Analogien vorgegangen werden kann, sondern stets zu prüfen ist, ob die für den fraglichen Schluss bzw. Regelanwendung nötigen Voraussetzung im gegebenen Fall vorliegen. Mathematisches Denken ist endlich **begründend**, und zwar **radikal begründend**, insofern – wie schon oben erwähnt – das Begründen in der Mathematik sein Ende nicht bei fraglos hinzunehmenden Fakten wie den Behauptungen von Autoritäten oder empirische Erfahrungen findet. Definitionen können als abkürzende Konventionen zum Zweck der leichteren Verständigung aufgefaßt werden, nötigen also nicht zur Annahme einer inhaltlichen Aussage, und Axiome können statt als deskriptiv als konstitutiv für den zu behandelnden Gegenstandsbereich angesehen werden, also als 'implizite Definitionen' (so etwa die Auffassung, nach der Hilbert in seinen "Grundlagen der Geometrie", [108], verfährt).[5] Insofern die Übung im mathematischen Beweisen für diese Art des Denkens förderlich ist, ist sie, weit über eine spezialisierte

[4]Das reduziert die Mathematik keineswegs auf ein steriles Deduzieren aus festliegenden Definitionen und schließt nicht aus, dass zwischen Definition und Beweis ein fruchtbares Wechselspiel stattfinden kann, wie es etwa Lakatos beschreibt [129].

[5]Zum Status von Axiomen in der Mathematik im Allgemeinen ist freilich deutlich mehr zu sagen, siehe etwa [143]. Indes ist die Frage nach einer Rechtfertigung der mengentheoretischen Axiome für die einführende Didaktik des Beweisens unerheblich. Die für die Elementarmathematik erforderlichen Axiomatisierungen etwa der Gruppentheorie, der Arithmetik oder der Geometrie sind jedenfalls entweder vollständig definitorisch – wie im Fall der Gruppentheorie – oder beschreiben den Gegenstandsbereich doch jedenfalls auf so elementare Weise, dass man sich recht gefahrlos auf den Standpunkt stellen kann, jemand, der etwa die Eindeutigkeit des Vorgängers in der Arithmetik leugnet, rede eben nicht über dasselbe, was der Proponent der Peano-Axiome unter "natürlichen Zahlen" verstanden haben will und steht insofern nicht in einem inhaltlichen Dissens mit diesem, sondern habe allenfalls andere Präferenzen hinsichtlich der Verwendung gewisse Ausdrücke.

methodische Fachausbildung hinaus, ein Beitrag zur Persönlichkeitsbildung und zur Heranbildung zur/zum mündigen BürgerIn.[6] Hier erfüllt das Beweisenlernen also eine wichtige Funktion, für die es abträglich wäre, lediglich unerklärte Rechenrezepte zu verordnen oder sich auf automatische Systeme zu verlassen.

Es gibt mindestens noch einen weiteren Grund, warum es verfehlt wäre, sich bei der mathematischen Ausbildung auch dort, wo die Mathematik "nur" die Rolle eine "Hilfswissenschaft" spielt, auf die Vermittlung von Lösungsverfahren zu beschränken, statt sich um ein konzeptionelles Verständnis von Grund auf sowohl der auftretenden Objekte und Operationen als auch der Verfahren selbst zu bemühen oder angesichts der Allgegenwart automatischer Systeme, die solche Verfahren auszuführen in der Lage sind, bloß noch deren Verwendung zu unterrichten; dieser Grund dürfte zugleich auch für jene überzeugend sein, die (wie etwa Andrew Hacker in seinem Angriff auf das tradierte mathematische Curriculum, [94]) eine allgemein persönlichkeitsbildende Wirkung der mathematischen Ausbildung bezweifeln oder diese zwar zugeben, sie aber als eine Hinführung zur weltfremden Spitzfindigkeiten geradezu für schädlich halten. Das bloße Vertrauen auf Verfahren bzw. deren Implementierung in elektronischen Rechnern birgt die Gefahr von Fehlanwendungen mit z.T. dramatischen Folgen. So weist der auf Systemsicherheit spezialisierte britische Informatiker Harold Thimbleby darauf hin, dass es in Krankenhäusern zu einer nicht unerheblichen Zahl vermeidbarer schwerer Nebenwirkungen bis hin zu Todesfälle durch die Fehlbedienung von Taschenrechnern kommt.[7] Auch für eine verantwortliche Anwendung der Mathematik in alltäglichen und beruflichen Kontexten – und davon gibt es von der Haushaltsführung über die Berechnung der richtigen Dosis eines Medikaments bis zur Baustatik viele – ist ein konzeptionell fundiertes Verständnis der verwendeten Begriffe erforderlich, die durch eine Einsicht in die Begründungszusammenhänge erreicht wird – und diese Einsicht wiederum erreicht man durch den mathematischen Beweis.[8]

Auch aus einer rein an ihrer "Anwendbarkeit" interessierten Perspektive auf die Mathematik ist es also wünschenswert, in der mathematischen Ausbildung die spezifischen Formen des mathematischen Begründens zu vermitteln. SchülerInnen und Studierende sollten sollten Mathematik als begründete und begründende Disziplin kennen und erleben lernen. Hierfür ist es insbesondere erforderlich, die

[6] Vgl. hierzu auch die Ausführungen in Wittenberg, [206], Kapitel 1.

[7] Siehe z.B. Thimbleby [193], wo es – mit Verweis auf [131] – heißt (S. 1): "more than 17% of medication errors involve miscalculation of doses, incorrect expression of units or incorrect administration rates".

[8] Zur Bedeutung von Ableitungsstrategien selbst beim Erlernen elementarer Rechentechniken in der Grundschule siehe auch Gaidoschik [84], S. 182f, wo darauf hingewiesen wird, dass die Verwendung von Taschenrechnern ohne eigenes Verständnis für Zahlen und Rechenoperationen fehleranfällig und zeitraubend ist.

Lehrkräfte entsprechend zu schulen – auch und gerade im Lehramtsstudium der Mathematik und Mathematik-anwendender Fachgebiete wie Physik, Informatik, Chemie etc. ist also Wert darauf zu legen, dass die Grundlagen des Beweisens verstanden sind und das mathematische Beweisen auch bis zu einem gewissen Grad selbstständig beherrscht wird.

2.1.1 Zur Relevanz des mathematischen Beweisens in der Lehramtsausbildung

"Es ist eine in mathematischen und verwandten Fächern durchaus falsche Vorstellung, daß man nicht viel mehr zu wissen brauche, als man zu lehren hat. (...) Aber ich behaupte: um die Elemente der Mathematik in bildender Weise zu lehren, muss der Lehrer durchtränkt sein von der Quintessenz des ganzen mathematischen Wissens. Dann erst wird sein Unterricht Klarheit gewinnen (...)."

Paul Du Bois–Reymond, zitiert nach Beutelspacher et al. [20]

Wir haben oben argumentiert, dass das Beweisen ein wesentlicher Bestandteil der Mathematik ist, der auch solche Bereiche der Mathematik betrifft, die, wie etwa die korrekte Anwendung von Rechenverfahren, vordergründig davon unabhängig zu sein scheinen. Wenn das so ist, muss das Beweisen auch und gerade im mathematischen Schulunterricht eine Rolle spielen.[9] Es ist damit selbstverständlich, dass das Beweisen als zentrales Element des zu erteilenden Unterrichts auch eine wichtige Rolle in der Ausbildung angehender Lehrkräfte zu spielen hat. Entsprechend werden sowohl das mathematische Argumentieren und Beweisen wie auch die kritische Betrachtung und Prüfung von Begründungs- und Beweistexten als Kompetenzbereiche in den Fachanforderungen für das Fach Mathematik angeführt ([78], S. 52). Über den offensichtlichen Aspekt hinaus, dass, was gelehrt werden soll, auch beherrscht werden muss gibt es indes noch eine Reihe weiterer Gründe, dem Beweisen in der Lehramtsausbildung eine prominente Stellung zu geben.[10]

[9] Diese Ansicht ist freilich nicht unumstritten. Siehe etwa die Einleitung von Gerwig [88] für eine ausführliche historische Diskussion der durchaus wechselhaften Geschichte des Beweisens im Mathematikunterricht in Deutschland. Zur Diskussion um die Bedeutung des Beweisens im Mathematikunterricht in den USA und eine kritische Behandlung der Argumente, die für eine Reduktion der Rolle von Beweisen oder sogar einen Verzicht auf Beweise im Unterricht vorgebracht wurden siehe Hanna [98].

[10] Der folgende Teil hat wesentliche Ansätze und Impulse aus persönlichen Gesprächen mit Hinrich Lorenzen erhalten. Für eine Liste von Funktionen, die Beweise im Mathematikunterricht haben können, die die folgenden Punkte umfasst, siehe etwa auch Hanna [97], S. 8 sowie deren Diskussion in [90], S. 189f.

- Der systematische Aspekt:[11] Indem Beweise deduktive Zusammenhänge zwischen den Sätzen eines mathematischen Gebietes aufzeigen, helfen sie dabei, ein In- und Nebeneinander von Einzeltatsachen in ein strukturiertes System von Aussagen zu überführen. Dadurch leisten sie einen wesentlichen Beitrag zur Konstitution eines Gebietes. Durch konsequentes Beweisen gliedern sich die Aussagen eines Wissensbereiches etwa in grundlegendere und abgeleitete, leichtere und schwierigere, offensichtliche und tiefliegende, allgemeinere und besondere, zentrale (etwa solche, die bei Beweisen anderer Ergebnisse immer wieder auftreten) und periphere; vordergründig disparate Aussagen erweisen sich als durch einfache Überlegungen ineinander überführbar, also in diesem Sinn als "äquivalent"; andere erweisen sich als Ausformulierungen eines einzigen Prinzips (etwa im Fall der "Strahlensätze", die sich letztlich auf den Satz reduzieren lassen, dass ähnliche Dreiecke ähnlich sind) usw. Insofern es die Aufgabe von Lehrkräften ist, den Wissensbestand eines Gebietes nachvollziehbar aufzubauen, ist die Kenntnis der Strukturen, die durch das Beweisen gestiftet werden, und es etwa erlauben, von einfachen Beobachtungen anfangend allmählich zu durch Beobachtungen kaum noch gestützten oder geradezu antiintuitiven Aussagen fortzuschreiten, unerlässlich.

- Der kulturelle Aspekt: Mathematische Beweise haben in mindestens zwei Hinsichten eine kulturelle Bedeutung. Einerseits sind sie ein essentieller Bestandteil der mathematischen Praxis und Methodik. Ein grundlegendes Verständnis mathematischer Beweise, zu dem auch die Erfahrung gehört, was es heißt, solche zu suchen, ist daher Voraussetzung dafür, die Mathematik als Wissensgebiet einordnen zu können. Dazu gehört dann insbesondere auch ein Wissen um und ein Verständnis für die historischen Entwicklungen und Motive, die zu jenen Standards für Begründungen führen, welche die Mathematik so deutlich von anderen, insbesondere den naturwissenschaftlichen, Disziplinen unterscheiden. Das steht nicht im Gegensatz zur Würdigung von Versuchen, diese Standards aus verschiedenen Perspektiven infrage zu stellen oder durch andere – etwa experimentelle – Begründungsstrategien zu ergänzen,[12] sondern ist im Gegenteil Voraussetzung dafür. Andererseits stellt die Mathematik infolge dieser Entwicklungen mit dem Konzept des mathematischen Beweises einen Standard für strenge und sichere Begründungen zur Verfügung, den man wohl ohne Übertreibung zu den größten kulturellen Errungenschaften der Menschheit rechnen kann; die kulturelle Bedeutung dieses Standards zeigt

[11] Vgl. etwa Hanna [97], S. 8, Listenpunkt 3.

[12] Für eine Diskussion solcher Ansätze und ihre Bedeutung für die Rolle von Beweisen im Mathematikunterricht siehe etwa Hanna, [98].

sich nicht zuletzt darin, dass er für so unterschiedliche Disziplinen wie die Physik,[13] die Psychologie[14], die Biologie,[15], die Wirtschaftswissenschaften[16], verschiedene Gebiete der Philosophie,[17] die Informatik,[18] die Theologie[19] etc. immer wieder zum Vorbild geworden ist (wobei die Wirkungen dieser zahlreichen Versuche, eine Disziplin nach dem Vorbild der Mathematik oder "more geometrico" (Spinoza, [187]) umzugestalten, trotz der wiederholten Fehlschläge nicht selten wesentliches zur Entwicklung dieser Gebiete beigetragen haben). Auch hier ist ein fundiertes Verständnis dafür, was es bedeutet, ein Gebiet zu "mathematisieren" oder "nach dem Vorbild der Mathematik zu behandeln" Voraussetzung dafür, an solchen Ansätzen qualifizierte Kritik üben und sie in ihrer Begrenztheit erkennen zu können. Zu wissen, was ein mathematischer Beweis ist, kann damit neben das Wissen um andere grundlegend kulturelle Ausdrucks- und Erscheinungsformen gestellt werden, etwa das Wissen, was ein Gedicht, ein Lied, ein Gebet oder ein Vertrag ist.

- Der ästhetische Aspekt: Mathematik, und besonders mathematische Argumentation, besitzt eine eigene, spezifische Form der Schönheit;[20]; diese stellte gegenüber anderen Merkmalen eines Beweises, wie etwa Einfachheit, Erklärungswert oder Nützlichkeit eine eigene Qualität dar.[21] Nun sollten LehrerInnen ihr Gebiet einerseits schätzen, wozu auch gehört, seine Schönheit selbst erlebt zu haben. Zugleich sollten sie dieses Erlebnis von Schönheit den SchülerInnen auch, soweit möglich, vermitteln können: Erleben geistiger Schönheit ist, wie das Erleben von Kunstschönheit, eine menschliche Grunderfahrung. Dass die Erfahrung von Schönheit in mathematischer Argumentation (und sogar das Erleben von Ähnlichkeit zwischen mathematischen Argumenten und Kunstwerken) keineswegs Spezialisten vorbehalten, sondern auch mathematischen Laien zugänglich ist,

[13]Ein Programm zur Axiomatisierung verschiedener Zweige der Physik findet sich etwa als sechstes Hilbertsches Problem, siehe Hilbert [109].

[14]Siehe z.B. Levy [135].

[15]Woodger, [208]

[16]Siehe etwa Debreu [57]; für eine kritische Diskussion dieses und verschiedener anderer axiomatischer Ansätze in den Wirtschaftswissenschaften siehe Clower [48].

[17]So etwa in der Ethik, siehe Spinoza [187]; ein neueres Beispiel wäre Zalta [214].

[18]Ein besonders einflussreicher Ansatz ist hier etwa Hoare, [112].

[19]Ein beispielhafter Vertreter aus der mittelalterlichen Theologie wäre Alanus ab Insulis; für einen Überblick über axiomatische Ansätze in der mittelalterlichen Theologie siehe Flasch [71], S. 247f.

[20]Siehe etwa Rota, [174] für eine phänomenologische Analyse mathematischer Schönheit.

[21]Diese Unterscheidung wird in Giaquinto [89] argumentativ plausibilisiert; in Inglis und Aberdein [116] wird empirisch festgestellt, dass sie von MathematikerInnen tatsächlich vorgenommen wird.

wird in Johnson und Steinerberger [124] empirisch demonstriert. Offenbar ist aber ein Verständnis für mathematische Argumentation die unabdingbare Voraussetzung dafür, ihre Schönheit zu erleben.

2.2 Überlegungen zum Übungsbetrieb zu universitären mathematischen Lehrveranstaltungen

Im Studium der Mathematik, einschließlich der Lehramtsausbildung, wird der Bedeutung des Beweisens durch einen aufwändigen Übungsbetrieb Rechnung getragen: Zu den meisten Lehrveranstaltungen, besonders den Grundvorlesungen wie der linearen Algebra oder der Analysis, gibt es gewöhnlich wöchentliche Übungsaufgaben, die selbstständig bearbeitet und eingereicht werden sollen. Die korrigierten Bearbeitungen der Aufgaben werden dann – typischerweise mit Abstand von etwa einer Woche – zurück gegeben und besprochen. Die Aufgaben dienen so einerseits der Wiederholung und Vertiefung des Lehrstoffs, andererseits aber eben auch dem Erwerb eigener Kompetenzen im mathematischen Beweisen und Argumentieren.

Ein offensichtlicher Nachteil an dieser Praxis besteht darin, dass die Rückmeldungen des oder der KorrektorIn auf diese Weise nicht mehr in den Lösungsprozess einfließen können. Der Eindruck etwa einer Lücke in der eigenen Beweisführung, des Hinweises auf einen Fehlschluß oder eine ungenaue Darstellung kann so nicht mehr verwendet werden, um die Lücke zu schließen, den Fehler zu korrigieren – ggf. die ganze Strategie zu überdenken – oder die Präsentation zu verbessern. Und für den sicherlich schon eher außergewöhnlich Fall, dass Studierende ihre Lösungen regelmäßig noch einmal überarbeiten, fehlt jedenfalls die Rückmeldung, inwiefern die Überarbeitung nun eine Verbesserung bzw. Behebung der früheren Mängel darstellt. Allenfalls kann eine "allgemeine Lehre", die aus einer gewissen Korrektur gezogen wurde, in die Bearbeitung späterer Übungsblätter einfließen, wobei es freilich eine Weile dauern mag, bis das nächste Mal eine Aufgabe vorkommt, bei der der fragliche Effekt zutage tritt. Ein sicherlich wünschenswerter kritischer Blick auf die eigene Lösung, die die Rolle der Korrektur letztlich auf die einer 'zweiten Expertenmeinung' reduzieren würde, kann gerade im Anfängerbereich noch nicht erwartet werden: Einen kritischer Blick auf den eigenen Beweis auszubilden, gehört zu den Zielen der mathematischen Ausbildung und kann also nicht schon zu Beginn vorausgesetzt werden.

Die Wichtigkeit, zu einer aktiven Auseinandersetzung mit Korrekturanmerkungen anzuregen um diese für den Erwerb der Fähigkeit zur Darstellung von Beweisen nutzbar zu machen, ist auch empirisch bestätigt worden; in einer Studie von Robert Moore etwa, in der der Umgang mit

Studierenden mit Anmerkungen zu von ihnen bearbeiteten Beweisaufgaben untersucht wurde, wird zunächst festgestellt, dass Studierende meist durchaus in der Lage sind, einen Beweistext aufgrund von Korrekturanmerkungen zu verbessern; in der Folge jedoch heißt es:

> "Six of the participants claimed to make use of their professor's comments when they were asked to revise and resubmit proofs, but they all noted that they were seldom asked for revisions. This evidence suggets that students generally do not meaningfully engage with the comments of their graded proofs unless required to do so. The language acquisition literature (...) and composition research (...) argue that such engagement is critical for learning from feedback." (Moore, [151])

Eine weitere Schwierigkeit des klassischen Übungsbetriebes besteht darin, dass Studierende mit der Bearbeitung der Übungsaufgaben oft weitgehend allein gelassen werden. Im typischen universitären Übungsbetrieb werden die Lösungen zwar in einer Übungsgruppe besprochen; dagegen findet der Lösungsprozess selbst unbegleitet statt. Studierende erhalten während der Bearbeitung entsprechend keine Hinweise zu Lösungsansätzen, nicht einmal im Bezug auf elementare Basisstrategien. Dies ist einerseits im Sinne des Ziels, einen selbstständigen Umgang mit mathematischen Problemen zu fördern. Andererseits kann sich der so angezielte Effekt in der Praxis leicht in sein Gegenteil verkehren: Bleiben Studierende beim Lösen "stecken", weil sie die erforderlichen strategische Ressourcen noch nicht in einem für die aktuelle Aufgabe erforderlichen Umfang beherrschen, haben sie kaum andere Optionen, als den Lösungsversuch abzubrechen. Fehlt es schon gleich zu Beginn an einer Strategie, die ganze Aufgabe in Angriff zu nehmen, unterbleibt so die selbstständige Beschäftigung mit der Aufgabe, während sich zugleich ein Gefühl der Überforderung und Frustration einstellen kann.

An der Europa-Universität Flensburg (wie inzwischen auch an zahlreichen anderen deutschen Universitäten) wird diesem Problem dadurch begegnet, dass Lehrveranstaltungen angeboten werden, in denen Studierende in kleinen Gruppen an Übungsaufgaben arbeiten, während Lehrkräfte den Lösungsprozess gezielt dort unterstützen, wo Schwierigkeiten auftreten. Die Erfahrung zeigt dabei, dass gerade im Anfängerbereich oft kleine strategische Anstöße genügen, um einen ins Stocken geratenen Lösungsprozess wieder in Gang zu bringen: So genügt häufig die Frage, wie Beweise von Allaussagen "meistens beginnen", um Studierende dazu zu bringen, ein Element aus dem quantifizierten Bereich mit "Sei" einzuführen und zu bezeichnen. Ebenso kann bei Beweisen zu Mengengleichheiten die Frage, wie "so etwas gewöhnlich gezeigt wird" ausreichen, um Studierende dazu anzuregen, die Aufgabe in zwei Teilaufgaben mit je einer Teilmengenbeziehung aufzuspalten.

Häufig genügen solche kleinen strategischen Impulse, die oft nicht mehr sind als ein recht allgemeiner Hinweis auf die Möglichkeit einer Strategiewahl, um einen im weiteren erfolgreichen Lösungsprozess anzustoßen (der ohne Intervention an dieser Stelle womöglich abgebrochen worden wäre).

Diese Art von Unterstützung kann also, wenn sie darauf verzichtet, Studierende allzu direkt zur Lösung "schubsen" zu wollen, gerade den Effekt haben, die selbstständige Auseinandersetzung mit Beweisaufgaben zu fördern. Allerdings können Studierende eben nur dann auf solche Hinweise zurückgreifen, wenn Lehrkräfte zur Verfügung stehen, die sie geben. Im Selbststudium, das außerhalb des Lehrbetriebes auch stattfindet (und auch stattfinden soll), treten also die oben genanten Schwierigkeiten weiterhin auf. Hier könnten also in begrenztem Umfang digitale Systeme Abhilfe schaffen, die entsprechende Hinweise "überall und jederzeit" automatisch generieren können.[22]

Ein dritter Nachteil des Übungsbetriebes betrifft die Wirksamkeit der Korrektur: Die Rückmeldung erfolgt im klassischen Übungsbetrieb mit einem so großem zeitlichen Abstand zur Bearbeitung (gewöhnlich liegt zwischen Abgabe und Rückmeldung eine Woche, wozu die Zeit zwischen Bearbeitung und Abgabe noch hinzukommt), dass daraus kaum ein Lernfortschritt im Hinblick auf Methodik, Strategie und Heuristik zu erwarten ist: Wenn die Überlegungen, die zu einer gewissen Lösungsdarstellung geführt haben, nicht mehr präsent sind, kann sich die Korrektur nur noch eingeschränkt fördernd (in Bezug auf erfolgreiche Strategien) oder hemmend (in Bezug auf Fehlstrategien) auswirken. Damit sinkt potenziell die didaktische Wirkung der Korrektur und es erhöht sich die Chance, dass gewisse Fehler immer wieder vorkommen.

Eine vierte Schwierigkeit einer vom Lösungsprozess entkoppelten Korrektur besteht darin, dass die Mathematik es erfordert, Darstellungsform, Inhalt und Methodik parallel zu erlernen. Nicht selten kommt es auf der Seite der Lernenden dabei zu Unsicherheiten bzw. Verwechselungen, wenn etwa in Bezug auf die Gültigkeit einer Schlussweise gefragt wird, ob "man das so schreiben darf" oder wenn das Beweisen insgesamt für eine Formsache und der Beweisbegriff rein formal verstanden wird.[23] Das erschwert nun wiederum die Korrektur, besonders,

[22]Wir wollen an dieser Stelle entschieden dem möglichen Missverständnis begegnen, wir hielten die menschliche Intervention für durch die digitale Intervention ersetzbar. Gerade bei Aufgaben zur inhaltlichen Mathematik, die strategische Kreativität ebenso erfordern wie den Appell an spezifisch menschliche Anschauungen und Grunderfahrungen (siehe dazu die Diskussion über die Abgrenzung zwischen inhaltliche und formaler Mathematik weiter unten) sind "menschengemachte" Hinweise unerlässlich, umso mehr, als diese auf die jeweiligen individuellen Besonderheiten der betreffenden Studierenden eingehen sollten und im Dialog mit diesen entwickelt werden. Des weiteren ist der psychologische, gewissermaßen "therapeutische" Effekt menschlicher Interventionen nicht zu unterschätzen, die sich auf den Aufbau von Motivation, den Abbau von Frustration und Ängsten etc. beziehen.

[23]So schreibt z.B. Weber in [202]: "For instance, many pre-service teachers believe a geometry

wenn eine Bewertung, etwa in Form einer Punktzahl, erfolgen soll: Da der oder die Studierende nur einen Beweistext zur Korrektur vorlegen kann und dieser als ganzer abschließend beurteilen muss, wäre es unnötig entmutigend, einen strategisch sinnvollen Ansatz mit Verweis auf eine formal unzureichende Darstellung zu verwerfen. So bleiben formale Fehler oft im Hintergrund, was die Möglichkeit blockiert, aus ihnen zu lernen. Das spricht dafür, im Rahmen der Korrektur verschiedenen Aspekte der Qualität von Beweistexten in der Bewertung deutlich zu trennen, etwa durch Einführung separater "Punktzahlen" für Ansatz und Beweisidee sowie die formale Korrektheit der Darstellung. Stünde dagegen die Rückmeldung bereits während der Bearbeitung zur Verfügung, ließe sich etwa ein zielführender und logisch korrekter Ansatz weiter verfolgen, während Darstellungsmängel sukzessive behoben werden.

2.3 Vorläufige Zielsetzung

Die oben genannten Gründe legen nahe, die Entwicklung von Beweiskompetenzen durch Lehrveranstaltungen zu fördern, in denen kompetente TutorInnen den Lösungsprozess beobachten und begleiten, für Fragen zur Verfügung stehen, zumindest auf Anfrage eine vorgeschlagene Lösung beurteilen etc. Solche Veranstaltungsformate sind denn auch tatsächlich entwickelt und eingesetzt worden, so z.B. an der Europa-Universität Flensburg die "Kolloquien" oder an der Universität Konstanz die "Mathewerkstatt". Indes begrenzt der Charakter einer Gruppenveranstaltung zwangsläufig die Aufmerksamkeit, die dem oder der einzelnen Studierenden dabei zuteil werden kann. Ferner ist auch der zeitliche Rahmen natürlicherweise begrenzt; das Beweisen muss letztlich viel in eigener (Heim)arbeit erlernt werden, wo kein/e TutorIn mehr zur Verfügung steht. Jeder und jedem Studierenden dauerhaft eine/n PrivattutorIn zur Seite zu stellen ist dagegen kaum realistisch. Es wäre daher äußerst wünschenswert, einige Funktionen, die eine Lehrperson bei der Begleitung eines Lösungsprozesses ausübt – also insbesonderes die Kontrolle und Begutachtung von Lösungsvorschlägen sowie die Möglichkeit, in schwierigen Lösungsphasen Hilfestellungen anzubieten – zu automatisieren.

Die vorliegende Arbeit ist ein Versuch, sich dieser Herausforderung zu stellen und ein System zu entwickeln, welches Studierende der Mathematik und anderer Fächer mit Mathematik-Bezug bei der Lösung von Beweisaufgaben von der Entwicklung einer Lösungsidee bis zur Ausformulierung unterstützen kann. Insbesondere soll das System einen vorgelegten Beweistext prüfen und informative Rückmeldungen geben können. Außerdem sollen Hinweise zu

argument must be in a two-column format to be a proof".

einzelnen Beweisschritten wenigstens in dem Umfang verfügbar sein, wie sie durch elementare mathematische Methodik nahegelegt wird, etwa durch einfache Heuristiken wie: "Wenn man eine Implikation $A \to B$ zeigen soll, nehme man A an und zeige B" oder "wenn man eine Allaussage über alle natürlichen Zahlen zeigen soll, versuche man es mit Induktion".

Der Name dieses hier entwickelten Systems zum didaktischen Beweisprüfen ist "Diproche", was für "DIdactical PROof CHEcking" steht. Der Name ist bewußt gewählt in Anlehnung an das Naproche-System, welches weiter unten besprochen wird und welches sowohl den Ansatz wie auch wesentliche Techniken von Diproche angeregt hat.

2.4 Formale Mathematik und automatische Beweisprüfer

In diesem Abschnitt wollen wir kurz auf den logischen und technischen Hintergrund automatischer Beweisprüfer eingehen, der auch für das Diproche-System die Grundlage bilden wird.

Formale Systeme in der Mathematik erlauben es, inhaltliche mathematische Schlüsse durch bloße Zeichenoperationen nach einem festen Regelsatz zu modellieren. Die Zielsetzung, das schlussfolgernde Denken durch solche Regelsysteme vollständig zu erfassen, ist alt: Als früher Ansatz dazu kann die Aristotelische Syllogistik gelten, wie sie in der "analytica priora" entwickelt wird [7]; spätestens mit Leibniz' "characteristica universalis" ist die Idee einer vollständigen Mechanisierung des Denkens klar formuliert.[24] Es sollte indes noch bis zum Ende des 19. Jahrhunderts dauern, bis die ersten erfolgreichen Versuche unternommen wurden, wesentliche Teile der Mathematik durch formale Systeme zu erfassen. Im Jahr 1879 brachte Frege seine "Begriffschrift, eine der arithmetischen nachgebildete Formelsprache des reinen Denkens" heraus, 1884 skizzierte er in "Die Grundlagen der Arithmetik" eine vollständig formale Zahlentheorie, mit deren Umsetzung er 1893 in "Die Grundgesetze der Arithmetik" begann.[25] Spätestens damit ist das logizistische Programm einer Zurückführung der gesamten Mathemaitk auf formale Logik klar formuliert. Freges Durchführung lief bekanntlich in Vorläufer der bekannten mengentheoretischen Antinomien, wie Russell (und vor ihm Zermelo) bemerkte. Russells Bemühungen um eine Korrektur führten zu den monumentalen "Principia Mathematica" von Russell und Whitehead, in denen 1910 wesentliche Teile der Mathematik durch vollständig formale Beweise erfasst wurden. In den 1920ern machte Hilbert im Rahmen des Hilbertschen Programms formale Systeme zur Basis seiner Strategie, die

[24] Zur *characteristica univeralis* siehe z.B. [185].

[25] Vgl. z.B. `https://de.wikipedia.org/wiki/Die_Grundlagen_der_Arithmetik` oder `https://plato.stanford.edu/entries/frege/` (zugegriffen 14.10.2021).

Mathematik einerseits gegen mögliche innere Widersprüche abzusichern und zweitens von philosophischen Deutungen unabhängig zu machen. Spätestens seitdem ist die formale Mathematik ein etabliertes Arbeits- und Forschungsgebiet mit einer kaum noch überblickbaren Vielfalt an Beiträgen.

Dass die Modellierung der Mathematik im Rahmen formaler Systeme die Möglichkeit zu einer automatisierten Überprüfung mathematischer Beweise ermöglicht, ist früh bemerkt worden – als geistiger Vorläufer kann Leibniz' Idee einer idealen Sprache gelten, in der beliebige Streitfragen durch rein mechanisches Rechnen gelöst werden können ("Calculemus"). Als erste technische Realisierung dieser Idee kann das Automath-System von N. de Bruijn angesehen werden, das 1967 begonnen wurde und unter anderem eine formal verifizierte Version von Edmund Landaus "Grundlagen der Analysis" ermöglichte, siehe etwa [6]. Inzwischen sind zahlreiche Systeme entstanden, in denen weite Teile der Mathematik in digitalen Bibliotheken vollständig formalisiert und digital verifiziert vorliegen und die, ähnlich wie Programmiersprachen, immer neue Pakete verfügbarer Begriffe und Sätze zur Verfügung stellen; wir nennen hier als Beispiele die Systeme Coq, Mizar und Isabelle, die aktuell in der Entwicklung eine zentrale Rolle spielen. Mit dem "Journal of Formalized Mathematics" gibt es eine eigene Zeitschrift für in Mizar automatisch verifizierbare Mathematik[26]. Zu den Sätzen mit automatisch verifizierten Beweisen gehören inzwischen der Primzahlsatz (Avigad et al, [12]), der Gödelsche Vollständigkeitssatz (Braselmann und Koepke, [27]) oder der Jordansche Kurvensatz, siehe Hales [95]. Mit der automatischen Verifikation von Hales' Beweis der Keplerschen Vermutung im Rahmen des "Flyspeck-Projektes" (siehe z.B. Hales [96]), an dessen vollständiger Überprüfung die Gutachter zuvor gescheitert waren, hat das automatische Beweisprüfen einen wesentlichen Beitrag zur allgemeinen Akzeptanz eines Beweises geliefert.

2.5 Automatische Beweisprüfer in didaktischen Kontexten

In der Mehrzahl seiner Erscheinungsformen wird das Erlernen einer Fähigkeit begünstigt durch ein von kompetenter Kritik begleitetes Versuchen. Speziell beim Beweisenlernen scheint es sich also anzubieten, automatische Beweisprüfer als Quellen solcher kompetenter Kritik einzusetzen und Lernende so instand zu setzen, eigene Lösungsversuche automatisch überprüfen zu lassen.

Die potenzielle Bedeutung der formalen Mathematik für die mathematische Ausbildung wird in der formalen Mathematik durchaus gesehen. Sie findet sich sogar explizit im "QED-Manifest", einer einflussreichen und vielzitierten programmatischen Schrift für die Entwicklung formaler und automatisch verifizierter Mathematik, als ein Motiv für die Förderung dieser Entwicklung:

[26] Siehe `http://mizar.org/JFM/` (zugegriffen 14.10.2021).

> *"The third motivation for the QED project is education. (...) The development of mathematical ability is notoriously dependent upon "doing" rather than upon "being told" or "remembering". The QED system will provide, via such techniques as interactive proof checking algorithms and an endless variety of mathematical results at all levels, an opportunity for the one-on-one presenting, checking, and debugging of mathematical technique, which it is so expensive to provide by the method of one trained mathematician in dialogue with one student. QED can provide an engaging and non-threatening framework for the carrying out of proofs by students (...)"* [QED-Manifesto, [171]]

In der Tat gibt es eine Reihe von Lehrprojekten, in denen automatische Theorembeweiser, Beweisprüfer oder Beweisassistenten in der Lehre eingesetzt werden (vgl. dazu die Diskussion verwandter Projekte weiter unten). Diese verfolgen aber zumeist weniger das oben festgelegte Ziel, AnfängerInnen den Einstieg in erste Erfahrungen mit eigenständigem Beweisen zu vermitteln, sondern dienen eher dem Zweck, formale Darstellungsformate für mathematische Beweise ein- und Studierende an formale und automatisierbare Mathematik heranzuführen. Die derzeit wohl bekanntesten automatischen Systeme zum automatischen Beweisprüfen bzw. Beweisassistenten sind Isabelle, Coq, LEAN und Mizar. Coq bildet u.a. die Grundlage des didaktischen Systems Edukera, während LEAN im Lehrprojekt von Avigad, Lewis, van Doorn (siehe [137]) eingesetzt wird. Beide werden unten ausführlicher besprochen.

Indes sind Systeme zur die automatische Beweisverifikation wie Mizar, Coq, Isabelle oder Lean für den Einsatz im Anfängerbereich ungeeignet. Ihre streng reglementierten, in Syntax und Erscheinung eher an eine Programmiersprachen erinnernden Eingabesprachen erfordern es von dem oder der BenutzerIn, sowohl diese Sprache wie auch die zugrundeliegende Mathematik sehr gut zu beherrschen. Nur so ist es möglich, einen mathematischen Gedankengang so gut zu durchdringen, dass man ihn in den jeweiligen konzeptionellen Rahmen des Systems übersetzen kannn. Schon ein kurzer Blick auf einfache, im Rahmen dieser Systeme formalisierte Beweise genügt, um zu sehen, dass sie in dieser Form für den oder die AnfängerIn ungeeignet sind – steht doch diesem oder dieser weder ein fundiertes Wissen über Beweisstrategien und deren Austauschbarkeit zur Verfügung (etwa die Möglichkeit, Beweise mit klassischer Induktion, starker Induktion oder mithilfe des Prinzips von der kleinsten Zahl durcheinander auszudrücken), noch die erforderliche Virtuosität in der Verwendung logischer Formalismen.[27]

[27] Andererseits können solche Systeme zum vertieften Verständnis und zur Schulung der Formalisierungsfähigkeit beitragen, wenn die ersten Hürden gemeistert sind, ein solider Grundstock an Stoff inhaltlich, methodisch und heuristisch beherrscht wird und es nun darum geht, diesen Wissenschatz in der Rückschau – etwa im Rahmen einer Logikvorlesung – wiederum

Für ein didaktisch im Anfängerbereich hilfreiches System ergibt sich damit folgende Anforderung: Die zu prüfenden Beweise sollten in einer Form geschrieben werden können, die einem/r AnfängerIn zugänglich ist, und zugleich nur in einer Form akzeptiert werden, die im Rahmen etwa eines Übungsbetriebs als formal korrekt gelten kann. Das Vorbild für die zu akzeptierenden Eingabetexte wären also formal und inhaltlich fehlerfreie Lösungen für Übungs- oder Klausuraufgaben. Das erfordert insbesondere, dass Eingabetexte in natürlicher Sprache eingegeben werden können. Dies mag auf den ersten Blick Anlass zur Sorge bezüglich der Umsetzbarkeit geben; ein Blick auf typische Bearbeitungen von Anfängerübungen zeigt aber, dass die erforderliche Sprache bezüglich Vokabular und grammatischer Form recht begrenzt ist.

Nun ist auch die Nähe zur natürlichen Sprache ein Ziel, das den EntwicklerInnen von Umgebungen für das automatische Beweisprüfen nicht fremd ist; beispielsweise sind Naproche und SAD automatische Beweisprüfer, die mit dem Ziel entwickelt werden, die Eingabesprache an die natürliche mathematische Sprache anzunähern und damit z.T. beeindruckende Erfolge zu verzeichnen haben. Aus Gründen, die wir weiter unten erläutern werden, eignen sich aber auch diese Systeme nicht für den Einsatz in der Lehre im Anfängerbereich. Das in der vorliegenden Arbeit entwickelte System ist gerade dadurch entstanden, dass das Grundkonzept von Naproche für den didaktischen Kontext angepasst wurde – was allerdings auf sämtlichen Ebenen so weitreichende Folgen hat, dass ein von Grund auf neues System geplant und implementiert werden musste.

systematisch einzuholen. Lehrveranstaltungen mit dieser Zielsetzung sind auch in der Tat konzipiert und mit Erfolg durchgeführt worden, siehe etwa den Abschnitt weiter unten zur Verwendung von LEAN in der Lehre durch Avigad et al. [137].

Kapitel 3

Abgrenzung zu einigen bestehenden oder in Entwicklung befindlichen Systemen

In diesem Abschnitt werden wir einige der zahlreichen Projekte betrachten, automatisches Beweisprüfen für die (universitäre oder auch schulische) mathematische Lehre fruchtbar zu machen. Wir streben dabei keine Vollständigkeit an; wohl aber versuchen wir, die vorhandenen Ansätze so weit abzubilden, dass sich klar zeigt, wo ggf. Lücken bestehen, die ein neues System füllen könnte. In jedem Fall besprechen wir im Anschluss an eine Darstellung des betreffenden Systems kurz, was aus dem jeweiligen Ansatz zu lernen ist und inwiefern Diproche gegenüber dem fraglichen System einen neuen Beitrag darstellt.

Die Kriterien, an denen sich die folgenden Betrachtungen orientieren, sind vor allem folgende:

- Nähe der einzugebenden Beweise zu "echten", informellen Beweisen; Natürlichkeit der Eingabesprache
- Integrierbarkeit in den Lehrbetrieb
- Erweiterbarkeit um weitere Aufgaben bzw. Themenkreise

Der Ansatz von Diproche ist es, die automatisierte Überprüfung natürlichsprachlicher mathematischer Beweise als didaktisches Hilfsmittel zur Unterstützung des Erlernens des eigenständigen mathematischen Beweises nutzbar zu machen. Naturgemäß wird das System sich also an solchen bereits bestehenden oder in Entwicklung befindliche Systemen messen lassen bzw. mit ihnen vergleichen müssen, die (i) die automatische Prüfung natürlichsprachlicher Beweise zum Ziel haben (wie VIP, Naproche oder SAD) oder (ii) automatisches Beweisprüfen im allgemeinsten Sinn zum Vermitteln von Beweiskompetenzen einsetzen, wie Concludio, QED, Edukera etc. (s.u.).

Unseres Wissens nach ist Diproche der erste volle Versuch einer Synthese der beiden Ansätze. Die uns bekannten Systeme der zweiten Art lassen sich grob in zwei Klassen einteilen: "Regelbasierte" Systeme stellen – gewöhnlich über ein Auswahlmenü – einen begrenzten Vorrat an formalen Beweisregeln oder Ableitungsmöglichkeiten zur Verfügung, die auf Voraussetzungen oder bereits abgeleitete Ergebnisse angewendet werden können. Von dieser Art sind etwa Edukera, GEOBEWEIS oder QED. Der zweite Typ sind die "Lückentextsysteme", bei denen ein bereits mehr oder weniger vorhandenes Beweistemplate an gewissen Stellen entweder mit vorgegebenen Textbausteinen auszufüllen oder durch freie Eingabe etwa eines Terms zu ergänzen ist (ggf. auch beides zugleich). Zu diesem Typ gehört z.B. E-Proofs. Eine Angabe von Herleitungsregeln ist bei diesen Systemen – durchaus in weitgehender Übereinstimmug mit der mathematische Darstellungspraxis – nicht oder nur dort erforderlich, wo es durch den Lückentext explizit gefordert wird. Ein System, das eine regelbasierte Vorgehensweise mit weitgehend frei handhabbaren Textbausteinen kombiniert ist "Concludio" (s.u.).

Quer zu dieser Einteilung verläuft die Unterscheidung zwischen zwei Richtungen, die wir die "logische" und die "inhaltliche" nennen wollen. "Logische" Systeme wie etwa Tao's QED beginnen mit und sind dominiert von der Herleitung rein logischer Formeln, typischerweise ausgehend von der Aussagenlogik und dann durch Einführung von Quantoren fortschreitend zur erststufigen Prädikatenlogik. "Inhaltliche" Systeme dagegen führen anhand eines Gegenstandsbereiches in das mathematische Argumentieren ein, der eine Form von herleitungsunabhängiger Einsicht bzw. Anschauung erlaubt: Im Fall von GEOBEWEIS und QED-Tutrix etwa der Elementargeometrie. Hier scheint in der Tat eine didaktische Grundsatzentscheidung vorzuliegen: Hat der zweite Ansatz den Vorteil, dass Argumentationsketten noch von der Anschauung gestützt, durch sie motiviert und an ihr überprüfbar sind, hilft der zweite dabei, die allgemeinen Strukturen, die mathematischen Beweisen aus allen Themengebieten gemeinsam sind, herauszuarbeiten und ins Bewußtsein zu heben – freilich um den Preis der Gefahr, dass das Beweisen als ein reines Formelspiel ohne inhaltlichen Sinn erscheint. Sicherlich sind beide Aspekte – das Aufmerken auf den Zusammenhang zwischen Anschauung und Versprachlichung ebenso wie die Kenntnis allgemeiner Argumentationsmuster – für ein fundiertes Verständnis des mathematischen Beweisens wichtig. In welcher Reihenfolge ihre Vermittlung zu erfolgen hat, ist eine Frage, auf die der letztliche Erfolg der verschiedenen in diesem Abschnitt vorgestellten Systeme sicherlich ein Licht werfen wird.

Jedenfalls ist jedes Logik-basierte System, das den Anspruch verfolgt, "natürliche" Beweise automatisch prüfen zu können, mit der Herausforderung konfrontiert, ein auf einem inhaltlichen Verständnis von Gegenständen bzw. Zusammenhängen basierendes einsichtiges Argumentieren durch das Anwenden

spezifischer formaler Regeln möglichst getreu abzubilden. Dieser Herausforderung stellt sich Diproche durch den Versuch, für möglichst viele (im Idealfall: alle) der im jeweiligen Themenfeld und Schwierigkeitsgrad zulässigen Beweisschritte eine entsprechende Regel zur Verfügung zu haben, anhand derer der entsprechende Schritt verifizierbar wird. Auf diese Weise wird es dem/der BenutzerIn ermöglicht, ohne einen Gedanken an Schlussregeln zu argumentieren und nicht nachvollziehbare Schritte ggf. durch Zwischenschritte zu ergänzen. Diese Möglichkeit bietet kein anderes der uns bekannten auf die Lehre ausgerichteten Systeme.

3.1 Diskussion einiger Systeme und Projekte zum natürlichsprachlichen Beweisprüfen und zum computergestützten Erwerb von Beweiskompetenzen

3.1.1 VIP

Das von Klaus Zinn im Rahmen seiner Dissertation "Understanding Informal Mathematical Discourse" [217] entwickelte System VIP ("Verifying Informal Proofs") darf wohl als der erste systematische Versuch gelten, natürlichsprachliche mathematische Beweise automatisch überprüfbar zu machen. Die konkrete Zielsetzung des Projektes war die automatische Verarbeitung von Beweisen, die in dem berühmten Lehrbuch der Zahlentheorie von Hardy und Wright [102] vorkommen.

Im Zuge des Projektes wurden wichtige Techniken entwickelt, die für weitere Projekte wie Naproche und auch Diproche richtungsweisend waren. Hierzu gehört z.B. die Verwendung von Techniken aus der formalen Linguistik zur Erzeugung eines Zwischenformates zwischen natürlichsprachlichem Text und Formalisierung durch eine Modifikation der Diskursrepräsentationsstrukturen (DRS) von Kamp und Reyle [128], die sogenannten "Proof Representation Structures" (PRS).[1] Von diesem Format aus wurden formale Ziele für einen automatischen Beweiser generiert, dessen Erfolg oder Misserfolg dann als Kriterium für die Korrektheit eines Beweisschrittes verwendet wird. Ein interessanter Ansatz ist die Verwendung von Beweisplänen: Hierbei werden aus einem gegebenen Text Hinweise auf die Verwendung einer gewissen Beweisform (Widerspruch, Fallunterscheidung, Induktion...) extrahiert und anschließend versucht, Textteile den Leerstellen des (bzw. der) zugehörigen "Frames" zuzuordnen.

[1] Auch für das Naproche-Projekt (s.u.) spielt ein als PRS bezeichnetes Zwischenformat eine wichtige Rolle. Allerdings sind beide Formate nicht identisch. In Diproche finden PRSs ihre Entsprechung im "annotierten Format" (s.u.).

Im Fazit muss indes festgestellt werden, dass das Projekt sein äußerst ehrgeiziges Ziel nicht erreicht hat. Trotz aller Vorteile, die die mathematische Fachsprache gegenüber der natürlichen Sprache als ganzer für eine automatische Erfassung bietet (z.B. semantische und syntaktische Präzision, grammatische Einfachheit, begrenztes Vokabular), ist ein Text wie der von Hardy und Wright dennoch von einer sprachlichen wie inhaltlichen Komplexität, die formale Ansätze jedenfalls derzeit überfordert. Eine Lektion, die aus dem Verlauf des VIP-Projektes gezogen werden kann und im Rahmen des Naproche-Projektes auch gezogen wurde ist die, dass eine automatische Erfassung sich auf ein klar definiertes und kontrolliertes Fragment der natürlichen Sprache, eine sogenannte "Controlled Natural Language" (CNL) beschränken und nicht versuchen sollte, sämtliche sprachlichen Figuren und Phänomene aus einem derart reichhaltigen Korpus abzubilden.

Was die didaktischen Einsatzmöglichkeiten von VIP angeht, so ist zunächst festzustellen, dass VIP eine solche Zielsetzung nie verfolgt hat. Da letztlich nur wenig Texte einer Überprüfung durch VIP zugänglich geworden sind und überdies für eine/n uninformierte/n BenutzerIn kaum abzusehen ist, ob ein Text sprachlich vom System erfasst werden kann, ist es als Instrument zum Beweisenlernen sicherlich ungeeignet.[2] Im Übrigen treffen auf VIP sämtliche Vorbehalte zu, die unten hinsichtlich eines didaktischen Einsatzes von Naproche besprochen werden.

3.1.2 Naproche

Das Naproche-Projekt ("Natural Language Proof Checking") ist ein interdisziplinäres Projekt mit Bezügen zu und Beteiligten aus den Gebieten der Mathematik (besonders der mathematischen Logik), der (formalen) Linguistik, der Informatik und der Philosophie. Ziel des Projektes ist die Entwicklung des Naproche-Systems, das Beweise, die in einem kontrollierten Fragment der natürlichen mathematischen Sprache (in Englisch) eingegeben werden, auf Korrektheit überprüfen soll. Ein wesentlicher Schritt zu diesem Ziel war die Entwicklung der Naproche Controlled Natural Language (CNL). Das Projekt wurde von Peter Koepke an der Universität Bonn initiiert und von Bernhard Schröder und Bernhard Fisseni von der linguistischen Seite entwickelt. Die konkrete Ausgestaltung und Implementierung des Systems ist vor allem durch Marcos Cramer im Rahmen seiner Dissertation [53] erfolgt; weitere wichtige EntwicklerInnen sind Daniel Kühlwein, Nikolay Kolev, Dörthe Arndt und andere.

[2] Siehe z.B. Zinn, "Understanding Informal Mathematical Discourse" [217], S. 182: "At the time of writing, we are only aware of Vip being able to completely process the two example constructions in ch. 7. Vip is thus a proof of concept, rather than a usable system ready for deployment. We have laid a solid ground work and demonstrated that its key technology works."

Das Naproche-System ist online verfügbar[3] und ist in zahlreichen Publikationen beschrieben, siehe etwa [70], [60], [45], [47], [8], [40], [46], [216], [44], [54]. Einer der ersten längeren Texte, die von Naproche verarbeitet werden konnten, war das erste Kapitel von Edmund Landaus "Grundlagen der Analysis" (siehe [130] bzw. [43] für den Naproche-Text), das bereits im Rahmen des Automath-Projektes (s.o.) für eine automatisierte Prüfung aufbereitet worden war (allerdings in einer formalen Fassung ohne die Verwendung natürlicher Sprache). Zuletzt hat das System durch die Verbindung mit SAD und der dazugehörigen CNL ForThel (siehe z.B. [159]) erhebliche Fortschritte gemacht.

Das Naproche-System war, wie schon in der Einleitung erwähnt, Motivation und Vorbild für Diproche; durch seine zeitweise Mitarbeit am Naproche-Projekt hat der Autor einen detaillierten Einblick in das Naproche-System erhalten. Entsprechend hat Diproche sowohl in seiner grundsätzlichen Zielsetzung – der Überprüfung von in einem kontrollierten Fragment der natürlichen Sprache eingegebenen mathematischen Beweisen auf Korrektheit – wie auch hinsichtlich der Systemarchitektur und der verwendeten Datenformate (wie den "Proof Representation Structures") wesentliche Ähnlichkeiten mit dem Naproche-System. Das angestrebte Einsatzgebiet von Diproche – das Erlernen elementarer Beweistechniken im Anfängerbereich des Hochschulstudiums von Mathematik, Lehramt Mathematik sowie angrenzenden Disziplinen wie Informatik, Wirtschaftswissenschaften oder Physik – weicht jedoch wesentlich von dem von Naproche ab; bei Naproche geht es zum einen um die Untersuchung des Verhältnisses zwischen formalen und natürlich(sprachlich)en Beweisen, zum anderen um eine Unterstützung von ExpertInnen bei der Herstellung automatisch verifizierbarer Beweise. Im Gegensatz dazu soll Diproche gegenüber im Beweisen unerfahrenen AnfängerInnen die Rolle eines "Tutors" übernehmen, der vom Studierenden vorgeschlagene Beweise prüft, Rückmeldungen gibt und auch Hinweise geben kann, wie weiter zu verfahren ist. Daraus ergeben sich wesentliche Unterschiede zu Naproche, von denen wir hier nur einige auflisten:[4][5]

[3] Nämlich unter `https://korpora-exp.zim.uni-duisburg-essen.de/naproche/`, zugegriffen 11.10.2021.

[4] Wir weisen an dieser Stelle darauf hin, dass Punkte wie die unten genannten, die Grenzen der didaktischen Verwendbarkeit von Naproche aufzeigen, keine Kritik am Naproche-System darstellen sollen; das Naproche-Projekt verfolgt schließlich keine didaktische Zielsetzung und es ist im Gegenteil bemerkenswert und ein Argument **für** den Ansatz von Naproche, dass die dort gewonnenen Ansätze und Einsichten in einer so deutlich anderen Domäne wie der didaktischen verwendbar sind. Dieser Hinweis gilt entsprechend für die übrigen hier besprochenen Systeme zum automatischen Beweisprüfen ohne spezifisch didaktische Zielsetzung, wie SAD, Naproche-SAD oder auch VIP (s.o.).

[5] Im Sommersemester 2021, nachdem die vorliegende Arbeit weitgehen abgeschlossen war, wurde Naproche-SAD an der Universität Bonn im Rahmen einer Vorlesung zur mathematischen Logik eingesetzt. Ähnlich wie in den weiter unten besprochenen Lehrprojekten unter Verwendung

- Da Diproche in deutschen Hochschulen im Anfängerstudium eingesetzt werden soll, ist die Eingabesprache für Diproche Deutsch. Für Naproche ist es Englisch.

- Naproche meldet zwar nicht verifizierbare Beweiszeilen, gibt aber keine differenzierte Rückmeldung. Diproche soll deutlich mehr Information geben, u.a.: Ist ein Schritt logisch schlüssig? Ist ein Satz im Rahmen der kontrollierten natürlichen Sprache korrekt gebildet? Ist er logisch sinnvoll (z.B.: hat das, was behauptet oder angenommen wird, überhaupt einen Wahrheitswert oder handelt es sich etwa um eine Menge oder eine Zahl? Sind alle auftretenden Referenten zuvor eingeführt worden)? Erreicht der Beweis das anvisierte Ziel?

- Naproche geht von einem/r BenutzerIn aus, der/die ein mathematisches Gebiet als Experte/in beherrscht und nun an einer Formalisierung von Beweisen interessiert ist; im Gegensatz dazu soll Diproche Hinweise geben, wie ein Beweis, abhängig vom Beweisziel und den bisherigen Schritten, fortgesetzt werden könnte, und das bereits auf sehr basalem Level ("wenn das Beweisziel eine Konjunktion ist, zeige ihre Glieder einzeln"; "wenn das Beweisziel eine Implikation ist, nimm die Prämisse an und versuche die Konklusion zu zeigen").

- Der Beweiser, der von Naproche zur Verifikation von Beweisschritten verwendet wird, ist stark genug, um viele der typischen Übungsaufgaben für AnfängerInnen ohne weitere Ergänzungen selbst zu lösen. Im Gegensatz dazu soll Diproche nur Schritte aus einem begrenzten Vorrat von elementaren Beweisschritten akzeptieren. Hierbei ist die Frage zu klären, was im Sinne einer Übungsaufgabe als elementarer Beweisschritt gelten kann.

- Naproche ist darauf ausgelegt, längere Texte mit zahlreichen Definitionen, Sätzen und Beweisen zu prüfen; dafür sind insbesondere eine dynamische Anpassung des Vokabulars, eine geschickte Auswahl der relevanten Prämissen und allgemein Behutsamkeit hinsichtlich Komplexitätsfragen erforderlich. Diproche ist auf Beweise von maximal einigen Dutzend Zeilen Länge in einem jeweils fixen Begriffsrahmen ausgelegt.

- Der Ansatz von Naproche ist es, einen Beweistext zu akzeptieren, sobald er dem im Hintergrund arbeitenden automatischen Theorembeweiser (ATP) genügend Hinweise liefert, um das Beweisziel zu erreichen. Dadurch werden auch Texte akzeptiert, die im herkömmlichen Sinn nicht als Beweise gelten

von LEAN oder Coq unterscheidet sich aber auch dieser Einsatz deutlich von der Verwendung im Anfängerbereich.

würden, etwa, weil sie abbrechen, ohne das Beweisziel explizit erreicht zu haben.[6] Im Gegensatz dazu soll Diproche den gegebenen Text sehr genau interpretieren und dabei auch auf Stilfragen achten: Wurden Beweisziele angekündigt? Wurde ihre Erreichung expliziert? etc.

- Die kontrollierte natürliche Sprache von Diproche ist speziell darauf ausgelegt, die Ausdrucksformen abzubilden, in denen Lösungen für Übungsaufgaben im Anfängerbereich typischerweise formuliert werden (sollten). Dazu gehören insbesondere Zeichen wie "$\rightarrow$" oder "$\subseteq$", um anzuzeigen, dass im Rahmen eines Äquivalenzbeweises nun eine Implikation in eine Richtung bewiesen werden soll oder im Rahmen eines Gleichheitsbeweises für Mengen eine Inklusion. Dazu gehört ferner die Integration von Textphänomenen wie Nebenrechnungen oder Termumformungs- bzw. Mengeninklusionsketten sowie ATP-Routinen, die für deren Verifikation zuständig sind. So sollte ein Satz wie "Es gilt $0 \leq (a-b)^2 = (a-b)(a-b) = a^2 - ab - ba + b^2 = a^2 - 2ab + b^2$" im Rahmen von Diproche ohne weiteres ebenso verwendbar sein wie etwa "Nun haben wir $2a + 6 = 7 \Leftrightarrow [-6]2a = 1 \Leftrightarrow [:2]a = \frac{1}{2}$".

- Um seiner Funktion als "Beweistutor" gerecht zu werden, verfügt Diproche über zahlreiche didaktische Zusatzfunktionen, die es in Naproche nicht gibt (und die auch nicht in den Rahmen von Naproche passen würden). Dazu gehören u.a. die Fehlschlussdiagnose, die bei nicht nachvollziehbaren Schritten prüft, ob diese durch einen bekannten formalen Fehlschluss erklärbar sind und ggf. eine entsprechende Rückmeldung liefert, ein "Tippgeber", der dem/der BenutzerIn Lösungshinweise gibt bzw. Zwischenschritte vorschlägt, eine Datenbank mit Übungsaufgaben sowie ein Aufgabengenerator versehen sein, der z.B. in den Bereichen Aussagenlogik, Boolesche Mengenlehre und vollständige Induktion auf Wunsch automatisch weitere Übungsaufgaben generiert.

Diproche kann also als didaktischer "Ableger" des Naproche-Systems betrachtet werden. Da die verschobene Zielsetzung sich auf sämtlichen Ebenen des Systems auswirkt, war es erforderlich, sämtliche Module von Diproche von Grund auf neu aufzubauen. Diproche verwendet also die Leitidee und Architektur, aber nichts von der Implementierung von Naproche.

[6] Peter Koepke, der das Naproche-Projekt initiiert hat und leitet, erwähnte in einem Gespräch einmal, dass etwa ein Satz der Analysis, der als solcher vom ATP noch nicht verifiziert werden konnte, von Naproche als bewiesen akzeptiert wurde, nachdem der Satz "Es sei $\varepsilon > 0$" ergänzt worden war – und sonst nichts.

3.1.3 SAD

SAD, das "System for Automated Deduction", ist ein von Andrey Paskevich u.a. im Rahmen seiner Dissertation [158][7] entwickeltes System zur automatischen Verifikation von in Englisch verfassten natürlichsprachlichen Beweisen. Wie Naproche stellt es ein kontrolliertes Fragment der (fachmathematischen) englischen Sprache zur Verfügung, in dem Beweise in einer Form verfasst werden können, die einer typischen Darstellung etwa im Rahmen von Lehrbüchern oder Vorlesungsskripten recht nahe kommt. Dieses Fragment ist "ForTheL", die "Formal Theory Language"; für eine Beschreibung von ForThel und einige Textbeispiele siehe Paskevich [158].

Das Ziel ist dabei ausdrücklich die Einführung einer "formal input language which must be close to natural mathematical language and easy to use" (siehe [199], S. 2). Auch hier ist der Ansatz eine Übersetzung von quasi-natürlicher Sprache in mathematischen Formalismus, der dann von einem automatischen Theorembeweiser weiterverarbeitet wird. Im Unterschiede zu Naproche findet allerdings kaum im engeren Sinn linguistisch zu nennende Verarbeitung statt; stattdessen wird mithilfe einer formalen Grammatik der Text recht unmittelbar in erststufige Prädikatenlogik übersetzt. Die Verifikation selbst dagegen erfolgt mit erheblicher methodischer Raffinesse: Neben den eigentlichen ATP kommen u.a. ein "evidence collector" und ein "reasoner" zum Einsatz, die hilfreiche Informationen (etwa Typenzuweisungen) und heuristische Hinweise sammeln und für die Verifikation nutzbar machen; siehe dazu etwa [199].

Wie Naproche, so verfolgt auch SAD keine didaktische Zielsetzung. Es gelten damit wiederum alle Einschränkungen, die oben bezüglich eines didaktischen Einsatzes von Naproche aufgeführt wurden.

Naproche-SAD

Naproche-SAD ist, wie der Name es nahelegt, eine Kombination der Systeme Naproche und SAD; für eine Übersichtsdarstellung siehe [59] oder [77]. Das Naproche-SAD-System hat einige der Beschränkungen des Naproche-Systems erfolgreich überwinden können und ermöglicht nun die automatische Verifikation von mathematischen Texten zu verschiedenen Themengebieten, die gewöhnlichen Lehrbuchtexten äußerst nahe kommen. Die Möglichkeiten für einen didaktischen Einsatz der hier intendierten Art unterliegen aber weiterhin sämtlichen Einwände, die oben vorgebracht wurden.

[7] Für eine englische Kurzfassung siehe [159]; siehe ebenfalls [199].

3.1.4 E-Proofs

Das E-Proofs-System[8] ist ein digitales System zum Erwerb grundlegender Beweiskompetenzen. Ein Prototyp ist online verfügbar unter [66]. Das Entwicklerteam besteht laut Homepage des Projekts aus Engelbert Niehaus, Kathrin Winter, Melanie Platz, Jörg Rapp, Matthias Größler, Miriam Krieger und Stefanie Buchheit. Das System wurde in zahlreichen Publikationen präsentiert, siehe z.B. [154], [162]. Das E-Proofs-Projekt teilt mit Diproche die Zielsetzung, ein digitales Werkzeug zur Übung des mathematischen Argumentierens bereitzustellen.

Dabei verfolgt das E-Proofs System folgenden Ansatz: Ein fertig vorliegender Beweistext in natürlicher Sprache wird in eine Art "Puzzle" verwandeln, indem er in einzelne Sätze bzw. kleinere Bestandteile zerlegt wird. Diese werden der/dem BenutzerIn dann in einer anderen Reihenfolge oder auch als ungeordneter "Haufen" präsentiert, der oder die nun herausgefordert ist, sie wieder in die richtige Reihenfolge zu bringen. Die Korrektheitsprüfung erfolgt hierbei einfach dadurch, dass die von dem/der BenutzerIn hergestellte Reihenfolge mit der ursprünglichen verglichen wird. Als zusätzliche Hürde werden Sätze und Satzbestandteile, die kein Teil der korrekten Lösung sind, als "Distraktoren" (siehe etwa [204]) hinzugefügt. Bisweilen können Terme und Formeln in dafür vorgesehene Lücken in den Textbausteinen eingefügt werden. Beim Übergang von einem Satz zum nächsten ist i.A. eine Begründung anzugeben; solche Rechtfertigungen für Beweisschritte (z.B. Umformungsregeln oder arithmetische Gesetze wie das Kommutativitätsgesetz der Addition) können aus einer Liste ausgewählt werden. Das System prüft dann, ob die Regel tatsächlich den Übergang zum fraglichen Satz ermöglicht. Es ist für die BenutzerInnen auch möglich, eigene Regeln als "Abkürzungen" zu definieren; die Korrektheit dieser benutzerdefinierten Regeln wird aber nicht mehr vom System überprüft, sondern obliegt einem/r menschlichen TutorIn. Das System prüft in diesem Fall nur noch, ob die Regeln korrekt angewendet wurden (siehe [66]).

Der/die BenutzerIn von E-Proofs hat die Möglichkeit, sich Vorschläge für weitere Beweisschritte anzeigen zu lassen. Da das System stets mit einer vorliegenden Lösung startet, auf die/der BenutzerIn "hinzuleiten" ist, können solche Hinweise problemlos generiert werden und sind stets aufgabenspezifisch und zielführend. Ferner verfügt E-Proofs über ein Bewertungssystem, das in Anspruch genommene Hinweise sowie fehlende, fehlerhafte und überflüssige Beweisschritte und Begründungen durch Vergleich mit der "Musterlösung" automatisch erfasst und zum einen eine entsprechende Rückmeldung an den/die

[8]Das E-Proofs-System, von dem hier die Rede ist, ist nicht zu verwechseln mit dem gleichnamigen Projekt zum Einsatz digitaler Medien zur Förderung des Beweis**verstehens** an der Universität Loughborough, siehe `https://www.educationaldesigner.org/ed/volume1/issue4/article14/` (zugegriffen 14.10.2021).

BenutzerIn erzeugt, zum anderen aber auch eine Prozentzahl ausgibt, die ein Maß für die Vollständigkeit und Korrektheit der bisher angegebenen Lösung darstellt.

Diese Möglichkeiten, beim Bearbeiten einer Aufgabe Hilfestellung ebenso zu erhalten wie eine Rückmeldung zum eigenen Fortschritt sind bemerkenswerte und didaktisch wertvolle Beiträge des E-Proofs-Systems.

Ein klarer Vorteil des E-Proof-Systems gegenüber den übrigen hier besprochenen Systemen ist die Möglichkeit, auf einschränkende Normierung bezüglich der behandelten Gegenstände sowie der verwendeten Sprache weitgehend zu verzichten: Der Ansatz, einen vorliegenden Beweistext in ein "Puzzle" zu verwandeln und mit Distraktoren anzureichern, ist an keine Domäne gebunden, weder argumentativ noch ontologisch oder sprachlich. Entsprechend ließen sich auf diese Weise prinzipiell von einfachen Termumformungen (wofür bereits einige Beispiele vorliegen) bis zu Aufgaben der internationalen Mathematik-Olympiade Beweise zu beliebigen Themen und von beliebiger Komplexität für ein digitales Lernen zugänglich machen. Zudem ist die Eingabe denkbar einfach und intuitiv und erfordert insbesondere nicht das Erlernen einer bestimmten Syntax. Lediglich die Auswahl der Rechtfertigungsschritte kann etwas umständlich werden, wenn die Listen der verfügbaren Möglichkeiten mit zunehmendem Fortschritt umfangreicher werden.

Sehr vorteilhaft für den praktischen Einsatz ist ferner die leichte Verwendbarkeit und Einsetzbarkeit von Seiten der Lehrkräfte: Insbesondere ist es ohne Programmierkenntnisse und ohne Wissen über formale Mathematik möglich, selbst Übungsaufgaben dieser Form zu erzeugen: Lehrkräfte, die E-Proofs im Unterricht einsetzen, können es also jederzeit selbst so erweitern, dass beliebige Themengebiete behandelt werden können.

Als weiterer Vorzug kann schließlich die leichte Übertragbarkeit in andere Sprachen gelten: Ist ein geeigneter Pool von Beispielaufgaben einmal aufgebaut, brauchen die auftretenden Sätze nur übersetzt zu werden. (Im Fall von Satzfragmenten entsteht ggf. die Notwendigkeit zu kleineren weiteren Anpassungen.) Änderungen an der Software selbst sind hingegen nicht erforderlich. Im Gegensatz dazu ist bei Systemen mit einer Eingabe in natürlicher Sprache – wie Diproche, Naproche oder SAD – stets eine Modifikation derjenigen Module erforderlich, die für die Sprachverarbeitung zuständig sind, insbesondere also der formalen Grammatik.[9]

Diesen Vorteilen stehen einige Nachteile gegenüber. Der erste, den die

[9]Eine potenzielle Erleichterung solcher Übersetzungsaufgaben in derartigen Systemen könnte sich durch die Verwendung des "Grammatical Framework" (GF) ergeben, einer speziell auf derartige Zwecke hin entworfenen Programmiersprache, siehe `https://www.grammaticalframework.org/(zugegriffen14.10.2021)`. Für in GF geschriebene Grammatiken ist eine Übersetzung in andere Sprachen mit minimalem zusätzlichen Programmieraufwand durchführbar.

EntwicklerInnen auch selbst in der Diskussion in [67] erwähnen, besteht in der starken Einschränkung des Bereichs zulässiger Lösungswege. Alternative Lösungen oder auch solche, die die vorgesehenen Schritte um einige "Umwege" anreichern bzw. in anderer Reihenfolge durchschreiten werden vom System als falsch angesehen. Es ist zu erwarten, dass das mit Behandlung komplexerer Beweise zu immer größeren Schwierigkeiten führen wird (wenn etwa bei einer Fallunterscheidung die Fälle nur in einer bestimmten Reihenfolge behandelt werden können). Das System könnte durch ein etwas "liberaleres" Prüfverfahren sicher gewinnen, das etwa nur Einbettbarkeit des richtigen Beweises in die vom Benutzer vorgeschlagene Lösung prüft und dabei gewisse "Blöcke" als vertauschbar behandelt. Siehe hierzu auch erneut [67] Hier könnte eine Kombination mit einfachen ATPs zu einer deutlichen Steigerung der Möglichkeiten führen.

Ein weiterer Kritikpunkt besteht in dem durch das System nahegelegten "konvergenten" Zugang zu Beweisaufgaben: In E-Proofs gibt es für eine Aufgabe im wesentlichen nur eine richtige Lösung. Für die Bearbeitung einer im Rahmen von E-Proofs gestellten "Textsortieraufgabe" dürfte es im Allgemeinen nicht einmal besonders hilfreich sein, die Aufgabe selbstständig gelöst zu haben, solange der eigene Lösungsweg nicht inhaltlich exakt der vorgesehenen Musterlösung entspricht und zusätzlich in ähnlicher Weise aufgeschrieben wurde. Die Bearbeitung einer Aufgabe erfolgt im Rahmen von E-Proofs stets entlang der Fragmente einer bereits vorliegenden Lösung. Das für das mathematische Arbeiten doch wesentliche selbstständige Erkunden, Planen, Probieren etc. (vgl. [164]) wird also durch den Ansatz von E-Proofs so weniger unterstützt, was den eingangs genannten Vorteil der Domänenunabhängigkeit zum Teil konterkariert. So dürfte E-Proofs eher zur Förderung des Beweislesens und -verstehens beitragen als zur Vermittlung eigenständiger Beweiskompetenzen.

Ferner kann E-Proofs den Erwerb der Fähigkeit zum korrekten Ausformulieren einer Lösung allenfalls indirekt über eine Art Vorbildwirkung unterstützen, da die zur Darstellung einer Lösung nötigen Formulieren ja bereits durchweg vorgegeben sind. Der Übergang von der Auswahl passender Schritte zum korrekten Gebrauch der Fachsprache, dem eigenständigen Schreiben und präzisen Formulieren dürfte indes ein erheblicher Schritt sein.

Ein analoges Problem findet sich auf höherer Textebene darin wieder, dass E-Proofs das einer sprachlichen Darstellung vorausgehende Strukturieren eines Beweisgedankens ebenfalls wenig fördert. Zwar erfordert die Aufgabe, die Bestandteile eines Beweises in die korrekte Reihenfolge zu bringen, ein Verständnis für den Aufbau dieses Beweises. Die schwierigere Aufgabe, einen noch nicht sprachlich ausformulierten Gedankengang so in Schritte zu gliedern, dass er einer nachvollziehbaren sprachlichen Darstellung fähig wird, stellt sich BenutzerInnen hingegen nicht.

Das E-Proofs-System dürfte sich nach unserer Einschätzung in gewissen Anwendungskontexten als durchaus nützlich erweisen und birgt im Übrigen ein interessantes technisches Entwicklungspotenzial. Für die Entwicklung einiger der Fähigkeiten, auf deren Förderung Diproche abzielt, scheint das System durch seine starke Beschränkung des Raums akzeptierter Lösungen jedoch nur wenig Spielraum zu bieten.

3.1.5 ETPS

Das System EPTS (Educational Theorem Proving System, siehe Andrews, Bishop, Brown, Pfenning, Xi [2]) ist ein automatisches System, mit dem BenutzerInnen eigene Beweise in einem formallogischen Ableitungskalkül auf Korrektheit überprüfen können. Beweise werden hierbei dargestellt als Folgen von Formeln, wobei Übergänge durch die Angabe einer passenden Deduktionsregel zu rechtfertigen sind. Das System kann somit primär zum Erlernen formaler Beweiskalküle verwendet werden. Der Mangel an einer natürlichen Eingabesprache und die strikte Regelbasiertheit sprechen gegen einen Einsatz im Anfängerbereich des Mathematikstudiums: Einerseits ist von der Benutzung kein verbesserter Erwerb der mathematischen Fachsprache zu erwarten, andererseits ist eine fundierte Beherrschung von Formalisierungstechniken für die Verwendung des Systems erforderlich. Ein didaktisch interessanter Aspekt von ETPS ist der interaktive Charakter: Das System kann sowohl naheliegende Beweisschritte vorschlagen als auch Rückmeldungen dazu geben, ob gewisse von dem/der BenutzerIn vorgeschlagene Beweisschritte voraussichtlich zielführend sind. Ferner verfügt das System über eine "'CLEANUP"-Funktion, die aus einem fertigen Beweis alle Schritte entfernt, die logisch nicht für den Beweis benötigt werden ([2], S. 6). Im Rahmen von Diproche gibt es die entsprechenden Funktionen des "Tippgebers" sowie den – bisher noch nicht implementierten – Ansatz zu einer automatischen "Beweisanalyse".

3.1.6 Edukera

"Edukera" ist ein System, das von einem gleichnamigen Unternehmen unter der Leitung von Benoit Rognier an der University of Nottingham entwickelt wurde. Auch das "Edukera"-System verfolgt das Ziel, StudienanfängerInnen durch einen automatischen Beweisprüfer den Einstieg in das eigenständige mathematische Beweisen zu erleichtern. In dieser Funktion wird es bereits an diversen Pariser Universitäten eingesetzt.

Der Formalismus, der dem Edukera-System zugrunde liegt, ähnelt Gentzens System des natürlichen Schließens [86], [87], angereichert durch einige zusätzliche Regeln. BenutzerInnen steht eine Reihe von formalen Beweisregeln zur Verfügung

– wie etwa die Einführung der Konjunktion oder der Modus Ponens – und ein Beweis wird dadurch aufgebaut, dass man zunächst eine der in einer Liste aufgeführten Regeln anklickt und danach die Beweiszeilen, auf die sie angewandt wird. Wären etwa die Formeln a und $a \to b$ bereits erzeugt worden und stünden in den Zeile 3 und 7, so könnte man auf "Modus Ponens" klicken, gefolgt von Zeile 3 und Zeile 7. Ist die Beweisregel auf die fraglichen Zeilen anwendbar, so wird die dadurch entstehende Formel als nächste Beweiszeile hinzugefügt. Falls nicht, wird eine Fehlermeldung ausgegeben. Der Beweis gilt als abgeschlossen, wenn die als Beweisziel vorgegebene Formel auf diese Weise erzeugt wurde.

Das Prüfsystem von Edukera ist eingebunden in ein interaktives Lernprogramm, das die Beweisregeln sukzessive einführt und ihr Verständnis durch passende Übungsaufgaben festigt und prüft. Spätere Lektionen können erst abgerufen werden, wenn die Übungen zu den frühren erfolgreich gelöst wurden. Das System hat ein Webinterface, das über `https://www.edukera.com/`[10] erreicht werden kann.[11]

Auf diese Weise wird die Funktion verschiedener Beweisregeln und -strategien (etwa Fallunterscheidung oder Widerspruchsbeweis) vermittelt. Darüber hinaus gibt es zahlreiche Übungen zu Themen wie vollständiger Induktion, Konvergenz und Grenzwerten, Folgen und Reihen etc.

Die Erläuterungen des Edukera-Systems sind in Englisch verfasst, allerdings erfolgt die Eingabe von Beweisen strikt formal und frei von natürlicher Sprache. Erst nach Eingabe wird die Auswahl in Textform übersetzt, indem etwa ein Text wie "by addition of 1 on both sides" neben einer Aussage als Rechtfertigung für eine Folgerung angezeigt wird. Das Edukera-System in andere Sprachen zu übertragen erfordert also nur eine Übersetzung der Lehrtexte.

Die Eingabe bei Edukera ist von der üblichen mathematischen Darstellungspraxis weit entfernt. Insbesondere wird zur Eingabe keine natürliche Sprache verwendet, so dass in Bezug auf die Handhabung der mathematischen Fachsprache nur ein geringer Übungseffekt zu erwarten ist. Auch umgekehrt gilt: Wer bereits "natürlich" beweisen kann, muss die Handhabung von Edukera erst noch lernen. Das "Zusammenklicken" des Beweises anhand eines vorgegebenen Satzes von Regeln unterscheidet sich deutlich von der üblichen Art, einen Beweis zu finden und sprachlich darzustellen. Insbesondere erfordert sie es, eine Reihe natürlicher Schritte in formale Teilschritte zu zerlegen, die jeweils einer benennbaren Regeln folgen. Ein Nachteil dieser Eingabeform besteht darin, dass BenutzerInnen im Grunde kaum Gelegenheiten haben, Fehler zu machen: Die

[10] Zugriff erfolgreich am 03.04.2021; beim letzten Zugriffsversuch am 14.10.2021 war diese URL nicht mehr aktiv.

[11] Einen Eindruck über die Funktionsweise des Systems gibt auch das folgende Lehrvideo, in dem ein Beweis der Gaußschen Summenformel mit Edukera vorgeführt wird: `https://www.youtube.com/watch?v=ek3WMCJThwI` (zugegriffen 08.07.2020).

Auswahl einer Schlussregel führt zu deren automatischer Anwendung auf die ausgewählten Prämissen, wodurch sich stets eine korrekte Folgerung ergibt. Somit ist es zwar möglich, nicht zielführende Folgerungsschritte einzugeben, nicht aber solche, die auf einer fehlerhaften Anwendung einer Schlussregel beruhen oder gar nicht durch eine Schlussregel gerechtfertigt sind. An den Stellen, wo Terme oder Formeln von dem/der BenutzerIn selbst eingegeben werden können, erfolgt die Eingabe wiederum menügeführt, so dass es nicht möglich ist, syntaktisch nicht wohlgeformte Ausdrücke zu bilden. Diese Leitung hat gewiss ihre Vorzüge; stehen mögliche Beweis- und Umformungsschritte jederzeit in einem Auswahlmenü gut sichtbar zur Verfügung, gerät der oder die BenutzerIn z.B. nicht so leicht in die Lage, dass sie oder er "nicht mehr weiter weiß" und in Untätigkeit verfällt oder aufgibt; es ist leichter, gewisse Schritte auszuprobieren und die Hemmung, das zu tun, mag dadurch vermindert sein, dass eine vorgegebene auswählbare Option als bereits vom System sanktioniert erscheint. Wo aber andererseits gewisse Fehler gar nicht erst gemacht werden können, besteht auch keine Möglichkeit, entsprechende Fehlvorstellungen zu erkennen und zu korrigieren; für das Beweisen ohne digitale Unterstützung bleibt damit nur ein Übungseffekt, der sich aus der Vorbildwirkung von Edukera ergibt. Wie weit dieser reicht, wäre eine Sache noch anzustellender empirischer Untersuchungen; vorerst jedenfalls ist es sicher sinnvoll, auf ein System hinzuarbeiten, das in einem größeren Umfang Fehler zuläßt als Edukera. In komplexeren Kontexten wird zudem der erforderliche Satz an konkreten Deduktionsregeln vermutlich einfach zu umfangreich sein, um mit einer expliziten Auflistung noch in einem übersichtlichen Rahmen zu bleiben. Edukera ist somit ein durchaus geeignetes Mittel, um gewisse Beweiskompetenzen zu entwickeln, fördert aber nicht den korrekten Gebrauch der natürlichen mathematischen Fachsprache und dürfte wegen der strikt kontrollierten Inferenz nur bedingt erweiterbar sein. Es scheint also sinnvoll, Systeme wie Edukera durch ein digitales Angebot mit einer freieren Eingabemöglichkeit zu *ergänzen*, indes ohne den Anspruch erheben zu wollen, sie "abzulösen" oder zu "ersetzen".

3.1.7 Concludio

Ein neueres System, das ein ähnliches Ziel wie Diproche verfolgt, ist das unter der Leitung von Fabian Grewing derzeit in der Entwicklung befindliche "Concludio". Concludio funktioniert in deutscher Sprache und ermöglicht es, Beweisaufgaben aus Bereichen der Hochschulmathematik wie etwa der Analysis zu bearbeiten. Angesichts der ähnlichen Zielsetzung bestand zeitweise eine Kooperation zwischen den Projekten, bei denen Ansätze und Ideen ausgetauscht werden.

Wie bei Edukera erfolgt die Beweiseingabe dadurch, dass Beweisschritte aus einem Menü mit Vorschlägen ausgewählt werden. Im Gegensatz zu Edukera sind diese Vorschläge aber als natürlichsprachliche Formulierungen sichtbar, etwa von

der Art “nach Induktion” oder “nun folgt”. Terme und Formeln können mithilfe eines Formeleditors auch direkt eingegeben werden, was die etwas mühsame Mauseingabe bei Edukera deutlich vereinfacht. Eine Freitexteingabe gibt es auch hier nicht, so dass viele Arten von Formulierungsfehlern nicht gemacht werden können. Beweisschritte müssen in den meisten Fällen, ähnlich wie bei Edukera, durch die Angabe expliziter Inferenz- oder Umformungsregeln gerechtfertigt werden; in einigen Kontexten können Schritte aber auch automatisch erkannt werden. Im Gegensatz zu Edukera ermöglicht die freie Eingabe von Formeln und Termen es, auch fehlerhafte Folgerungsschritte anzugeben, die dann als solche gemeldet werden. Im Zuge der Kooperation mit dem Diproche-Projekt wurde den Concludio-Entwicklern die Idee des Anti-ATP kommuniziert, die nun auch in Concludio zum Einsatz kommt.

3.1.8 GEOLOG und GEOBEWEIS

Eine auf die Vermittlung von Beweiskompetenzen im Bereich der Elementargeometrie an SchülerInnen der Mittelstufe ausgerichtetes Software ist das Programm GEOBEWEIS von Gerhard Holland, welches ein Teil des GEOLOG-Systems ist. Darstellungen finden sich z.B. in [103], [104] und [105].[12] Bei GEOBEWEIS werden dem/der BenutzerIn einfache geometrische Beweisaufgaben angezeigt, die nun mithilfe eines fixierten Vorrats an verfügbaren geometrischen Lehrsätzen zu lösen sind. Dazu können die anzuwendenden Lehrsätze mit der Maus aus einem Menü ausgewählt werden. Die Verfügbarkeit von Sätzen in den jeweiligen Aufgaben wird dabei von der Person bestimmt, die die Aufgabe gestellt hat. Die bisher vollzogenen Beweisschritte werden dabei auf dem Bildschirm in Form eines Baumes visualisiert. Ferner evaluiert das System vorgeschlagene Beweisschritte im Hinblick auf ihre Nützlichkeit und logische Folgerichtigkeit und gibt die Ergebnisse der Evaluation in Form von Punkten zurück, die der/die BenutzerIn sammeln kann (vgl. [197]).

GEOBEWEIS ist im Unterricht der Mittelstufe erfolgreich eingesetzt worden; ferner sind GEOBEWEIS-basierte Unterrichtsvorlagen entwickelt worden, siehe z.B. Lorenzen [138].

Im Vergleich zu Diproche fällt zunächst die thematische Beschränkung auf ein spezifisches Themengebiet, die Elementargeometrie, auf. Diese zeichnet sich u.a. dadurch aus, dass eine Vielzahl von inhaltlich interessanten Aufgaben mithilfe eines recht begrenzten Vorrats an Lehrsätzen (wie etwa den Kongruenzsätzen für Dreiecke) in verhältnismäßig kurzen und strategisch einfachen Beweisen

[12] Unglücklicherweise war es uns nicht möglich, das GEOLOG-System selbst auszuprobieren. Unsere Darstellung der Funktionsweise beruht daher auf den obigen Darstellungen sowie der Beschreibung in [197].

lösbar sind. (GEOBEWEIS ist auf das Vorwärtsschließen ausgelegt, also darauf, aus gegebenen Vorausetzungen durch iterierte Anwendung von Lehrsätzen solange Folgerungen abzuleiten, bis das Beweisziel erreicht ist; es gibt weder Widerspruchsbeweise, noch Fallunterscheidungen, Beweise durch Kontraposition o.ä.) Die in GEOBEWEIS zum Einsatz kommenden Techniken sind wesentlich von diesen Eigenschaften der gewählten Domäne abhängig. In anderen Bereichen, wie etwa der Aussagenlogik, wird rasch eine solche Vielzahl an Schlussweisen verwendet, dass ihre vollständige Auflistung zu einem kaum noch bedienbaren System führen würde. Schon für GEOBEWEIS bemerken Didaktiker der Universität Freiburg trotz insgesamt sehr positiver Beurteilung, dass das Auswahlsystem die Bedienbarkeit einschränkt:

"Hauptnachteil dieses Programms: Es verwendet Kürzel über Kürzel. So haben die zur Lösung erlaubten Sätze z.B. Namen wie "dk_sss" oder "wk_dk". Was sich dahinter verbirgt, kann man sich vor dem Beweis zwar anzeigen lassen, beispielsweise verbirgt sich hinter "dk_sss" der Kongruenzsatz SSS für Dreiecke, steckt man aber erst einmal mitten im Beweis, und muss nun einen passenden Satz aus der (manchmal recht langen) Liste auswählen – und die Kürzel sind zum Teil doch recht ähnlich – dann kann man manchmal einfach nur noch raten, und nimmt Minuspunkte in Kauf." ([197])

Diese Schwierigkeiten darf man sich für fortgeschrittene Aufgaben oder komplexere Kontexte potenziert denken. Dennoch wird in [197] wie auch an anderen Stellen darauf hingewiesen, dass der Umgang mit GEOBEWEIS für die SchülerInnen ein motivierender Anreiz zur Befassung mit elementargeometrischen Beweisaufgaben ist.

Ein weiterer deutlicher Unterschied zu Diproche ist der recht restriktive Eingabemodus; insbesondere ist keine Freitexteingabe möglich. Eine ganze Reihe von Fehlern, die bei der Beweisdarstellung vorkommen, sind dadurch schon von vornherein ausgeschlossen, weshalb in diesen Hinsichten auch kein Übungseffekt zu erwarten ist. Insbesondere gibt es im Rahmen von GEOBEWEIS keine Typenfehler, keine formalen Fehlschlüsse und keine Fehler bezüglich der logischen Struktur (insbesondere der Frage, welche Aussagen unter welchen Annahmen bewiesen werden).

3.1.9 QED-Tutrix

Bei dem französischen System QED-Tutrix, das von Nicolas Leduc im Rahmen seiner Dissertation [75] entwickelt wurde, handelt es sich, ähnlich wie bei GEOBEWEIS, um ein System zum Erwerben von Beweiskompetenzen im Rahmen der Elementargeometrie in der Schule.

Die Zielsetzung von QED-Tutrix beschreiben die AutorInnen wie folgt:

"1) to allow the student to freely explore the problem and its figure,
2) to accept proofs [sic!] elements in any order,"
3) to handle a variety of proofs, which can be customized by the teacher, and
4) to be able to help the student at any step of the resolution of the problem, if the need arises." ([73])

Während zumindest der "figure"-Teil von (1) ein klar Geometrie-spezifisches Ziel darstellt und Diproche sich als Beweisprüfer und Hinweisgeber, nicht aber als den einem Beweis vorangehenden Untersuchungsprozess unterstützende Software versteht, gehören (3) und (4) klar zur Zielsetzung von Diproche: Auch Diproche soll eine möglichst große Freiheit im Führen eines Beweises geben innerhalb eines methodischen Rahmens, den die Lehrperson setzen kann und in der Lage sein, sinnvolle Hilfestellungen anzubieten. Punkt (2) bezieht sich auf die Möglichkeit des/der Benutzers/In, im Rahmen von QED-Tutrix Beweisschritte in beliebiger Reihenfolge angeben zu können, wobei das System weiterhin erkennt, dass alle nötigen Schritte für einen korrekten Beweis vorliegen. Diese Möglichkeit wird sich aus der unten kurz skizzierten Beschreibung der Funktionsweise von QED-Tutrix ergeben. Für die Unterstützung eines durch die Anschauung gestützten und bisweilen "kreuz und quer" verlaufenden Untersuchungsprozesses mag das eine hilfreiche Eigenschaft des Systems sein. Im Hinblick auf das von Diproche verfolgte Ziel, insbesondere das Verfassen logisch und formal einwandfreier Beweistexte zu fördern, muss diese Möglichkeit indes eher als Fehler denn als Vorteil angesehen werden: Sofern ein Beweis dazu dient, eine/n kompetent Mitdenkende/n von der Richtigkeit einer Behauptung zu überzeugen, spielt die Reihenfolge der Darstellung sicherlich eine wichtige Rolle, um statt einer Reihe von Gedanken**sprüngen** überhaupt einen Gedankengang nachvollziehbar zu machen.

Die Funktionsweise von QED-Tutrix ähnelt der von GEOBEWEIS: Auf formalisierte geometrische Aussagen wie ("$\overline{AB}||\overline{CD}$") wird – aus einem Menü auswählbar – ein geometrischer Lehrsatz angewendet, was dann zu einem Schluss führt. Neu gegenüber GEOBEWEIS sind u.a. der automatische Tutor "Turing", der Beweisschritte kommentiert und insbesondere Fortschritte vermerkt sowie die Möglichkeit, den zunächst formal geführten und ggf. logisch nicht völlig kleinschrittig vorgehenden Beweis in eine natürlichsprachliche Fassung zu konvertieren und anzeigen zu lassen. Der automatische Tutor beurteilt dabei einerseits sowohl, ob ein vorgeschlagener Beweisschritt zielführend ist, gibt Hinweise zum Grad des Fortschritts (wie "Du hast die Hälfte geschafft" oder "Du bist fast fertig") und registriert andererseits nicht zielführende Schritte und Schwierigkeiten in der Lösung, denen er mit suggestiven Hinweisen, etwa auf Definitionen oder Lehrsätze, begegnet ([73]).

Diese bemerkenswerte Form der Interaktivität wird ermöglicht durch den Einsatz einer "Mikrowelt" ([73]), d.h. eine scharfe Begrenzung der Domäne: In jeder einzelnen Beweisaufgabe steht nur ein sehr begrenzter Vorrat an Objekten und Sätzen zur Verfügung. Das ermöglicht die Konstruktion des sogenannten "HPDIC"-Graphen, eines gerichteten Graphen, der verfügbare Annahmen, Sätze, Definitionen, Zwischenergebnisse und das Beweisziel auflistet und ihre logischen Zusammenhänge repräsentiert, etwa, dass aus einem möglichen Zwischenergebnis, auf das man im Verlauf eines Beweises stoßen könnte, mithilfe eines verfügbaren Lehrsatzes ein weiteres Zwischenergebnis gefolgert werden kann ([73]). In dem vom System gesetzten Rahmen kann dieser Graph tatsächlich vollständig generiert (wenn auch mitunter sehr groß) werden, wodurch ein Überblick über alle in diesem Rahmen möglichen Beweise einer Aussage möglich ist ([69], S. 691–692). Dadurch kann jeder von dem/der BenutzerIn vorgeschlagene Beweisschritt als Bewegung im zugehörigen Graphen aufgefasst und hinsichtlich seiner Nützlichkeit evaluiert und kommentiert werden: Das System findet die jeweiligen Zwischenschritte im Graphen wieder, sucht nach Beweisen – also Wegen durch den gerichteten Graphen, die zum Knoten mit dem Beweisziel führen – die möglichst viele der bisher verwendeten Elemente enthalten (erzeugt der/die BenutzerIn eine Aussage, die im Graphen nicht vorkommt, wird zurückgemeldet, dass der Schritt nicht zielführend war), "rät" daraus den von dem/der BenutzerIn verfolgten Beweisplan und richtet die Rückmeldungen danach, wie gut sie sich in diesen einfügen lassen ([73], S. 47-48).

Besonders durch die Einführung des HDPIC-Graphen und die dadurch ermöglichte interaktive "Betreuung" durch einen digitalen Tutor stellt QED-Tutrix einen didaktisch interessanten Fortschritt gegenüber GEOBEWEIS dar. Die Beschränkung auf einen begrenzten Vorrat an Annahmen und Voraussetzungen und die Auffassung eines Beweises als Weg durch einen Graphen ähnelt der Herangehensweise bei den EProofs; zugleich ist es durch die behutsame Verwendung eines automatischen Theorembeweisers im Hintergrund möglich, einige der Schwierigkeiten der EProofs – z.B. die Abhängigkeit der Lösungskontrolle von der Reihenfolge der Schritte – zu lösen.

Das System QED-Tutrix ermöglicht kein eigenständiges Formulieren, und insbesondere keine Freitexteingabe; die Beweisdarstellung kann hier lediglich durch die Vorbildwirkung des durch das System automatisch aus den Eingaben der/des BenutzerIn erzeugten Beweistextes erfolgen. Ferner sind die in QED-Tutrix verwendeten Techniken an starke Voraussetzungen bezüglich des thematischen und methodischen Rahmens gebunden, die in anderen Kontexten (z.B. in erststufiger Logik) und bei fortgeschritteneren Aufgaben nicht mehr gegeben sind; so dürfte es kaum gelingen, einen vollständigen Überblick über alle Beweise eines Satzes der elementaren Zahlentheorie zu gewinnen, sobald dieser über ein sehr elementares

Niveau hinausgeht. Ob der Aufbau eines HDPIC-Graphen und die damit verbundenen Möglichkeiten sich etwa auch in begrenztem Umfang in den Anfängen der Aussagenlogik (solange nur wenige Sätze und Beweisstrategien bekannt sind), der elementaren Zahlentheorie (etwa der einfachen Teilbarkeitstheorie mit Aussagen wie der Transitivität der Teilbarkeitsrelation) o.ä. realisieren lassen, ist dennoch eine wichtige Frage, muss aber an dieser Stelle offen bleiben. Vielversprechend scheinen Anwendungen etwa auf die axiomatische Geometrie oder auf Beweisaufgaben wie "Linksinverse sind Rechtsinverse", die typischerweise am Anfang der Auseinandersetzung mit einer axiomatisch eingeführten Domäne wie Gruppen, Ringen, Körpern oder Vektorräumen auftreten.

3.1.10 Advanced Geometry Tutor

Der "Advanced Geometry Tutor" (siehe etwa Matsuda und VanLehn [147]) ist ein weiteres System, das auf strikt formale Ableitungen im Bereich der Elementargeometrie spezialisiert ist. Er unterstützt die Strategien des Vorwärts- und Rückwärtsarbeitens und wurde in [147] dazu eingesetzt, die didaktischen Vor- und Nachteile der Vermittlung beider Strategien gegeneinander abzuwägen. Das System unterliegt denselben Beschränkungen wie das zuvor besprochene QED-Tutrix, insbesondere erfolgt die Darstellung von Beweisen im Rahmen des Systems frei von natürlicher Sprache und Herleitungsschritte erfordern die explizite Angabe von Regeln.

3.1.11 Terence Taos QED

Ein weiteres Programm zum interaktiven Beweisenlernen stammt von Terence Tao, einem der größten lebenden Mathematiker, und trägt den Titel "QED – an interactive textbook", siehe [170].

Die Absicht hinter QED ist es, eine spielerische Annäherung an das mathematische Beweisen zu ermöglichen, in Taos Worten "to gamify the rules of inference of propositional logic" [Tao, [170]]. Der Ablauf von QED besteht aus einer Sequenz von Aufgaben, die in der vorgegebenen Reihenfolge bearbeitet werden müssen. Regelmäßig werden dabei neue Schlussregeln eingeführt, die auf die vorliegende Aufgabe angewendet werden können bzw. zu ihrer Lösung erforderlich sind.

Die Beweise werden dabei strikt formal geführt, in einem Format, das Gentzens "natürlichem Schließen" [86], [87] ähnelt. Der/die BenutzerIn erzeugt nacheinander Beweiszeilen, indem er/sie entweder auf eine zur Verfügung stehende Voraussetzung oder auf zwei bereits erzeugte Beweiszeilen (die auch identisch sein dürfen) klickt; dadurch öffnet sich ein Menü, das die auf die beiden gewählten Zeilen anwendbaren Schlussregeln anzeigt. Aus diesen kann der/die BenutzerIn

nun wiederum eine wählen, was dann dazu führt, dass die durch die gewählte Regel aus den gewählten Zeilen ableitbare Aussage als nächste Beweiszeile hinzugefügt wird. Die verwendeten Beweisregeln müssen also explizit angegeben werden; durch die Auswahlmöglichkeit ist es aber nicht erforderlich, sie sich alle namentlich zu merken. Die Eingabe erfolgt bei QED also ausschließlich über Mausbewegungen und -klicks, ohne dass Terme oder Ausdrücke natürlicher Sprache eingegeben werden könnten oder müssten. Die Anzeige des so erzeugten Beweises erfolgt dann in einer leicht mit natürlicher Sprache angereicherten Form. Von der Funktionsweise her ähnelt das System QED also den oben besprochenen auf das geometrische Beweisen abzielenden Systemen wie QED-Tutrix und unterscheidet sich von diesen hauptächlich in der Wahl des behandelten Gebietes, nämlich der Aussagenlogik.

Konzeptionell interessant an QED ist sicherlich die Verbindung eines Beweisprüfers mit einem systematisch aufgebauten Übungsmaterial, mit dem Beweisregeln und -techniken sukzessive erlernt, erprobt und eingesetzt werden können. Ein beachtenswerter Aspekt ist ferner die Hilfestellung für den/die BenutzerIn durch die Anzeige verfügbarer Schlussweisen bei ausgewählten Voraussetzungen; auf diese Weise kann in Situationen, in denen man "nicht mehr weiter weiß", womöglich ein entscheidender Hinweis gegeben werden ("man könnte z.B."), wodurch das frustrierte Abbrechen des Lösungsversuches aufgrund von Rat- oder Ideenlosigkeit vermindert werden dürfte.

Für den Einsatz in der Didaktik des Beweisens ist QED dennoch nur von begrenztem Nutzen. Einerseits beschränkt sich QED in seiner derzeitigen Form auf Aussagenlogik, womit ein großer Teil mathematisch wesentlicher Argumentationsfiguren und -strategien entfällt. Andererseits ist der streng formale, regelorientierte Aufbau weit von der Art entfernt, in der Mathematik üblicherweise dargestellt wird. Schließlich besteht die Gefahr, dass die auf der einen Seite hilfreiche Lösbarkeit durch Auswahl aus einer Reihe vorgeschlagener Optionen zu einer Auffassung von Beweisen als reiner Formelspielerei verführt, von "höherstufigen" Beweisplänen gerade ablenkt und dadurch die Entwicklung eines verständnisbasiertes Vorgehens sogar behindert. Als Teil einer umfangreicheren Lernumgebung könnte ein QED-ähnliches System aber fraglos einen echten Mehrwert darstellen.

3.1.12 Lehrprojekte unter direkter Verwendung professioneller Beweisprüfer bzw. Beweisassistenten

Beweisenlernen mit LEAN

Ein konkretes universitätes Lehrprojekt, bei dem automatische Beweiser zum Einsatz kommen, beschreibt Jeremy Avigad in [11]. Nach [11], p. 1 wurde der

Kurs von ihm selbst, Robert Lewis und Floris van Doorn entwickelt und versteht sich als:

> "'(...) an undergraduate introduction to mathematical proof, symbolic logic, and interactive theorem proving."

Ausdrücklich ist dabei auch die sprachliche Sensibilisierung der TeilnehmerInnen Ziel des Kurses ([11], p.1):

> "Students are required to master three different languages: informal mathematical language, formal symbolic logic, and a computational proof language that lies somewhere in between."

Der wesentlich originelle Ansatz des Kurses besteht nun darin, zugleich mit dem logischen Formalismus die Eingabesprache des automatischen Theorembeweisers "Lean" zu vermitteln ([11], p. 3). Für Überblicksdarstellung zu Lean siehe z.B. [132], [133].

Die Eingabesprache von Lean erfordert z.T. von dem/der BenutzerIn bisweilen eine erhebliche Abweichung von der üblichen mathematischen Alltagssprache. Dies wird in [11] auf S. 8 am Beispiel einer einfachen mengentheoretischen Identität demonstriert, für die ein Teil eines Beweises in natürlichem Englisch und ein weiterer in der Eingabesprache von Lean gegeben werden. Der Beispieltext lautet wie folgt:

> *"Theorem 4.1. Let A, B, and C be any three sets. Then $(A\backslash B)\backslash C = A\backslash(B\backslash\cup C)$.*
>
> *Proof: Suppose x is any element of $(A\backslash B)\backslash C)$. Then x is in $A\backslash B$ but not C, and hence x is in A but not B. But this means that x is in A but not B or C, and hence not in $B \cup C$. So x is in $A\backslash(B \cup C)$, as required. (...)"* [[11], S. 8]

Wie Avigad im Anschluss an dieses Beispiel bemerkt: "a corresponding proof in Lean is not as concise" ([11], S.8). Die in [11] gegebene Fassung erfordert es, zunächst die Folgerung von $x \notin B$ und $x \notin C$ auf $x \notin (B \cup C)$ als eigenes Lemma zu beweisen; zudem hat die Darstellung wenig Ähnlichkeit mit dem obigen mathematischen Text. Wir geben hier als Beispiel nur die Formulierung des gerade erwähnten Hilfslemmas und verweisen Interessenten für die vollständige Formalisierung auf [11]. Die Aussage des fraglichen Lemmas hat in Lean folgende Gestalt:

"lemma ex41a $\{AB : setU\}\{x : U\}(h1 : x \notin A)(h2 : x \notin B) : x \notin A \cup B :=$"

[[11], S. 8]

Wir stellen dem an dieser Stelle eine Wiedergabe des obigen Argumentes in Diproche gegenüber, die von der zum Zeitpunkt der Niederschrift aktuellen Version von Diproche als logisch korrekt verarbeitet wurde:

> Es seien A, B und C Mengen. Wir zeigen: Dann gilt $((A \setminus B) \setminus C) = (A \setminus (B \cup C))$.
> Beweis:
> $\subseteq$ Es sei $x \in ((A \setminus B) \setminus C)$. Dann folgt $x \in (A \setminus B)$. Ferner folgt $\neg x \in C$. Also gilt $x \in A$. Weiter folgt $\neg x \in B$. Damit haben wir $\neg x \in (B \cup C)$. Damit folgt $x \in (A \setminus (B \cup C))$. qed.
> $\subseteq$ Nun sei $x \in (A \setminus (B \cup C))$. Dann ist $x \in A$. Ferner ist $\neg x \in (B \cup C)$. also ist $\neg x \in B$ und $\neg x \in C$. Damit ist $x \in (A \setminus B)$. Also gilt $x \in ((A \setminus B) \setminus C)$. qed.
> Damit folgt nun $((A \setminus B) \setminus C) = (A \setminus (B \cup C))$.
> qed.

Der Ansatz, die natürliche mathematische Sprache und ihr Verhältnis zum Formalismus zum expliziten Gegenstand eines Kurses zu machen, in dem voll formale Beweise durch einen automatischen Beweisprüfer verifiziert werden können, ist als Weg zur Vermittlung von Beweiskalkülen originell und didaktisch sicherlich sinnvoll. Als erste Beweiseinführung dürfte dieser Zugang jedoch ungeeignet sein und auch ein falsches Bild vom mathematischen Beweisen als einem formalen Herleiten statt als einem vernünftigen Begründen vermitteln. Insbesondere für StudienanfängerInnen, die große Schwierigkeiten bereits mit elementaren Eigenschaften des Formalismus haben wie etwa der Unterscheidung zwischen Objekten, Mengen von Objekten und Aussagen über Objekte, ist von diesem Zugang wenig Hilfe zu erwarten. Der Lean-basierte Kurs von Avigad et al. ist also ein didaktisch interessantes Projekt, das auch ein im wesentlichen ähnliches Ziel verfolgt wie Diproche, aber mit wesentlich anderen Mitteln. Welche sich in welchen speziellen Lehrkontexten als die wirksameren erweisen, bleibt abzuwarten. Indes wird auch von Avigad et al. die Möglichkeit einer natürlicheren Eingabespräche ausdrücklich als Desiderat erwähnt:

> "Over time, we expect Lean to evolve to the point where we will be able to use automation to support writing more complicated mathematical proofs in a style that is close to the informal ones we expect students to write." [[11], p. 4]

Es ist ein Ziel von Diproche, sich dieser Herausforderung zu stellen und im Rahmen eines digitalen Systems das korrekte Darstellen in kontrollierter natürlicher Sprache zu vermitteln. Dabei ist es gerade nicht das Ziel, StudienanfängerInnen an einen strikten logischen Formalismus heranzuführen, sondern diesen "strengen" Stil letztlich auf dem Weg zum kreativen Beweisen zu überschreiten. Formale Logik ist ein im Hintergrund operierendes Werkzeug von Diproche, nicht das (wesentliche) Vermittlungsziel.

Beweisenlernen mit Coq

Ein weiteres Lehrprojekt mit dem Ziel, das mathematische Beweisen mithilfe eines automatischen Beweisassistenten zu vermitteln, wurde von S. Böhne und C. Kreitz an der Universität Hamburg durchgeführt, siehe [24]. Die Veranstaltung wandte sich an Studierende der Informatik. Der Ansatz bestand hier darin, zunächst das Beweisen im Rahmen des automatischen Beweisassistenten Coq, einem System zur automatischen Verarbeitung formaler Mathematik, zu lehren und dann das allmähliche "Deformalisieren": Dazu wurde eine Reihe von Zwischenformaten zwischen Coq-Beweisen und "Lehrbuch-Beweisen" definiert und die Übergänge allmählich vermittelt. Das Projekt führte, gemessen an den abschließenden Klausurergebnissen, zu einem guten Lehrerfolg, siehe [24], Abschnitt 8.

Der Ansatz, mit formalen Beweisen im Rahmen eines automatischen Beweisassistenten zu beginnen, hat sicherlich gerade für Studierende der Informatik seine Vorteile; Beweise in Coq und ähnlichen Systemen haben starke Ähnlichkeit mit Computerprogrammen, so dass Studierende mit Programmiererfahrung hier sicher bekannte Strukturen wiederentdecken und in dieser Form auch in höherem Maß motiviert sein mögen, sich mit Beweisen zu befassen. Für Studierende anderer Fachrichtungen – etwa im Lehramt Mathematik – mit wenig inhärentem Bezug zu formalen Methoden hingegen erscheint dieser Zugang zum Beweisen künstlich zu sein. Ferner ist auch im Rahmen dieses Ansatzes der Einsatz natürlicher Sprache in der Mathematik gerade kein Gegenstand der automatischen Prüfung.[13]

3.1.13 Vellemans Proof Designer

Als Begleitsoftware zu seinem einführenden Lehrbuch "How to Prove it: A Structured Approach" ([209]) hat Daniel Velleman das Programm "Proof Designer" implementiert, in dem sich einfache mengentheoretische Identitäten beweisen lassen. Beweise erfolgen durch eine sukzessive Auswahl per Mausklick

[13] Tatsächlich bemerken die Autoren von [24] auf S. 14 sogar "some unwillingness to switch back to natural language proofs" auf Seiten der Studierenden im Anschluss an die Einübung im Beweisen mit Coq.

von Schritten und Zwischenzielen aus Menüs; es gibt insbesondere keine Freitexteingabe in natürlicher Sprache. Insofern ähnelt das System dem oben besprochenen QED von T. Tao. Ein interessanter Ansatz ist die Verwendung von Beweis'templates', in denen diverse einfache Argumentationsfiguren bereits vorformuliert sind. Eine kurze Einführung findet sich in Anhang 2 von [209].

3.1.14 Lurch

Das System "Lurch", welches von Nathan Carter (Bentley University) und Ken Monks (University of Scranton) entwickelt wird, verfolgt das Ziel, einen "word processor that can check your math"[142] zu entwickeln, der speziell für hochschuldidaktische Anwendungen ausgelegt ist. Das System ist online frei verfügbar und wird von seinen Urhebern in Lehrveranstaltungen regelmäßig eingesetzt. Ferner gibt es ein einführendes Lehrbuch zur Logik mit Übungsaufgaben, die mit Lurch bearbeitet werden können (siehe [144].

Wie Diproche stellt Lurch einen Texteditor zur Verfügung, der eine Freitexteingabe erlaubt. Allerdings erlaubt Lurch durch eine LaTeX-Einbindung einen deutlich angenehmeren und breiteren Umgang mit Sonderzeichen. Die Verwendung eines what-you-see-is-what-you-get-Editors, der LaTeX-Befehle umgehend in die entsprechenden Zeichen umsetzt und anzeigt, macht die Eingabe sehr angenehm und erlaubt es, einen Überblick über den bisher geführten Beweis zu behalten. Die im intendierten Einsatzgebiet des Systems vorkommenden Sonderzeichen sind überdies über eine Reihe von Submenüs mit der Maus auswählbar, so dass der Editor auch ohne LaTeX-Kenntnisse gut verwendet werden kann.

Im Unterschied zu Diproche nimmt Lurch keinerlei Verarbeitung natürlicher Sprache vor. Die Aufgabe einer solchen Verarbeitung liegt vor allem darin, jedem Satz eine Funktion (Annahme, Behauptung, Annotation, Deklaration,...) zuzuordnen und den jeweils behaupteten/angenommenen/... Inhalt zu ermitteln. Diese Aufgabe wird in Lurch dem/der AnwenderIn übertragen: 'Inhaltlich relevante' Teile des Textes, bei denen es sich um Formeln handeln muss, welche der Formelparser von Lurch verarbeiten kann, werden von dem/der BenutzerIn markiert und annotiert (etwa als Aussage, Variable, deklarierte Konstante etc.), wodurch intern die Struktur eines formalen Beweises in einem Kalkül wie dem natürlichen Schließen entsteht. Dieser in den Gesamttext gewissermaßen "eingebettete" formale Beweis wird dann vom System geprüft.[14]

Ansonsten steht es dem/der BenutzerIn frei, Text nach Belieben einzugeben. So würde z.B., wenn eine Transitivitätsregel für die Gleichheit zur Verfügung steht

[14] Allerdings kennt die Lurch-Annotation keinen Unterschied zwischen "Annahme" und "Behauptung"; worum es sich jeweils handelt, wird aus der weiteren Verwendung erschlossen, die, wenn sie inkonsistent (etwa zirkulär) ist, vom System moniert wird.

und mit “Transitivitätsregel” benannt wurde, folgender Satz akzeptiert, sofern das erste $a = b$ und das erste $b = c$ sowie $a = c$ als “meaningful expressions” markiert werden, weiter “Transitivitätsregel” als “reason” mit einem Pfeil nach links und schließlich die zweiten Vorkommen von $a = b$ und $b = c$ als “premise” mit einem Pfeil nach links:

Es gelte $a = b$. Außerdem gelte $b = c$. Dann folgt $a = c$ wegen Transitivitätsregel, $a = b$ und $b = c$.

Bei gleicher Markierung der erwähnten Bestandteile könnte man aber ebenso gut schreiben:

Folglich gilt $a = b$. Folgendes ist falsch: $b = c$. Es gelte $a = c$, Schuh Transitivitätsregel, $a = b$ Blume $b = c$ Dackel.

Oder auch:

$a = b$ $b = c$ $a = c$ Transitivitätsregel $a = b$ $b = c$

Beweisschritte müssen stets begründet werden, auch Begründungen werden als solche markiert und mit einem Pfeil wird angegeben, ob sie sich auf die letzte oder die folgende Aussage beziehen. Diese Begründungen erfolgen stets durch die explizite Angabe einer Schlussregel und einer Auswahl von früheren Aussagen, auf die sie angewendet wird. Das begrenzt die Anwendbarkeit aus den schon oben genannten Gründen. Allerdings ist der Regelsatz bei Lurch nicht fest, sondern kann – auch aufgabenspezifisch – von der jeweiligen Lehrperson festgelegt werden. Ebenso ist es möglich, Regeln in den Kanon der bei einer Aufgabe verfügbaren Regeln aufzunehmen bzw. davon auszuschließen, sie [49]. Insofern arbeitet Lurch mit einem aufgabenspezifisch kontrollierten ATP, wie auch Diproche.

Als erheblicher Vorteil von Lurch ist sicherlich die Verbindung aus einer Freitexteingabe in natürlicher Sprache mit einer letztlich völlig sprachunabhängigen Verarbeitung anzusehen, wodurch das System jederzeit in jedem Sprachkreis eingesetzt werden kann, für den der Zeichenvorrat des Texteditors geeignet ist; auch entsprechende Erweiterungen etwa in das chinesische Alphabet wären keine ernsthafte Schwierigkeit. Die Verwendung eines gewöhnlichen Texteditors macht die Eingabe sehr angenehm.[15] Wie die Urheber des Systems ausdrücklich hervorheben (siehe [49], S. 7) dürfte die Notwendigkeit,

[15]Einige wünschenswerte Möglichkeiten, die das Lurch-Interface derzeit nicht bietet, sind die Verwendung von Indizes und Exponenten sowie die dynamische Erweiterung der Notation durch Einführung neuer Symbole (letztere wird auch von den Entwicklern selbst erwähnt). Die letzere Funktion sind in Systemen wie Naproche bereits realisiert; hier läge womöglich ein Ansatz für eine fruchtbare Kooperation.

einen Beweistext zu annotieren, nicht unerheblich zu einer vertieften Beschäftigung mit Beweistexten und den einzelnen Funktionen ihrer Teile beitragen und so das Beweisverstehen fördern. Neben reinen Beweisaufgaben sind auch andere Arten von Aufgabe möglich, z.B. die, einen fertigen Beweistext so zu annotieren, dass Lurch ihn verifiziert, was es insbesondere erfordert, die logische Struktur herauszuarbeiten. Ferner ist die Eingabe neuer Regeln durch die Lehrperson ohne Programmierkenntnisse problemlos möglich.

Im Gegensatz zu vielen der hier besprochenen Systeme – und auch im Gegensatz zu Diproche – verfügt Lurch nicht über eine Komponente, die Hinweise zum Vorgehen bei Beweisaufgaben generiert.[16] Die Beweiskonstruktion ganz dem/der BenutzerIn zu überlassen ist hierbei eine didaktisch motivierte Designentscheidung, siehe z.B. [42], S. 2. Ebenso erfordert Lurch bei Schlüssen stets die Angabe einer Schlussregel; das für natürliche Beweise typische – und ab einer gewissen Komplexität wohl unentbehrliche – implizite Folgern wird also von Lurch nicht unterstützt. Beides sind freilich (didaktisch begründete und gewollte) Eigenschaften der aktuellen Implementierung von Lurch, nicht des Ansatzes; man könnte diese Funktionen ohne weitere ergänzen.

Ein prinzipieller Unterschied zu Diproche besteht darin, dass Lurch bei der Prüfung die natürlichsprachlichen Textanteile ignoriert und folglich diesbezüglich keine Rückmeldungen liefert.[17] Die Verarbeitung natürlicher Sprache, die für Diproche wesentlich ist – einschließlich der damit einhergenden Standardisierung der Formulierungen und des Vokabulars – gibt es hier also nicht. Eine Freitexteingabe ist zwar möglich, hat aber gerade nicht die didaktischen Effekte, die dadurch angestrebt waren; lediglich die Lesbarkeit des Dokumentes für menschliche Leser wird gegenüber den rein formalen Systemen deutlich verbessert (inklusive jenen, die, wie QED-Tutrix, den formalen Beweis automatisch in natürliche Sprache konvertieren können). Somit ist Lurch letztlich ein System zum formalen Beweisen mit einer geschickt integrierten und extensiven Kommentarfunktion. Insbesondere kann eine Fehlverwendung der Fachssprache weiterhin nur durch eine/n menschlichen KorrektorIn festgestellt werden. Auch die formale Korrektheit der Darstellung wird durch eine erfolgreiche Lurch-Prüfung nicht gewährleistet.

Gerade für Bereiche, in denen das Arbeiten in einer ontologisch wie inferentiell streng begrenzten und stark formalisierungsnahen Domäne im Vordergrund steht, stellt Lurch einen sehr kreativen Ansatz dar, von dem zu erwarten ist, dass er die Arbeit mit Diproche in Lehrveranstaltungen produktiv ergänzen kann.

[16] Siehe z.B. Carter und Monks [42], S. 2: “No help with proofs”.

[17] Siehe z.B. das Lehrvideo “Student’s View” `http://lurchmath.org/videos/` (zugegriffen: 07.10.2020), wo es heißt: ‘I can’t expect Lurch to understand English”).

3.1.15 Elfe

Ein System, das sich explizit die Verifikation natürlichsprachlicher mathematischer Beweise mit didaktischer Absicht zum Ziel gesetzt hat, ist "Elfe", das von Maximilian Doré und Krysia Broda entwickelt wurde, siehe [62] und [17]; das System ist online verfügbar unter [19].

Die Motivation für Elfe ist mit der hinter Diproche weitgehend identisch, nämlich der Einsatz eines automatischen Beweisprüfers für natürlichsprachliche Texte zur Verbesserung der Rückmeldungen im Anfängerbereich des Mathematikstudiums (siehe [17]). Das System ermöglicht eine freie Texteingabe; die Rückmeldung erfolgt durch die unterschiedliche Einfärbung verifizierbarer (grün) und nicht verifizierbarer (rot) Beweiszeilen. Zusätzlich ist Elfe in der Lage, in gewissen Fällen automatisch Gegenbeispiele zu fehlerhaften Beweisschritten zu generieren, also eine Situation auszugeben, in der die Voraussetzungen erfüllt sind, die vermeintliche Folgerung aber nicht. Das System wurde in einem Versuch mit 12 Studierenden eingesetzt und von diesen insgesamt positiv evaluiert, siehe [17].

Das System unterstützt linguistische Konstrukte für diverse Beweisstrategien, so etwa die Aufgliederung in Teilbeweise oder Fallunterscheidungen. Wie in diversen Programmiersprachen wird die Beweisstruktur u.a. durch Einrückungen gekennzeichnet. Ähnlich wie Naproche und SAD erlaubt das System die Verifikation von Texten, die über einzelne Beweise hinausgehen und etwa Definitionen und notationelle Konventionen sowie mehrere Lemmata mit Beweisen erlauben.

Die von Elfe akzeptierte Eingabesprache ist ein stark kontrolliertes Fragment des Englischen; entsprechend haben typische Beweise in Elfe einen stark formalen Charakter.[18] Folgerungen werden stets durch "Then" oder "Hence" eingeführt, wobei zwischen beiden ein semantischer Unterschied besteht: Während 'then' eine einfache Folgerung anzeigt, wird durch 'hence' die letzte offene Voraussetzung zurückgezogen (siehe [17]; 'hence' hat somit eine ähnliche Funktion wie 'thus' in Diproche). Damit verwendet Elfe semantische Konventionen, die in der mathematischen Fachsprache nicht üblich sind. Eine in diesem Fragment verfasste Lösung dürfte für einen nicht mit dem System vertrauten Mathematiker zwar lesbar sein, ist von der tatsächlich verwendeten Fachsprache – auch im Bereich von Anfängerübungen – recht weit entfernt.[19]

[18] Siehe z.B. die Beispieltexte unter `https://elfe-prover.org/examples` (zugegriffen. 07.10.2020).

[19] Für einfache mengentheoretische oder aussagenlogische Beweise reduziert sich der Unterschied allerdings de facto häufig darauf, dass Diproche ein breiteres Spektrum an Formulierungen für Annahmen und Folgerungen (also zusätzlichen "syntaktischen Zucker") erlaubt und andererseits unter diesen keine semantischen Unterschiede einführt, da die

Ein weiterer Unterschied zu Diproche besteht in der Verwendung "professioneller" ATPs für die Verifikation; das System versucht nicht, die zulässigen Inferenzregeln kontextabhängig auf Basis didaktischer Überlegungen zu erfassen, sondern läßt – wie etwa auch Naproche – zur Verifikation von Beweisschritten mehrere etablierte ATPs parallel mit einer Zeitschranke laufen und wertet einen Schritt als verifzierbar, wenn einer der ATPs einen Erfolg meldet (siehe [17]). Die damit einhergehenden didaktischen Vor- und Nachteile wurden bereits oben in den Abschnitten zu Naproche und SAD diskutiert.

Die Unterschiede zu Diproche liegen also zum einen im Grad der Annäherung an die natürliche Sprache (im Fall von Diproche: des Deutschen) und in der Verwendung kontrollierter Inferenz in Diproche, zum anderen aber auch in der Art der Rückmeldung: So verfügt Elfe im Gegensatz zu Diproche über einen "Gegenbeispielgenerator", dessen Ausgabe bei entsprechend nutzerfreundlicher Anzeige einen erheblichen didaktischen Zugewinn darstellen düfte. Gegenbeispiele gibt es bei Diproche aktuell nur in den Bereichen "Aussagenlogik" sowie mit Einschränkungen im Bereich "Boolesche Mengenlehre"; andererseits bietet Diproche über den "Anti-ATP" (s.u.) eine Diagnose der möglichen Fehlvorstellungen hinter einem nicht verifizierbaren Inferenzschritt an.

3.2 Der angezielte didaktische Mehrwert von Diproche

Wir wollen nun die oben erfolgte Auseinandersetzung mit einer Reihe verschiedener Systeme dafür nutzen, die didaktische Zielsetzung für das zu konstruierende System präziser herauszuarbeiten.

3.2.1 Erwartete didaktische Vorteile des automatischen Beweisprüfens im Allgemeinen

Wir beginnen damit, die Vorteile in der Vermittlung von Beweiskompetenzen, die vom Einsatz automatischer Beweisprüfer zu erhoffen sind, noch einmal zusammenzufassen.

(1) Der offensichtlichste Vorteil gegenüber dem klassischen Übungsbetrieb besteht in der Etablierung einer Feedback-Schleife im Gegensatz zur üblichen "Feedback-Sackgasse": Rückmeldungen bezüglich eines Beweisversuches kommen im unmittelbaren zeitlichen Zusammenhang zu diesem Versuch und können so einerseits dabei helfen, Fehlvorstellungen also solche zu erkennen und zu korrigieren

Beweisstruktur durch die Absatzstruktur repräsentiert wird (s.u.). Aber auch hier stehen dem/der BenutzerIn bereits einige Ausdrucksmöglichkeiten – etwa Folgerungs- und Umformungsketten, begründete Behauptungen etc. – zur Verfügung, die eine stärkere Annäherung an die üblichen Darstellungsweise zumindest ermöglichen.

und außerdem in den Bearbeitungsvorgang einfließen; die so erhaltene verbesserte Lösung kann wiederum korrigiert werden etc.

(2) Insbesondere durch den Einsatz von Aufgabengeneratoren, aber auch bereits durch die Bereitstellung von Korrekturen, verbessern automatische Systeme die Möglichkeiten zum Selbststudium und ermöglichen es, individuell intensiver auf Punkte einzugehen, an denen es noch Schwierigkeiten gibt.

(3) Durch die unmittelbare Meldung von Fehlern, aber auch durch das Interesse an der Interaktion mit dem System ist zu erwarten, dass insgesamt mehr Zeit auf die Beschäftigung mit mathematischen Beweisaufgaben verwendet wird, was sich in einem verbesserten Lernerfolg niederschlagen sollte.

(4) Auf der anderen Seite bietet das System im Erfolgsfall eine "instant gratification" in Form einer sofortigen positiven Rückmeldung; es ist zu erwarten, dass solche häufigeren und unmittelbareren positiven Bestätigungen dazu geeignet sind, die Motivation zur Beschäftigung mit mathematischen Beweisen zu erhöhen.

(5) Ein weiterer positiver Effekt, der von dem Einsatz solcher Systeme zu erhoffen ist, ist eine Verschiebung des Verhältnisses zwischen Studierenden und Lehrenden: Durch die automatische Korrektur, die ein hohes Maß an Kohärenz aufweist[20] sollte sich der Eindruck verstärken, dass man es bei der Korrektheit von Beweisen mit einer objektiven Qualität zu tun hat statt mit einer "verordneten Norm". Lehrende können sich also darauf konzentrieren, bei der Produktion korrekter Beweistexte zu helfen und sind nicht zugleich "Richter", die vorgeschlagene Texte kritisieren und bewerten. Dies gilt umso mehr, als Studierende gerade im Anfängerbereich häufig selbst noch nicht erkennen können, ob ein Beweis richtig ist (siehe etwa [179]). Somit erscheint die Rückmeldung der Lehrenden für Studierende potenziell nicht als Hinweis auf eine objektiven Eigenschaft ihrer Beweistexte – wie etwa Korrektheit oder Lückenhaftigkeit –, die sie, einmal darauf hingewiesen, auch selbst erkennen können; vielmehr legt sich der Eindruck nahe, die Lehrenden setzten diese Kriterien autoritativ fest. Es ist zu hoffen, dass Studierende sich durch den Einsatz eines Systems, dass einen externen Begriff von Beweiskorrektheit anbietet, stärker als mit den Lehrenden "auf derselben Seite stehend" erleben.[21]

Ein Vergleich mag helfen, diesen Punkt deutlicher zu sehen: Ob ein Salto gelungen ist, kann jede/r unmittelbar sehen. In der Leichtathletik kann sich ein/e TrainerIn also darauf konzentrieren, SchülerInnen bei der Verbesserungen ihrer Technik zu unterstützen. Beim Beweisen dagegen ist – wie die Studie von Selden und Selden [179] unterstreicht – oft schon der Standard nicht entwickelt, an dem

[20] Im Gegensatz zur menschlichen Korrektur – siehe die Diskussion zur Kohärenz von Bewertungen von Beweisen durch verschiedene Lehrende weiter unten.

[21] Diese Erfahrung mit dem Einsatz automatischer Beweisprüfer in der Lehre ist dem Autor von verschiedenen Seiten zugetragen wurde, u.a. durch P. Koepke, den Leiter des Naproche-Projektes (s.o.).

der Beweis zu messen wäre – die Entwicklung dieses Standards ist selbst Teil des Lernprozesses. Damit besteht die Gefahr, dass die Korrektheit von Beweisen als rein formale oder soziale Norm missverstanden wird: Man lernt Umgangsformen, die festgelegt werden von dem, der sie einem beibringt, um von diesem zum Schluss etwas (etwa eine Note, einen Schein oder einen Abschluss) zu erhalten. In dieser fundamental unsachlichen Haltung fehlt der für eine wissenschaftliche Ausbildung zentrale Bezug zum Gegenstand. Dass TutorInnen und Lernende gemeinsam dem System als einem Kriterium gegenüber stehen, mag helfen, mathematische Beweisaufgaben stärker als gemeinsam zu meisternde sachliche Herausforderungen sichtbar zu machen.

3.2.2 Erwartete didaktische Vorteile durch den Einsatz von Diproche gegenüber anderen Systemen

Wie wir gesehen haben, existieren zahlreiche automatische Systeme zum Beweisprüfen und Beweisenlernen.

Ein gemeinsames Merkmal aller oben besprochenen Systeme besteht darin, dass sie eine automatische Beweisprüfung durchführen, dabei aber die natürlichsprachliche Darstellung auf verschiedene Weisen umgehen, oder, wie im Fall von Elfe, so stark normieren, dass sie sich deutlich von der auch im Anfängerbereich üblichen Darstellungspraxis entfernt. Diese Systeme können also nur bedingt dazu beitragen, den korrekten Gebrauch der Fachsprache, der ein wichtiger Bestandteil des Beweisenlernens ist, durch Übung zu fördern. Weiter haben BenutzerInnen sich entweder an einem strikten Regelsatz zu orientieren, anhand dessen Beweisschritte explizit zu rechtfertigen sind, oder die Verifikation erfolgt mithilfe eines professionellen ATP, wodurch Schritte zulässig werden, die im Kontext der Lehre im Anfängerbereich als rechtfertigungsbedürftig anzusehen sind.

Aus diesen Betrachtungen entstehen für ein System zur Unterstützung des Beweisenlernens folgende Desiderata, um die bereits bestehenden Systeme produktiv ergänzen zu können:

- Das System sollte eine Freitexteingabe in natürlicher Sprache ermöglichen; diese akzeptierte Sprache sollte möglichst umfassend die Wendungen, Formulierungen und Ausdrücke abbilden, die zur korrekten Bearbeitung von Beweisaufgaben im Anfängerbereich erforderlich sind.

- Das System sollte sich in der Evaluation eines Beweistextes möglichst eng an diesen Text halten. Das heißt, es sollte natürlichsprachliche Formulierungen nach Möglichkeit so interpretieren wie ein/e kompetente/r menschliche/r LeserIn und die Auswertung auf dieser Interpretation basieren lassen.

- Die Verifikation von Beweisschritten sollte möglichst genau die Standards abbilden, die im jeweiligen Bereich üblich sind, mit der Möglichkeit, auf Veränderungen und Entwicklungen dieser Standards flexibel zu reagieren. Die explizite Erwähnung von Inferenzregeln sollte dort möglich sein, wo sie es im jeweiligen Kontext auch in der üblichen Darstellungspraxis ist, zwingend erforderlich aber nur dort, wo das auch nach den Maßstäben menschlicher KorrektorInnen nötig ist; auch diesbezüglich sollte das System eine weitreichende Flexibilität in der Festlegung von Standards bieten.

Ziel ist es also, zusammengefasst, das System möglichst weit an die in der Lehre übliche Praxis anzunähern. Dadurch werden andere Systeme, die gerade in der Abweichung von dieser Praxis verschiedene Arten von Hilfestellung leisten oder sonst eher im Hintergrund präsente Aspekte (wie etwa die Formalisierbarkeit oder die Möglichkeit eines strikt regelbasierten Vorgehens) am Beweisen hervorheben, nicht ersetzt, wohl aber wird – so jedenfalls unsere Hoffnung – ihnen ein System hinzugefügt, das einige relevante Fertigkeiten in einem größeren Umfang als diese fördert.

Kapitel 4

Vorüberlegungen zur Machbarkeit: Möglichkeiten und Grenzen[1]

An Sinn und Möglichkeit automatischer Beweisprüfer lassen sich eine Reihe von Zweifeln geltend machen. In diesem Abschnitt wollen wir aus verschiedenen Blickwinkeln der Frage nachgehen, ob und inwieweit eine maschinelle Prüfung natürlichsprachlicher mathematischer Beweise, wie sie in mathematischen Diskurssituationen (etwa einer Vorlesung, der Publikation eines Fachartikels, der Bearbeitung einer Klausur- oder Übungsaufgabe) auftreten, möglich und sinnvoll ist. Daraus wird sich eine gegenüber den oben stehenden Überlegungen zugeschärfte Zielsetzung ergeben, die dann im weiteren Verlauf der Arbeit in Angriff genommen wird. Wir befassen uns daher unten mit einigen Einwänden, die das Konzept eines automatischen Beweisprüfers sehr kritisch beurteilen; wir begrüßen solche Einwände, wo sie gut begründet sind, und finden in ihnen einen wertvollen Beitrag zu unserer Arbeit, weil sie zwar nicht das Projekt als unmöglich erweisen, wohl aber helfen, den Fokus in eine Richtung zu lenken, die Erfolg verspricht – und dort zu lassen; im Übrigen helfen sie dabei, Grenzen des Systems klar zu sehen und somit Vereinseitigungen durch den Einsatz des Systems dadurch vorzubeugen, dass entsprechende Gegenakzente gesetzt werden.

[1] Viele der in diesem Abschnitt dargelegten Gedanken wurden angeregt durch Vorträge, Diskussionen und persönliche Gespräche im Rahmen des Workshops "Mathematical Language and Practical Type Theory", der im Frühjahr 2020 am Hausdorff Center for Mathematics in Bonn abgehalten wurde; besonders zu nennen ist an dieser Stelle der Vortrag "Technology adoption in mathematical research: automated proof-checking as a case study" von Benedikt Löwe, sowie persönliche Gespräche mit Peter Koepke, Benedikt Löwe, Marcos Cramer und Deniz Sarikaya.

4.1 Die naive Vorstellung automatischer Beweisverifikation

Beweise zu prüfen gehört zum mathematischen Alltagsgeschäft. In der Lehre werden studentische Lösungen in Form von Abgaben von Übungsaufgaben oder Klausuren sowie in den Übungen mündlich vorgetragene Beweisvorschläge geprüft. Zur Forschung gehört einerseits die Teilnahme am "peer review"-Prozess, die die Begutachtung von Fachartikeln auf Korrektheit umfaßt; andererseits ist die Prüfung eigener Beweise und Beweisansätze ein integraler Bestandteil mathematischer Forschung: Fast jeder erfolgreiche Beweis geht hervor aus einer Abfolge von versuchten, als unzureichend erkannten und verbesserten oder auch als undurchführbar verworfenen Versionen. Schließlich gehört das gegenseitige Prüfen auch wesentlich zum kooperativen mathematischen Forschen, dem sogenannten "Diskutieren".

Nun sind solche Prüfungen erfahrungsgemäß sowohl aufwändig als auch unsicher. Selbst bei einem kurz und präzise dargestellten Beweis kann es Stunden oder Tage dauern, bis "der Groschen fällt", sich also ein Durch- und Überblick in Bezug auf das Argument einstellt.[2] Überdies werden beim gedanklichen Nachvollzug eines Beweises immer wieder Lücken und Mängel übersehen und es gibt zahlreiche Beispiele für Beweise, die viele Jahre nach ihrer Publikation als lücken- oder fehlerhaft erkannt wurden, wie etwa der ursprüngliche Beweis des 4-Farben-Satzes durch Kempe, dessen Fehlerhaftigkeit erst 11 Jahre nach der Publikation bemerkt wurde [52], [80]. In Zeiten großer kooperativer Projekte (wie etwa der Klassifikation der einfachen Gruppen, siehe [93], [65]) entstehen zunehmend Beweise von einer Länge und Komplexität, bei der diese Probleme begründete Zweifel an deren Verläßlichkeit wecken.[3] Entsprechend gibt es auch Beispiele für Beweise, die zumindest für längere Zeit "umstritten" waren. Ein prominentes Beispiel ist Hales Beweis der Keplerschen Vermutung (s.o.), ein weniger bekanntes aus neuerer Zeit die Kontroverse um Mochizukis Arbeiten zur "interuniversellen Teichmüller-Theorie", siehe [125].

Angesichts dieser Schwierigkeiten scheint das automatische Prüfen natürlichsprachlicher Beweise ein faszinierender Ausweg zu sein: Statt

[2]Bei diesem Phänomen scheint mehr im Spiel zu sein als ein bloßer Nachvollzug einer Kette von Deduktionen; eher kommt der Beweis "als ganzes" in den Blick. Man mag hier mit Bezug auf die Gestalttheorie von dem Erkennen der "Gestalt" eines Beweises sprechen. Tatsächlich spricht z.B. in der Studie von Moore [151] zur Bewertung von Beweisen durch akademische Lehrkräfte eine der Probandinnen explizit von einem "gestalt point of view" auf die Bepunktung. Die sicherlich lohnende Verfolgung dieses Gedankens liegt aber außerhalb des Skopus dieser Arbeit.

[3]Siehe z.B. Elwes [65], wo Michael Aschenbacher, einer der Hauptbeteiligten des Projektes der Klassifikation der endlichen einfachen Gruppen, nach dessen Abschluss mit folgendem Satz zitiert wird: "to my knowledge the main theorem [of our paper] closes the last gap in the original proof, so (for the moment) the classification theorem can be regarded as a theorem". (Zitiert nach [65], Originalzitat [9], S. 739.))

menschlichen Mathematikern mit ihrer begrenzten Zeit, Konzentration und Motivation, die dadurch überdies von der (vermeintlich) "eigentlichen" mathematischen Forschung abgehalten werden, übergibt man Beweise zur Prüfung einem Computerprogramm, das den Text entgegennimmt und nach kurzer Zeit ein klares "Ja" oder "Nein" ausgibt, in letzterem Fall vorzugsweise unter Nennung zumindest eines konkreten Fehlers.

Dieses Projekt nun ist in der Tat so faszinierend wie undurchführbar. Die "Zweifelhaftigkeit", die der rechnergestützte Ansatz beseitigen soll, ist, so werden wir weiter unten argumentieren, ein wesentliches Merkmal natürlicher mathematischer Beweise und kann also allenfalls dadurch beseitigt werden, dass wir unser Verständnis von der Korrektheit eines Beweises an das Kriterium maschineller Verifizierbarkeit anpassen. Wir werden weiter begründen, warum eine solche Anpassung nicht wünschenswert wäre.

4.2 Die (vermeintliche) Nutzlosigkeit automatischer Beweisprüfung: Das Argument von Rav

In seiner Arbeit "A Critique of a Formalist-Mechanist Version of the Justification of Arguments in Mathematicians" [173] formuliert der Mathematiker und Philosoph Yehuda Rav einen grundsätzlichen Einwand gegen die Sinnhaftigkeit automatischen Beweisprüfens, den wir hier sinngemäß so wiedergeben wollen: Da einem Computer ein inhaltlicher Bezug zu den in einem Beweis verwendeten Begriffen fehlt, muss ein Beweis, um von einem Computer nachvollzogen werden zu können, vollständig formalisiert sein. Dieser Prozess der Formalisierung ist von dem oder der BenutzerIn des Systems zu leisten; in seinem Verlauf wird das inhaltliche Verständnis vollständig durch formale Inferenzen ersetzt. Um diesen Übergang vollziehen zu können, muss der Mensch den Beweis aber so gut verstanden haben, dass die Korrektheit oder Fehlerhaftigkeit bereits nach Abschluss der Formalisierung offenbar ist – die eigentliche Ausgabe des automatischen Prüfers bestätigt somit nur, was bereits eingesehen wurde, als die Eingabe erzeugt wurde. Folglich kann automatisches Beweisprüfen, in Ravs Worten, keinen "epistemic gain", keinen Erkenntnisgewinn, zur Folge haben.

Dieses durchaus nicht unplausible Argument wurde bereits in Carl [31] mit dem Hinweis angegriffen, dass Formalisierung und Prüfung auf unterschiedlichen Verständnisstufen stattfinden können und dass ein für eine Formalisierung erforderliches Verständnis für eine Verifikation nicht unbedingt ausreicht. Auch zeigt die praktische Erfahrung im Umgang mit Systemen wie Naproche oder SAD ein anderes Bild (siehe auch hierzu etwa [31]).

Speziell für den didaktischen Kontext gibt es indes weitere Gründe, Ravs

Einwand für nicht stichhaltig zu halten[4]. Denn zum einen stellt gerade für AnfängerInnen oft schon der korrekte Gebrauch des mathematischen Formalismus eine Herausforderung dar, so dass eine automatische Rückmeldung schon auf rein syntaktischer Ebene durchaus informativ sein kann. Zum zweiten kann gerade zu Beginn auch das Argumentieren selbst auf einer formalen und kleinschrittigen Ebene sinnvoll geübt werden, wie etwa der erfolgreiche Einsatz einiger der oben besprochenen Systeme zum automatischen Beweisprüfen in der Lehre auch ganz konkret zeigt.[5] Zum dritten gibt es für den Anfängerbereich genug Übungsfelder, auf denen inhaltliches Verständnis zunächst nur in so grundlegender Form verwendet wird, dass dieses sich formal durchaus umfassend repräsentieren läßt (siehe dazu etwa die Ausführen zu den "Spielwiesen" zur elementaren Zahlentheorie und zur axiomatischen Geometrie weiter unten). Eine entwickelte inhaltliche Intuition für einen weitläufigen, abstrakten Gegenstandsbereich ist in dem hier relevanten Bereich im Allgemeinen noch nicht vorhanden (und wo sie vorhanden ist, bleibt zu lernen, diese in Textform schlüssig auszudrücken), so dass die Differenz zwischen formalen Beweistexten und solchen, die auf eine solche Intuition zurückgreifen, keine Schwierigkeiten bereitet.

4.3 Die Fragwürdigkeit eines einheitlichen Verständnisses von "korrekter Beweistext"

Wir kehren nun zurück zur eingangs erwähnten Fiktion eines Programmes P, das das mathematische Beweisprüfen im allgemeinsten Sinn vornehmen können soll – P soll also beliebige argumentative mathematische Texte im Hinblick auf ihre Korrektheit beurteilen können. Formal gesprochen: P soll eine Funktion berechnen, die (mathematische) Texte auf 0 oder 1 abbildet, je nachdem, ob sie einen korrekten Beweis beschreiben oder nicht. Eine Minimalbedingung für die Möglichkeit eines solchen Programmes ist damit die Existenz einer solchen Funktion. Wenn es i.A. keine eindeutige Antwort auf die Frage gibt, ob ein Text einen korrekten Beweis darstellt, ist automatisches Prüfen offenbar unmöglich.

Nun scheint dieser Umstand so offensichtlich gegeben zu sein, dass es fast pedantisch erscheinen mag, darauf hinzuweisen: Auf der ganzen Welt und über die Kluft von Jahrtausenden hinweg können etwa Menschen, wenn sie nur über die

[4]Wir bemerken an dieser Stelle, dass Rav, obzwar er das nicht explizit anmerkt, mit seinem Argument auch nicht StudienanfängerInnen im Blick hat, sondern forschende FachmathematikerInnen. Unsere Entgegnung ist also kein Angriff auf Ravs philosophische Position, sondern zeigt lediglich, dass diese ihrerseits keinen Einwand gegen das hier verfolgte Projekt darstellt.

[5]Siehe etwa Carter und Monks, [49], S. 8-9 zum Erkenntnisgewinn, den Studierende aus der Verwendung von Lurch gezogen haben.

nötige Vorbildung verfügen, Euklids Beweis für die Unendlichkeit der Primzahlen als richtig erkennen. Über die Kulturen und Zeitalter hinweg besteht in der Mathematik ein Konsens darüber, was ein korrekter Beweis ist. Der britische Philosoph Jody Azzouni spricht in diesem Zusammenhang von einer "benign fixation of mathematical practice" ([14]).]

Indes ist der Bezugspunkt dieser Einigkeit nicht leicht dingfest zu machen; heute Mathematiker sind mehrheitlich nicht des Altgriechischen so weit mächtig, dass sie Euklids Originaltext lesen könnten. So gibt es unzählige verschiedene sprachliche Objekte, die Euklids Beweis für verschiedene Zeitalter, Kulturen und Kontexte aufbereiten. Es ist damit klar, dass das, was unter dem Namen "Euklidischer Beweis" als korrekt beurteilt wird, kein einzelner, konkreter Text ist; vielmehr scheint es sich um etwas zu handeln, was selbst kein Text ist, sondern von dem die vielen verschiedenen Texte "Darstellungen" sind. Auch in einer als fehlerhaft erkennbaren Darstellung kann zuweilen der Euklidische Beweis als ein richtiger "Grundgedanke" ausgemacht werden.

Es lohnt sich gewiss, der Frage nachzugehen, was dieses etwas sein mag. Für die Zwecke dieser Arbeit genügt es aber, festzustellen, dass Azzouni's "benign fixation" jedenfalls keineswegs garantiert, dass MathematikerInnen sich in Bezug auf die Korrektheit eines konkreten *Textes* einig sein werden oder dass für die Korrektheit von argumentativen mathematischen Texten auch nur kultur-, kontext- und zeitunabhängige Standards bestehen. Anders gesagt: Selbst wenn die vielgepriesene Einigkeit der MathematikerInnen in Bezug auf *Beweise* gegeben wäre,[6] wäre sie es darum noch nicht zwangsläufig auch in Bezug auf *Beweistexte*. In der Tat werden wir nun einige empirische Ergebnisse besprechen, die diese Vorstellung – z.T. mit explizitem Bezug auf Azzouni – zumindest stark erschüttern, wenn nicht widerlegen.

4.3.1 Empirische Ergebnisse: Die Studie von Inglis et al.

Die These, hinsichtlich der Korrektheit mathematischer Beweistexte bestünde unter Mathematikern eine weitreichende Einigkeit, ist, wenn man sie so konkret versteht wie in der oben eingeführten "naiven Auslegung", einer empirischen Überprüfung zugänglich, indem man ein und denselben Beweistext von einer Reihe

[6] Auch hier gibt es sicher genug Anlass zum Zweifel; man denke etwa an die Ablehnung indirekter Beweise durch die Intuitionisten, die Auseinandersetzungen über die Korrektheit des Auswahlaxioms, die alternativen Begründungsstrategien der "experimentellen Mathematik" (die etwa in der Ankündigung eines "death of proof", siehe [107] ihren Ausdruck finden) oder die Diskussion zur Zulässigkeit von maschinengenerierten Beweisen, die für den menschlichen Nachvollzug schlicht zu lang sind, siehe etwa [196].

von Mathematikern unabhängig voneinander beurteilen läßt.[7] Eine entsprechend Studie ist tatsächlich durchgeführt worden, siehe Inglis et al. [117]. Wir wollen die Ergebnisse dieser Studie hier kurz erläutern und im Hinblick auf ihre Aussagekraft über die Möglichkeit einer automatischen Prüfung von natürlichsprachlichen Beweisen in einem didaktischen Kontext betrachten.

In der Studie von Inglis et al. wurde ein recht kurzer Beweis einer Grundaussage der Analysis – die Stammfunktionen der Funktion $x \mapsto \frac{1}{x}$ sind gerade die Funktionen der Form $\ln(x) + c$ mit $c \in \mathbb{R}$ – einer Gruppe von 109 MathematikerInnen (darunter Doktoranden und akademische Angestellte) vorgelegt, die ihn jeweils hinsichtlich seiner Korrektheit überprüfen sollten. Das Ergebnis ist frappierend: 29, also knapp 30 Prozent, der TeilnehmerInnen beurteilten den Beweis als korrekt, gut 70 als unzureichend, während einige erklärten, ohne weiteren Kontext darauf keine Antworte geben zu können. Überdies blieben die von der Korrektheit des Beweises Überzeugten auch dann bei ihrer Ansicht, wenn sie mit den Gründen konfrontiert wurden, die die anderen dazu gebracht hatten, den Beweis als unzureichend anzusehen (siehe [117], S. 1; Inglis et al. berichten, dass nur in einem einzigen Fall ein Proband seine oder ihre Einschätzung geändert hat.).

Auf den ersten Blick scheint dieses Ergebnis niederschmetternd. Hätte die Mehrheit der befragen ExpertInnen mit ihrer Beurteilung recht, hätten etwa 70% von ihnen den Beweis korrekt beurteilt.Man mag hier einen Grund zum Zweifel finden, ob es überhaupt einen allgemeinverbindlichen Begriff davon gibt, was ein korrekter Beweis ist – und tatsächlich führen Inglis et al. ihre Studie auch als Argument gegen diese Ansicht an. Wenn es aber keinen solchen Begriff gibt, kann es auch kein Programm geben, das prüft, ob ein gegebener Beweistext unter diesen – nicht existenten – Begriff fällt – nicht aus technischen Gründen sondern darum, weil schon die Spezifikation auf einer falschen Annahme beruhen würde: "Was *so* gar nicht erst sein kann, *wie* ein Erfahren das Dasein zu erfassen prätendiert, entzieht sich grundsätzlich der Erfahrbarkeit." (Heidegger, [190]).

Dennoch sollte man sich unseres Erachtens nach vor voreiligen philosophischen Schlussfolgerungen hüten: Azzouni's "benign fixation of mathematical practice" (siehe [15]), die Inglis et al. vermeintlich zurückweisen, mag eine andere Praxis betreffen als die hier beschriebene. Beweisbeurteilungen werden sich selten in einem einfachen "Ja oder nein" erschöpfen, es gibt auch ein "im Grunde richtig,

[7] Die z.T. sehr unterschiedlichen Reaktionen von verschiedenen Gutachtern für mathematische Fachzeitschriften in Bezug auf dieselbe Arbeit sind nicht ohne weiteres zur Überprüfung der naiven These geeignet, da sie neben der Korrektheit der Beweistexte noch eine Reihe anderer Aspekte betreffen, wie etwa die Relevanz des Themas, die Bedeutung der Ergebnisse oder die zu erwartende Anzahl interessierter LeserInnen. Entsprechend ausgewertet könnten sie aber durchaus Gegenstand einer weiteren empirischen Prüfung der naiven These werden, die einige der unten genannten Kritikpunkte an der Studie von Inglis et al. vermeidet.

aber da fehlt noch etwas – aber dieses Fehlende ist leicht zu ergänzen" oder "offenbar ein Tippfehler, hier sollte es A statt a heißen" etc. All das mögen Gründe sein, den vorliegenden Beweistext für in einem strikten Sinn inkorrekt zu halten, während er trotzdem für jeden kompetenten Leser erkennbar die Korrektheit der fraglichen Behauptung darlegt. Es muss aber eben auch festgestellt werden, dass sich das "im Grunde richtig" offenbar nicht auf konkrete Texte bezieht, sondern etwas (deutlich) abstrakteres, was Anlaß zu der Frage gibt, worauf denn eigentlich. Hiermit mag auch die völlig ungeklärte Frage zusammenhängen, wann zwei Beweise als gleich gelten sollten.[8][9][10]

Aber auch wenn der erkenntnistheoretische Status mathematischer Beweise durch die Ergebnisse von Inglis et al. nicht erschüttert wird, scheint sie doch in Bezug auf die Möglichkeit eines automatischen Beweisprüfers für natürlichsprachliche Texte, wie schon oben erwähnt, einem Unmöglichkeitsbeweis nahe zu kommen.

4.4 Die Kontextabhängigkeit des Verständnisses von "korrekter Beweis"

Nun zeigt die alltägliche Erfahrung in der mathematischen Praxis indes, dass bezüglich der Korrektheit von Beweisen Schwierigkeiten der von Inglis et al. beschriebenen Art verhältnismäßig selten auftauchen. Von einigen spektakulären Einzelfällen wie etwa Mochizukis Arbeiten zur interuniversellen Teichmüllertheorie (s.o.) abgesehen gibt es zumindest wenig Diskussionen darüber, welche Aussagen als Theoreme gelten und welche nicht. Ein Grund hierfür mag sein, dass mathematische Texte in der Praxis selten oder nie derart isoliert auftreten wie im Experiment von Inglis et al.: Ein Beweis steht nicht für sich, sondern er steht in einem Lehrbuch, das für ein gewisses Zielpublikum gedacht ist, welches zu einem bestimmten sozialen Kontext gehört (etwa Studierende der Physik ab dem dritten Semester); er steht in einem Zeitschriftenartikel, in dem Notation und Herangehensweise zuvor eingeführt wurden, und der seinerseits Teil einer Zeitschrift ist, die sich mit einem gewissen Anspruch an LeserInnen aus einem gewissen Kreis richtet; er steht im Kontext eines Forschungsprogrammes,

[8] Ein Versuch einer Antwort ist die "Homotopie von Beweisen, siehe etwa [113], Kap. 7.2.

[9] Argumente dafür, dass die analoge Frage nach der Identität von Algorithmen unbeantwortbar ist, finden sich in Blass et al. [18]. Wir halten diese Argumente nicht für wasserdicht, doch ist für eine detaillierte Auseinandersetzung mit dieser Frage hier nicht der Ort.

[10] Wir bemerken ferner, dass eine Vorstellung von der Identiät von Beweisen einer Vermutung zugrunde liegt, die Hilbert ursprünglich in seine Liste richtungsweisender Probleme aufnehmen wollte, nämlich die These, dass jede Behauptung im wesentlichen nur einen Beweis hat, siehe Thiele und Wos [192].

das auf gewisse Methoden und gewisse Ziele ausgerichtet ist; usw. Auch die Lösung zu einer Übungsaufgabe steht im Kontext einer Lehrveranstaltung, in der gewisse Grundlagen und Techniken zu Beginn vorausgesetzt, andere vermittelt, wieder andere womöglich explizit ausgeschlossen wurden (etwa analytische Lösungen in einer Vorlesung über Elementargeometrie).[11]. So besteht auch bei einer Klausurkorrektur erfahrungsgemäß zwischen verschiedenen Korrigierenden zumeist weitgehende Einigkeit darüber, was als korrekte Lösung anzusehen ist – und wo nicht, lassen sich diese Uneinigkeiten meist zufriedenstellend klären (wenn etwa eine durchaus korrekte Lösung nicht verstanden oder andererseits ein Fehler übersehen wurde). Das führt auf die These: Faktisch findet Mathematik in sozialen, historischen, akademischen usw. Kontexten statt, und im Rahmen eines fixierten Kontextes besteht weitgehende Einigkeit.[12] Die Ergebnisse von Inglis et al. würden dann eben zeigen, dass diese Kontexte bei der Beurteilung der Korrektheit eines Beweises eine zentrale Rolle spielen.

Damit läßt sich die Auffassung der Mathematik als eines Bereiches mit objektiven und daher auch intersubjektiven Wahrheitskriterien zwar (zumindest vorläufig) retten; die Möglichkeit eines automatischen Beweisprüfers im oben besprochenen Sinn ist dadurch indes nicht wahrscheinlicher gemacht, sondern zusätzlich erschüttert: Denn neben dem Text ist nun ein Kontext (oder sind mehrere Kontexte) in den Blick geraten, der für die Korrektheit mitverantwortlich ist, selbst wenn MathematikerInnen relativ zum Kontext sich einig sein sollten. Was in einem Artikel in den "Annals of Mathematics" ein akzeptabler Schluss sein mag, ist darum noch lange keine akzeptable Lösung in einer Klausur für Anfängerstudierende.[13] Unser fiktives Programm müsste entsprechend nicht allein einen Beweistext als Eingabe erhalten, sondern zudem eine Beschreibung des Kontextes relativ zu dem der Beweis beurteilt werden soll. Nun sind soziale Kontexte weder statisch, noch klar gegeneinander abgegrenzt und – schon aus diesen Gründen – auch nicht erschöpfend auflistbar. Ein sozialer (historischer, methodischer, akademischer...) Kontext ist kein Parameter, den man einem Programm übergeben könnte; er ist etwas, in das "man sich hineinfindet". Ein Programm zur Beurteilung von Beweisen müsste in der Lage sein, einen gegebenen Beweis in den jeweiligen Kontext einzuordnen und die in diesem Kontext implizit geltenden Standards zu erkennen und anzuwenden. Das zu verlangen hieße aber

[11]Dies wird auch durch die einschlägige Forschungsliteratur bestätigt; so heißt es etwa in Weber [201], S. 451: "The educational implications of this study concern the finding that the processes involved in proof validation are highly dependent on contextual factors, including the mathematical domain in which the proof is situated, the community that is evaluating the proof, and the author of the proof."

[12]Für die These der Kontextabhängigkeit mathematischen Wissens, und insbesondere der Rolle, die Beweise dabei spielen, siehe etwa Löwe und Müller, [141].

[13]Vgl. etwa Löwe und Müller, [141], S. 96.

kaum weniger als einen "künstlichen Menschen" herzustellen – und liegt jedenfalls weit jenseits unserer derzeitigen Möglichkeiten.[14]

Wir gelangen damit zu folgendem Fazit: natürlichsprachliche mathematische Beweise sind zu sehr sozial konstituierte Objekte, als dass eine definitive Entscheidung über "richtig" oder "falsch" bloß auf Basis des Textes selbst möglich wäre. Entsprechend gibt es auch kein Programm, das das leisten kann. Ein solches System müsste insbesondere in der Lage sein, sich in soziale Kontexte "hineinzufinden"; deren Standards sind aber implizit und nie völlig zu klären. Selbst wenn es also gelänge, eine derart starke künstliche Intelligenz zu realisieren, wäre ihre Ausgabe nur ein weiterer Standpunkt bezüglich der Qualität (Klarheit, Verständlichkeit, Vollständigkeit...) eines Beweistextes, das System also kein definitiver "Schiedsrichter". Damit wäre aber die Eindeutigkeit und Objektivität der Beurteilung, die den Einsatz automatischer Beweisprüfer zunächst attraktiv erscheinen läßt, gerade preisgegeben.

Für das angestrebte System bedeutet das: Es muss im Vorhinein der Anwendungskontext klar festgelegt werden; weiter ist das System so aufzubauen, dass auf Verschiebungen innerhalb dieses Kontextes flexibel reagiert werden kann. Der Einsatz ist stets auf diesen Kontext zu begrenzen; die Rückmeldungen des

[14]Die Kritik läßt sich über diesen Punkt hinaus treiben: Auch innerhalb eines gegebenen Kontextes ist ein Beweistext kein abgeschlossenes Objekt, das hinsichtlich seiner Korrektheit für sich genommen beurteilt würde, sondern Teil eines kommunikativen Vorgangs, der über die Abfassung und Publikation eines Beweises hinaus andauert. Beweisschritte werden von verschiedenen Seiten infrage gestellt; ob der Beweis als richtig gilt, hängt von der Fähigkeit des/r Autors/In oder eines Umfeldes von "SchülerInnen" oder "Angehörigen einer Schule" ab, auf diese Angriffe befriedigend zu reagieren; darin wird dann der Unterschied zwischen einer Verständnisschwierigkeit auf Seiten der/des LeserIn und einer Lücke im Beweis gesehen. Damit läge das Kriterium für die Akzeptanz eines Beweises als korrekt gar nicht allein im den Beweis darstellenden Text, sondern hinge zusätzlich von der Fähigkeit und Bereitschaft eines oder mehrerer "BefürworterInnen" des Beweises ab, im Verlauf eines offenen Kommunikationsprozesses Begründungen und Erklärungen "nachzuliefern". Wir übergehen diesen an sich durchaus interessanten Punkt an dieser Stelle, um im nächsten Abschnitt darauf zurück zu kommen. Wir bemerken allerdings bereits hier, dass dieses Phänomen auch in der Lehre durchaus vorkommt, etwa wenn TutorInnen ein abschließendes Urteil über eine Bearbeitung erst nach prüfender Rücksprache mit dem/der betreffenden Studierenden abgeben. Siehe dazu etwa die Studie von Moore [151], wo über eine der befragten Professorinnen gesagt wird: "She went on to explain that "context" meant her observation of the student's knowledge and performance in similar situations in a course and that she might ask the student what he or she was thinking with respect to this particular error". Ein solcher Zugang zur Beweiskorrektheit ließe sich womöglich durch die dialogischen Logiken modellieren, die u.a. durch Kuno Lorenz und Paul Lorenzen [121] sowie Jaakko Hintikka et al. [114] beschrieben wurden. Ob die Automatisierung auch in diesem Rahmen eine Rolle spielen kann, ist sicher eine wichtige Frage für die Weiterentwicklung derartiger Systeme. Erste Ansätze mag man in der Möglichkeit sehen, im "Dialog" mit dem Diproche-System einen Beweis bis zu dessen Akzeptanz zu verfeinern, ferner in dialogischen Aspekten von Systemen wie dem weiter oben besprochenen QED-Tutrix.

Systems sind immer relativ zum aktuellen Einsatzbereich zu lesen.

4.4.1 Beweisbewertung in Lehrkontexten: Moores Lehren

Es stellt sich nun freilich die Frage, ob die Festlegung eines Kontextes tatsächlich ausreicht, um zumindest relativ zum Kontext bezüglich der Frage, was als korrekter Beweis gilt, zu einer ausreichenden intersubjektiven Übereinkunft zu kommen, um zumindest eine Spezifikation für ein automatisches System zu haben. Auch hierzu gibt es empirische Untersuchungen: In einer Studie von Robert Moore [151] wurde untersucht, wie von Studierenden als Lösungen für Übungsaufgaben eingereichte Beweistexte von verschiedenen universitären Lehrenden bewertet wurden. Auch hier zeigten sich z.T. deutliche Abweichungen: auf einer Bewertungsskala von 1 bis 10 kam es beim gleichen Beweis immerhin zu Abweichungen von fast 5 Punkten ([151], S. 257). Diese Bewertungen bezogen sich auf mehrere Aspekte des Beweises zugleich, darunter logische Korrektheit und Korrektheit der Darstellung. Aber auch bei einer Einschränkung auf den Aspekt der Gültigkeit kam es zu erheblichen Uneinigkeiten: So wurden für die Gültigkeit eines Beweises auf einer Skala von 1 bis 5 alle Punktzahlen von 1 bis 4 vergeben ([151], S. 260). Die Schwierigkeiten, die sich bereits aus den Ergebnissen von Inglis et al. ergeben, scheinen sich damit selbst in diesem sehr verengten Kontext und bei einer Präzisierung der Hinsicht der Korrektur zu wiederholen.

Was bedeutet das nun für die Möglichkeit einer automatischen Beweisevaluation in der Lehre? An dieser Stelle ist es lohnend, die Gründe anzusehen, die Moore für diese Abweichungen angibt. Diese lagen nämlich nicht in unterschiedlichen Einschätzungen bezüglich der sachlichen Richtigkeit einer Lösung und auch nicht primär an Uneinigkeit darüber, was eine gute Darstellung ausmacht, sondern in der Frage, wie einzelne Fehler oder Mängel zu gewichten seien:

> "In sum, the professors agreed that the overall logical framework of Proof 2 was essentially correct, the student understood the proof and its key ideas reasonably well, the clarity could be improved, and surface features such as layout and punctuation should carry little, if any, weight in the overall score. But they disagreed on the seriousness of the student's error (...), the need for additional justifications, and the necessity of writing the proof in complete sentences." (Moore, [151], S. 257)

Der Grund für die unterschiedlichen Gewichtungen wiederum besteht laut Moores Untersuchung darin, dass die Rückmeldungen sich nicht auf den Text selbst, "not just the proof as a decontextualized mathematical artifact" (Moore,

[151], S. 256) bezogen, sondern auf "the student's understanding and proof-writing skills" (ebd.), also auf innere kognitive Eigenschaften der AutorInnen, auf die die KorrektorInnen vom Text aus zu schließen versuchten. So wurde dieselbe Textstelle einmal als ein bloßer Schreibfehler ausgelegt und ein anderes Mal als Ausdruck eines wesentlichen Mangels an Verständnis.

Dagegen bestand, wie oben deutlich wurde, inhaltlich durchaus Einigkeit bezüglich der Frage, was ein Fehler ist – und wo sie nicht bestand (etwa, weil ein Fehler übersehen wurde), ließ sie sich meist herstellen. Die von Inglis beschriebenen Schwierigkeiten scheinen hier also nicht aufgetreten zu sein. Gegen ein automatisches Prüfen natürlichsprachlicher Beweise im Lehrkontext scheint also hierin kein stichhaltiges Argument zu liegen. Im Gegenteil kann im Rahmen einer ständigen Interaktion zwischen Studierendem/r und System darauf verzichtet werden, durch eine "theory of mind" die kognitiven Eigenschaften einer/s Studierenden aus einem Text erschließen zu wollen; ob ein wesentlicher Verständnis- oder lediglich ein Tippfehler vorliegt, zeigt sich daran, ob eine als fehlerhaft gemeldete Stelle im nächsten Versuch adäquat verbessert werden kann. Allerdings ziehen wir aus den Ergebnissen von Moore folgende Lehren:

1. Die Aspekte an einem Beweistext, zu denen eine Rückmeldung erfolgen soll, sind zu trennen und die Rückmeldungen zu den jeweiligen Aspekten sind als zu den jeweiligen Aspekten gehörig kenntlich zu machen; insbesondere wünschenswert wären separate Rückmeldungen zum korrekten Gebrauch der Sprache, zur Typenkorrektheit, zur Korrektheit der logischen Schlüsse sowie dem Erreichen des Beweisziels.

2. Nicht verifizierbare Stellen sollten der/dem BenutzerIn gemeldet, aber nicht gewichtet werden. Rückmeldungen sollten den Charakter kritischer Nachfragen haben, auf die im Zuge einer Revision des Beweistextes reagiert werden kann, nicht den einer abschließenden Bewertung eines festen und fertigen Textes.

4.5 "Inhaltliche" und "formale" Mathematik

In seinem Hauptwerk "Beweise und Widerlegungen" [129] führt Imre Lakatos den Unterschied von "formaler" und "inhaltlicher" Mathematik ein. Dabei nimmt die "inhaltliche" Mathematik durch ihre Begriffe in irgend einer Form auf "Objekte" Bezug, die jenseits der reinen Argumentationsform auch in einer Art von Anschauung gegeben sind – und ist in ihren argumentativen Standards an diese Bedeutungen ihrer Begriffe gebunden. Die "formale" Mathematik dagegen hat den Gegenstandsbezug aus ihren argumentativen Standards eliminiert; Begriffe treten nur noch als Abkürzungen innerhalb eines Formalismus auf, in dem auch die Frage,

was ein korrektes Argument ist, durch eine rein syntaktische Antwort geklärt wird: Ein korrekter Beweis ist eine Zeichenfolge mit diesen und jenen Eigenschaften. Ein wesentlicher Unterschied zwischen inhaltlicher und formaler Mathematik liegt damit in der Rolle der Definitionen: Während diese in der formalen Mathematik Festlegungen mit rein konventionellem Charakter sind, durch den der fragliche Begriff überhaupt erst konstituiert wird, sind sie in der inhaltlichen Mathematik Versuche einer charakterisierenden Bezugnahme auf einen unabhängig von allen Definitionen vorliegenden Gegenstand oder Gegenstandsbereich, die insbesondere fehlerhaft und damit auch korrigierbar sein können. Dies wird in [129] eindrucksvoll am Beispiel des Begriffs des "Polyeders" herausgestellt.

Lakatos arbeitet erhebliche wissenschaftstheoretischen Unterschiede zwischen inhaltlicher und formaler Mathematik heraus: So ist inhaltliche Mathematik, ähnlich wie die Naturwissenschaft, in einer ständigen dialektischen Bewegung begriffen, in der Definitionen, Sätze und Beweise sich fortlaufend gegenseitig informieren und revidieren. Im Gegensatz dazu schreitet die formale Mathematik lediglich linear fort in der stetigen Vergrößerung eines Bestands an jeweils endgültig abgesicherten Sätzen – um den Preis freilich, dass diese "Sätze" keinerlei Gegenstandsbezug oder Bedeutung mehr haben. Wendet man einen rein formal erreichten Satz über einen formal definierten Polyederbegriff auf ein anschaulich – oder "handgreiflich" gegebenes Polyeder an, geht man zur inhaltlichen Mathemaitk über, wodurch der formale Begriff samt den zugehörigen Herleitungen ihren Status als sichere und abschließende Wahrheiten einbüßen.

Nun genügt ein kurzer Blick in die in [129] dargestellte Gedankenbewegung zum Begriff des Polyeders und dem Beweis des Eulerschen Polyedersatzes, um einzusehen, dass diese Art des Umgangs mit Beweisen sich jedenfalls einer Automatisierung der hier angestrebten Art entzieht:[15] Die TeilnehmerInnen des dort präsentierten Dialoges beziehen sich auf eine Vorstellung von Polyedern, die sie zwar irgendwie "haben", aber eben nicht in explizierter oder auch nur abschließender Form. Dieser zunächst anschaulich gegebene und dann unter dem Druck zahlreicher Beispiele immer wieder veränderte und durchaus kontrovers verhandelte Begriff bildet die Grundlage für die Bewertung der jeweils vorgeschlagenen Beweise des Eulerschen Polyedersatzes, wobei aber auch umgekehrt diese Beweise einen Einfluss auf den Begriff haben. Der Beweis spielt hier – in den Worten von Lakatos – die Rolle eines "Gedankenexperimentes", anhand dessen Vorstellungen von Begriffen allererst geklärt bzw. entwickelt werden. Ist das Gedankenexperiment anschaulich überzeugend, wird der Beweis zum Kriterium für eine Definition des Polyeders: Diese muss so gewählt sein, dass der Beweis mit ihr funktioniert.

[15] Ein voll formalisierter und damit maschinell überprüfbarer Beweis für den Eulerschen Polyedersatz wurde von Jesse Alama im Rahmen seiner Dissertation konstruiert, siehe [4].

Wollte man ein automatisches System an dieser Art von Bewegung teilnehmen lassen, müsste es zunächst über anschauliche Vorstellungen der relevanten Begriffe verfügen, die zudem an diejenigen der anderen DialogteilnehmerInnen anschlussfähig sind: Die anschaulichen, prä-explikativen Vorstellungen von Begriffen wie "Zahl", "Kugel" oder eben "Polyeder" werden z.B. aus Alltagserfahrungen mit diversen Objekten hervorgegangen sein, sowie aus einer gewissen Funktion dieser Begriff in der Alltagssprache. Dadurch werden diese Begriffe aber wesentlich menschliche: Sie sind zunächst eingebunden in menschliche Handlungs- und Kommunikationskontexte. Ein System, das sich hier beteiligen soll, müsste also eine vorbegriffliche Alltagserfahrung aufbauen können, und dazu eine solche, die für Menschen nachvollziehbar und relevant ist; zudem müsste das System als Dialogpartner anerkannt werden. Schon an dieser Stelle sieht man, dass ein solches System aufzubauen letztlich darauf hinausliefe, den Menschen technisch zu reproduzieren. Selbst wenn das aber gelänge, wäre für unsere Zwecke wenig gewonnen: Menschliche TutorInnen gibt es ja bereits.

Nun findet sich verschiedentlich der Einwand, Lakatos' Arbeit habe wenig Bezug zur mathematischen Praxis. Auseinandersetzungen wie die in [129] dargestellten mag es zwar im Verlauf der Geschichte gegeben haben (aber auch als historische Rekonstruktion der Entstehung des Polyederbegriffs ist Lakatos' Darstellung nicht sonderlich präzise, siehe z.B. [127]) kommen aber inzwischen kaum noch, wenn überhaupt, vor in einer Zeit, in der die formale Mathematik zum fachlichen Standard geworden ist. Es ist also nicht offensichtlich, ob der Unterschied zwischen inhaltlicher und formaler Mathematik für den für uns relevanten Bereich – das anfängliche Beweisen – überhaupt relevant ist.

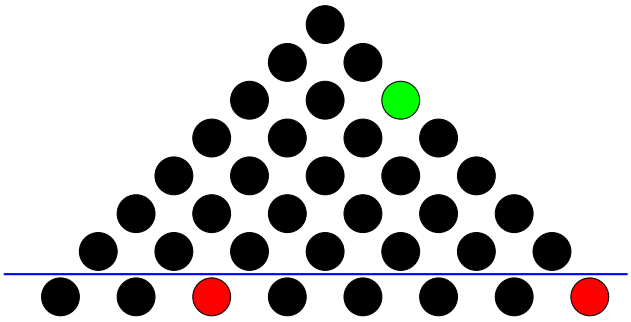
Wir nehmen zu dieser Frage Stellung, indem wir einige Beweise vorstellen, die beim Beweisenlernen im Anfängerbereich zweifellos vorkommen könnten – und unseres Erachtens nach auch sollten.

Unser erstes Beispiel ist ein "Beweis ohne Worte" für die Gaußsche Summenformel für die Summe der ersten n natürlichen Zahlen.[16]

Beispiel 1: Zeige, dass $\Sigma_{i=1}^{n} i = \binom{n+1}{2}$, wobei $\binom{n+1}{2}$ die Anzahl der Möglichkeiten bezeichnet, aus einer $(n+1)$-elementigen Menge zwei Elemente auszuwählen.

Lösung:

[16] Auf diesen Beweis bin ich durch MathOverflow aufmerksam geworden, siehe `https://mathoverflow.net/questions/8846/proofs-without-words(zugegriffen14.10.2021)`, wo ihn der Benutzer "Mariano Suárez-Álvarez" gepostet hat. An der gleichen Stelle wird seine Erfindung Loren Larson zugeschrieben. Eine zweibändige Sammlung solcher "Bildbeweise" ist [168], [169].



Die Lösung ist ebenso unmittelbar überzeugend wie elegant: Oberhalb der Linie befinden sich gerade $\Sigma_{i=1}^{n} i$ Punkte, darunter $(n+1)$, und wie im Diagramm angedeutet, besteht zwischen den Paaren von Elementen der Punktmenge unterhalb der Linie und der Menge der Punkte darüber eine Bijektion – folglich gibt es von beiden gleich viele. Obwohl unmittelbar und zurecht überzeugend und zudem im Hinblick auf die kombinatorische Deutung der Gaußschen Summe überdies auch noch sehr erhellend, mag man zweifeln, ob man hier überhaupt von einem "Argument" sprechen kann. Weit entfernt davon, eine klare deduktive Struktur zu besitzen, in der Annahmen aufgestellt und sukzessive Folgerungen abgeleitet werden, hat man es hier nicht einmal mit einem im engeren Sinne sprachlichen Gebilde zu tun, sondern lediglich mit einem Bild. Dennoch spricht wenig dagegen, diesen "Beweis" Studierenden vorzulegen; ihn zu verstehen und sich von ihm überzeugen zu lassen heißt, das Allgemeine in der speziellen Figur "sehen", d.h. anhand einer einzelnen Figur eine unendliche Anzahl von Fällen in den Blick zu bekommen – diese erstaunliche Fähigkeit ist ein wichtiges mathematisches Werkzeug.[17] Anderseits sollte man diese Lösung, würde sie von einer/m Studierenden vorgelegt, gewiss nicht als unzureichend kritiseren, sondern im Gegenteil die oder den Betreffende/n dazu ermutigen, diese ebenso brilliante wie kreative Denkweise weiter zu entwickeln.

Dass in der Möglichkeit derartiger Beweise ein Problem für Ansätze – wie etwa das Hilbertsche Programm – besteht, alles korrekte mathematische Argumentieren durch formale Schlussregeln einzufangen, wurde z.B. in A. Bundy und M. Jamnik [23] bemerkt. Jedenfalls liegt hier ein Typ von Beweis vor, der für die automatische Verifikation weitgehend unzugänglich ist, solange jedenfalls, wie es nicht gelingt, einen Rechner mit einer menschlichen Form von "Anschauung" auszustatten – womit wohl sämtliche Probleme verbunden wären, die wir oben in Bezug auf die inhaltliche Mathematik erwähnt haben. Zwar werden eben in der gerade zitierten Arbeit [23] Ansätze zu einer automatischen Verifikation diagrammatischer Beweise gemacht, aber diese beruhen eben auf einer Formalisierung gewisser Arten von Diagramm (nämlich solcher "Punktkonfigurationen" wie der oben

[17] Für eine Behandlung der mathematik-philosophischen Implikationen der Möglichkeit solcher Beweise siehe R. Brown [28], Kapitel 3.

abgebildeten) und gewisser Arten, mit ihnen umzugehen (verschieben, drehen, zusammenfügen...). Von einer Erschöpfung der Möglichkeiten, die visuelle Anschauung mathematisch korrekt überzeugungsbildend einzusetzen, ist man jedenfalls weit entfernt.

Beispiel 2:[18] Gandalf bricht am Montag um 8.00 im Auenland auf und reitet nach Bruchtal, wo er um 19.00 ein Date mit Galadriel hat. Am Dienstag bricht er um 8.00 in Bruchtal auf und reitet denselben Weg zurück ins Auenland, wo er um 19.00 ankommt. Zeige, dass es eine Uhrzeit gibt, zu der er am Montag und Dienstag am gleichen Ort war.

Lösung: Wir stellen uns den Gandalf vom Dienstag als einen zweiten Gandalf vor, der am Montag reitet. Offenbar müssen sie sich irgendwo treffen und sind damit zur selben Zeit am selben Ort.

Hier haben wir als Begründung zwar einen Text, der aber von einer deduktiven Herleitung oder gar einer Formalisierung weit entfernt ist. Man mag Gandalfs Weg ebenso wie die verstreichende Zeit als reelle Intervalle auffassen, die Situation damit durch den Verlauf reeller Funktionen modellieren und so in der Behauptung den Zwischenwertsatz wiedererkennen – aber von all dem ist im obigen Argument nicht die Rede. Man findet überhaupt wenig von einem schrittweisen Fortschreiten, einem Anwenden von Schlussregeln etc., sondern stattdessen ein anschauliches Gedankenexperiment, das freilich sofort überzeugt. Dieses Argument zu verstehen erfordert es, sich die beschriebene Situation wie auch deren in der Lösung angegebene Verschiebung bildlich zu vergegenwärtigen und wiederum in dieser Vorstellung das allgemeine zu sehen. Dieses Gedankenexperiment zu finden erfordert ein gehöriges Maß an Kreativität und Fantasie, um überhaupt darauf zu kommen, die zeitlich getrennten Prozesse zu einem einzigen zu "überlagern". Auch dieses Argument gibt einen guten Eindruck davon, was mathematisches Argumentieren erfordert und sollte, wo es von einer oder einem Studierenden vorgeschlagen wird, zur Ermutigung Anlass geben, nicht aber zur Kritik an einem Mangel an formaler Stringenz.

Die oben angegebenen Beispiele für schwer oder gar nicht adäquat formalisierbare Beweise mögen künstlich oder jedenfalls von der universitären Lehrpraxis recht weit entfernt wirken. Wir weisen daher an dieser Stelle darauf hin, dass Aufgaben, bei denen der entscheidende Schritt zur Lösung im Einnehmen oder Wechsel zu einer gewissen "Sichtweise" liegt, mindestens im Kontext der endlichen Kombinatorik – und damit auch der Wahrscheinlichkeitsrechnung –

[18]Siehe z.B. Hemme [101], Aufgabe 44; Hemme verweist seinerseits auf eine Arbeit aus der Psychologie von 1935.

häufig vorkommen und folglich auch als für die universitäre Lehre von Belang sind. Wir betrachten hierzu eine typische Aufgabe aus der elementaren Kombinatorik.

Beispiel: 10 Personen sitzen in einer Reihe. Zwei davon, Anna und Bernd, sitzen nebeneinander. Bestimme, wie viel Möglichkeiten es dafür gibt.

Lösung: Wir fassen Anna und Bernd als eine Person auf. Damit hat man 9 Personen anzuordnen, wofür es 9! Möglichkeiten gibt. Da Anna und Bernd auf 2 Arten angeordnet werden können, ergibt sich die Lösung $2 \cdot 9!$.

In der folgenden klassischen[19] Aufgabe ist eine elementare quantitative Anschauung entscheidend, während eine formale – rechnerische – Lösung deutlich umständlicher ist:

Beispiel: Gegeben sind zwei Gläser mit je 500 ml Fassungsvermögen. Das eine Glas enthält 300 ml Apfelsaft, das andere enthält 300 ml Wasser. Es werden 100 ml Wasser in den Apfelsaft geschüttet und verrührt. Anschließend werden 100 ml des Apfelsaft-Wasser-Gemisches in das Wasserglas zurückgegossen. Ist nun mehr Wasser im Saftglas oder mehr Saft im Wasserglas?

Lösung: Offenbar ist gerade so viel Wasser im Saftglas, wie im Wasserglas an Wasser fehlt – und dieses wiederum wurde durch Saft ersetzt. Folglich sind beide Größen gleich.

Hier ist eine formal-rechnerische Lösung natürlich möglich: Mit etwas Aufwand lassen sich zunächst die Mischverhältnisse und Mengenvolumen in beiden Gläsern nach dem ersten Umschüttvorgang ermitteln und daraus dann auch die nach dem zweiten. Der Reiz der oben beschriebenen Lösung besteht nun aber gerade darin, diese rechnerische "Nachahmung" des beschriebenen Prozesses durch eine simple anschauliche Überlegung zu umgehen und den Lösungsweg auf diese Weise deutlich abzukürzen.[20]

[19] Siehe z.B. `https://www.spiegel.de/wissenschaft/mensch/raetsel-der-woche-mehr-wasser-im-wein-a-1035004.html` (zugegriffen 14.10.2021).

[20] Ein weiteres bekanntes Beispiel derselben Art ist etwa folgendes (siehe z.B. `https://www.matheplanet.com/default3.html?call=viewtopic.php?topic=69410&ref=https\%3A\%2F\%2Fwww.google.com\%2F` (zugegriffen 14.10.2021)): Zwei Züge fahren mit einer konstanten Geschwindigkeit von 50 km/h auf einem Geraden Schienenstück der Länge 100km/h aufeinander zu. Zwischen ihnen fliegt eine Fliege mit 30km/h hin und her, wobei sie stets umkehrt, wenn sie die Spitze eines der Züge erreicht hat. Wie weit ist die Fliege in dem Moment geflogen, in dem die Züge kollidieren? Die rechnerisch-"nachahmende" Lösung würde hier auf eine unendliche (geometrische) Reihe führen, wohingegen es leicht einzusehen ist, dass die Zeitspanne bis zur Kollision genau eine Stunde beträgt und die Fliege sich in dieser Zeit also genau 30km weit bewegt haben wird.

Die soeben betrachteten Beispiele mögen automatisches Beweisprüfen unrealistisch erscheinen lassen. Offenbar ist es in jedem Fall erforderlich, dem jeweils dargestellten Gedankenexperiment in der Vorstellung zu folgen, damit das Argument seine überzeugende Wirkung entfalten kann. Dazu scheint eine Anschauung von Zeiten, Wegen, Personen, Wassergläsern etc. erforderlich zu sein, die einem Rechner kaum zu vermitteln sein dürfte. Überdies ist die erforderliche Anschauung von recht delikater Natur: Nicht um präzise und partikuläre (etwa bildliche) Vorstellungen kann es hier gehen (diese wären allenfalls geeignet, in einem Spezialfall eine berechtigte Überzeugung hervorzubringen), sondern um eine "allgemeine" oder "schematische" Anschauung[21], die an einzelnen Operationen zugleich deren Durchführbarkeit in jedem Fall erkennt. Dies ist im Fall des oben gegebenen "Bildbeweises" besonders deutlich. Diese Form der Anschauung auf eine für einen automatischen Beweisprüfer hinreichend verlässliche Art einem Rechner zu vermitteln dürfte kaum wesentlich einfacher sein als die Herstellung einer starken künstlichen Intelligenz und liegt jedenfalls außerhalb der Reichweite der hier in Erwägung gezogenen Methoden.

Als Gegensatz diene uns das folgende Beispiel, in dem eine klare, sprachlich vermittelte, deduktive Struktur erkennbar ist.

Beispiel: Zeige: Für alle Mengen A, B, C ist $A \cap (B \cup C) = (A \cap B) \cup (A \cap C)$.

Lösung:

"$\subseteq$": Es sei $x \in A \cap (B \cup C)$. Dann ist $x \in A$ und $x \in B \cup C$. Also ist $x \in B$ oder $x \in C$. Falls $x \in B$, folgt $x \in A \cap B$ und also $x \in (A \cap B) \cup (A \cap C)$. Falls $x \in C$, folgt, $x \in A \cap C$ und also $x \in (A \cap B) \cup (A \cap C)$. In jedem Fall gilt $x \in (A \cap B) \cup (A \cap C)$.

"$\supseteq$": Es sei $x \in (A \cap B) \cup (A \cap C)$. Dann ist $x \in (A \cap B)$ oder $x \in (A \cap C)$. Falls $x \in A \cap B$, ist $x \in A$ und $x \in B$, also auch $x \in B \cup C$ und also $x \in A \cap (B \cup C)$. Falls $x \in A \cap C$, ist $x \in A$ und $x \in C$, also auch $x \in (B \cup C)$, also wiederum $x \in A \cap (B \cup C)$.

QED.

Dieser Aufgabentyp, und entsprechend dieser Typ von Beweis, der aus einer sprachliche dargestellten, strukturierten, deduktiven Kette besteht, kommt in der universitären Mathematik häufig vor. Darin liegt nun keineswegs eine akademische Pedanterie, die gegen die Originalität, Ästhetik und Anschaulichkeit von Argumenten wie den zuvor gegebenen auszuspielen wäre; für den Aufbau größerer Theorien, für die Behandlung abstrakter Gegenstandsbereiche, für die schlicht keine einfache Anschauung zu haben ist und ferner für solche, an denen

[21] Zum Begriff des Schematismus der Anschauung und seiner Bedeutung für die Mathematik, besonders die Rolle von Diagrammen in der Geometrie, siehe z.B. Kant, [120].

die Anschauung aus reinen Ressourcengründen scheitert oder gar – wie etwa häufig in der Wahrscheinlichkeitsrechnung – in die Irre führt ist diese Art von Argument unerlässlich. Sie stellt damit zurecht einen wichtigen Bestandteil der universitären Mathematikausbildung dar.

Mit Bezug auf das zu entwickelnde System ergibt sich damit folgendes Fazit: Das System kann sinnvollerweise nur verwendbar sein für Beweise mit klarer begrifflicher Grundlage, klarer deduktiver Struktur und einer festen Menge von formal präzise formulierbaren Voraussetzungen – mit anderen Worten, für Beweise aus dem Bereich der formalen Mathematik, auf die sich das System also wird beschränken müssen. Diese steht aber nicht für sich, sondern muss durch inhaltliche Aspekte ergänzt werden. Das System sollte also für formale Mathematik entwickelt werden, muss im Einsatz aber stets durch Beispiele und Übungen aus der inhaltlichen Mathematik ergänzt werden.

4.6 Exkurs: Zur Hermeneutik mathematischer Texte

Ein automatischer Beweisprüfer für natürlichsprachliche Beweise entnimmt einem natürlichsprachlichen Beweis seinen formalisierbaren Gehalt und wendet auf diesen Methoden der formalen Verifikation an. Dieses sinnentnehmende Verarbeiten modelliert bis zu einem gewissen Grad eine/n menschliche/n LeserIn. Wird ein Beweis zunächst einmal verstanden als ein rationales Argument für eine/n menschliche/n LeserIn, so kann auch der automatische Beweisprüfer in einem gewissen Sinn als "Leser" eines Beweistextes aufgefasst werden (de facto ist in Gesprächen um solche Systeme auch oft die Rede davon, dass das System den Beweis "liest"). Legt man das eben erwähnte Verständnis von Beweisen zugrunde, ist es also die Aufgabe des automatischen Prüfers, zu ermitteln, ob ein Beweis für einen Menschen (eine/n Angehörige/n der Zielgruppe des Beweises) überzeugend sein wird. In diesem Sinn stellt die maschinelle Verarbeitung ein Modell des menschlichen Lese- und Auslegungsprozesses dar. Wir wollen diesen Abschnitt dazu nutzen, kurz die Begrenztheit dieser Modellierung aufzuzeigen – und damit auch der Möglichkeiten, die automatisches Prüfen bereitstellen kann.

Automatische Beweisprüfer wie Naproche, SAD oder auch Diproche ermitteln für jede Textstelle eine Liste von Aussagen, die an diesem Punkt entweder als wahr angenommen oder bereits bewiesen sind. Diese geben die Basis für die Verifikation weiterer Beweisschritte ab: Eine an dieser Stelle folgende Behauptung läßt sich entweder mit einem gewissen Methodenvorrat aus den Elementen dieser Liste ableiten oder sie wird als nicht verifizierbar gemeldet. Diese Liste verfügbarer Aussagen ist somit ein Modell für das Wissen, das einem/r menschlichen LeserIn an dieser Stelle des Textes zur Verfügung steht. Nun gibt es viele Hinsichten, in denen diese Modellierung zu kurz greift:

1. Zunächst die Beschränkung des "Wissens" auf einen Vorrat vorliegende Aussagen; mit der Lektüre eines Textes erwerben menschliche LeserInnen neben explizit formulierten Aussagen auch Methoden, Weisen der Begriffsverwendung, Schlussweisen etc. Was zu Beginn eines Lehrbuchs noch überfordern mag, kann nach einigem Fortschritt einfach sein, nicht weil es explizit erwähnt wurde, sondern weil z.B. bereits eine Reihe analoger Fälle behandelt wurde, so dass sich ein entsprechender Schritt einfach ergänzen läßt. Dieser Fortschritt im Hinblick auf die verfügbaren Inferenzen wird in längeren Texten auch genutzt, indem Schritte, die derart zur Routine geworden sind, kurz gehalten oder fortgelassen werden. Ein solcher Erwerb von deduktiver Kompetenz wird durch die oben genannten Systeme nicht abgebildet.

2. Auch mathematische Texte sind häufig auf zahlreiche Weisen ambig.[22] Ein besonders offensichtliches Beispiel sind Argumente wie das folgende: "Angenommen, n ist gerade. Dann ist $n(n+1)$ auch gerade. Angenommen, n ist ungerade. Dann ist $n(n+1)$ wiederum gerade. Also ist $n(n+1)$ in jedem Fall gerade.". Hier ist jedem/r kompetenten LeserIn klar, dass die Annahme "n ist gerade" in dem Moment zurückgenommen wird, wo "n ist ungerade" angenommen wird – es wäre ja auch kaum sinnvoll, unter explizit widersprüchlichen Voraussetzungen zu argumentieren. Betrachten wir nun zum Vergleich folgendes Argument: "Angenommen, n ist gerade. Dann ist n durch 2 teilbar. Weiter sei n prim. Dann ist n nur durch 1 und sich selbst teilbar. Also ist $n = 2$.". Hier bleibt die erste Voraussetzung "n gerade" offenbar bis zum Ende des Textes in Geltung. Was die reine Abfolge von Sätzen des Typs "Annahme" bzw. "Folgerung" angeht, stimmen beide Texte überein. Trotzdem führen beide zu deutlich unterschiedlichen logischen Repräsentationen. Die Basis, auf der menschliche LeserInnen sich hier offenbar für eine Lesart entscheiden, ist pragmatisch: Davon ausgehend, dass der Text ein Ziel verfolgt, wird als dieses Ziel im ersten Text die Begründung der Aussage ausgemacht, dass $n(n+1)$ für $n \in \mathbb{N}$ stets gerade ist, im zweiten hingegen das Beweisziel, dass 2 die einzige gerade Primzahl ist; dann wird die logische Struktur des Textes so gebildet, dass dieses Ziel erreicht wird. Menschliche LeserInnen wählen also aus verschiedenen Lesarten von Beweistexten unter anderem nach *pragmatischen* Kriterien aus.

3. Zusätzlich zur einen Auffassung der jeweils besprochenen Sache nimmt ein/e menschliche/r LeserIn Bewertungen des Gelesenen vor, insbesondere hinsichtlich der Bedeutsamkeit des jeweiligen Inhalts. Diese Bewertung

[22] Für eine ausführliche Darstellung von Ambiguitäten in der Sprache der Mathematik siehe Ganesalingam [83], Kapitel 4.

wiederum kann sich im Laufe der Lektüre verschieben und zu einem nichtlinearen Leseprozess führen, bei dem immer wieder vorgeblättert wird (z.B., um die Bedeutsamkeit des gerade vorliegenden Inhalts einschätzen zu können) und früheres wiederholt wird. Ebenso werden Inhalte vergessen, die als weniger wichtig eingestuft werden. Dieses "Vergessen" ist einerseits keineswegs ein reines Defizit des Menschen gegenüber einer maschinellen Verarbeitung, in der alle bisherigen Schritte gespeichert werden. Vielmehr liegt in dieser Art Steuerung gerade ein positives Phänomen, das das Verstehen erst ermöglicht: Kämen etwa in einem Beweis am Ende eines Lehrbuches von einigen hundert Seiten sämtliche bisher bewiesenen Aussagen – einschließlich von Folgerungen, die lediglich im Beweis der explizit benannten Aussagen auftreten – als mögliche Gründe für eine Folgerung zur Verfügung, wäre der Nachvollzug nahezu unmöglich.[23] Andererseits muss diese Eigenschaft menschlicher LeserInnen beim Verfassen von Beweistexten berücksichtigt werden: Soll ein Beweistext seine Funktion als rational überzeugendes Argument erfüllen, ist also ein gezielter Einsatz von Mitteln zur Steuerung der Aufmerksamkeit erforderlich, wozu etwa explizite Vorblicke ("ehe wir den Beweis in Angriff nehmen, beweisen wir einige Hilfsresultate"), Bewertungshinweise ("dieses zentrale Lemma werden wir häufig benutzen") oder Erinnerungen ("nach Lemma 3") gehören.

4. Auch im Verstehen von Beweistexten gibt es den Eindruck, dass etwas "mehr oder weniger", "im Grunde" oder "immer besser" verstanden ist. Oft führt erst die wiederholte Lektüre eines zunächst unverständlichen Beweistextes schließlich zu der Überzeugung, dass hier ein gültiges Argument vorliegt. Dieses "mehr" ist im Rahmen des von den hier betrachteten Systemen abgebildeten Verstehensmodells kaum abzubilden. Ein möglicher Grund mag sein, dass sich auch im Verstehen mathematischer Beweistexte ein "hermeneutischer Zirkel" einstellt: Einerseits wird ein grober Beweisplan, eine Beweisabsicht, ein Eindruck davon, "wie es gehen soll", aus den vorliegenden einzelnen Sätzen geformt; andererseits werden diese Sätze dann aus diesem Gesamtplan in ihrer Funktion, und damit mittelbar auch in ihrem Inhalt, verständlich. Hiermit liegt ein Ansatz zur Verbesserung und Verfeinerung (oder ggf. auch zur Korrektur) des erkannten Beweisplanes vor und die Bewegung setzt sich fort.[24]

[23] Diese Schwierigkeit hat sich auch im Naproche-Projekt gezeigt, sobald versucht wurde, längere Texte zu verarbeiten. Ein erster Ansatz zur Lösung war, einen Algorithmus für die Vorselektion potenziell relevanter Prämissen zu implementieren, siehe etwa Kühlwein et al. [46], was als ein Versuch angesehen werden kann, die positiven Aspekte der "Vergesslichkeit" für das System nutzbar zu machen.

[24] Vgl. z.B. Gadamer, "Wahrheit und Methode", [82], S. 275: "Wir erinnern uns hier der

Den vorstehenen Aspekten ist eines gemeinsam: Sie sorgen dafür, dass menschliches Beweisprüfen zunächst in Bezug auf die Frage der Korrektheit zu einer Reihe von Urteilen führen kann, die sich nicht auf ein einfaches "korrekt" oder "fehlerhaft" reduzieren lassen. Sie haben überdies zur Folge, dass sich das Ergebnis einer solchen Prüfung mit der Zeit ändern (wenn etwas erneute Lektüre des gleichen Textes zur Klärung führt) und zwischen verschiedenen Individuen variieren kann. Damit stehen sie jedoch im Gegensatz zu dem, was von einer automatischen Prüfung erwartet wird: Deren Ergebnis sollte binär sein (Akzeptanz oder Meldung eines Fehlers) und deterministisch (erneute Prüfung sollte zum gleichen Ergebnis führen). Selbst wenn sich etwa die Arbeitsweise von Diproche durch eine realistische Simulation menschlichen "Vergessens" an die eines/r menschlichen LeserIn annähern ließe, würde der Nutzen einer maschinellen Prüfung dadurch reduziert: Mit einem "mehr oder wenig" wird der/die Studierende des Anfängerbereichs ebenso wenig anfangen können wie die/der professionelle MathematikerIn, der oder die sich um die Absicherung eines aufgrund seiner Komplexität umstrittenen Beweises bemüht. (Nicht auszuschließen ist allerdings, dass eine solche Simulation ihren Sinn im Rahmen einer automatischen "Stilberatung" für mathematische Texte finden könnte, die etwa darauf hinweisen könnte, dass an einer Stelle auf die Verwendung eines früheren Ergebnisses explizit hingewiesen werden sollte.) Insofern sie andererseits als spezifische und positive Eigenschaften menschlichen Verstehens anerkannt werden müssen, stellen sie prinzipielle Beschränkungen für die Möglichkeit des automatischen Prüfens natürlich(sprachlich)er Beweise dar.

Für die Entwicklung eines beweisprüfenden Systems sind die vorstehenden Überlegungen insofern nützlich, als sie eine bescheidene Herangehensweise nahelegen. Der Abstand zu von einer/m menschlichen LeserIn, der etwa durch dessen begrenzte Gedächtniskapazität entsteht, kann dadurch klein gehalten werden, dass man sich auf kurze, einfache und überschaubare Beweise beschränkt. Das Wechselspiel von Teil und Ganzem im menschlichen Verstehen hat nur einen geringen Einfluss, wenn man sich auf bestimmte Themengebiete und Beweistypen konzentriert, so dass das "Gesamtbild" und die Rolle der Teile darin "unmittelbar" einsichtig ist. Insbesondere gilt es bei der Entwicklung des Systems, Tendenzen zu allzu großer Allgemeinheit zu widerstehen und sich bewusst auf einige überschaubare und beherrschbare Kontexte zu beschränken.

---

hermeneutischen Regel, dass man das Ganze aus dem Einzelnen und das Einzelne aus dem Ganzen verstehen müsse. (...) Die Antizipation von Sinn, in der das Ganze gemeint ist, kommt dadurch zu explizitem Verständnis, daß die Teile, die sich vom Ganzen her bestimmen, ihrerseits auch dieses Ganze bestimmen. (...) So läuft die Bewegung des Verstehens stets vom Ganzen zum Teil und zurück zum Ganzen. Die Aufgabe ist, in konzentrischen Kreisen die Einheit des verstandenen Sinnes zu erweitern."

4.6.1 Andere Aspekte des Verstehens[25]

Abschließend zu diesem Abschnitt wollen wir noch einige Aspekte des Beweisverstehens bemerken, die über die obigen noch deutlich hinausgehen und z.T. mit einer Prüfung des Beweises nur noch eingeschränkt zu tun haben. Wir tun dies hauptsächlich, um einer möglicherweise durch maschinelle Modelle des Lesens nahegelegten Verengung des Verstehensbegriffs entgegen zu wirken.

Eine deutlich anders akzentuierte Perspektive auf das Verstehen wurde etwa von Gadamer [82] im Ausgang von Heidegger [190] vorgestellt: Nach Heidegger ist das Verstehen ein fundamentaler Aspekt der menschlichen Existenz, der keineswegs auf Texte beschränkt ist, sondern jeden Umgang mit etwas betrifft. Zum "In-der-Welt-sein" gehört es wesentlich und ursprünglich, die Dinge im Rahmen eines planenden Entwurfs (und herkommend aus einer Tradition) als sinnhaft auszulegen. So ist die Welt ein "Horizont", in dem die Einzeldinge aus zweckhaften Bezügen ihren Sinn erhalten; Heideggers Paradebeispiel ist die Werkstatt.[26] Zum Verständnis eines Beweistextes gehört in diesem Sinn z.B. auch immer ein Erfassen des Sinnes von Beweisen überhaupt, wozu etwa eine Einsicht in den deduktiven Aufbau und die Methodik ebenso gehören wie eine Erfahrung von der Unzuverlässigkeit von Intuition oder der Verallgemeinerung aufgrund der Beobachtung von Einzelfällen; insbesondere gehört zum Beweisverstehen so auch stets ein "Beweisbedürfnis"[27].

In Bezug auf das Verstehen mathematischer Texte würde sich hieraus ergeben, dass zum Verstehen die Einordnung in einen (lebensweltlichen) Gesamtkontext gehört, sowie die Möglichkeit, den Text innerhalb dieser als sinnvoll zu erfahren.[28][29]

Diesem Ansatz folgt Gadamer in seiner Analyse des Textverstehens, indem er dessen historische Dimension betont.

So kann es etwa zum Verstehen eines mathematischen (Beweis-)Textes

[25] Auf den Ansatz, die Verstehensbegriffe von Heidegger und Gadamer auf das Verstehen mathematischer (Beweis-)texte anzuwenden, wurde ich im Zuge einer gemeinsamen Arbeit durch Bernhard Schröder aufmerksam gemacht.

[26] Siehe z.B. Heidegger [190], S. 68-69: "*Ein* Zeug ≪ist≫ strenggenommen nie. Zum Sein von Zeug gehört je immer ein Zeugganzes, darin es dieses Zeug sein kann, das es ist. (...) In solchem gebrauchenden Umgang unterstellt sich das Besorgen dem für das jeweilige Zeug konstitutiven Um-zu; (...) Der nur ≪theoretisch≫ hinsehende Blick auf Dinge entbehrt des Verstehens von Zuhandenheit."

[27] Siehe etwa [205] zum Begriff des Beweisbedürfnisses und seiner Bedeutung für die Dikaktik des Beweisenlernens.

[28] Es ist insofern eine konsequente Wendung dieser Haltung ins Performative, wenn Heidegger in Vorträgen und Vorlesungen immer wieder auf die konkrete Vortragssituation Bezug nimmt, siehe etwa "Grundbegriffe der Metaphysik", [111] S. 500f oder "Gelassenheit", [110] S. 11f.

[29] Einen ähnlichen Aspekt hat, aus didaktischer Perspektive, Martin Wagenschein unter dem Begriff der "Einwurzelung" beschreiben, siehe etwa Wagenschein, "Ursprüngliches Verstehen und exaktes Denken", [200].

wesentlich dazu gehören, ihn in einen historischen Kontext einzuordnen. Betrachten wir als Beispiel folgenden Satz ([91], S. 187):

> "Zu jeder ω-widerspruchsfreien rekursiven Klasse κ von Formeln gibt es rekursive Klassenzeichen r so, daß weder $v\mathrm{Gen}r$ noch $\mathrm{Neg}(v\mathrm{Gen}r)$ zu $\mathrm{Flg}(x)$ gehört (wobei v die freie Variable aus r ist)."

Gewiss ist dieser Satz in einem gewissen Sinn verständlich, wenn man sich ggf. die Mühe macht, einige Definitionen nachzulesen; mindestens wird man wohl, wenn man die Definitionen der verwendeten Begriffe kennt, imstande sein, einen Beweis für diese Aussage als solchen zu erkennen. Dennoch erscheint dieser Modus des Verstehens des Gödelschen Satzes, in dem er als ein technisches Resultat gleichberechtigt neben einer Reihe anderer technischer Resultate steht, die man ebenso gut hätte formulieren können, defizitär: Seinen "eigentlichen" Sinn erkennt man erst, wenn man ihn in seiner Bedeutung als (starken, nahezu vernichtenden) Einwand gegen einen gewissen Ansatz (das Hilbertsche Programm) erkennt, auf eine historische Problemsituation – die mathematische Grundlagenkrise – zu reagieren.[30] Aus diesem historischen Kontext heraus erhält der Satz eine Bedeutungsdimension, die über die Einsicht in seine Wahrheitsbedingungen bzw. die Fähigkeit, ein korrektes Argument für ihn als solches zu erkennen, deutlich hinausgeht. Schon das Verständnis eines mathematischen Satzes kann also seine *historische Einordnung* einschließen.

Nicht anders verhält es sich mit Beweisen: Wenn etwa Russell und Whitehead in der Principia Mathematica auf S. 379 der *Principia Mathematica* [176] das zentrale Lemma beweisen, aus dem viele Seiten später folgt, dass $1 + 1 = 2$ ist, so ist "Verständnislosigkeit" eine verständliche Reaktion auf diesen Text, solange

[30] Mehr noch: Der Gödelsche Unvollständigkeitssatz ist ausdrücklich für rekursive Axiomsysteme formuliert. Dass VertreterInnen des Hilbertschen Programmes hierauf nicht einfach mit dem Einwand reagieren können, es sei dann eben nach einem nicht-rekursiven System Ausschau zu halten, hat seinen Grund in der Gleichsetzung von "rekursiv" und "durch einen idealisierten Agenten effektiv entscheidbar", welche durch die Church-Turing-These ausgedrückt wird (siehe etwa [55]). Für die Bedeutung von Gödels Satz ist also weiterhin wesentlich, dass man die Formalisierung der Entscheidbarkeit durch die Rekursivität akzeptiert, was allerdings sich allerdings erst durch die Arbeiten von Turing, besonders das 1936 – die fünf Jahre nach Gödels Unvollständigkeitssatz – publizierte "On Computable Numbers, with an Application to the Entscheidungsproblem" [195] durchgesetzt hat, siehe etwa [10]. (Dieser Umstand wird von Gödel in einem 1965 verfassten Zusatz zum Unvollständigkeitssatz ([92], S. 71f) auch bemerkt, siehe [99].) Auf derselben Gleichsetzung beruht auch die Akzeptanz von Turings und Matiyasevichs Arbeiten als Lösungen für das 1928 durch Hilbert formulierte Entscheidungsproblem und das zehnte der 1929, also 7 Jahre vor Turings Arbeit, formulierten Hilbertschen Probleme, siehe etwa [99]. Dass diese Arbeiten ihre Funktion als Lösungen erfüllen können, beruht also auf historischen Entwicklungen, die erst deutlich nach der Formulierung dieser Probleme eingetreten sind; man kann also sogar sagen, dass sich ihre Bedeutung – und damit die Art, wie sie zu verstehen sind – mit der historischen Entwicklung verschoben hat.

kein weiterer Kontext vorliegt. Als überzeugendes Argument ist dieser Beweis eben nur für den oder die wirksam, der oder die versteht, auf welches Problem hier mit welchem Ansatz reagiert wird und wie sich die "Spielregeln" des Beweises aus dieser Situation ergeben.

Wie sich zeigt, ist Gadamers Analyse des Verstehens ([82]) auch für mathematische Texte zutreffend: Auch für das Verstehen mathematischer Texte und Sätze (und selbst einzelner Begriffe; siehe dazu etwa Lakatos [129], besonders die Ausführungen zum Begriff der stetigen Funktion) ist es von Bedeutung, sie vor dem Hintergrund einer historischen Problemsituation zu lesen und ebenso, sie dann wiederum in einen Bezug zur aktuellen Situation zu setzen, in der der/die LeserIn steht: Da das logizistische Programm in der Mathematik insgesamt als gescheitert anzusehen ist, wird man etwa auch Russells und Whiteheads in dessen Kontext geleistete Arbeit kaum als "tieferen Grund" ansehen wollen, an die Wahrheit von $1 + 1 = 2$ zu glauben oder gar ein erhöhtes Vertrauen in diese Aussage daraus ziehen.[31]

Ein weiterer Aspekt, den Gadamer in "Wahrheit und Methode" betont, ist der, dass das Verstehen eines Textes immer auch in gewisser Hinsicht ein "Anwenden" ist, in dem der Text als Antwort auf eine Frage aufgefasst wird.[32] Auch dieser Aspekt findet sich im Verstehen mathematischer Texte wieder. Betrachten wir als Beispiel den Satz von Silver:

Es sei κ eine singuläre Kardinalzahl mit überabzählbarer Konfinalität. Angenommen, es gilt $2^\lambda = \lambda^+$ für jede unendliche Kardinalzahl $\lambda < \kappa$. Dann gilt $2^\kappa = \kappa^+$.

Dieser Satz findet sich z.B. in Devlins einführendem Lehrbuch zur Mengenlehre [58]. Sicherlich ist der Satz in einem gewissen, begrenzten Sinn allein dadurch "verstehbar", dass man sich an die entsprechenden Definitionen erinnert. Indes provoziert dieser Satz für sich genommen die Frage, "was das soll". Der volle Sinn des Silverschen Satzes erschließt sich, wenn man die Frage kennt, auf die er antwortet: Zum Zeitpunkt, zu dem er bewiesen wurde, war (durch die Ergebnisse von Easton) bekannt, dass die Kontinuumsfunktion (die einer Kardinalzahl die Kardinalität ihrer Potenzmenge zuordnet) auf den regulären Kardinalzahlen durch die Zermelo-Fraenkelsche Mengenlehre ZFC nur in sehr geringem Ausmaß

[31] Vgl. Gadamer, Wahrheit und Methode [82], S. 274: "Das Verstehen selber ist nicht so sehr als eine Handlung der Subjektivität zu denken, sondern als Einrücken in ein Überlieferungsgeschenen, in dem sich Vergangenheit und Gegenwart beständig vermitteln."

[32] Siehe z.B. Gadamer [82], S. 352: "Wer verstehen will, muß also fragend hinter das Gesagte zurückgehen. Er muß es als Antwort von einer Frage her verstehen, auf die es Antwort ist. (...) Man versteht den Text ja nur in seinem Sinn, indem man den Fragehorizont gewinnt, der als solcher notwendigerweise auch andere mögliche Antworten umfaßt. Insofern ist der Sinn eines Satezs relativ auf die Frage, für die er eine Antwort ist (...)".

festgelegt wird – jeder Funktionsverlauf, der einigen wenigen trivialen Bedingungen genügt, ist mit ZFC vereinbar. Silvers Satz gibt nun eine negative Antwort auf die damit naheliegende Frage, ob sich dieses Ergebnis auch auf singuläre Kardinalzahlen ausdehen läßt (vgl. etwa die Diskussion in [58], S. 116). Mit diesem Hintergrundwissen erschließt sich eine Sinndimension am Satz von Silver, die in der reinen Formulierung dieses Satzes verborgen bleibt.

Es wäre ein ebenso verdienstvolles wie schwieriges Projekt, die Verstehensbegriffe etwa der hermeneutischen Tradition Gadamers und Heideggers auf mathematische Texte anzuwenden; wir beschränken uns auf diese kurzen Andeutungen, die jedenfalls genügen mögen, um voreiligen Beschränkungen der Vorstellung vom Verstehen von Beweisen – und ggf. darauf fußenden Operationalisierungen – entgegen zu treten.[33]

4.7 Nachteile eines Beharrens auf einem rein formalen Verständnisses von "korrekter Beweis"

> "(...) dann aber trägt auch jeder mathematische Begriff das nötige Korrektiv in sich selbst einher; ist er unfruchtbar oder unzweckmäßig, so zeigt er es sehr bald durch seine Unbrauchbarkeit und er wird alsdann wegen mangelnden Erfolgs fallen gelassen. Dagegen scheint aber jede überflüssige Einengung des mathematischen Forschungstriebes eine viel größere Gefahr mit sich zu bringen und eine um so größere, als dafür aus dem Wesen der Wissenschaft wirklich keinerlei Rechtfertigung gezogen werden kann; denn das *Wesen* der *Mathematik* liegt gerade in ihrer *Freiheit.* "
>
> G. Cantor ([35], S.182)

Wir haben oben argumentiert, dass die aktuelle mathematische Praxis mit dem Einsatz automatischer Systeme anstelle menschlicher LeserInnen zum Zweck der Prüfung auf Korrektheit nicht vereinbar ist. Gründe dafür waren u.a. die Instabilität und Kontextabhängigkeit in der Beurteilung von Texten durch menschlicher LeserInnen, die Bedeutung der "inhaltlichen" Mathematik sowie verschiedene Aspekte der Hermeneutik mathematischer Texte. Faktisch weist die mathematische Praxis so Eigenschaften auf, die mit einer automatischen Verifikation unvereinbar sind, jedenfalls soweit diese definitive Standards von

[33] Dass Mathematik von ihrer Geschichte bzw. von den Fragen her verständlich wird, die sie motiviert haben, ist ein Gedanke, der sich in der Didaktik und Philosophie der Mathematik an verschiedenen Stellen findet; so etwa in der genetischen Didaktik Wagenscheins [200], in Toeplitz' "genetischer" Einführung in die Analysis [194] oder in der Philosophie von Lakatos [129].

"richtig" und "falsch" etablieren soll, die die Möglichkeiten menschlicher Begutachtung ersetzen oder gar im einem wesentlichen Sinn überschreiten.

Indes wäre eine andere Art denkbar (und durch die oben genannten Hinderungsgründe auch geradezu nahegelegt), wie das automatische Prüfen Bedeutung für die mathematische Praxis erlangen könnte: Nicht dadurch nämlich, dass die entsprechenden Programme sich der mathematischen Praxis annähern, sondern umgekehrt dadurch, dass die Akzeptanz durch automatische Verifikationsverfahren als verbindlicher Standard für die Korrektheit von Mathematik normativ gesetzt wird.[34] So könnte etwa die Korrektur von Übungsaufgaben und Klausuren oder Abschlussarbeiten wie auch die Begutachtung von Einreichungen bei mathematischen Fachzeitschriften maschinell erfolgen, nicht zuletzt mit dem Ziel, die Instabilität und Kontextabhängigkeit der menschlichen Prüfung zu überwinden; wie es nahezu unmöglich geworden ist, eine mathematische Arbeit in einer Fachzeitschrift zu publizieren, die nicht in LaTeX gesetzt ist,[35] könnte die Bestätigung der Korrektheit durch einen automatischen Beweisprüfer zum Kriterium für die Annahme werden – was entsprechend hieße, die Arbeit maschinenlesbar zu verfassen.[36]

Die Etablierung eines automatischen Systems, welches als Kriterium für die Korrektheit mathematischer Argumente gelten soll, ist ein Beispiel für eine Reihe von Vorhaben der Art, die Standards dafür, was als "gute Wissenschaft", "richtige Wissenschaft" oder überhaupt als "Wissenschaft" gelten kann, für ein bestimmtes Wissenschaftsgebiet oder gar für die Wissenschaft als ganze klar und endgültig zu bestimmen. Solche Vorhaben unterliegen einer Kritik, die der Wissenschaftstheoretiker Paul Feyerabend in seiner Arbeit "Against Method" ("Wider den Methodenzwang") [76] kritisiert hat. Die Strategie seiner Kritik

[34]Siehe hierzu etwa Heintz [100], S. 9: "Der Erfolg der künstlichen Intelligenz hat weniger mit der Qualität von Programmen zu tun als vielmehr mit dem Verhalten jener, die simuliert werden.".

[35]Für die Analogie zu LaTeX, vgl. etwa Löwe, [140], S. 1.

[36]Ein solches Szenario wird z.B. am Ende von Avigad und Harrison [5] angedeutet: "We expect within a couple of decades seeing important pieces of mathematics verified will be commonplace and by the middle of the century may even become the new standard for rigorous mathematics." Deutlicher wird sie im QED-Manifest [171] ausgesprochen, insbesondere in den Punkten 4 und 5, wo es mit Bezug auf das QED-System, ein angestrebtes System zur formalen Verifikation von Mathematik, heißt: "Fifth, the QED system may help preserve mathematics from corruption. (...) The QED system could offer an antidote to such a tendency. The standard, impartial answer to the question "Has it been proved?" could become "Has it been checked by the QED system?". Such a mechanical proof checker could provide answers immune to pressures of emotion, fashion, and politics.". Wie in Punkt 4 erwähnt, wird hiermit ausdrücklich ein kultureller Einfluss angestrebt, der weit über die Praxis der Mathematik hinausgeht: "The QED system will provide a beautiful and compelling monument to the fundamental reality of truth. It will thus provide some antidote to the degenerative effects of cultural relativism and nihilism.", ([171], Punkt 4). Siehe auch die Diskussion in Koepke [126] und Löwe [140].

besteht dabei unter anderem darin, für zahlreiche plausible Beispiele einer Maxime "guter Wissenschaft" historische Fälle aufzuzeigen, in denen sich ihre konsequente Befolgung hemmend auf die Entwicklung der Wissenschaft ausgewirkt hätte bzw. die Entwicklung gerade dadurch zustande kam, dass dieser Maxime zuwider gehandelt wurde.[37] Wir sind der Überzeugung, dass Feyerabends Kritik auch auf die Idee eines einheitlichen formalen Standards für "die Mathematik" anwendbar ist. Die Ausführung dieser Überzeugung würde den thematischen Rahmen dieser Arbeit sprengen. Wir skizzieren daher nur kurz einige Punkte einer solchen Anwendung.

Wir wollen also kurz darlegen, warum wir eine Entwicklung hin zur automatisierten Verifikation als Kriterium für die Korrektheit von Mathematik nicht für wünschenswert, ja im Hinblick auf die Entwicklung der Mathematik für geradezu schädlich halten. Für den Einsatz von Systemen wie Diproche folgt hieraus, dass sie stets als Hilfe zum Erlernen eines Teilaspektes des Beweisens angesehen werden sollten und ihr Einsatz in Lehrveranstaltungen entsprechend durch die Darlegung und Übung anderer Aspekte "ausbalanciert" werden sollten. Dazu gehört insbesondere, anschauliche und inhaltliche Formen des Begründens und Beweisens sowie suggestive und heuristisch erklärende Formen der Darstellung vorzuführen, zu ermutigen, einzufordern (z.B. durch Übungs- und Klausuraufgaben) und zu bestärken.[38]

In der Tat gibt es zahlreiche historische Beispiele von mathematischen Theorien, die in ihrer Entstehungsphase nicht den Voraussetzungen für eine formale Verifikation genügt haben und bei denen diese Möglichkeit auch erst aufkam, nachdem sie bereits verbreitet, ausgearbeitet und angewendet worden waren. Somit würde ein Beharren auf formalen Standards für die Mathematik deren Entwicklung gerade dort behindern, wo neue Ideen und

[37] Siehe z.B. Feyerabend, [76], p. 14: "The idea of a method that contains firm, unchanging and absolutely binding principles for conducting the business of science meets considerable difficulty when confronted with the results of historical research. We find, then, that there is not a single rule, however plausible, and however firmly grounded in epistemology, that is not violated at some time or other. It becomes evident that such violations are not accidental events (...). On the contrary, we see that they are necessary for progress."

[38] Zur Bedeutung der "Freiheit der Mathematik" in der Lehre siehe etwa Wittman, [207], S. 9: "Die didaktischen Zwänge, die von Didaktikern fixiert und von Lehrerinnen und Lehrern nachvollzogen werden, beruhen weniger auf mangelndem psychologisch-didaktischem Verständnis für das Denken von Kindern und weniger auf mangelnder pädagogischer Sensibilität, sondern zu allererst auf mangelnder Einsicht in das wahre Wesen von Mathematik. Nicht nur das Lernen, sondern das Fach wird verfälscht, wenn die in der Mathematik wesenhaft gegebene Freiheit durch fachfremde Zwänge eingeschränkt wird. Wer das tut, hat nicht verstanden, was Mathematik ist."

Begriffe entstehen.[39][40] Schon ein so vermeintlich selbstverständliches Kriterium wie Widerspruchsfreiheit ist potenziell geeignet, mathematische Entwicklungen zu blockieren: Tatsächlich gibt es eine Reihe von historischen Beispielen für Fälle, in denen MathematikerInnen über längere Zeit in einem System gearbeitet haben, das ihnen als widersprüchlich bekannt war (siehe etwa [50]), etwa in der naiven Mengenlehre oder der ursprünglichen Fassung des Infinitesimalkalküls.[41] Erst die längere Arbeit mit einem solchen System hat allmählich eine als stabil erkannte Praxis etabliert, aus der dann schließlich der Aufbau einer konsistenten Theorie hervorgehen konnte. Solche Entwicklungen hätten nicht stattfinden können, hätte man bereits im Anfangsstadium auf Widerspruchsfreiheit bestanden oder gar die Kommunikation, Verbreitung und Verwendung solcher Gebiete dadurch beschränkt, dass die entsprechenden Verbreitungskanäle – heute primär die Publikation in Fachzeitschriften – an formale Verifizierbarkeit gebunden worden wären.[42]

Auch auf methodischer Ebene könnte sich das Szenario einer verbindlichen automatischen Verifikation hemmend auswirken. Zwar betonen etwa die AutorInnen des QED-Manifests: "Although there are a variety of logics, there is little doubt that one can describe all important logics within an elementary logic, such as primitive recursive arithmetic, about which there is no doubt, and within which one can reliably check proofs presented in the more controversial logics. We plan to build the QED system upon such a "root logic" (...)" ([171]);

[39] Vgl. etwa Lakatos, [129], S. IX: "Keine einzige der 'schöpferischen' Perioden und kaum eine der 'kritischen' Theorien würden Aufnahmen in den formalistischen Himmel finden (...). Auf solche Begriffe musste Newton vier Jahrhunderte warten, bis ihm schließlich Peano, Russell und Quine in den Himmel halfen, indem sie die Analysis formalisierten."

[40] Dieser Punkt wird von den AutorInnen des QED-Manifests in folgender Bemerkung gestreift ([171]): " Let us add that we need take no position on the question whether mathematicians can or should profit from the use of formal notations in the discovery of serious, deep mathematics. The QED system will be mainly useful in the final stages of proof reporting, similar to writing proofs up in journals, and perhaps possibly never in the discovery of new insights associated with deep results.". Nun gehört der Austausch von Ergebnissen aber zum kollektiven Entdeckungsgeschehen der Mathematik wesentlich dazu. Die vermeintlichen "final stages" schließen eine Entwicklung eben nicht ab, sondern sind ein Teil von ihr.

[41] Ein häufig genanntes Beispiel für eine Inkonsistenz im ursprünglichen Infinitesimalkalkül, in dem statt mit Grenzwerten mit aktual "unendlich kleinen" Größen Δ gearbeitet wurde, entsteht etwa in der Ableitung der Quadratfunktion (siehe etwa Mortensen, [149], S. 8 oder [50], S. 27): Man bildet den Differenzenquotienten $\frac{(x+\Delta)^2-x^2}{\Delta}$ und formt zu $2x$ um, wobei man einerseits Δ kürzt und andererseits im letzten Schritt Δ als Summanden fortläßt, also Δ einmal als von 0 verschieden und einmal als gleich 0 behandelt. (Ich bedanke mich an dieser Stelle bei den OrganisatorInnen der Lesegruppe zur Philosophie der Mathematik an der Universität Konstanz, in deren Verlauf ich auf die Arbeiten zur inkonsistenten Mathematik aufmerksam gemacht wurde).

[42] Siehe z.B. Byers [30] zur Rolle von Widersprüchen und Paradoxien für das mathematische Denken.

doch auch eine "root logic" sollte innerhalb der Mathematik noch revidier- und angreifbar sein. So hat sich z.B. als Reaktion auf die Verwendung widersprüchlicher Theorien in der Mathematik eine "inkonsistente Mathematik" gebildet, die in einer parakonsistenten Hintergrundlogik arbeitet, in der sehr grundlegende logische Prinzipien außer Kraft gesetzt sind, siehe etwa Mortensen [149]. Es steht zu vermuten, dass eine "root logic", die keiner möglichen Kritik unterliegen könnte, welche selbst als Teil der Mathematik anzusehen wäre, schlicht leer sein müßte. Und auch die vermeintlich universell konsensfähige primitiv-rekursive Arithmetik wird etwa in der ultrafinitistischen Mathematik abgelehnt[43] – womit die ultrafinitistische Mathematik durch das avisierte QED-System nicht mehr als Mathematik gelten könnte.

Als abschließender Punkt sei die Frage gestellt, welche Rolle die Publikation und Akzeptanz fehlerhafter Beweise für die Entwicklung der Mathematik spielt oder historisch gespielt hat. So scheint der Weg zu einem korrekten Beweis bisweilen so zu erfolgen, dass zunächst ein fehlerhafter Beweis gefunden und für richtig gehalten wird, der dann erst deutlich später korrigiert und vervollständigt wird – ggf. auch mit Methoden, die zum Zeitpunkt der Abfassung des fehlerhaften Beweises noch nicht zur Verfügung standen. Ein Beispiel für eine solche Entwicklung scheint der Beweis des 4-Farben-Satzes zu sein, siehe Appel und Haken [13][44] oder Connor und Robertson [52]: Ein Beweisversuch des Mathematikers Alfred Kempe vom Ende des 19. Jahrhunderts wurde erst über zehn Jahre nach seiner Publikation als fehlerhaft erkannt, steuerte aber letztlich wesentliche Ideen zur Lösung durch Appel und Haken bei. In diesem Fall hat also die Publikation des fehlerhaften Beweises die Beweisidee bis zum Zeitpunkt ihrer korrekten Durchführbarkeit "aufbewahrt" und so einen wesentlichen Schritt auf dem Weg zum korrekten Beweis dargestellt. Gewiß sind weitere Weisen denkbar und in der Entwicklung der Mathematik auch anzutreffen, in denen Fehler sich letztlich als fruchtbar erwiesen haben.[45] Derartige Formen des Austausches würden durch zu starke Ansprüche an die Korrektheit von Beweisen unterbunden. Eine auf automatisierter Prüfung basierende mathematische Praxis, die eine solche Hemmung ihrer kreativen Potenz nicht hinnehmen will, müßte eine Kultur der Kommunikation von Beweistexten etablieren, die als lücken-

[43] So beweist die primitiv-rekursive Arithmetik etwa die Totalität der Exponentialfunktion, die von Ultrafinitisten i.A. abgelehnt wird; siehe z.B. [39].

[44] Vgl. z.B. Appel und Haken [13], S. 2: "Kempe's argument was extremely clever, and although his "proof" turned out not to be complete it contained most of the basic ideas that eventually led to the correct proof one century later."

[45] So schreibt etwa Goro Shimura in seinem Nachruf auf Yutaka Taniyama, [181], S. 190: "Though he was by no means a sloppy type, he was gifted with the special ability of making many mistakes, mostly in the right direction. I envied him for this, and tried in vain to imitate him, but found it quite difficult to make good mistakes".

und fehlerhaft bekannt sind. Nun ist einerseits fraglich, ob solche Beweistexte eine ähnliche Wirkung entfalten könnten wie solche, die zunächst für korrekt gehalten und erst später als problematisch erkannt wurden; wichtiger indes ist die Frage, wie dann die Begutachtung solcher Beweistexte erfolgen sollte. Von den Möglichkeiten automatischer Verifikation ist es jedenfalls weit entfernt, einen fehler- oder lückenhaften Beweistext daraufhin zu beurteilen, ob er "interessant", "vielversprechend", "ausbaufähig" etc. ist.

4.8 Präzise Zielsetzung

Aus den oben besprochenen Kritikpunkten leiten wir für ein realistisches und praktisch einsetzbares System zur automatischen Prüfung natürlichsprachlicher Beweise folgende Maximen ab:

- Der Einsatz ist auf klar festgelegte und eng begrenzte Anwendungskontexte anzulegen.
- Die zu verifizierenden Beweise müssen einem deduktiv-formalen Stil haben und insbesondere Kontexten angehören, in denen eine solche Vorgehensweise auf natürliche Weise möglich ist.
- Beim Einsatz des Systems muss darauf geachtet werden, die Kontextbezogenheit der jeweiligen Rückmeldungen herauszustellen und insbesondere den Eindruck zu vermeiden, diese stellten eine objektive, absolute und abschließende Beurteilung eines Beweistextes dar.

Zusammen mit den oben angeführten Betrachtungen führt diese Zielsetzung und sogleich auf einige Hinsichten, in denen der Einsatz des Systems didaktisch zu ergänzen ist:

(1) Der Einsatz von Diproche kann sich nur auf formale Mathematik beziehen, während doch der Erwerb von Kompetenzen im Bereich der inhaltlichen Mathematik insofern wichtiger ist, als die formale Mathematik erst von dieser her ihren Sinn erhält. Neben Diproche-tauglichen Beweisaufgaben sind also stets solche zu stellen, die jenseits von einem formal-deduktiven Vorgehen den Einsatz der Anschauung, das heuristische Erkunden, das kreative gedankliche Experimentieren etc. fordern und fördern.

(2) Wie die Studie von Moore et al. [146] zeigt, führen Korrekturanmerkungen in Bearbeitungen von Beweisaufgaben zwar durchaus in einem wesentlichen Anteil der Fälle dazu, dass die Studierenden in der Lage sind, ihre Bearbeitung im intendierten Sinn zu verbessern (und, so darf man hoffen, diese Verbesserung womöglich auch in weiteren Bearbeitungen beizubehalten). Deutlich weniger

häufig sind diese aber nach Moore et al. auch in der Lage, den den Korrekturen zugrunde liegenden Grund korrekt zu erkennen und damit eine Einsicht in das Problem einer Schlussweise, den Fehler in einer gewissen Notation etc. zu erlangen oder etwa zu erkennen, warum eine Formulierung irreführend ist. Im Sinne des Erwerbs des mathematischen Beweisens als einer begründeten und motivierten Praxis, im Gegensatz zu einem formalisierten sprachlichen Ritualsystem, ist diese Einsicht aber zweifellos wünschenswert. Hier sind also weiter menschliche Lehrende gefragt, das Verständnis für den Sinn hinter den Korrekturen zu fördern.

Wir gelangen damit zu folgender Zielsetzung:

Zu entwickeln ist ein automatisches System, das formal-deduktive Beweistexte in einer kontrollierten natürlichen Sprache vor dem Hintergrund jeweils klar spezifizierter Kontexte mit nach verschiedenen Aspekten untergliederten Rückmeldungen versieht, die für die intendierte Zielgruppe bei einer Entwicklung dieser Beweistexte hilfreich sind; der Charakter einer Bewertung soll bei den Rückmeldungen vermieden werden. Die Eingabe soll dabei als Freitext mit möglichst wenig Reglementierungen erfolgen. Die Kontextabhängigkeit der Rückmeldungen ist zu betonen; weiter ist der Einsatz des Systems durch Aufgaben zu ergänzen, die den Einsatz von Kreativität und Anschauung fördern.

Wir werden uns nun der Beschreibung des Diproche-Systems zuwenden, das ein Ansatz ist, dieses Ziel zu erreichen.

Kapitel 5

Funktionen und Aufbau des Systems

In diesem Kapitel werden wir die Konzeption des Diproche-Systems vorstellen. Da heißt insbesondere, dass wir aufzeigen, welche Komponenten es gibt, welche Aufgaben die einzelnen Komponenten erfüllen und wie diese im Diproche-System zusammenwirken. Die Funktions*weise* der einzelnen Komponenten und ihre Implementierung werden hier *nicht* besprochen, sondern erst im nächsten Kapitel. Ebenfalls ausgespart werden hier eher technische Komponenten wie etwa die Vorverarbeitung des Eingabestrings oder die Bildschirmausgabe der Rückmeldungen, die für die didaktisch wesentlichen Funktionen kaum relevant sind.[1]

5.1 Prüfen

Die Hauptfunktion des Diproche-Systems ist das Prüfen von Beweistexten, einschließlich entsprechender Rückmeldungen an den/die BenutzerIn. Wie im letzten Kapitel beschrieben, sollten diese nach Kategorien gegliedert erzeugt und angezeigt werden.

5.1.1 Sprachliche Prüfung

Das Modul zur sprachlichen Prüfung dient der Kontrolle, ob ein eingegebener Beweistext sich formal so weit an die Eingabesprache von Diproche hält, dass eine weitere Verarbeitung möglich ist. Die Prüfung erfolgt in vier Stufen:

1. Im ersten Schritt wird geprüft, ob alle verwendeten Zeichen zum Ausdrucksrahmen der jeweiligen Spielwiese gehören. Falls nicht, wird die

[1] Eine kurze Überblicksdarstellung der Funktionsweise des Diproche-Systems wurde in Carl und Krapf [34], gegeben.

weitere Verarbeitung mit einer entsprechenden Fehlermeldung (Liste der unbekannten Zeichen) abgebrochen. Andernfalls wird mit dem zweiten Schritt fortgefahren.

2. Im zweiten Schritt wird geprüft, ob alle verwendeten Wörter zum Lexikon gehören, also gültige Schlüsselwörter in der Eingabesprache von Diproche (s.u.) sind. Falls nicht, wird die weitere Verarbeitung mit einer entsprechenden Fehlermeldung (Liste der unbekannten Wörter) abgebrochen. Andernfalls wird mit dem dritten Schritt fortgefahren.

3. Im dritten Schritt wird überprüft, ob das Textannotationsmodul jeden Satz einer der Hauptkategorien "Behauptung", "Annahme" und "Annotation" zuordnen kann. Falls nicht, wird die weitere Verarbeitung mit einer entsprechenden Fehlermeldung (nicht kategorisierbarer Satz und Hinweis auf die bekannten Formulierungen) abgebrochen. Andernfalls wird mit dem vierten Schritt fortgefahren.

4. Im vierten Schritt wird bei Behauptungen und Annahmen überprüft, ob ihr (bereits formalisierter) Inhalt eine Formel im Sinne der Formelsyntax darstellt. In einigen Fällen, wo das nicht der Fall ist, wird ein Kandidat für den vermutlich intendierten Inhalt ermittelt.[2] Wird eine solche vermutete Korrektur vorgenommen, erfolgt nach einer entsprechenden Hinweisemeldung die Weiterverarbeitung. Kann trotz Korrektur kein wohlgeformter Inhalt erkannt werden – etwa bei Eingabe des Satzes "Es gilt $(A \wedge \wedge \rightarrow B)$ – wird die weitere Verarbeitung mit einer entsprechenden Fehlermeldung (nicht verarbeitbarer Inhalt) abgebrochen. Andernfalls ist die sprachliche Prüfung damit abgeschlossen und der Text wird an die Module zur inhaltlichen Verifikation weitergegeben.

5.1.2 Zielverfolgung

Die bloße Prüfung eines Argumentes auf die logische Korrektheit der Folgerungsschritte entscheidet noch nicht, ob das Argument auch die These stützt, für die es ein Argument sein soll. Ein Beweis ist stets ein Beweis für etwas, das sogenannte "Beweisziel". Im Verlauf eines mathematischen Beweises können sich Teilziele ergeben; typische Beispiele für Aufspaltungen in Teilziele sind etwa die Aufteilung einer Äquivalenzaussage in zwei Implikationen, die Aufteilung einer Mengengleichheit in zwei Inklusionen, die Aufteilung eines Induktionsbeweises in Induktionsanfang und Induktionsschritt etc. Solchen Zwischenziele geben

[2]So würde als Inhalt des Satzes "Es folgt folgt $(A \rightarrow B)$ etwa "folgt $(A \rightarrow B)$ identifiziert, was dann zu dem vermutlich intendierten Inhalt "$(A \rightarrow B)$" weiterverarbeitet würde.

ihrerseits zu Beweisen Anlaß, die wiederum Teilziele enthalten können. So entsteht eine Hierarchie von Zielen, der sich im Verlauf des Beweises ändert: Zu jedem Zeitpunkt gibt es ein "aktuelles" Ziel, das im weiteren etwa zu einem Teil-Teilziel führt, welches dann den Status des aktuellen Zieles einnimmt, oder erreicht wird und somit als Beweisziel für den weiteren Verlauf des Argumentes verschwindet.

Für das mathematische Beweisen, wie auch für das Verstehen mathematischer Beweise, ist es eine grundlegende Fähigkeit, das Ziel bzw. diese Zielhierarchie im Blick zu behalten, also insbesondere: Zu wissen, was das aktuelle Ziel ist und welche Funktion es im Gesamtbeweis einnimmt. Gerade bei AnfängerInnen ist es daher sinnvoll, Beweise so darzustellen, dass Ziele und Teilziele explizit genannt werden und deutlich wird, an welcher Stelle sie erreicht sind – und andererseits auch solche Darstellungen zu verlangen. Die Sprache von Diproche enthält daher explizite Marker für die Deklaration von Zielen sowie den Anfang und das Ende von Beweisen (s.u.). Die von Diproche unterstützte Darstellungsform folgt also der Maxime *"Sage was Du tun willst, tue es, und dann sag, dass Du es getan hast"*.[3].

Die Verschiebung des aktuellen Beweiszieles im Verlauf Beweistexten ist nicht allein von expliziten Zieldeklarationen abhängig, sondern kann sich auch dadurch ergeben, dass Annahmen gemacht werden oder nicht explizit als solche genannte Teilziele erreicht werden. So könnte etwa die Annahme "Es sei n gerade." das aktuelle Beweisziel von "Wenn n gerade ist, dann ist auch n^2 gerade" verschieben zu "n^2 ist gerade" oder die Herleitung der Aussage a könnte das aktuelle Beweisziel von $a \wedge b$ ändern zu b.

Die Zielverfolgung, in Diproche realisiert durch das Goaltracing-Modul, hat die Aufgabe, die Entwicklung des aktuellen Beweiszieles zu verfolgen und zu melden, ob an der Stelle, an der ein Beweis durch einen Beweisendmarker wie "qed" geschlossen wird, das zugehörige Ziel erreicht wurde. Falls nicht, wird die Position ("Zeilennummer") des entsprechenden Beweisendmarkers zusammen mit dem offenen Beweisziel zurückgemeldet.

Dabei verfährt die Zielverfolgung in drei Schritten: Zunächst wird für einen (jeden) Beweisendmarker M das Ziel G des durch M geschlossenen Beweises ermittelt. Ist nun G das zu einem Beweisendmarker M gehörige Beweisziel, so wird zunächst geprüft, ob G im Text die letzte vor M stehende Behauptung ist, ob der Beweis also zumindest der Textform nach auf das richtige Ziel hinausläuft. Im dritten, logischen, Schritt wird schließlich geprüft, ob G ausschließlich unter den Annahmen bewiesen wurde, die laut Behauptung zur Verfügung stehen. An dieser Stelle würde es also bemerkt, wenn beim Beweis von G zusätzliche Annahmen eingeführt worden wären, womit die Allgemeinheit des Beweises (potenziell)

[3]Es war mir leider nicht möglich, für diese ausgezeichnete Maxime eine Quelle ausfindig zu machen. Wir haben sie im Verlauf unseres Mathematikstudiums in Bonn immer wieder zu hören bekommen. Vermutlich ist es eine Anlehnung an die Maxime "Tell the audience what you're going to say, say it; then tell them what you've said." von Dale Carnegie.

beschränkt wäre.

Nun gibt es zahlreiche Arten, wie ein Beweisziel erreicht werden kann ohne dass es explizit gennant würde. Im Beweise einer Implikation $\phi \rightarrow \psi$ könnte man etwa ϕ annehmen und ψ beweisen; oder man könnte $\neg\psi$ annehmen und daraus $\neg\phi$ herleiten etc. In solchen Fällen noch auf einer expliziten Wiederholung des ursprünglichen Zieles zu bestehen, wäre zwar im Sinne der aus didaktischen Gründen angestrebten Explikation der Zielstruktur, andererseits aber auch für die übliche mathematische Darstellungspraxis ungewohnt "sperrig". Aus diesem Grund sind Beweisstrategien wie etwa der Implikationsbeweis durch Annehmen und Folgern oder die Kontraposition in die Zielverfolgung integriert, so dass auch in solchen Fällen das Beweisziel als erreicht gilt. Das hat allerdings zur Folge, dass es i.A. nicht eine, sondern mehrere Aussagen gibt, mit denen das Beweisziel als erreicht betrachet werden kann. Um dem zu entsprechen arbeitet Diproche mit einer Liste von aktuellen Beweiszielen statt mit einer einzelnen Aussage. Die Zielhierarchie wird also in jedem Punkt des Beweises repräsentiert durch eine Liste von Listen von Aussagen.

Der Goaltracer verfährt bei der Aktualisierung der Zielhierarchie nach folgenden Regeln:

- Durch eine Zieldeklaration ("Wir zeigen, dass A") wird B der Beweiszielliste als neues erstes Element hinzugefügt.

- Durch eine Fallannahme A innerhalb einer Fallunterscheidung, vor deren Beginn das aktuelle Beweisziel B war, wird $(A \rightarrow B)$ der Beweiszielliste als erstes Element hinzugefügt.

- Durch einen Beweisendmarker (qed) oder einen Marker für das Ende einer Fallunterscheidung ("In jedem Fall gilt, dass...") wird das erste Element der GoalListe gelöscht.

- Durch die Ankündigung einer Beweisrichtung (=>, <=) wird, sofern das erste Element der GoalListe derzeit eine Biimplikation $(A \leftrightarrow B)$ ist, die entsprechende Richtung (also $(A \rightarrow B)$ bzw. $(B \rightarrow A)$) der GoalListe als neues erstes Element hinzugefügt.

- Durch die Ankündigung einer Teilmengrelation ($\subseteq$, $\supseteq$) als isolierter Annotation wird, sofern das erste Element der GoalListe derzeit eine Mengengleichheit $(A = B)$ ist, die entsprechende Richtung (also $(A \subseteq B)$ bzw. $(B \supseteq A)$) der GoalListe als neues erstes Element hinzugefügt.

- Ist das aktuelle Beweisziel eine Implikation $(A \rightarrow B)$ und wird A angenommen, so wird B der GoalListe als neues erstes Element hinzugefügt.

- Ist das aktuelle Beweisziel eine Teilmengenbeziehung $A \subseteq B$ und wird $x \in A$ für eine Variable x angenommen, so wird $x \in B$ zum neuen aktuellen Beweisziel.

- Ist das aktuelle Beweisziel eine Konjunktion $(A\&B)$ und wird A bzw. B behauptet, so wird das jeweils andere Konjunktionsglied der GoalListe als neues erstes Element hinzugefügt.

- Durch die Ankündigung eines Widerspruchsbeweises ("Durch Widerspruch") wird "falsum" der GoalListe als neues erstes Element hinzugefügt.

- Ist das aktuelle Beweisziel von der Form $\forall x_1, ..., x_n(\phi(x_1, ..., x_n) \rightarrow \psi(x_1, ..., x_n))$ und wird eine Deklaration mit inhaltlicher Annahme der Form "Es seien $a_1, ..., a_n$ so, dass $\phi(a_1, ..., a_n)$" gemacht, so wird $\psi(a_1, ..., a_n)$ zum neuen aktuellen Beweisziel.

- Ist das aktuelle Beweisziel von der Form $\forall x\phi$, so wird durch eine einen Induktionsanfang markierende Annotation ("Induktionsverankerung" bzw. "Induktionsanfang") die Aussage $\phi[\frac{1}{x}]$ (bei der alle freien Vorkommen von x in ϕ durch 1 ersetzt wurden) zum neuen aktuellen Beweisziel. Bei Aussagen der Form $\forall x(x \geq n \rightarrow \phi)$ bzw. $\forall x(x > n \rightarrow \phi)$ mit festem n wird entsprechend zu $\phi[\frac{n}{x}]$ bzw. $\phi[\frac{n+1}{x}]$ aktualisiert.

- Ist das aktuelle Beweisziel von der Form $\forall x\phi$, so wird durch eine einen Induktionsschritt markierende Annotation ("Induktionsschritt" bzw. "Induktionsschluss") die Aussage $\forall x(\phi \rightarrow \phi[\frac{x+1}{x}])$ zum neuen aktuellen Beweisziel.

- Wird durch eine der obigen Regeln ein aktuelles Beweisziel erzeugt und ist dieses gleich dem derzeit vorliegenden aktuellen Beweisziel, so ändert sich das aktuelle Beweisziel nicht. So ist es z.B. wirkungslos, nach der Annotation $\Rightarrow$ noch einmal die entsprechende Implikation explizit als Beweisziel zu deklarieren.

Um einen Eindruck von der Arbeitsweise der Zielverfolgung zu geben, geben wir hier einen Diproche-Beispieltext mit den jeweils zugehörigen Ziellisten.[4] Dabei wird jeder Beweiszeile (links) die nach dieser Zeile aktuelle Zielhierarchie (rechts) zugeordnet. Das aktuelle Beweisziel steht in der Zielhierarchie ganz links. Der Übersichtlichkeit halber wurden einige Zeilen fortgelassen, in denen die Zielhierarchie sich nicht ändert.

[4]Ein solches Beispiel wurde bereits in [40] behandelt.

Beweis	Zielhierarchie
Es sei n eine ganze Zahl.	[]
Wir zeigen: Dann ist n gerade gdw n^2 gerade ist.	$[[2\|n \leftrightarrow 2\|n^2]]$
	$[[2\|n \leftrightarrow 2\|n^2]]$
	$[[2\|n \leftrightarrow 2\|n^2]]$
Beweis:	$[[2\|n \leftrightarrow 2\|n^2]]$
	$[[2\|n \leftrightarrow 2\|n^2]]$
$\Rightarrow$	$[[2\|n \rightarrow 2\|n^2],[2\|n \leftrightarrow 2\|n^2]]$
Es sei n gerade.	$[[2\|n \rightarrow 2\|n^2,\ 2\|n^2],[2\|n \leftrightarrow 2\|n^2]]$
...	...
Also ist n^2 gerade.	$[[2\|n \rightarrow 2\|n^2,\ 2\|n^2],[2\|n \leftrightarrow 2\|n^2]]$
(an dieser Stelle wird	das aktuelle Beweisziel erreicht)
qed.	$[[2\|n \leftrightarrow 2\|n^2]]$
$\Leftarrow$	$[[2\|n^2 \rightarrow 2\|n],[2\|n \leftrightarrow 2\|n^2]]$
Es sei n^2 gerade.	$[[2\|n^2 \rightarrow 2\|n,\ 2\|n],[2\|n \leftrightarrow 2\|n^2]]$
...	...
Also ist n gerade.	$[[2\|n^2 \rightarrow 2\|n,\ 2\|n],[2\|n \leftrightarrow 2\|n^2]]$
(an dieser Stelle wird	das aktuelle Beweisziel erreicht)
qed.	$[[2\|n \leftrightarrow 2\|n^2]]$
Also ist n gerade gdw n^2 gerade ist.	$[[2\|n \leftrightarrow 2\|n^2]]$
(an dieser Stelle wird	das aktuelle Beweisziel erreicht)
qed.	[]

5.1.3 Type-Checking

Häufige Fehler bei Beweistexten im Anfängerbereich sind die Verwendung von nicht zuvor eingeführten Variablen sowie die Verwendung von Variablen in einer Weise, die zur Bildung undefinierter Ausdrücke führt, etwa die Verknüpfung von Mengen durch aussagenlogische Junktoren oder arithmetische Operatoren. Die Typenprüfung hat die Funktion, solche Fehler zu erkennen und zu melden.

In der Sprache von Diproche existiert zu diesem Zweck eine gesonderte Kategorie von Sätzen, die "Deklarationen". Mit einer Deklaration werden eine oder mehrere Variablen eingeführt und es werden ihnen "Typen" zugewiesen, also Bereiche, aus denen die durch sie bezeichnete Objekte stammen. Beispiele für Deklarationen in Diproche sind "Es sei x eine ganze Zahl.", "Es seien a, b und c Punkte." oder "Es seien x, y reelle Zahlen und n eine natuerliche Zahl.". Ebenso ist es möglich, eine Deklaration mit einer weiteren Annahme über das eingeführte Objekt zu verbinden, etwa durch Formulierungen wie "Es sei k eine ganze Zahl mit $n = 2 * k$." (siehe dazu auch den Abschnitt über die Sprache von Diproche weiter unten). Dabei sind die Zugänglichkeitsregeln für Deklarationen die gleichen wie für Annahmen, siehe auch dazu den Abschnitt zur Sprache von Diproche weiter

unten.

Die Typenprüfung hat die Aufgabe, jede Verwendung eine Variablen darauf zu prüfen, ob sie an der verwendeten Stelle deklariert ist und falls ja, ob ihre Verwendung im Sinne der Deklaration möglich ist. Dazu sind die in Diproche verwendeten Operationen und Prädikate intern mit den Typen der Objekte assoziiert, die in sie eingesetzt werden können. Beispielsweise kann das Prädikat "gerade" sinnvoll nur von einer natürlichen oder ganzen Zahl ausgesagt werden, während seine Anwendung etwa auf einen Punkt oder eine Aussage zur Meldung eines Typenfehlers führt. Bei Verwendung komplexer Terme wird dabei der Typ des zusammengesetzten Objektes aus den Typen der auftretenden Variablen sowie den verwendeten Operationen automatisch ermittelt. Überdies werden doppelte Deklarationen erkannt und gemeldet.

Wir betrachten aus hier einen Beispieltext, um die Funktion der Typenprüfung zu verdeutlichen:

Beweis	Verfügbare Deklarationen	Status Type-Checking
Beweis:	[]	OK
Es sei n eine ganze Zahl.	[[n,int]]	OK
	[[n,int]]	OK
Es sei n gerade	[[n,int]]	OK
Weiter sei $n = m$	[[n,int]]	Fehler[68]
Es seien m, n reelle Zahlen	[[n,int],[m,int],[n,real]]	Fehler[69]
Dann ist $m^2 - 1 = (m - 1) * (m + 1)$.	[[n,int],[m,int],[n,real]]	OK
Ferner ist m parallel zu n.	[[n,int],[m,int],[n,real]]	Fehler[70]
	[[n,int]]	OK[71]

5.1.4 Logische Prüfung

Die logische Prüfung ist das Herzstück von Diproche. Die Aufgabe dieses Moduls ist es, festzustellen, ob alle Schritte in einem deduktiven Argument logisch folgerichtig sind. Wird eine Behauptung aufgestellt, so muss diese auf der Basis der am Ort ihres Vorkommens im Text verfügbaren Annahmen sowie der bisher aus diesen Annahmen gefolgerten Aussagen mithilfe einer logischen Schlussregel folgen. Das gleiche gilt, wenn eine Behauptung zwar nicht explizit aufgestellt, aber im Rahmen eines anderen sprachlichen Ausdrucks implizit angenommen, "präsupponiert", wird. So führt etwa die Deklaration mit inhaltlicher Annahme

[68] m ist nicht deklariert.

[69] n doppelt deklariert

[70] "parallel" erfordert Geraden als Objekte

[71] Durch den Absatz fallen die Deklarationen aus dem letzten Absatz fort, nur die Deklaration von n bleibt; vgl. dazu die Zugänglichkeitsregeln weiter unten. Auf diese Weise ist es möglich, derselben Variable in logisch voneinander unabhängigen Teilen eines Beweistextes verschiedene Typen zuzuweisen.

"Es sei k eine ganze Zahl mit $n = 2 \cdot k$." das k als Bezeichnung für eine ganze Zahl mit $n = 2 \cdot k$ ein, was voraussetzt, dass eine solche Zahl existiert; diese Voraussetzung muss also an dieser Stelle verifiziert werden.

Um die einzelnen Beweisschritte möglichst unabhängig voneinander beurteilen zu können und dem/der BenutzerIn die Möglichkeit zu geben, zielgenau einzelne Schritte zu ergänzen bzw. zu verbessern, erfolgt die Rückmeldung zu jedem Schritt "relativ" zu den früheren Schritten, d.h. unter der Annahme, dass alle zuvor aufgestellten Behauptungen korrekt sind. So würde etwa in dem Text

Es gilt 2=5. Also gilt 2=5.

nur die erste Behauptung als fehlerhaft gemeldet, wohingegen die zweite sich aus der ersten ergibt. Dagegen würden bei Eingabe von

Es gilt 2=5. Es gilt 2=6.

beide Behauptungen als nicht verifizierbar gemeldet, weil die zweite mit keiner dem ATP derzeit zu Verfügung stehenden Regel aus der ersten zu folgern ist. (Um diese Art von Korrektur sinnvoll zu ermöglichen, ist eine zu starke Version von "ex falso quodlibet" im ATP zu vermeiden.)

Die logische Prüfung zerfällt in zwei Teile, nämlich einerseits die Erzeugung formaler Beweisziele auf der Basis des Eingabetextes, der ihrerseits die Ermittlung der Zugänglichkeitsrelation zugrundeliegt, und andererseits die Prüfung dieser Ziele durch den automatischen Theorembeweiser (ATP). Diese Komponenten betrachten wir nun separat.

Die Zugänglichkeitsrelation

Für Beweistexte ist es wesentlich, dass eine einmal getroffene Annahme nicht unbedingt für den gesamten Rest des Textes in Geltung bleiben muss. So wird etwa bei einer Fallunterscheidung eine Fallannahme mit der Behandlung des entsprechenden Falles abgeschlossen und beim Beweis einer Äquivalenzaussage $A \leftrightarrow B$ durch zwei Implikationen $A \rightarrow B$ und $B \rightarrow A$ ist die zum Beweis von $A \rightarrow B$ eingangs getroffene Annahme A bei Beginn des Beweises von $B \rightarrow A$ nicht mehr in Geltung. Da es in Beweistexten üblicherweise keine expliziten Formulierungen für das Zurückziehen von Annahmen gibt,[5] ist die Markierung bzw. Ermittlung des Geltungsbereiches von Annahmen ein Problem, dem sich Systeme zur automatischen Prüfung natürlichsprachlicher Beweise stets stellen

[5] Vgl. etwa Cramer [53], S. 255: "The reason for this unnatural aspect in the Naproche CNL is that the language of mathematics does not have clear signs for marking the retraction of an assumption."

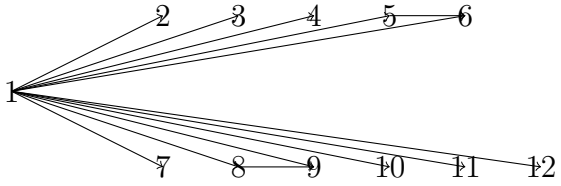
müssen (siehe dazu etwa in Bezug auf das Naproche-Projekt die Besprechung in Cramer [53], S. 255).

In der Diproche-Sprache sind die Geltungsbereiche von Annahmen und Deklarationen durch die Absatzstruktur reguliert: Eine Annahme gilt generell bis zum Ende des Absatzes, in dem sie eingeführt wird; folgt dieser Absatz unmittelbar auf einen Beweisanfangsmarker (wie "Beweis:"), so gilt sie außerdem bis zum Ende des dort begonnen Beweises. Andernfalls endet ihr Geltungsbereich mit dem Ende des Absatzes, in dem sie steht (siehe dazu den Abschnitt über die Sprache von Diproche unten).

In Diproche wird die so definierte Zugänglichkeitsrelation durch ein gesondertes Modul ermittelt, das sie in Form eines gerichteten Graphen auf der Menge der Zeilenindizes des Beweises repräsentiert. So wäre etwa der zu dem (der Übersichtlichkeithalber unvollständigen) Beweistext

(1) Es sei n eine ganze Zahl.
(2) Wir zeigen: Dann ist $n^2 + n$ gerade.
(3) Beweis:
(4)
(5) Angenommen, n ist gerade.
(6) Dann ist $n^2 + n$ ebenfalls gerade.
(7)
(8) Nehmen wir nun an, n ist ungerade.
(9) Dann ist $n^2 + n$ wiederum gerade.
(10)
(11) Also ist $n^2 + n$ gerade.
(12) qed.

gehörige Digraph der folgende:



Erzeugung formaler Beweiszielen

Die Erzeugung formaler Beweisziele hat die Aufgabe, zu jeder Textstelle in einem Beweistext den an dieser Stelle behaupteten Inferenzschritt zu erzeugen. Dazu werden einerseits die verfügbaren Voraussetzungen ermittelt, andererseits die aufgestellte Behauptung. Aus Gründen, die wir weiter unten erläutern werden,

werden Voraussetzungen unterschieden nach Annahmen, Behauptungen, bisher erreichten Folgerungschritten und Deklarationen.

Für die Ermittlung der verfügbaren Annahmen wesentlich ist die Zugänglichkeitsrelation, die wir im letzten Abschnitt erläutert haben. Eine Annahme oder Deklaration ϕ gilt als für eine Behauptung ψ verfügbar, wenn die Zeile, in der ϕ steht, im Zugänglichkeitsgraphen mit der Zeile verknüpft ist, in der ψ steht. Eine Behauptung ϕ gilt für eine Behauptung ψ als verfügbar, wenn ϕ im Text vor ψ kommt und alle Annahmen, die für ϕ verfügbar sind, auch für ψ verfügbar sind. Zusätzlich zu den verfügbaren Annahmen, Deklarationen und Behauptungen stehen dem Beweiser auch die bisherigen Folgerungsschritte als Implikationen zur Verfügung; wurde etwa zuvor aus der Annahme ϕ die Behauptung ψ gefolgert, so steht $\phi \to \psi$ als "früherer Schritt" zur Verfügung. Die Implikation kann dabei auch iterativ geschachtelt werden: Wird zuerst ϕ_1 angenommen, dann ϕ_2 und dann ϕ_3 gefolgert, so steht von nun an $\phi_1 \to (\phi_2 \to \phi_3)$ zur Verfügung etc.

Die Behandlung eines Beweisschrittes durch die Beweiszielerzeugung hängt dann in folgender Weise von der Art des Schrittes ab:

- Bei Annotationen, Annahmen oder einfachen Deklarationen wird kein Beweisziel erzeugt.
- Bei einer einfachen Behauptung ϕ werden die verfügbaren Annahmen A, Behauptungen B, Deklarationen D und früheren Schritte S ermittelt. Das Beweisziel ist dann das Tupel $((A, B, S, D), \phi)$.
- Bei einer begründeten Behauptung ("Wegen ϕ folgt ψ") stehen lediglich die in der Begründung explizit genannten Aussagen zur Verfügung sowie die Annahme oder Behauptung im der begründeten Behauptung vorausgehenden Satz. (Zusätzlich wird an dieser Stelle geprüft, ob alle Teile der Begründung an der nämlichen Stelle im Text verfügbare Aussagen sind.)
- Deklarationen mit inhaltlicher Annahme ("Es sei k eine ganze Zahl mit $n = 2 \cdot k$") werden behandelt wie Behauptungen, wobei die Rolle der Behauptung von der Aussage eingenommen wird, dass Objekte mit den genannten Eigenschaften existieren.
- Definitionen ("Es sei p der Mittelpunkt der Strecke s") werden behandelt wie Deklarationen mit inhaltlicher Annahme.

Der Diproche-ATP

Der letzte Schritt der logischen Prüfung besteht in dem Versuch, das im zweiten Schritt ermittelte Beweisziel logisch zu verifizieren. Hierzu dient

ein automatischer Theorem-Beweiser ("automated theorem prover", ATP). Im Gegensatz zu Systemen wie Naproche, die professionelle ATPs mit möglichst großer Beweisstärke einsetzen ist, der Diproche-ATP so konstruiert, dass er die im Rahmen einer Übungsaufgabe im Anfängerbereich akzeptablen Schlussformen möglichst vollständig abbildet. Der Diproche-ATP wurde dadurch entwickelt, dass die in typischen Übungsaufgaben erforderlichen Schlussregeln sukzessive hinzugefügt wurden bis für den angezielten Typ von Aufgabe eine "Sättigung" erreicht war, also die meisten weiteren Aufgaben sich in der intendierten Weise lösen ließen, ohne neue Schlussregeln hinzuzufügen. Typische Regeln des Diproche-ATP sind etwa der modus ponens ("aus A und $A \to B$ läßt sich B folgern") oder die Fallunterscheidungsregel ("aus $A \to B$ und $\neg A \to B$ schließe B").

Wie schon oben erwähnt – und wie die Erfahrung mit der Konstruktion des ATPs gezeigt hat – hängen die akzeptablen Schlussformen vom Kontext ab, insbesondere dem Fortschritt der Vorlesung. Aus diesem Grund sind die ATP-Regeln mit Kennungen versehen; Aufgaben in Diproche sind dann mit sogenannten "Schwierigkeitsgraden" assoziiert, also Mengen von ATP-Regeln, die für die betreffende Aussage verwendet werden dürfen. Arbeitet der ATP im Kontext der Bearbeitung einer solchen Aufgabe, verwendet er entsprechend nur Regeln aus dem entsprechenden "Schwierigkeitsgrad".

Überdies gibt es für jedes inhaltliche Teilgebiet spezifische Schlussweisen. So kann etwa im Rahmen einer zahlentheoretischen Aufgabe eine Fallunterscheidung nach der Parität einer ganzen Zahl durchgeführt werden, in der Geometrie können Kongruenzsätze benutzt werden etc. Zu jedem von Diproche abgedeckten Teilgebiet gibt es daher Spezial-ATPs, die z.T. noch auf weitere Teilmodule zurückgreifen. Derzeit besteht der ATP von Diproche aus 10 Submodulen: dem allgemein logischen ATP, der aussagen- und quantorenlogische Regeln enthält und bei allen Aufgaben eingesetzt wird; dann Spezial-ATPs für boolesche Mengenlehre, axiomatische Geometrie, Gruppentheorie sowie den Themenbereich "Funktionen und Relationen"; und schließlich dem arithmetischen ATP, der seinerseits auf Sondermodule für Teilbarkeitsregeln, Kongruenzregeln, natürliche Zahlen und Termumformungen zurückgreift.[6] Der "empirische" Anspruch an den ATP, die im Anfängerbereich akzeptablen Schlussweisen möglichst vollständig zu erfassen hat zur Folge, dass die ATPs über einen recht umfangreichen Regelvorrat verfügen. So verfügt der "allgemein logische" ATP allein über rund 50 Regeln für den Umgang mit Quantoren und rund 190 aussagenlogische Regeln.

Wird der ATP mit einem Beweisziel konfrontiert, versucht er, ob dieses mit einer der für die fragliche Aufgabe zugelassenen Beweisregeln erreichbar ist. Falls ja, gilt der entsprechende Schritt als verifiziert; andernfalls wird dem/der

[6]Für eine detailliertere Besprechung des Geometrie-ATPs und des Arithmetik-ATPs siehe die Abschnitte über die entsprechenden "Spielwiesen" weiter unten.

BenutzerIn die betreffende Textstelle als nicht verifizierbar gemeldet.

5.2 Hilfestellung

Die Rückmeldungen der Zielverfolgung, der Typenprüfung sowie der logischen Prüfung haben den Charakter einer Korrektur, die Mängel an einem gegebenen Beweistext aufzeigt und so Anlass zur Verbesserung des Textes – oder auch des zugrundeliegenden Gedankenganges – geben kann. Diese Art der Hilfestellung kann offenbar nur dann zum Einsatz kommen, wenn überhaupt ein Beweistext vorliegt, der von Diproche verarbeitet werden kann. Treten Schwierigkeiten schon in der Beweissuche selbst auf, hat die Prüfung keinen Gegenstand und kann somit auch keine Hilfestellung geben. Wer gar keine Idee hat, wie eine Aufgabe anzugehen ist (und auch keine Idee, wie man zu einer Idee kommen könnte), hat kaum eine andere Möglichkeit, als aufzugeben, was nicht im Sinn der Förderung der eingehenden Beschäftigung mit Beweisaufgaben ist. So haben menschliche TutorInnen denn neben der Korrektur- und ggf. Vorbildfunktion oft auch die Aufgabe, in Präsenzübungen während des Beweisens Hilfestellungen und Tipps zu geben. Aus dem gleichen Grund ist es sinnvoll, die Möglichkeit der Hilfestellung beim Beweissuchen auch in Diproche vorzusehen. Tatsächlich verfügen auch eine Reihe der oben besprochenen Systeme zur Förderung des mathematischen Beweisens über die eine oder andere Form der Hilfestellung.

Im Rahmen von Diproche sind derzeit drei Arten von Hilfestellung bei der Beweissuche vorgesehen und implementiert, von denen allerdings zum Zeitpunkt der Niederschrift dieser Arbeit nur die erste bereits über das User-Interface abrufbar ist: Erstens direkte aufgabenspezifische Hinweise von Seiten der Lehrkraft, zweitens allgemeine strategische Hinweise, drittens die Generierung potenziell hilfreicher Zwischenschritte.

Direkte Hinweise

Die erste Art Hinweis besteht in Lösungshinweisen, die die Lehrkraft beim Erstellen und Einpflegen der Aufgabe eingeben kann und die dem/der BenutzerIn auf Verlangen angezeigt werden. Hier können also Lösungstipps gegeben werden, wie man sie typischerweise im Lösungsteil von Lehrbüchern findet. Ein Vorteil dieser Art Hinweis ist die weitgehende Freiheit bezüglich Form und Inhalt. Die offensichtlichen Nachteile sind zum einen, dass solche Hinweise nur für explizit eingepflegte Aufgaben verfügbar sind (und etwa nicht für automatisch generierte oder selbst gewählte) und weiter darin, dass sie i.A. nur einen vorgesehenen Lösungsweg nahelegen, aber auf eine bereits begonnene (und evtl. vielversprechende) Lösung kaum (allenfalls im Rahmen einer Antizipation

verschiedener Lösungswege, für die dann jeweils separate Hinweise gegeben werden) eingehen können.

Allgemeine strategische Hinweise

Die zweite Art von Hinweis sind die "allgemein strategischen" Hinweise. Teil des grundlegenden Handwerkszeugs des mathematischen Beweisens ist eine Reihe von Routinen und Ansätzen, die in mathematischen Beweisen immer wieder auftreten. Dazu gehört es etwa, beim Beweis einer Implikation $\phi \rightarrow \psi$ die Annahme ϕ zu machen und dann aus ψ hinzuarbeiten; beim Beweis einer allquantifizierten Aussage $\forall x \phi$ ein beliebiges Objekt aus dem Bereich des Quantors einzuführen und zu benennen; den Beweis einer Äquivalenzaussage in den Beweis zweier Implikationen und den Beweis einer Mengengleichheit in den Beweis zweier Teilmengenbeziehungen aufzuteilen; bei einer Disjunktion unter den Voraussetzungen – falls nötig – eine Fallunterscheidung einzuführen, usw. Solche grundlegenden Strategien sind, wie die Lehrerfahrung zeigt, zu Beginn des Studiums keineswegs selbstverständlich und werden mitunter auch nur mit einer gewissen Zurückhaltung angewendet oder wieder "vergessen". Daher ist es häufig zielführend, in schwierigen Lösungsphasen auf die fragliche Strategie noch einmal hinzuweisen. Auf diese Weise besteht weiter die Aussicht, dass sich die Verbindung zwischen Aufgabentyp und Lösungsansätzen allmählich einprägt – jedenfalls sicher eher, als wenn der Lösungsversuch aufgrund von Ideenlosigkeit abgebrochen wird. In der Tat zeigt die Erfahrung mit der expliziten Vermittlung von Beweisstrategien etwa in Seminaren zum mathematischen Problemlösen, dass diese dazu beitragen, einen Blick für die strategisch relevante Struktur der Aufgabe zu entwickeln und sich weniger von oberflächlichen Eigenschaften ablenken zu lassen.[7]

Diese Art der Hilfestellung ist automatisierbar. Hierzu wird zunächst von der Zielverfolgung das aktuelle Beweisziel ermittelt. Anschließend wird das Tippgeber-Modul aktiviert, das eine Reihe von Regeln der oben genannten Art enthält, wie etwa "Wenn das Beweisziel eine Implikation ist, nimm die Prämisse an" oder "Wenn sich unter den Voraussetzungen eine Disjunktion befindet und keine andere Regel anwendbar ist, führe eine Fallunterscheidung durch". Dabei werden Hinweise, die sich auf die Struktur des Beweiszieles ergeben, priorisiert. Ist aufgrund des Beweisziels kein Hinweis mehr zu finden, folgen Hinweise, wie die

[7] Siehe Schoenfeld [178], Kapitel 8, etwa S. 243: "(...), the mathematical experts appear to base their perceptions of problem relatedness upon principles of the discipline or an archetypal method of solution, which have been called the problems' *deep structure*. Novices tend to classify problems by their *surface structure*, focusing on the words or subjects that are prominent in the problem statements.". Im weiteren Verlauf des Kapitels wird aufgezeigt, wie die gezielte Förderung mathematischer Problemlösekompetenzen auch die Wahrnehmung von Aufgaben und ihre Einteilung in Kategorien beeinflussen kann.

zur Verfügung stehenden Voraussetzungen zu nutzen sind; etwa die Teile einer Konjunktion abzuspalten, aus A und $(A \to B)$ noch B zu folgern o.ä.; ist auch in dieser Hinsicht kein Hinweis mehr verfügbar, gibt es noch allgemeine Hinweise wie "versuche es durch Widerspruch". Diese Hinweise kann der/die BenutzerIn sich auf Anfrage anzeigen lassen. Sie sind allerdings derzeit nur für den Bereich der Aussagenlogik verfügbar.

Beispielsweise generiert das System für folgenden Text:

> Wir zeigen: Es gilt $(a\&b) \leftrightarrow (b\&a)$.
> Beweis:

die folgende Ausgabe:

> Identifiziertes Beweisziel: Leite aus den Voraussetzungen her, dass $(a\&b) \leftrightarrow (b\&a)$.
> Zeige zunächst $((a\&b) \to (b\&a))$.

Ergänzt man am Ende des Textes noch einen Rechtspfeil $\Rightarrow$, ändert sich dieser Hinweis in:

> (1) Versuche, $(a\&b)$ anzunehmen und $(b\&a)$ zu zeigen.
> (2) Versuche es durch Kontraposition: Nimm $\sim (b\&a)$ an und versuche, $\sim (a\&b)$ zu zeigen.

Ergänzung von Zwischenschritten

Eine dritte Möglichkeit, wie Studierende bei der Beweissuche unterstützt werden können, ist der Vorschlag geeigneter Zwischenziele und -schritte. Auch diese Art von Hinweis läßt sich in begrenztem Umfang automatisieren: So kann zunächst, wie bei der Generierung strategischer Hinweise, die Zielverfolgung eingesetzt werden, um das aktuelle Beweisziel zu identifizieren. Anschließend versucht der ATP, den Beweis zu vervollständigen. Gelingt dies, wird ein geeigneter Zwischenschritt – also eine im Verlauf des gefundenen Beweises aufgetretende Aussage – als Hilfestellung ausgegeben. In Diproche sucht der ATP derzeit nach Beweisen mit einer maximalen Schrittzahl von 10 und gibt im Erfolgsfall einen Schritt aus der Mitte des Beweises aus. Damit ist indes dem Kriterium eines "geeigneten" Zwischenschrittes nur sehr ungenügend Rechnung getragen. Hier ergibt sich ein weites Feld von Verbesserungsmöglichkeiten, siehe dazu die Erörterungen im letzten Teil dieser Arbeit.

5.3 Diagnose von Fehlschlüssen

Bearbeitungen von Übungsaufgaben enthalten weit mehr Information, als in einem bloßen "richtig" oder "falsch" enthalten ist. Im Falle von Rechenfehlern gibt es eine umfangreiche Literatur dazu, welche konzeptionellen Schwierigkeiten durch welche Arten von Fehlern nahegelegt sind, siehe etwa Brandle [22] oder Wehrmann [203]. Fehler sind somit wertvolle diagnostische Information, die eine erfahrene Lehrkraft erkennen und, etwa zur gezielten Förderung oder zur Beseitigung von Fehlkonzeptionen, nutzen kann.[8]

Ähnliches wie für die Rechenfertigkeit läßt sich auch für das Beweisen sagen. Ist man in der Lage, zu erkennen, was die Absicht hinter einem fehlerhaften Beweis war, so kann auch eine qualifizierte Vermutung bezüglich der (Fehl)vorstellungen angestellt werden, die zu einem Fehler geführt haben; das wiederum ist die Basis für Rückmeldungen, die Schwierigkeiten gezielt adressieren. Folgert jemand etwa aus $\phi \to \psi$ und $\sim \phi$, dass $\neg\psi$ gelten muss, so liegt vermutlich eine falsche Auffassung der Implikation zugrunde, die dann auf verschiedene Arten (etwa durch Erinnerung an die Definition der Implikation, alltägliche oder anderweitig anschauliche Beispiele) angegangen werden kann.

Auch diese Art von Fehlschlussdiagnose ist zumindest teilweise automatisierbar. In Diproche übernimmt diese Funktion der "Anti-ATP". Der Anti-ATP funktioniert ähnlich wie der ATP: Er versucht, auf der Basis gegebener Voraussetzungen und einer Behauptung in einem Regelvorrat eine Regel zu finden, die es erlaubt, die Behauptung aus den Voraussetzungen abzuleiten. Im Unterschied zum ATP besteht der Regelvorrat beim Anti-ATP aber nicht aus korrekten Schlussregeln, sondern aus gängigen Fehlschlüssen. In der Tradition automatischer Systeme zur Fehlerdiagnose basiert der Anti-ATP damit auf einer "mal-rule library", siehe etwa Beller und Hoppe [21], Payne und Squibb [167] oder Sleeman [183].[9]

Die Wirksamkeit des Einsatzes von "mal-rule libraries" in einer gewissen Domäne hängt stark davon ab, ob einerseits Fehler typischerweise systematischen Charakter haben, also auf Fehlvorstellungen beruhen und entsprechend wiederholt auftreten werden, andererseits davon, ob sich unter den systematischen Fehlern

[8] Vgl. etwa Payne und Squibb [167], S. 1: "Theoreticians have long recognized that important insights into the nature of cognitive skills and its acquisition can be gained by examining errors. This approach also has potential practical implications for instruction, because achieving a "cognitive diagnosis" of a learner's errors may be an important step towards meaningful individualized tutoring." Payne und Squibb untersuchen Fehler in elementaren algebraischen Umformungen; die Bemerkung scheint aber für "höhere" Aktivitäten wie das Beweisen ebenso zuzutreffen.

[9] Die genannten Arbeiten beziehen sich durchweg auf die Diagnose von Rechen- oder Umformungsfehlern. Ein entsprechender Ansatz zur Diagnose von Schlussfehlern in Argumentationsketten ist uns nicht bekannt.

solche finden, die bei einer nennenswerten Anzahl von Studierenden auftreten bzw. ob eine überschaubare und damit realistischerweise implementierbare Menge von Fehlerregeln existiert, die einen nennenswerten Anteil der faktisch auftretenden Fehler erklärt und schließlich davon, ob diesen Fehlern jeweils eindeutig eine Fehlvorstellung zugeordnet werden kann, aus der sie sich ergeben.[10] Wie weit diese drei Bedingungen für den vorliegenden Kontext gegeben sind, kann ohne umfangreiche empirische Untersuchungen nicht entschieden werden. In der Literatur zur Fehlerdiagnose wird üblicherweise zwischen den auf systematischen Fehlvorstellungen oder falsch gelernten Verfahren beruhenden "bugs" und den unsystematischen, etwa aufgrund einer kurzfristigen Ablenkunge auftretenden "slips" unterschieden, siehe z.B. Payne und Squibb [167], S. 452. Zur automatisierten Diagnose und sogar Generierung von Fehlermustern bei der schriftlichen Subtraktion existiert eine umfassende empirische Untersuchung, siehe VanLehn [198]. In [167] beziehen sich Payne und Squibb kritisch auf die Behauptung, die Mehrzahl der Fehler im Bereich der elementaren Arithmetik sei durch "bugs" erklärbar; insbesondere weisen sie darauf hin, dass zwischen "slips" und sehr selten auftretenden "bugs" kaum eine klare Abgrenzung möglich ist ([167], S. 452). Aber auch in [167] ergibt sich, dass zehn "mal-rules" immerhin zur Diagnose von 67 Prozent der auftretenden Fehler genügten ([167], S. 452). Dennoch sind die in [167] beobachteten Einschränkungen für die Wirksamkeit einer mal-rule-basierten Diagnose potenziell auch für den Einsatz des Anti-ATP im Diproche-System relevant: So beobachten Payne und Squibb durch einen Vergleich der von SchülerInnen verschiedener Schulen verwendeten mal-rules, dass deren Vorkommen und Häufigkeiten sich zwischen den Schulen unterscheiden und somit offenbar vom erhaltenen Mathematikunterricht abhängen ([167], S. 478); da Studierende beim intendierten Einsatz von Diproche zwar die jeweilige Lehrveranstaltung gemeinsam besuchen, aber sehr unterschiedliche schulische Hintergründe haben, könnte sich hierdurch eine Schwierigkeit in Bezug auf die Diagnose von Fehlern ergeben, die bereits den Schulstoff betreffen, insbesondere also die Diagnose fehlerhafter Termumformungen (s.u.). Ermutigend in Bezug auf die intendierte Zielgruppe, nämlich Studierende mathematischer Studiengänge, ist hingegen die Beobachtung, dass der diagnostische Wert von mal-rules mit der Kompetenz der Probanden steigt ([167], S. 478). Wie weit diese in einem deutlich anderen Bereich angestellten Beobachtungen sich auf den vorliegenden Kontext übertragen lassen, wäre Gegenstand einer eigenen Untersuchung, die den Rahmen dieser Arbeit sprengt.

[10] Dies ist i.A. nicht unbedingt der Fall, was eine Diagnose allein aufgrund eines vorliegenden Fehlers erschwert, vgl. etwa Payne und Squibb [167], p. 451.

5.3.1 Feststellbare Fehlschlüsse

In diesem Abschnitt wollen wir eine Auswahl der Fehlschlüsse darstellen, die vom Anti-ATP aktuell erkannt werden können. Sie ist aus der Korrektur von Klausur- und Übungsaufgaben hervorgegangen und durch gelegentliche Hinweise von KollegInnen ergänzt worden.

Eine offene Herausforderung ist ein systematischerer und empirisch fundierterer Aufbau des Anti-ATP. So existieren in der Argumentationstheorie umfassende Systematiken von Fehlschlüssen, die für diesen Zweck nutzbar gemacht werden könnten, siehe etwa Walton et al. [211]. Weiter wäre es nützlich, die Erfassung von Schlussfehlern anhand von Übungsabgaben im Rahmen einer systematischen empirischen Studie durchzuführen. Darauf basierend könnten dann auch die Rückmeldungen optimiert werden. Wir diskutieren diese und weitere Ansätze im Ausblick dieser Arbeit.

Für die Systematik des Anti-ATP unterscheiden wir drei Formen von Fehlschlüssen: Formallogische Fehlschlüsse sind fehlerhafte aussagen- oder quantorenlogische Schlussformen, die unabhängig vom behandelten Gegenstand auftreten können. Gebietsspezifische Fehlschlüsse sind solche, die sich auf Begriffe und Relationen eines bestimmten Themenbereichs beziehen (etwa ein transitiver Gebrauch der Elementrelation). Ein dritter Typ sind fehlerhafte Umformungen arithmetischer Terme, wie etwa der "Klassiker", $(a+b)^2$ "umzuformen" zu a^2+b^2.

Formallogische Fehlschlüsse

- (Invertierte Kontraposition) $\dfrac{A \to B}{\neg A \to \neg B} \qquad \dfrac{\neg A \quad (A \to B)}{\neg B}$
- (Umkehrschluss) $\dfrac{A \to B}{B \to A}$
- (Exklusive Interpretation der Disjunktion) $\dfrac{A \qquad A \vee B}{\neg B}$
- (Fehlinterpretation der Implikation)[11] $\dfrac{\neg A}{\neg(A \to B)}$

[11] Aus Diskussionen mit Studierenden, die dieses Schlussmuster zeigten, ging hervor, dass es sich hierbei bisweilen um eine Verwechselung bzw. eine unterbleibende Unterscheidung zwischen "$(A \to B)$ ist nicht wahr" und "$(A \to B)$ ist nicht anwendbar" handeln könnte, also um die

- (Distributive Verwendung der Negation)[12] $\dfrac{\neg(A \wedge B)}{\neg A \wedge \neg B}$
- (Fehlerhaftes Umklammern), z.B. $\dfrac{((A \vee B) \wedge C)}{(A \vee (B \wedge C))}$
- (Quantorentausch) $\dfrac{\forall x \exists y \phi}{\exists y \forall x \phi}$
- (Fehlerhafte Quantorennegation) $\dfrac{\neg \forall x \phi}{\forall x \neg \phi}$ $\dfrac{\neg \exists x \phi}{\exists x \neg \phi}$

Gebietsspezifische Fehlschlüsse

- (Verwechselung von Teilmengen- und Elementrelation) $\dfrac{A \subseteq B}{A \in B}$ $\dfrac{A \in B}{A \subseteq B}$
- (Transitiver Gebrauch der Elementrelation)[13] $\dfrac{A \in B \quad B \in C}{A \in C}$

Diagnose fehlerhafter Termumformungen

Ein häufiger Fehlertyp in Beweisen sind Umformungsfehler im Rahmen von arithmetischen Term- oder Äquivalenzumformungsketten. Auch wenn es wünschenswert wäre, dass die diesbezüglich in der Schule erworbenen Kompetenzen das systematische Auftreten solcher Fehler wenigstens im Bereich der reellen Zahlen verhindern (womit jedenfalls als weitere Fehlerquelle die allzu eilfertige Übertragung von aus diesem Kontext bekannten Gesetzen etwa in

Vermischung zweier Vorstellungen von "nicht hilfreich".

[12]Für die Disjunktion und die Biimplikation sind diese Folgerungen richtig; ihre Anwendung kann natürlich im Einzelfall trotzdem aufgrund einer fehlerhaften Überzeugung von der generellen Distributivät der Negation erfolgen.

[13]Dieser Fehler geht möglicherweise auf eine anschauliche Deutung von "A ist/befindet sich in B" bzw. eine "Containersicht" auf Mengen (siehe etwa Lakoff und Nunez [136]) zurück: Befindet sich mein Brot in der Brotdose, die Brotdose sich im Rucksack und der Rucksack sich im Kofferraum, würde man wohl durchaus sagen, dass sich das Butterbrot im Kofferraum befindet.

allgemeine Gruppen oder Körper bliebe), zeigt die Erfahrung, dass typische Fehler wie etwa eine lineare Verwendung der Exponentiation ("$(a+b)^2 = (a^2+b^2)$"; häufig auch in der "versteckten" Form über das Wurzelziehen als $\sqrt{a+b} = \sqrt{a}+\sqrt{b}$) oder sogar das komponentenweise Addieren von Brüchen ("$\frac{a}{b} + \frac{c}{d} = \frac{a+c}{b+d}$") regelmäßig auftreten. Eine wirksame Fehlerdiagnose sollte daher in der Lage sein, möglichst viele derartige Fehler zu erkennen und entsprechende Rückmeldungen zu liefern. Dies gilt umso mehr, als Untersuchungen zu Umformungsfehlern dafür sprechen, dass diese in der Mehrzahl nicht auf Flüchtigkeit, sondern auf Fehler in den entsprechenden erlernten Routinen, sogenannte "mind bugs" zurückzuführen sind, siehe etwa erneut VanLehn [198].

In Diproche übernimmt das ein auf Termumformungen spezialisiertes Submodul des Anti-ATP, genannt "Antiterms". Im Gegensatz zum eher mageren Forschungsstand bei formalen Fehlschlüssen im Bereich der Aussagenlogik gibt es zu Fehlumformungen zahlreiche Untersuchungen, zumal dann, wenn man die Vermutung wagt, dass im numerischen Bereich auftretende Fehler sich auch im "Buchstabenrechnen" wiederfinden werden, also dass etwa, wer $\frac{1}{2} + \frac{2}{3} = \frac{3}{5}$ rechnet, auch dazu neigen wird, $\frac{a}{b} + \frac{c}{d} = \frac{a+c}{b+d}$ zu schreiben. Neben eigenen Beobachtungen beim Korrigieren von Übungsaufgaben und Klausuren sowie dem, was mir von KollegInnen diesbezüglich zugetragen wurde, basiert das Antiterms-Modul vor allem auf den in den mit "häufige Schülerfehler" überschriebenen Abschnitten von Padberg und Wartha "Didaktik der Bruchrechnung" [157] sowie Brandles äußerst reichhaltigen "Analyse von Rechenfehlern im Grundschulbereich" [22] und Gaidoschiks "Wie Kinder rechnen lernen – oder auch nicht" [84].

Zur automatischen Erkennung von algebraischen Umformungsfehlern durch mal-rule libraries existiert bereits eine umfangreiche Literatur, siehe etwa Burton [29] oder Sleeman [184]. Diese bezieht sich aber, soweit uns bekannt, primär auf eng begrenzte und sehr elementare Bereiche, etwa die schriftliche Subtraktion. Unseres Wissens nach stellt das Termumformungsmodul des Anti-ATP den ersten Versuch dar, dieses Konzept im Rahmen eines digitalen Systems zur Vermittlung von Beweiskompetenzen einzusetzen.

Das Modul Antiterms erfasst derzeit 30 fehlerhafte Arten der Umformung; beispielhaft seien zur Verdeutlichung die folgenden aufgeführt:

- "Formales Kürzen": $\frac{n^2}{n^3} = \frac{2}{3}$, $\frac{n-1}{n^2-1} = \frac{n}{n^2}$.
- Komponentenweises Addieren von Brüchen: $\frac{a}{b} + \frac{c}{d} = \frac{a+c}{b+d}$; bei Addition einer Bruchzahl zu einer Nicht-Bruchzahl bisweilen auch in beiden Komponenten $\frac{a}{b} + 1 = \frac{a+1}{b+1}$.
- Verwechselung von Exponentiation und Multiplikation; Kürzen der Exponenten: $\frac{a^3}{a^6} = \frac{1}{a^2}$

Von Antiterms werden weiterhin auch solche Fehlumformungen erfasst, die sich lediglich auf einen Subterm beziehen und bei denen ggf. sogleich einige korrekte Vereinfachungsschritte vorgenommen wurden; so würde etwa $3 \cdot (a + \frac{a}{4}) + 1 = 3 \cdot (\frac{2 \cdot a}{4}) + 1)$ als Folge des Fehlers $a + \frac{b}{c} = \frac{a+b}{c}$ (vgl. Padberg und Wartha [157], S. 93) erkannt. Nicht erfasst werden dagegen derzeit Fehler, die sich nicht auf die Gleichheit, sondern die Größe von Ausdrücken beziehen, wie etwa falsche Monotonieannahmen in Schritten wie $\frac{a}{b} < \frac{a+1}{b+1}$

Für den/die BenutzerIn erfolgt dann eine Rückmeldung in Form einer kurzen Benennung des Fehlers. Als zusätzliche Hilfestellung könnten dann kurze Erläuterungstexte zu diesen Fehlern dienen, die etwa Gegenbeispiele und weiteres Übungsmaterial anbieten; siehe dazu auch den Abschnitt "Ausblick" am Ende der Arbeit.

Eine mögliche Erweiterung des Antiterms-Moduls könnte durch eine systematische Klassifikation der Fehler und eine entsprechend allgemeinere Implementierung erreicht werden. Viele der typischen Fehler scheinen auf die Übertragung von aus einem Bereich bekannten Rechengesetzen auf einen anderen zurückzugehen; so dürften etwa sowohl $(a + b)^2 = (a^2 + b^2)$ als auch $\frac{a}{b} + n = \frac{a+n}{b+n}$ und $((A \wedge B) \rightarrow C) \leftrightarrow ((A \rightarrow C) \wedge (B \rightarrow C))$ Ausdrücke eines vermeintlichen "allgemeinen Linearitätsgesetzes" der Form $(a \circ b) * c) = ((a * b) \circ (a * c))$ sein. Eine systematische Erfassung und Erkennung von Fehlermustern dieser Art könnte das Antiterms-Modul auch für Gegenstandsbereiche nutzbar machen, für die gar keine explizite Analyse von Fehlertypen vorliegt.

5.4 Erzeugung von Gegenbeispielen

Die Fehlschlussdiagnose kann den Lernprozess dadurch unterstützen, dass die Verwendung gewisser bekannter Fehlschlüsse erkannt und angezeigt wird. Bei Fehlern, die keinem bekannten Muster entsprechen, liefert sie hingegen keine Information. In diesem Fall erhält der/die BenutzerIn lediglich die Meldung, dass der entsprechende Schritt nicht nachvollzogen werden konnte. Wie häufig die Fehlschlussdiagnose Informationen liefern wird und wie sehr diese dem intendierten BenutzerInnenkreis dabei helfen, Fehler zu erkennen und zu verbessern (und ggf. künftig zu vermeiden) ist eine Frage, die letztlich nur empirisch untersucht werden kann. In jedem Fall wäre es vorteilhaft, auch dann genauere Information über die Art des Fehlers zur Verfügung zu stellen, wenn dieser keinem bekannten Fehlermuster entspricht. Insbesondere dürfte es für die Verbesserung eines Beweistextes nützlich sein, zu wissen, ob ein Folgerungschritt inhaltlich falsch ist – also die jeweilige Behauptung sich aus den verfügbaren Voraussetzungen nicht folgern läßt – oder ob der Schritt zwar logisch zulässig, aber noch nicht ausreichend begründet ist: Während im ersten Fall grundsätzlich neu angesetzt werden müsste,

wird der Beweistext sich im zweiten Fall im Allgemeinen durch die Ergänzung einiger Zwischenschritte verbessern lassen. Überdies dürfte es die Einsicht in die Fehlerhaftigkeit eines Schrittes fördern, wenn dessen Fehlerhaftigkeit nicht nur behauptet, sondern auch begründet wird.

Diese Funktionen erfüllt die Angabe von Gegenbeispielen zu Schlussfolgerungen, also von Situationen, in denen sämtliche verfügbaren Voraussetzungen wahr, die behauptete Schlussfolgerung aber falsch ist. Einige Systeme zum automatischen Beweisprüfen, besonders solche mit didaktischer Zielsetzung wie etwa das weiter oben besprochene "Elfe", verfügen daher über einen Gegenbeispielgenerator, der sich meist auf den zugrundeliegenden ATP oder Beweisassistenten stützt und dessen Ausgabe in einer benutzerfreundlichen Form anzeigt.

Im Rahmen von Diproche ist diese durchaus nützliche Funktion derzeit nur partiell realisiert, nämlich einmal für den Bereich der Aussagenlogik sowie, in begrenztem Umfang, für den Bereich der Booleschen Mengenlehre.

Wird in einem aussagenlogischen Bereich eine Behauptung B durch die logische Prüfung als auf der Basis der Liste Vss verfügbarer Voraussetzungen nicht verifizierbar gemeldet, so wird durch eine Wahrheitstafel geprüft, ob $\bigwedge \text{Vss} \to B$ eine Tautologie ist, ob also B unter Voraussetzung aller verfügbaren Annahmen logisch folgt. Falls ja, erfolgt keine weitere Verarbeitung. Andernfalls wird eine Belegung der auftretenden Aussagevariablen zurückgemeldet, unter der alle Elemente von Vss wahr, B aber falsch wird.

Für die Boolesche Mengenlehre ist dieselbe Funktion aktuell nur in dem Fall verfügbar, dass B eine Kette von mindestens zwei Teilmengen- bzw. Gleichheitsbeziehungen ist und kein Element von Vss die Elementrelation verwenden; in diesem Fall lassen sich sämtliche auftretenden Ausdrücke in entsprechende aussagenlogische Formeln "übersetzen", so dass der eben beschriebene Wahrheitstafelalgorithmus angewendet werden kann. Das "Gegenbeispiel" besteht in diesem Fall aus einem Objekt, das in gewissen der durch die auftretenden Mengenvariablen bezeichneten Mengen enthalten und in anderen nicht enthalten ist, so dass diese Kombination mit allen Elementen von Vss vereinbar, mit B aber unvereinbar ist. Auf die Eingabe

> Es seien A, B, C Mengen. Es sei $A = B$. Dann folgt $A = B = C$.

generiert der Gegenbeispielgenerator etwa folgende Ausgabe:

> Zeile 4: $(B = C)$ ist z.B. dann falsch, wenn ein Objekt existiert, das: – in A liegt – in B liegt – nicht in C liegt

Den Gegenbeispielgenerator auf andere Bereiche auszuweiten ist eine Aufgabe für künftige Verbesserungen des Systems.

5.5 Ausblick: Beweisanalyse

Die Beurteilung Qualität eines Beweises (bzw. Beweistextes), und damit auch das, was sich an einem Beweisversuch lernen läßt, beschränkt sich nicht auf die bloße Feststellung seiner logischen Korrektheit, der Erreichung oder Nichterreichung des Beweiszieles und Anmerkungen zur Form der Darstellung. Nicht umsonst wurden und werden viele Lehrsätze immer wieder bewiesen, so dass wir inzwischen für einige, wie etwa den Satz des Pythagoras, eine dreistellige Anzahl von Beweisen besitzen.[14] Wichtige weitere Kriterien sind z.B.:

1. Die Nachvollziehbarkeit der Darstellung: Ist die Beweisdarstellung übersichtlich, sind strategisch zentrale Schritte des Argumentes von Hilfsargumenten etc. zu unterscheiden? Ist die Beweisdarstellung heuristisch motiviert, d.h. ist es für den/die LeserIn nachvollziehbar, was warum wann gemacht wurde und worauf es hinauslaufen soll?

2. Die Angemessenheit der Mittel: Im Allgemeinen wird man es für sinnvoll halten, einen Satz mit möglichst wenig Voraussetzungen zu beweisen, sowohl hinsichtlich der verwendeten Axiome als auch hinsichtlich der Vorarbeiten.[15] Ein ganzer Zweig der Logik, die sogenannte reverse Mathematik (siehe z.B. Simpson [182]), ist damit beschäftigt, für die Sätze aus verschiedenen Gebieten die minimalen axiomatischen Voraussetzungen zu bestimmen. Insbesondere wird man dazu neigen, Beweise zurückzuweisen, die Hilfssätze verwenden, welche ihrerseits nur mit deutlich komplizierteren Methoden zu beweisen sind als der Satz selbst oder gar eine mögliche Beweismethode für den fraglichen Satz verallgemeinern oder erweitern. Im Hinblick auf das Verständnis eines Beweises als eines Mittels, vernünftige Wesen von der Richtigkeit einer Behauptung zu überzeugen, ist eine solche Zurückweisung auch durchaus zu rechtfertigen: Der Versuch etwa, die Unendlichkeit der Primzahlen durch Rekurs auf den Primzahlsatz zu beweisen, fordert den Kommentar heraus, dass wohl niemand, der den Primzahlsatz und seinen Beweis verstanden hat, an der Existenz unendlich vieler Primzahlen

[14] Für eine Sammlung von Beispielen für solche "Mehrfachbeweise" und einige der folgenden Kriterien für die Qualität von Beweisen vgl. Dawson, [56].

[15] Hiervon kann natürlich abgewichen werden, insbesondere dann, wenn es darum geht, die Reichweite eines tiefliegenden Satzes anhand der Vielfalt seiner Konsequenzen aufzuzeigen.

noch Zweifel hegen wird – und das Argument insofern ein Argument 'für niemanden' ist.[16]

3. Die Effizienz des Argumentes: Wenn man den verwendeten Rahmen hinsichtlich verwendeter Ressourcen (Sätze) und Methoden (Schlussweisen) als gegeben ansieht, kann das Argument noch immer Umwege gehen und entsprechend Verkürzungen erlauben. Dies zeigt sich im Lehrbetrieb z.B. an Stellen, wo Fallunterscheidungen vorgenommen werden ohne zu beachten, dass einige Fälle gar nicht auftreten können oder einige Fallannahmen für die fragliche Aussage unerheblich sind.[17]

4. Die Reichweite des Argumentes: Ein Beweis zeigt selten genau das, was als Beweisziel angegeben wird. In vielen Fällen zeigt eine genauere Analyse im Nachhinein, dass die Voraussetzungen abgeschwächt oder die Folgerung verstärkt werden kann.

Diese Liste ist bei weitem nicht abschließend[18], gibt aber immerhin eine Andeutung der Gesichtspunkte, die eine umfassende Beweisbegutachtung berücksichtigen sollte. Hinsichtlich der Möglichkeit einer Automatisierung in einem System wie Diproche ist hierzu folgendes zu sagen:

1. Eine stilistische Evaluation ist zumindest im Prinzip teilweise realisierbar: So könnte z.B. darauf geachtet werden, ob verwendete Voraussetzungen an den Stellen, an denen sie verwendet werden, explizit erwähnt oder in hinreichender Nähe zuvor erwähnt wurden. Dazu müsste der ATP die verwendeten Voraussetzungen auflisten und anschließend würde deren Position im Text festgestellt und ausgewertet. Ein Schritt in Richtung strategisch durchsichtiger Beweise könnte die Integration von Beweisplänen sein, wie sie in sogenannten Proof-Planning-Systemen verwendet werden, wie etwa Coq (siehe etwa Bertot und Casteran, [16]) oder Ωmega, siehe

[16]Patrick Lahr verdanken wir den Hinweis auf ein weiteres Beispiel, nämlich den folgenden Beweis der Irrationalität von $\sqrt[3]{2}$, der unter Mathematikern aus guten Gründen als Witz verbreitet wird: Wäre $\sqrt[3]{2}$ rational, so gäbe es von 0 verschiedene natürliche Zahlen p und q mit $p^3 = 2q^3$, also $p^3 = q^3 + q^3$, was aber dem großen Satz von Fermat widerspricht.

[17]Michael Schmitz machte uns auf folgendes Beispiel aufmerksam: Gerade im Anfängerbereich ist immer wieder eine Neigung zu beobachten, eine Unterscheidung in Teilfälle, die in einem Fall getroffen wurde, auch in allen anderen durchzuführen: Soll etwa gezeigt werden, dass $a \cdot b$ genau dann eine gerade Zahl ist, wenn a oder b gerade ist, könnte so ein Argument etwa dadurch erfolgen, dass eine Fallunterscheidung nach der Parität von a durchgeführt wird und, da sich im Fall, dass a ungerade ist, eine weitere Fallunterscheidung nach der Parität b als nötig erweist, eine solche auch in dem Fall vorgenommen wird, dass a gerade ist – wo sie freilich unnötig ist.

[18]Für zahlreiche weitere Kriterien siehe z.B. die Einleitung des schon erwähnten Buches von Dawson, [56]

Siekmann et al. [156]. Eine auf Verständlichkeit angelegte automatische Evaluation könnte dann auf längere 'planlose' Textpassagen hinweisen und ggf. anhand von linguistischen und logischen Informationen (verwendete Schlussformen etc.) passende Beweispläne vorschlagen. Der derzeit intendierte Anwendungsbereich von Diproche sind freilich Beweise von einer solchen Kürze und Einfachheit, dass der Mehrwert eines solch aufwändigen Moduls zweifelhaft erscheinen muss.

2. Die verwendeten Voraussetzungen und Methoden werden in Diproche über explizite Kontexte (Schwierigkeitsgrade und Spielwiesen) kontrolliert. Darüber hinaus ist es ohne größere Schwierigkeiten möglich, zu einer gegebenen Aufgabe einen Minimalvorrat an Ressourcen anzugeben, die zu ihrer Lösung erforderlich sind, und eine Rückmeldung zu geben, wie weit dieser eingehalten wurde. Es könnte dann eine zusätzliche Herausforderung für den/die BenutzerIn darin liegen, dieses Minimum zu erreichen. Auch hier ist aber fraglich, wie weit solche Betrachtungen im intendierten Einsatzbereich von Diproche sinnvoll sind.

3. Eine Bewertung hinsichtlich der Verkürzbarkeit eines gegebenen Beweises ist dagegen auch bei einfachen Beweisen bereits sinnvoll und technisch auch umsetzbar, zumindest im Prinzip sogar recht leicht: Um zu prüfen, ob alle Beweisschritte erforderlich sind, verwendet zunächst den ATP, um festzustellen, welche bisher vollzogenen Schritte zur Ableitung des Beweiszieles nötig sind. Ergibt sich bereits hier, dass etwa gewisse Zwischenergebnisse unnötig waren (z.B. wenn es zwischen dem letzten zur Ableitung des Beweiszieles verwendeten Zwischenresultat noch weitere Zeilen gibt), erfolgt eine entsprechend Rückmeldung; andernfalls wird mit den benutzten Zwischenresultaten auf die gleiche Weise verfahren.

   De facto hat dieses Vorgehen allerdings einige Schwierigkeiten: Zum einen wird es im Allgemeinen nicht eine minimale Menge von nötigen Zwischenergebnissen geben, sondern mehrere, was in der Rückverfolgung zu einer kombinatorischen Explosion führen kann. Zum anderen wird diese Form der Evaluation echte Umwege nicht von Beweisabschnitten unterscheiden, die lediglich in höherem Detailgrad ausgeführt wurden als für den ATP erforderlich. Der zweiten Schwierigkeit ließe sich dadurch zumindest teilweise begegnen, dass bei höheren Beweisregeln, die mehrere elementare Schritte in einem zusammenfassen, ihre möglichen 'Ausfaltungen' mitgeführt und ggf. mit dem Text verglichen werden, so dass Passagen, die einen durch eine solche Regel vollziehbaren Schritt in Teilschritte zerlegen, identifiziert werden können. Bei der ersten Schwierigkeit wäre zu untersuchen, wie weit sie in der Praxis überhaupt relevant ist.

4. Auch hier könnte eine Rückverfolgung wie im Punkte 3 beschrieben Hinweise liefern, ob alle angegebenen Voraussetzungen verwendet wurden. Darüber hinaus wäre es im Prinzip möglich, einige naheliegende Abschwächungen automatisiert auszuprobieren (etwa, ob ein Beweis zugleich zeigt, dass eine für rationale Zahlen formulierte Aussage auch für reelle Zahlen gilt).

Keiner der obigen Punkte ist in der aktuellen Version von Diproche umgesetzt; sie sollten daher als Ausblick auf und Anregung zu möglichen künftigen Erweiterungen und Verbesserungen betrachtet werden.

5.6 Integration von Kontexten: Die "Spielwiesen"

Wie bereits in der Diskussion zur Machbarkeit, besonders im Abschnitt zur Studie von Inglis et al., festgestellt wurde, muss für den erfolgreichen Einsatz eines Systems wie Diproche stets der Verwendungskontext sehr klar festgelegt werden. Das betrifft zunächst die Festlegung, sich auf Beweisaufgaben im Anfängerbereich eines Mathematikstudiums zu konzentrieren, weiter die "Schwierigkeitsgrade", also die flexible Gestaltung der zugelassenen Inferenzregeln in Abhängigkeit von der jeweils zu bearbeitenden Aufgabe; vor allem aber betrifft es die Themenfelder bzw. Teilgebiete, zu denen die Aufgaben gestellt werden. In der aktuellen Diproche-Version sind u.a. die Themenfelder Aussagenlogik, Boolesche Mengenlehre, Funktionen und Relationen, Gruppentheorie, Induktion, elementare Zahlentheorie und axiomatische Geometrie implementiert. Mit jedem dieser Themenfelder – die wir, einem Vorschlag von Jörn Schnieder folgend, auch als "Spielwiesen" bezeichnen werden – sind besondere Teile der Implementierung verbunden. Zu jedem Themengebiet gehören:

- Eine den Erfordernissen des jeweiligen Gebietes angepasste kontrollierte natürliche Sprache, die im System repräsentiert wird durch spezifische Regeln für das Textparsing sowie die Formalisierung natürlichsprachlicher Formulierungen. Jedes Gebiet hat ein spezifisches Vokabular (wozu etwa in der Zahlentheorie etwa Begriffe wie "Quadratzahl" oder "ungerade" gehören, in der Geometrie z.B. Begriffe wie "parallel" oder "Raute") sowie spezifische komplexere Formulierungen ("ist die Parallele zu g durch p")). Dazu gehört ein entsprechender Satz von Parsing-Regeln für die linguistische Verarbeitung. Für diejenigen Gebiete, in denen das in besonderem Maß auftritt, ist das Parsing in separate Module ausgelagert; in der aktuellen Version ist das der Fall für die Gebiete "Funktionen und Relationen", "Gruppentheorie" und "axiomatische Geometrie".

- Eine den Erfordernissen des jeweiligen Gebietes angepasste Formelsyntax, im System repräsentiert durch spezifische Regeln für das Formelparsing. In

jedem Teilgebiet gibt es eine spezifische Symbolik, wobei gleiche Symbole in verschiedenen Kontexten verschiedene Bedeutungen haben können (so bedeutet "e" etwa in der Gruppentheorie etwas anderes (neutrales Element) als in der Analysis (Eulersche Zahl) oder in der Geometrie (hier kann e einfach als Variable, z.B. für Geraden, verwendet werden, weil der Kontext Verwechselungen nahezu ausschließt); in Diproche ist z.B. zu unterscheiden, ob "$*$" im Rahmen der Zahlentheorie für die gewöhnliche Multiplikation, im Rahmen der Gruppentheorie für eine beliebige binäre Verknüpfung oder im Rahmen von "Funktionen und Relationen" für die Komposition von Funktionen steht. Daher gibt es zu jedem Teilgebiet einen Satz an spezifischen Regeln für das Parsing von formalen Ausdrücken und deren Übersetzung in das interne Listenformat, auf dem die weitere Verarbeitung beruht.

- Typische, mit den für das Gebiet wichtigen Begriffen in Verbindung stehende Schlussweisen und Definitionen, im System repräsentiert durch spezielle ATP-Regeln und -Routinen. Diese sind für einige Teilgebiete, wie etwa die elementare Zahlentheorie oder die axiomatische Geometrie, sehr umfangreich (vgl. die detaillierten Ausführungen zu diesen Spielwiesen weiter unten) und sind in separate Module ausgelagert, die ihrerseits auf weitere Submodule zugreifen. So hat etwa das Zahlentheorie-Modul Zugriff auf Module mit Spezialregeln zu Termumformungen, Teilbarkeit sowie Kongruenzen (siehe auch die Besprechung des ATPs weiter oben).

- Ein Typensystem, im System repräsentiert durch besondere Regeln für das Typechecking. In der Zahlentheorie wird zwischen verschiedenen Zahlbereichen unterschieden, in der Geometrie zwischen Punkten, Geraden, Kreisen und Figuren etc.; ferner müssen die im jeweiligen Gebiet auftretenden Funktionen und Relationen im Hinblick auf ihr Verhalten auf der Typenebene spezifiziert werden. Für die elementare Zahlentheorie, die axiomatische Geometrie und die Boolesche Mengenlehre existieren eigene Module, die das Typechecking für Texte aus diesen Bereichen übernehmen. Im Fall der Geometrie kommt es zudem durch die Mehrsortigkeit der zugrundeliegenden Sprache zu Interaktionen von Inferenz und Typensystem, weil die Anwendbarkeit von Inferenzregeln vom Typ der auftretenden Variablen und Terme abhängen kann; z.B. ist "Es existiert eine Parallele zu g durch p" nur dann sinnvoll und wahr, wenn g eine Gerade und p ein Punkt ist, nicht aber, wenn es sich etwa umgekehrt verhält.

- Aufgaben und Schwierigkeitsgrade. Zu jedem Gebiet gibt es spezielle Übungsaufgaben; zu diesen gehört jeweils die Angabe einer Menge zulässiger Inferenzregeln. Für jedes Gebiet entsteht damit die Herausforderung,

Mengen von Inferenzregeln festzulegen, mit denen eine nennenswerte Anzahl von Aufgaben hinreichend natürlich bearbeitet werden kann. Im Fall der Zahlentheorie etwa gibt es zahlreiche Aufgaben zu Paritäten der Art "Zeige: Wenn n gerade ist, so ist n^2+5 ungerade", bei denen die Voraussetzung "n ist gerade" jeweils durch Anwendung der Definition, also die Einführung einer ganzen Zahl k mit $n = 2k$, ins Spiel zu bringen ist, während später Regeln wie "wenn a und b ungerade sind, so auch ab" einfach zur Verfügung stehen. Besonders für die axiomatische Geometrie liegt hier eine Herausforderung, wie wir weiter unten sehen werden.

5.7 Weitere Funktionen: Formalisierungsübungen, "Beweispuzzles" und mehr

Auch wenn wir nicht der Ansicht sind, dass eine Schulung in formaler Logik ein geeigneter Einstieg in das mathematische Beweisen ist, setzt das Verstehen und Verfassen mathematisch begründender Texte die Beherrschung zumindest der Ausdrucksmittel der erststufigen Prädikatenlogik voraus und wird dadurch gefördert. Diese Kompetenzen können im Rahmen der oben besprochenen Funktionen von Diproche nicht gefördert werden: Ein formal fehlerhafter Ausdruck wird zwar als solcher gemeldet, doch die Fähigkeit, eine gegebene Aussage zu formalisieren bzw. einer quantorenlogischen Formel ihren Sinn zu entnehmen (wie es z.B. für das Verständnis so zentraler Begriffe wie Konvergenz oder Stetigkeit erforderlich ist), läßt sich dadurch allein kaum trainieren. Aus diesem Grund verfügt Diproche über zwei Komponenten, die speziell das Formalisieren üben sollen: Einerseits die "Mathediktate", andererseits das "Game of Def".

Ferner enthält Diproche noch eine Reihe anderer Übungsprogramme zu verschiedenen Aspekten des Beweisens, so die "Beweispuzzles", das "Markup"-Spiel, bei dem Sätzen eines Beweistextes ihre Funktion (Annahme, Folgerung, Deklaration, Ziel etc.) zugeordnet werden soll, das "Korrekturspiel", bei dem BenutzerInnen fehlerhafte Beweisschritte identifizieren sollen, "Diagnose", bei dem BenutzerInnen die Rolle des Anti-ATP einnehmen und Fehlschlussmuster erkennen sollen, "Vereinfachen", bei dem aussagenlogische Ausdrücke und arithmetische Terme möglichst weit vereinfacht werden sollen, "Reformulate", bei dem aussagenlogische Ausdrücke mit einem beschränkten Vorrat an Junktoren wiedergegeben werden sollen und das aktuell noch nicht ins Interface eingebundene "DefCheck", bei dem es um die Anwendung formaler Definitionen geht.

Mathediktate

Dem Konzept des "Mathediktats" sind wir erstmals an der Universität Konstanz begegnet, wo Michael Junk es im Rahmen seiner Einführungskurse eingesetzt hat. Bei einem Mathediktat wird Studierenden die natürlichsprachliche Formulierung einer mathematischen Aussage vorgelegt, die diese nun formalsprachlich reformulieren sollen. So könnte z.B. die Aussage "Die reelle Funktion f ist streng monoton steigend" zu übersetzen sein in

$$\forall x \forall y (x < y \to f(x) < f(y)).$$

Auch Edukera (s.o.) verfügt über einen Übungsteil mit Formalisierungsübungen, in dem natürlichsprachliche Aussagen zu formalisieren sind, z.T. aus Alltagsdomänen, aber auch aus dem Bereich "Funktionen und Relationen". (Das "Mathediktate"-Modul wurde unabhängig von, zeitlich aber deutlich nach diesem Edukera-Modul konzipiert und entwickelt.) Ein System, das das Formalisieren natürlichsprachlicher Aussagen in erststufiger Logik entlang eines kleinschrittigen Verfahrens vermittelt, ist das NLtoFOL ("natural language to first-order logic")-System, siehe etwa Perikos, Grivokostopoulou und Hatzilygeroudis [160], [161]. NLtoFOL verfügt neben einem systematischen Aufbau des Formalisierungsprozesses auch über ein umfangreiches Hinweissystem, das sogar metakognitive Analysen des Nutzerverhaltens verwendet ([161]) und wurde bereits ausführlich evaluiert, wobei sich signifikante Lerneffekte zeigten (ebd.). Paradigmatisch scheint dieses System jedoch eher für Formalisierungen geeignet zu sein, wie sie in der formalen Wissensrepräsentation, etwa in Bereichen wie der Informatik, insbesondere der künstlichen Intelligenz, eine Rolle spielen; in solchen Feldern besteht die Herausforderung typischerweise darin, eine zum jeweiligen Kontext passende Ontologie erst zu entwerfen. Entsprechend leitet NLtoFOL BenutzerInnen dazu an, aus einem natürlichsprachlichen Satz zunächst die logische zentralen Begriffe zu extrahieren und als Funktionen, Konstanten, Variablen, Relationen etc. zu klassifizieren (siehe [161], Anhang A). In der Mathematik dagegen ist der begriffliche Rahmen gewöhnlich fixiert und die Aufgabe besteht darin, mit diesen begrenzten Mitteln Aussagen zu formulieren, die erst nach einiger inhaltlicher Arbeit einen Bezug zu diesem Rahmen aufweisen. So hat etwa die Aussage "f hat mindestens zwei Nullstellen" wenig syntaktische Ähnlichkeit zu der Formel $\exists x \exists y (x \neq y \wedge f(x) = 0 \wedge f(y) = 0)$, und es ist zweifelhaft, ob der in [161] beschriebene Prozess geeignet ist, um zu dieser Formalisierung zu gelangen. (Zudem arbeitet das NLtoFOL-System mit einem englischen Sprachparser und wäre daher nur mit erheblichen Anpassungen für deutschsprachige Formalisierungsaufgaben zu verwenden.) Es ist daher unklar, wie weit dieser Ansatz auch für die in der Mathematik üblichen Formalisierungsaufgaben trägt.

Das Mathediktate-Modul von Diproche automatisiert die Auswertung solcher Mathediktate auf naheliegende Weise. Eine Aufgabe besteht aus einer natürlichsprachlichen mathematischen Aussage aus dem Bereich "reelle Funktionen", wobei als Relationszeichen nur $=, <, >, \leq, \geq$ zur Verfügung stehen sowie Variablen für einstellige Funktionszeichen und für reelle Zahlen.

Präzise ist die Syntax der akzeptierten Sprache wie folgt definiert:[19]

- Kleine lateinische Buchstaben werden als Variablen und Konstantenzeichen verwendet; sowohl Variablen als auch Konstantenzeichen sind Terme.
- Jede natürliche Zahl (in Dezimaldarstellung, ohne führende Nullen) ist ein Term.
- Sind a und b Terme und ist a keine Zahl, so ist $a(b)$ ein Term, der die Anwendung der Funktion a auf b beschreibt (was offenbar nur dann sinnvoll ist, wenn a eine Funktion ist).
- Sind a und b Terme, so sind $a < b$, $a \leq b$, $a > b$, $a \geq b$ und $a = b$ Formeln.
- Sind ϕ und ψ Formeln, so auch $(\phi \& \psi)$, $(\phi v \psi)$, $(\phi \rightarrow \psi)$ und $\sim \phi$.
- Sind ϕ und ψ Formeln und ist x ein kleiner lateinischer Buchstabe, so sind auch $Ax : \phi$ und $Ex : \phi$ Formeln.

Formal besteht eine Aufgabe aus einem 4-Tupel (Id,Nat,Formal,FreeVars), wobei Id die Kennung der Aufgabe ist, Nat die natürlichsprachliche Formulierung, die der/dem BenutzerIn angezeigt wird, Formal eine Liste von korrekten Formalisierungen dieser Aussage und FreeVars die Liste der freien Variablen, die in einer korrekten Lösung vorkommen müssen (so wird "f ist streng monoton steigend" durch keine Formel korrekt formalisiert, die f nicht als freie Variable oder darüber hinaus noch andere enthält). Der Grund, warum "Formal" aus einer Liste von Formalisierungen statt einer einzelnen besteht, ist folgender: Bisweilen gibt es einige wesentlich verschiedene Ansätze zur Formalisierung einer Aussage, deren Äquivalenz nicht ohne Hintergrundwissen über die Domäne einzusehen ist und sich daher den Möglichkeiten des hier verwendeten ATP entzieht (und auch entziehen soll: siehe dazu die Ausführungen weiter unten). In diesem Fall können alle diese Möglichkeiten angegeben werden.[20]

[19] Die folgenden Erläuterungen sind im wesentlichen eine deutsche Fassung von Teilen aus der Arbeit "Automatized Evaluation of Formalization Exercises in Mathematics", [33].

[20] Die prinzipielle Möglichkeit, dass unüberschaubar viele oder gar unendlich viele wesentlich verschiedene Formalisierungsansätze möglich sind, ignorieren wir hier – sollte so ein Fall tatsächlich vorkommen, ist er eben als Übungsaufgabe für das gegebene Übungsformat ungeeignet.

Über ein Eingabefenster gibt der/die BenutzerIn einen String ein, aktiviert die Prüfung und erhält Rückmeldungen darüber, ob die Eingabe ψ eine wohlgeformte Formel im Sinne der oben angegebenen Syntax ist und ob sie zur gegebenen Behauptung ϕ äquivalent ist oder zumindest eine notwendige bzw. hinreichende Bedingung für sie darstellt. Letzteres wird dadurch geprüft, dass die Formeln $\phi \rightarrow \psi$ und $\psi \rightarrow \phi$ an einen automatischen Theorembeweiser für die Prädikatenlogik erster Stufe übergeben werden, der im wesentlichen der Beschreibung eines Tableau-Beweisers durch Fitting [70] folgt.

Einige Beispiele für Aussagen, die im Rahmen der automatisierten Mathediktate als Übungen dienen können, sind folgende:

1. Die Funktion f hat nur positive Werte.
2. f ist eine streng monoton wachsende Funktion.
3. f hat überall dort eine Nullstelle, wo g eine Nullstelle hat.
4. f dominiert g global.
5. f ist nach oben unbeschränkt.

Auch in diesem Modul ist der ATP strikt kontrolliert; nicht jede Formel ϕ, die zu einer gegebenen Aussage logisch äquivalent ist, kann schließlich auch als "Formalisierung" dieser Aussage gelten. Hier stellt sich das Adäquatheitsproblem von Übersetzungen: So ist "Zu je zwei verschiedenen reellen Zahlen existiert eine dritte, die echt zwischen ihnen liegt" als wahre Aussage zwar in der Tat logisch äquivalent zu der Formel $1 = 1$, letztere sollte aber sicherlich nicht als Formalisierung dieser Aussage angesehen werden. Vielmehr sollte die Entsprechung zwischen natürlichsprachlichem Ausdruck und Formel "unmittelbar sichtbar sein", was insbesondere heißt: Ohne weiteres Wissen über den Gegenstandsbereich – reelle Zahlen und Funktionen – zu verwenden. Das einzige spezifische Domänenwissen, das dem verwendeten ATP zur Verfügung steht, ist die Austauschbarkeit von $a < b$ mit $b > a$, von $a \leq b$ mit $b \geq a$ sowie von $a = b$ mit $b = a$. Ferner ist auch die Inferenz stark kontrolliert: So dürfen nicht mehr als drei verschiedene Instantiierungen einer allquantifizierten Formel im Beweis verwendet werden. Wie weit diese technischen Beschränkungen den naturgemäß etwas vagen Begriff der "adäquaten Formalisierung" einfangen und ob hier ggf. weitere Einschränkungen oder auch Erweiterungen nötig sind, muss die praktische Erfahrung im Einsatz des Systems zeigen. Ein rein technischer Grund, die Möglichkeiten des ATP zu beschränken, ist der, die Laufzeiten für die Prüfung gering zu halten (und insbesondere, angesichts der prinzipiellen Unentscheidbarkeit der erststufigen Prädikatenlogik, endlich).

Mathediktate lassen sich natürlich auch für andere Themengebiete implementieren; besonders geeignet sind dabei solche Bereiche mit angenehmen modelltheoretischen Eigenschaften, die eine einfache Entscheidbarkeit von Formeln zur Folge haben, wie etwa o-minimale Theorien oder solche, die – wie etwa die Presburger Arithmetik – eine effektive Quantorenelimination erlauben, siehe etwa Marker [145], S. 81f.

Aussagenlogik-Diktate Die Aussagenlogik-Diktate dienen der Übung des Umgangs mit den aussagenlogischen Junktoren $\neg$, $\wedge$, $\vee$, $\rightarrow$ und $\leftrightarrow$. Bei einem aussagenlogischen Diktat sind einige Aussagen in natürlicher Sprache gegeben, die durch aussagenlogische Variablen repräsentiert werden. Aufgabe der/des Benutzers/In ist es dann, eine komplexe natürlichsprachliche Aussage unter Verwendung der aussagenlogischen Junktoren durch diese aussagenlogischen Variablen auszudrücken.

Eine typische Aussagenlogik-Diktataufgabe ist etwa folgende:

> Es bezeichne s die Aussage "Peter geht an den Strand", m die Aussage "Peter trägt einen Mantel" und h die Aussage "Peter hat einen Hut auf". Druecke die Aussage "Geht Peter an den Stand, so hat er einen Mantel an, aber keinen Hut auf" in der Sprache der Aussagenlogik aus.

Die Syntax für die Eingabe bei den aussagenlogischen Diktaten ist folgende:

- Kleine lateinische Buchstaben werden als aussagenlogische Variablen verwendet; aussagenlogische Variablen sind gültige Eingaben.
- Sind ϕ und ψ gültige Eingaben, so auch $(\phi\&\psi)$, $(\phi \vee \psi)$, $(\phi \rightarrow \psi)$, $(\phi \leftrightarrow \psi)$ und $\sim \phi$.

Vom System erfolgt eine Rückmeldung, ob die eingegebene Aussage eine korrekte Formalisierung der gegebenen natürlichsprachlichen Aussage darstellt. Tut sie es nicht, wird zudem eine Belegung der auftretenden Aussagevariablen mit Wahrheitswerten angegeben, unter der sich die eingegebene von der richtigen Lösung unterscheidet. Andere Kriterien (etwa Einfachheit, Natürlichkeit oder Kürze) werden derzeit nicht geprüft.

Die Idee der Automatisierung aussagenlogischer Diktate findet sich an verschiedenen Stellen, u.a. auch im Edukera-System. Die unseres Wissens nach erste Verwendung dieser Idee erfolgte im "Mathematical Logic Tutor" von A. Moreno, siehe [25]. Für Diproche wurde ein "Formalisierungskurs Aussagenlogik" entwickelt, der durch eine Serie von 27 Aufgaben schrittweise in die

Verwendung aussagenlogischer Junktoren zur Übersetzung natürlichsprachlicher Zusammenhänge einführt.[21]

Mengendiktate Analog zu den aussagenlogischen Diktaten dienen die "Mengendiktate" der Übung im Umgang mit mengentheoretischen Grundoperationen und -relationen. Es werden einige Mengen durch natürlichsprachliche Ausdrücke gegeben und durch Variablen bezeichnet; die Aufgabe der/des Benutzers/In ist es dann, einen ebenfalls in natürlicher Sprache formulierten Zusammenhang dieser Mengen mit den gegebenen mengentheoretischen Symbolen und Variablen auszudrücken.

Eine typische Aufgabe für die Mengendiktate ist etwa:

> Es bezeichne b die Menge der Bäume, h die Menge der Häuser, g die Menge der grünen Dinge. Drücke "Einige Bäume sind keine grünen Häuser" in der Sprache der Mengenlehre aus.

Die Syntax für die Eingabe bei den Mengendiktaten ist durch folgende Regeln gegeben:

- $\emptyset$ ist ein Mengenterm.
- Kleine lateinische Buchstaben werden als Mengenvariablen verwendet; Mengenvariablen sind Mengenterme.
- Sind a und b Mengenterme, so auch $(-a)$ (das Komplement von a), $(a \cap b)$, $(a \cup b)$ und (a/b) (das relative Komplement von a in b)[22].
- Sind a und b Mengenterme, so sind $(a = b)$, $(a \subset b)$ und $(a \supset b)$ gültige Eingaben.[23].
- Sind ϕ und ψ gültige Eingaben, so auch $(\phi \& \psi)$, $(\phi \vee \psi)$, $(\phi \rightarrow \psi)$, $(\phi \leftrightarrow \psi)$ und $\sim \phi$.

Die Rückmeldung besteht, wie bei den aussagenlogischen Diktaten, in der Mitteilung, ob die eingegebene Formel korrekt ist. Als Antworten auf die oben gegebene Aufgabe würden etwa $\sim (b \subset (h \cap g))$ oder $\sim (b \cap (-(g \cap h)) = \emptyset)$ akzeptiert. Im Fall einer falschen Eingabe wird überdies eine Charakterisierung für

[21] Dieser Kurs ist online verfügbar unter `https://eufmath.uni-flensburg.de/FormalisierungskursAussagenlogik.html` (zugegriffen 14.10.2021).

[22] Die Verwendung von "/" statt des üblichen \ hat technische Gründe. Es ist geplant, diese sicher etwas irritierende Notation zu ändern.

[23] Der Einfachheit halber können äußere Klammern auch fortgelassen werden.

Objekte angegeben, für die die Eingabe und die in der Aufgabenstellung gegebene Aussage unterschiedliche Wahrheitswerte liefern. Würde man etwa als Antwort auf die obige Beispielaufgabe $(g \cap h) = b$ eingeben, erhielte man die Rückmeldung, dass das für Objekte, die in b, in g und in h liegen, nicht funktioniert.

Auch für dieses Übungsformat wurde ein "Formalisierungskurs Mengenlehre" entwickelt, der aus 25 Aufgaben besteht und systematisch in die Verwendung mengentheoretischer Symbolik einführen soll.[24]

Game of Def

Die Rückmeldungen der Mathediktat-Module sind für eine Verbesserung einer fehlerhaften Lösung unter Umständen nur bedingt hilfreich: zwar erfährt man, dass die Eingabe z.B. keine notwendige Bedingung für die fragliche Aussage ist; für eine Korrektur wäre es aber darüber hinaus hilfreich, auch ein konkretes Gegenbeispiel zu kennen. Obwohl es prinzipiell möglich sein dürfte, solche Beispiele für den von den Mathediktaten derzeit abgedeckten Gegenstandsbereich automatisch zu generieren, wäre das jedenfalls eine anspruchsvolle Aufgabe. Zudem sind reelle Funktionen relativ abstrakte Objekte, die gerade für AnfängerInnen, die häufig sowohl mit dem Funktionsbegriff als auch mit grundlegenden Anordnungseigenschaften der reellen Zahlen (Unbeschränktheit, Homogenität, Dichtheit) Schwierigkeiten haben, eine Herausforderung darstellen, die von dem eigentlichen Ziel – dem Einüben des Formalisierens – ablenken mögen.

Diesen Schwierigkeiten begegnet das "Game of Def": Hier wird der/dem BenutzerIn ein quadratisches 21×21-Feld gezeigt, von dem einige Teilquadrate gelb gefärbt sind. Die Aufgabe besteht dann darin, eine Formel anzugeben, die genau auf die gefärbten Quadrate zutrifft. Als Grundrelationen stehen dabei zur Verfügung:

- "ueber(x, y)" bzw. "unter(x, y)", was genau dann erfüllt ist, wenn das Quadrat y in derselben Spalte und echt oberhalb bzw. unterhalb von x liegt.
- "rechts(x, y)" bzw. "links(x, y)", was genau dann erfüllt ist, wenn das Quadrat y in derselben Zeile echt rechts bzw. links von x liegt.
- "nachbar(x, y)", was genau dann zutrifft, wenn die Quadrate x und y genau eine gemeinsame Kante besitzen.
- "dist(x, y)=dist(a, b)", was genau dann zutrifft, wenn die Quadrate x und y in derselben Zeile bzw. Spalte liegen, die Quadrate a und b in derselben

[24] Dieser Kurs ist unter `https://eufmath.uni-flensburg.de/FormalisierungskursMengenlehre.html` online verfügbar (zugegriffen 14.10.2021).

Zeile bzw. Spalte liegen und der Abstand von x zu y gleich dem Abstand von a zu b ist.

- "ed(a, b)=ed(x, y), was genau dann zutrifft, wenn die Mittelpunkte der Quadrate a und b denselben euklidischen Abstand haben wie die Mittelpunkte der Quadrate x und y.

- "taxi(a, b)=taxi(x, y), was genau dann zutrifft, wenn der Taxi-Abstand der Quadrate a und b – also die Summe der Beträge der Differenzen ihrer x-Koordinaten und ihrer y-Koordinaten – gleich dem Taxi-Abstand der Quadrate x und y ist.

Formal akzeptiert das "Game of Def" folgende Ausdrücke als wohlgeformte Formeln:[25]

- Kleine lateinische Buchstaben sind Variablen bzw. Konstantenzeichen.

- Sind a, b, x, y Variablen oder Konstantenzeichen, so sind rechts(a,b), links(a,b), ueber(a,b), unter(a,b), nachbar(a,b), dist(a,b)=dist(x,y), ed(a,b)=ed(x,y) und taxi(a,b)=taxi(x,y) Formeln.

- Sind ϕ und ψ Formeln, so auch $(\phi \& \psi)$, $(\phi \vee \psi)$, $(\phi \rightarrow \psi)$, $(\phi \leftrightarrow \psi)$ and $\sim \phi$.

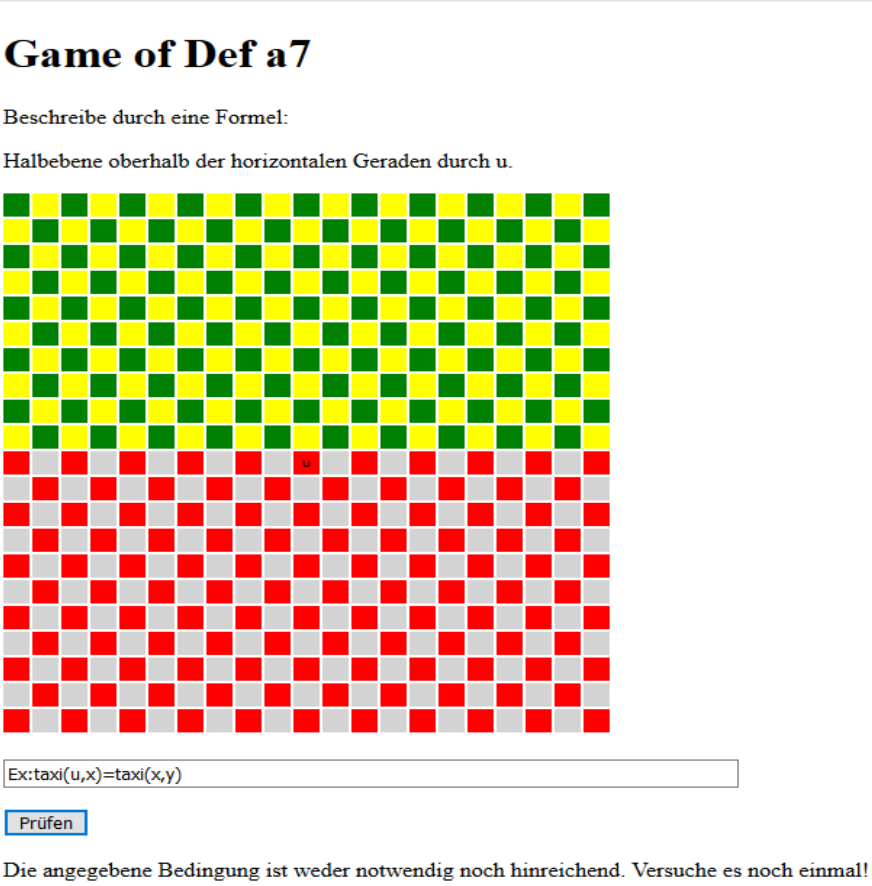
- Ist ϕ eine Formel und x ein kleiner lateinischer Buchstabe, so sind auch $Ex : \phi$ und $Ax : \phi$ Formeln.

Junktoren und Quantoren funktionieren in ihrer üblichen Bedeutung, wobei zu beachten ist, dass die Quantoren sich auf die Quadrate im Gitter beziehen, nicht auf die gesamte Ebene; entsprechend gibt es Quadrate ohne rechten Nachbarn etc. Bei der derzeitigen Auswertungsroutine ist überdies zu beachten, dass die Verarbeitung einer Formel mit mehr als zwei geschachtelten Quantoren zu lange braucht und solche Formeln daher vermieden werden müssen.

Damit lassen sich dann Aufgaben stellen, die zunächst die Wirkung der logischen Junktoren, im weiteren dann auch der Quantoren sowie deren Interaktion mit den Junktoren oder anderen Quantoren einführen. Jede Aufgabe besteht dabei aus der Anzeige eines 21×21-Feldes, in dem die Elemente der fraglichen Menge gelb markiert sind, sowie einer sprachlichen Beschreibung dieser Menge. Optional können auch gewisse Felder mit Konstantensymbolen markiert sein, die dann in der Beschreibung als Parameter verwendet werden können; stets ist das mittlere Feld mit "u" markiert.

[25] Die folgenden Erläuterungen sind eine deutsche Version von Teilen der Arbeit "Evaluation of Formalization Exercises in Mathematics" [33].

Die Aufgabe der/des BenutzerIn besteht nun darin, einen String in ein Textfeld einzugeben und auf “Check” zu klicken. Falls die Eingabe keine wohlgeformte Formel ist oder keine Menge von Quadraten beschreibt (also nicht genau eine freie Variable enthält), wird eine entsprechende Fehlermeldung ausgegeben. Andernfalls werden alle Quadrate, auf die die eingegebene Formel zutrifft und die in der fraglichen Menge liegen, grün markiert, alle Quadrate, auf die die eingegebene Formel zutrifft, die aber nicht in der fraglichen Menge liegen, rot und alle Quadrate, die zwar zur fraglichen Menge gehören, auf die die eingegebene Formel aber nicht zutrifft, bleiben gelb. Zusätzlich erfolgt eine Rückmeldung in Textform, ob die eingegebene Formel eine korrekte Charakterisierung der fraglichen Menge darstellt oder lediglich eine notwendige bzw. eine hinreichende Bedingung für das Enthaltensein in dieser Menge oder keines von beidem. Das folgende Bild zeigt ein Beispiel für eine Rückmeldung des “Game of Def”.



Einige Beispielaufgaben, die mit dem “Game of Def” bearbeitet werden können, finden sich in Anhang B.

Neben dem Übergang vom natürlichsprachlichen Ausdruck bzw. einer intuitiven Vorstellung zur quantorenlogischen Formel ist oft auch der umgekehrte

Vorgang, also das Verstehen formaler Ausdrücke, eine Schwierigkeit. Dies ließe sich durch eine – derzeit aber noch nicht implementierte – "inverse" Variante des "Game of Def" fördern, die wir "Malen nach Formeln" nennen wollen: Gegeben ist in diesem Fall nicht eine Menge markierter Felder, zu denen ein logischer Ausdruck anzugeben ist, sondern ein logischer Ausdruck, wobei die Aufgabe nun darin besteht, alle Felder zu markieren, auf die er zutrifft. Wir planen, diese Variante in einer späteren Version verfügbar zu machen.

Sowohl für das "Game of Def" als auch für "Malen nach Formeln" sollte eine automatische Aufgabenerzeugung möglich sein, einfach dadurch, dass ein (hinreichend kurzer) logischer Ausdruck generiert und die zugehörigen Felder ermittelt werden. Hier kommt es nur darauf an, triviale (tautologische oder kontradiktorische) Ausdrücke zu vermeiden. Auch das ist für spätere Versionen geplant.

5.7.1 Beweispuzzles

Bei einem "Beweispuzzle" geht es darum, einen bereits vorhandenen, aber in sinntragende Bestandteile zerlegten Beweis wieder in die bzw. eine korrekte Reihenfolge zu bringen. Dieser Aufgabentyp wurde z.B. in den die Analysis- und Geometrievorlesungen im Frühjahrssemester 2020 begleitenden Übungen an der EUF regelmäßig eingesetzt und war auch Bestandteil der Klausur. Das Konzept digitaler Beweispuzzles ist z.B. auch im E-Proofs System realisiert, das wir oben besprochen haben.

Die Implementierung von Beweispuzzles in Diproche kann als konzeptionelle Ausdifferenzierung und Verfeinerung der E-Proofs angesehen werden. Es besteht aus zwei separaten Programmen, von denen das erste eine einfache Darstellung der Aufgabe bei einer weniger differenzierten Rückmeldung anbietet und das zweite eine deutlich detailliertere Rückmeldung an die BenutzerInnen ermöglicht, allerdings um den Preis, dass die Eingabe neuer Aufgaben deutlich aufwändiger wird.

Einfache Beweispuzzles

Bei einfachen Beweispuzzles ist eine Aufgabe ein 5-Tupel (Id,Text,Digraph,Blocks,Order), wobei Id eine Kennung ist und Text der Aufgabentext als Liste der anzuordnenden Textteile. Angenommen, dieser Text besteht aus n Teilen. Dann ist "Digraph" ein gerichteter Graph auf der Menge $\{1, 2, ..., n\}$, der die Abhängigkeiten der Zeilen voneinander codiert; die Kante (i, j) gehört also dann zum Digraphen, wenn in jeder richtigen Lösung die i-te Zeile vor der j-ten Zeile kommen muss. Dabei wird nicht differenziert, welche Art von Fehler sonst vorläge. Auf diese Weise lassen sich also u.a. sowohl

deduktive Beziehungen "C folgt aus A und B, muss also danach kommen" wie auch die Typenkorrektheit ("A muss vor B kommen, weil B Variablen verwendet, die in A deklariert werden") codieren. Allerdings ist das für viele Fälle noch nicht ausreichend, etwa dann nämlich, wenn ein Beweis mehrere unabhängige Teilbeweise umfasst (wie bei einem Äquivalenzbeweis oder einer Fallunterscheidung). Betrachten wir als Beispiel ein Beweispuzzle zum Beweisziel, dass $A \to (B \to A)$ eine Tautologie ist.

1. Fall 1: Es gelte A.
2. Dann folgt $B \to A$.
3. Also gilt $A \to (B \to A)$, weil die Konklusion wahr ist.
4. Fall 2: Nun gelte $\neg A$.
5. Dann folgt sofort $A \to (B \to A)$, weil die Prämisse falsch ist.
6. Folglich gilt $A \to (B \to A)$.

Der Digraph enthält hier also die Kanten $(1, 2)$, $(2, 3)$, $(4, 5)$, $(2, 6)$ und $(5, 6)$. Allein aufgrund dieser Angabe wäre aber etwa auch folgende Anordnung eine korrekte Lösung:

1. Fall 1: Es gelte A.
2. Fall 2: Nun gelte $\neg A$.
3. Dann folgt $B \to A$.
4. Also gilt $A \to (B \to A)$, weil die Konklusion wahr ist.
5. Dann folgt sofort $A \to (B \to A)$, weil die Prämisse falsch ist.
6. Folglich gilt $A \to (B \to A)$.

Diese Anordnung ist jedoch fehlerhaft: Ab Zeile 3 werden sowohl A als auch $\neg A$ angenommen, die folgenden Zeilen werden also unter widersprüchlichen Annahmen abgeleitet. Damit ist das Beweisziel $A \to (B \to A)$ aber nicht erreicht.

Um dieser Schwierigkeit zu begegnen, gibt es als zusätzliche Spezifikation die "Blocks", eine Liste $[[Z_1^1, ..., Z_{n(1)}^1], [Z_1^2, ..., Z_{n(2)}^2], ..., [Z_1^k, ..., Z_{n(k)}^k]]$ von Listen von Zeilen. Die dadurch codierte Anforderung an eine Anordnung ist, dass für jede dieser Zeilenlisten die darin enthaltenen Zeilen direkt aufeinander folgen müssen. Im vorliegenden Fall wären die Blocks $[1, 2, 3]$ sowie $[4, 5]$, womit die oben angegebene Anordnung ausgeschlossen wäre.

"Order" schließlich ist eine Permutation π auf $\{1, 2, ..., n\}$, die angibt, in welcher Reihenfolge die Textbestandteile dem/der BenutzerIn angezeigt werden.

Eine Lösung besteht in diesem Fall also aus einer weiteren Permutation der Zahlen $\{1, 2, ..., n\}$, die über ein Interface eingegeben werden kann. Klickt der/die BenutzerIn auf den Check-Button der betreffenden Aufgabe, erhält sie/er zum einen die Rückmeldung, ob die Eingabe formal korrekt war (also eine Permutation von $\{1, 2, ..., n\}$) und, falls ja, über die Anzahl der in der falschen Reihenfolge eingegebenen Zeilenpaare sowie nicht berücksichtigen Blöcke. Zusätzlich wird als Hilfe zur Korrektur der Text in der eingegebenen Reihenfolge angezeigt.

Für viele Fälle ist diese Darstellung ausreichend. Allerdings hat sie einige Mängel:

- Die reine Meldung der Anzahl von Fehlstellungen bzw. Blockverletzungen hat wenig Bezug zum logischen Gehalt des Textes; es ist nicht zu erkennen, welche Art von Fehler vorliegt.

- Das Phänomen, dass mehrere Textteile austauschbar sind, etwa wenn es mehrere Teilbeweise gibt und daher mehrere "qed" als Textteile verfügbar sind, läßt sich nicht abbilden.

- Ebenso wenig sind Textabschnitte zu erfassen, die in einer Reihenfolge oder deren Umkehrung kommen müssen, etwa bei Äquivalenzumformungen oder bei Gleichungsketten.

- Ferner zwingt logische Folgerbarkeit nicht notwendigerweise dazu, ein gewisses Zeilenpaar auf eine bestimmte Weise anzuordnen; so könnte z.B. eine Aussage ϕ sowohl aus den Elementen einer Aussagenmenge Φ_1 wie auch denen einer anderen, zu Φ_1 disjunkten, Aussagenmenge Φ_2 folgen; die Forderung aber, dass alle Elemente von Φ_1 **oder** alle Elemente von Φ_2 im Text vor ϕ kommen müssen, ist wiederum in dem gegebenen Rahmen nicht abzubilden.

- Die Beschränkung auf Permutationen der vorgegebenen Zeilen ist restriktiv; es wäre vorteilhaft, auch nicht zum Beweistext gehörige Textbestandteile als "Distraktoren" einzufügen, wie es etwa im E-Proofs-System möglich ist.

Diese Mängel sind in der Umgebung für differenzierte Beweispuzzles behoben. Im Vergleich beider Umgebungen ist allerdings zu beachten, dass die einfachen Beweispuzzles für viele Aufgaben ausreichen, in denen die genannten Schwierigkeiten nicht auftreten und die erforderliche Analyse zur Erfassung eines Textes im Rahmen eines differenzierten Beweispuzzles deutlich höher ist. Ob der Zugewinn an Ausdruckskraft diesen erhöhten Aufwand rechtfertigt oder ob

der Übungseffekt von Beweispuzzles sich mit Aufgaben, die im Rahmen einfacher Beweispuzzles darstellbar sind, bereits vollständig abgeschöpft werden kann, muss die Erfahrung im Einsatz zeigen.

Differenzierte Beweispuzzles

Eine Aufgabe zu den differenzierten Beweispuzzles ist ein 11-Tupel

(Id,Text,VssLogik,BlocksLogik,VssTypes,VssStil,BlocksStil,Ketten,Equiv,Zielindex,Order).

Dabei sind Id und Text wie bei den einfachen Beweispuzzles die interne Kennung der Aufgabe sowie der anzuzeigende Text als Liste von Textteilen; ebenso ist Order die Reihenfolge, in der die Textteile angezeigt werden sollen; allerdings handelt es sich bei Order nun nicht mehr um eine Permutation aller Zeilenindizes, sondern lediglich einer Teilmenge davon – auf diese Weise wird also der Einsatz von Distraktoren möglich.

VssLogik ist eine Liste, die zu jedem Zeilenindex I eine Liste

$$[[I^1_1, ..., I^1_{n(1)}], [I^2_1, ..., I^2_{n(2)}], ..., [I^k_1, ..., I^k_{n(k)}]]$$

von Listen von Zeilenindizes enthält; diese Listen sind sämtliche minimalen Mengen von Textbestandteilen, aus denen der Textbestandteil I logisch folgerbar ist. In einer logisch korrekten Lösung müssen also alle Elemente von mindestens einer dieser Listen dem Textbestandteil mit dem Index I vorangehen.

BlocksLogik hat die gleiche Syntax, Bedeutung und Funktion wie "Blocks" bei den einfachen Beweispuzzles.

VssTypes funktioniert analog zu VssLogik, jedoch in Bezug auf die Deklaration von Variablen, gibt also eine Übersicht darüber, welche (Mengen von) Zeilen einer Zeile vorausgehen müssen, damit alle darin vorkommenden Variablen so deklariert sind, dass man sie auf die dort verwendete Weise benutzen kann. Zur Einübung korrekter Typenverwendung wäre es so etwa möglich, verschiedene Deklarationen einer Variablen anzubieten, von denen aber nicht alle sinnvoll sind, etwa A und B einmal als Mengen und einmal als Zahlen zu deklarieren und dann den Ausdruck $A + B$ zu verwenden.

VssStil funktioniert ebenfalls analog zu VssLogik, jedoch in Bezug auf Zeilen, die einer gegebenen Zeile im Sinne der Lesbarkeit vorangehen sollten. (So sollte z.B. das Beweisziel am Ende eines Beweises stehen, auch wenn noch Schritte verfügbar sind, die darauf logisch korrekt folgen könnten.)

BlocksStil funktioniert analog zu BlocksLogik, listet aber Zeilen auf, die stilistisch gesehen aufeinander folgen sollten; werden etwa in einem Beweis zunächst A und B separat gezeigt und aus beidem zusammen dann C gefolgert,

wäre es sinnvoll, die Beweise von A und B separat zu führen statt die zugehörigen Textbestandteile durcheinander zu mischen – auch dort, wo das, logisch gesehen, durchaus möglich wäre.

Ketten ist eine Liste von Listen von Zeilenindizes, die in einer Lösung entweder in der angegebenen oder in der genau umgekehrten Reihenfolge auftreten müssen und dient der Erfassung der oben angesprochenen Möglichkeit, etwa eine Gleichungs- oder Äquivalenzumformungskette in beide Richtungen aufschreiben zu können.

Equiv ist eine Liste von Listen von Zeilenindizes, die angibt, welche Zeilen austauschbar verwendet werden können; diese werden dann in den Prüfungen aller bisher aufgeführten Punkte als gleichwertig behandelt (wenn z.B. 1 und 2 laut BlocksLogik in einem Block kommen sollen und 2 als gleichwertig zu 3 gilt, so gilt diese Bedingung als erfüllt, wenn 1 und 3 aufeinanderfolgen).

Zielindex ist die Nummer des Textbestandteils, der das Beweisziel und den Beweisabschluss enthält. In jeder korrekten Lösung muss dieser Index vorkommen. Auf diese Weise wird vermieden, dass – stilistisch, logisch und in Bezug auf die Typenverwendung einwandfreie – Anordnungen als korrekt gewertet werden, die aber nicht auf das gegebene Beweisziel herauslaufen.

5.7.2 "Markup"

"Markup" ist ein den Beweispuzzles ähnliches Übungsprogramm, das die strukturierte Erfassung bereits vorliegender Beweistexte fördern soll. In jeder Aufgabe wird dem oder der BenutzerIn ein korrekter Beweis angezeigt, zusammen mit einer Liste von Erläuterungen der Beweisbestandteile (wie "Fallannahme", "Folgerung aus Zeile 3 und 7", "Erklärung des Beweiszieles" etc.); die Aufgabe besteht nun darin, jeder Zeile eine korrekte Erläuterung zuzuweisen. Dabei können auch Erläuterungen in der Liste enthalten sein, die zu keiner der gegebenen Textbestandteile passen.

Formal ist eine Aufgabe ein 6-Tupel

(Id,Text,ItemListe,Loesung,Equiv,DisplayOrder),

wobei Id die Kennzeichnung der Aufgabe ist, Text der gegebene Beweis als Liste von Textbestandteilen, ItemListe die Liste der verfügbaren Erläuterungen (ebenfalls als Liste von Texten), Lösung eine Liste von Indizes von Erläuterungen in der richtigen Reihenfolge, Equiv eine Liste von Erläuterungen, die untereinander austauschbar sind und DisplayOrder die Reihenfolge, in der die Erläuterungen angezeigt werden.

Die Rückmeldung besteht in den Indizes der fehlerhaft markierten Zeilen und deren Anzahl.

5.7.3 Das "Korrekturspiel"

Beim Korrekturspiel geht es um die kritische Aufmerksamkeit auf einen gegebenen Beweis. Insofern das mathematische Beweisen ein ständiger Kreislauf aus der Konstruktion von Beweisen, kritischer Prüfung und verbesserter Konstruktion ist, gehört auch diese zu den Kompetenzen, die im Rahmen einer Unterstützung des Beweisenlernens gefördert werden sollten. Empirische Untersuchungen zeigen, dass Studierende im Anfängerbereich mit der Bewertung von Beweisen große Schwierigkeiten haben, siehe etwa Selden [179] oder Inglis und Alcock [115].

Wie schon bei den Beweispuzzles oder bei "Markup" werden werden der/dem BenutzerIn im Korrekturspiel Beweistexte angezeigt, und zwar als durchnummerierte Folge von Textteilen. Diese Beweistexte sind aber sowohl bezüglich der logischen Folgerbarkeit als auch bezüglich der Verwendung von Typen mitunter fehlerhaft, und die Aufgabe besteht nun darin, separat die logisch nicht schlüssigen und die im Hinblick auf die Typenverwendung fehlerhaften Zeilen herauszufinden und einzugeben. Als Rückmeldung werden die fehlerhaft markierten Zeilen und deren Anzahl, aufgeschlüsselt nach logischen Fehlern und Typenfehlern, ausgegeben.

Offenbar gibt es beim "Korrekturspiel" zwei mögliche Arten von Fehlern: Einmal kann eine korrekte Zeile als falsch markiert sein, andererseits eine fehlerhafte als korrekt. Zwischen diesen beiden Fehlertypen wird zwar intern unterschieden, nicht aber in der Rückmeldung, um Lösungsversuche durch bloßes Ausprobieren und Raten zu erschweren.

5.7.4 "Diagnose"

"Diagnose" ist eine Verfeinerung des "Korrekturspiels", die auf eine Anregung von Regula Krapf zurückgeht. Anstatt lediglich die fehlerhaften Zeilen als solche zu markieren, besteht hier die Aufgabe darin, die fehlerhaften Zeilen – wo das möglich ist – als Instanzen eines Typs von formalem Fehlschluss oder Umformungsfehler zu erkennen. Dabei steht eine Liste von solchen Fehlern zur Verfügung, in der jedem Fehler eine Kennung zugeordnet ist; die Aufgabe besteht dann in einer korrekten Zuordnung der Kennungen zu den Beweiszeilen. In gewisser Weise spielt also hier der oder die BenutzerIn eine ähnliche Rolle wie der Anti-ATP in Diproche. Wir hoffen, dass diese Erweiterung gegenüber dem Korrekturspiel dazu beitragen kann, die Aufmerksamkeit für gewisse Arten von Fehlern zu schärfen, was dann weiter zu deren Vermeidung beitragen sollte. Überdies dürfte diese Art von Übung gerade für Lehramtsstudierende eine hilfreiche Vorbereitung dafür sein, Fehler nicht lediglich als solche zu erkennen, sondern auch ein Gespür für die möglicherweise zugrundeliegenden (Fehl-)Vorstellungen zu entwickeln. Derzeit steht "Diagnose" zwar als Übungsformat zur Verfügung, ist aber hinsichtlich

der Anzahl verfügbarer Aufgaben und der diagnostizierbaren Fehlschlüsse noch stark eingeschränkt. Wir planen, das Konzept in späteren Versionen deutlich zu erweitern.

5.7.5 "Vereinfachen"

Mit "Vereinfachen" stellt das Diproche-System ein Werkzeug zur Verfügung, mit dem der Umgang sowohl mit aussagenlogischen Formeln wie auch mit mengentheoretischen Termen und Formeln sowie mit arithmetischen Termen geübt werden kann. Die Lehrerfahrung zeigt, dass besonders letzteres zu Unrecht als aus der Schule hinlänglich bekannt vorausgesetzt wird und den Studierenden häufig große Schwierigkeiten bereitet, mitunter bis in das Masterstudium. Nun sind zu diesem Thema bereits zahlreiche digitale Angebote verfügbar, etwa die "Online-Übung"[26], bei "Gut Erklärt"[27] oder vom Studienkreis[28] oder, in besonders bemerkenswerter Ausführung, u.a. unter Verwendung einer automatischen Handschriftenerkennung, "Correct"[29]. Im Gegensatz zu vielen dieser Angebote ist "Vereinfachen" nicht Multiple-Choice-basiert, sondern kann beliebige syntaktisch korrekte Eingaben überprüfen; aber auch in dieser Hinsicht reicht es nicht über die Möglichkeiten professioneller Algebra-Systeme wie Mathematica hinaus oder auch nur heran. Das Ziel von "Vereinfachen" besteht lediglich darin, BenutzerInnen von Diproche unkompliziert eine Übungsmöglichkeit zu diesen Themen zur Verfügung zu stellen.

Beim aussagenlogischen Teilsystem von "Vereinfachen" wird dem/der BenutzerIn ein aussagenlogischer Ausdruck angezeigt und die Aufgabe besteht nun darin, einen logisch äquivalenten, aber möglichst kurzen Ausdruck anzugeben. Als Rückmeldung wird angezeigt, ob der eingegebene Ausdruck eine wohlgeformte Formel darstellt und falls ja, ob er zum angegebenen Ausdruck äquivalent ist und, falls auch das der Fall ist, ob er so kurz wie möglich ist, wobei die Länge eines Ausdrucks sich hier an der Zahl der in ihm auftretenden Junktoren bemißt.

Die Syntax ist dabei die folgende:

- Lateinische Buchstaben werden als Aussagevariablen verwendet und sind wohlgeformte Formeln.
- Sind ϕ und ψ wohlgeformte Formeln, so auch $\sim \phi$, $(\phi \& \psi)$, $(\phi \vee \phi)$, $(\phi \to \psi)$ und $(\phi \leftrightarrow \psi)$.

[26] `https://onlineuebung.de/mathe/termumformung/` (zugegriffen 14.10.2021)

[27] `https://www.gut-erklaert.de/mathematik/termumformung-aufgaben-uebungen.html` (zugegriffen 14.10.2021)

[28] `https://www.studienkreis.de/mathematik/termumformungen-klammern/` (zugegriffen 14.10.2021)

[29] `https://digitallearninglab.de/tools/correct` (zugegriffen 14.10.2021)

Insbesondere ist die Klammersetzung strikt zu beachten, auch äußere Klammern sind erforderlich. Die Überprüfung der Korrektheit erfolgt mithilfe eines aussagenlogischen Tableau-Beweisers (siehe etwa Fitting [70]).

In den mengentheoretischen Termvereinfachungsaufgaben ist ein mengentheoretischer Term – etwa $(A \cup \bar{A}) \cap B)$ – gegeben und die Aufgabe besteht wiederum darin, einen möglichst kurzen Term anzugeben, der die gleiche Menge repräsentiert (im genannten Beispiel: B). Die Rückmeldung ist dieselbe wie für die aussagenlogischen Vereinfachungsaufgaben.

Bei Vereinfachungsaufgaben für mengentheoretische Formeln geht es entsprechend darum, eine mengentheoretische Formel – also eine Boolesche Kombination aus Gleichheits- und Teilmengenbeziehungen zwischen mengentheoretischen Termen – zu einer – hinsichtlich der Anzahl der auftretenden mengentheoretischen Operatoren und logischen Junktoren – möglichst kurzen äquivalenten Formel umzuformen. Eine einfache Aufgabe dieser Art wäre etwa, $(A \subseteq B) \wedge (B \subseteq A)$ durch $A = B$ auszudrücken. Auch hier sind die Rückmeldungen bezüglich Wohlgeformtheit der Eingabe sowie Korrektheit und Optimalität der Eingabe die gleichen wie bei aussagenlogischen Formeln.

Im arithmetischen Teil hingegen wird ein arithmetischer Term vorgelegt, ein Ausdruck also, der aus natürlichen Zahlen (in Dezimaldarstellung) und Variablen (lateinischen Buchstaben) durch iterierte Anwendung von Addition, Subtraktion, Multiplikation, Division und Exponentiation entsteht; wiederum besteht die Aufgabe der/des Benutzers/In darin, einen möglichst kurzen Ausdruck anzugeben, der zum angegebenen Ausdruck für jede Instantiierung der auftretenden Variablen wertgleich ist; die Länge eines Ausdrucks wird dabei als Anzahl der auftretenden Operationszeichen erfaßt. Die Prüfung auf Gleichheit der Ausdrücke erfolgt, wie die Überprüfung von Termumformungen beim Beweisprüfen, probabilitisch: Die Variablen beider Terme werden mehrfach mit Zufallszahlen instantiiert, die Werte der Terme werden berechnet und verglichen. Ergibt sich – bis auf Rundungsfehler, die der Ungenauigkeit der Maschinenarithmetik zugerechnet werden können – dabei stets das gleiche Ergebnis, werden die Terme als gleichwertig akzeptiert, sonst nicht.[30]

[30]Dieses Verfahren hat sich praktisch als zuverlässig erwiesen; die theoretische Fundierung hinter der Zuverlässigkeit liegt darin, dass die Nullstellenmenge eines Ausdrucks der hier relevanten Form entweder eine Nullmenge ist oder eine Menge von Maß 1, die Auswertung an einer "zufälligen" Stelle also im Hinblick auf die Gleichwertigkeit von Termen mit einer Wahrscheinlichkeit von 1 das richtige Ergebnis liefert. Eine empirische Verifikation war freilich trotzdem erforderlich, da das Rechnen mit reellen Zahlen durch die begrenzte Genauigkeit der Maschinenarithmetik nur unzureichend eingefangen wird.

5.7.6 "Reformulate"

Ein im Rahmen von Übungsaufgaben zur Aussagenlogik häufig auftretender Typ von Übung besteht darin, gegebene Ausdrücke (möglichst kurz) mithilfe eines beschränkten Vorrats an Junktoren äquivalent zu reformulieren. Lösungen zu solchen Übungen können im Rahmen von "Reformulate"[31] überprüft werden, wobei im wesentlichen dieselben Techniken wie bei "Vereinfachen" verwendet werden; zusätzlich wird erfasst, ob der eingegebene Ausdruck tatsächlich nur die in der jeweiligen Aufgabe zugelassenen Junktoren verwendet und ggf. eine entsprechende Meldung ausgegeben. Außerdem stehen neben den "üblichen" Junktoren, also $\neg$, $\&$, $\vee$, $\rightarrow$ und $\leftrightarrow$ auch die etwas selteneren XOR (ein exklusives "Oder", repräsentiert durch $*$) sowie NAND (ein negiertes "Und", repräsentiert durch $+$) zur Verfügung, um eine größere Abwechslung bei den Übungen zu ermöglichen. Ein besonderes Interesse des NAND liegt dabei darin, dass sämtliche anderen Junktoren, einschließlich der Negation, hierdurch ausgedrückt werden können. Das XOR hingegen wurde aufgenommen, um die Abgrenzung gegen die gewöhnliche, inklusive Disjunktion ("$\vee$") deutlicher zu machen.

5.7.7 "DefCheck"

Eine Schwierigkeit, die bei Studierenden im Anfängerbereich immer wieder zu beobachten ist, besteht im Umgang mit Begriffen, die durch Definitionen gegeben sind. Gerade (aber nicht nur) beim Beweisen spielt diese Fähigkeit oft eine Schlüsselrolle (so hängen z.B. viele der oben behandelten Beispiele zur elementaren Zahlentheorie davon ab, von der Annahme, eine Zahl x sei gerade, überzugehen zur Darstellung von x in der Form $2k$). (Zum Umgang mit Definitionen im mathematischen Anfängerbereich siehe z.B. die Studie von Zaslavsky und Shir, [215].) DefCheck dient dazu, den Umgang mit Definitionen zu üben. Dazu wird eine Eigenschaft natürlicher Zahlen generiert, indem aus einer Liste von Eigenschaften natürlicher Zahlen (wie "ist im Dezimalsystem ein Palindrom", "enthält die Ziffer 1" oder "ist durch 3 teilbar") drei Eigenschaften zufällig ausgewählt werden; diese neue Eigenschaft wird dann mit einem automatisch generierten Zufalls"wort" benannt und dem/der BenutzerIn angezeigt. Die Aufgabe besteht dann darin, aus einer Liste gegebener Zahlen diejenigen auszuwählen, die die fragliche Eigenschaft erfüllen.

[31] Die Idee zu "Reformulate" verdanken wir Regula Krapf.

Kapitel 6

Das System Diproche

In diesem Kapitel beschreiben wir kurz den Aufbau des Diproche-Systems, seine einzelnen Komponenten und deren Implementierung. Es ist nicht das Ziel dieses Teils, den gesamten Code oder auch nur einzelne Stellen davon im Detail zu erläutern. Das vorliegende Kapitel soll nur eine Orientierungshilfe bieten und einen groben Einblick in die Funktionsweise derjenigen Module geben, deren Funktionsweise – wie etwa bei der Zielverfolgung – nicht bereits bei ihrer Einführung im letzten Kapitel erläutert wurde.

Die Programmiersprache, in der Diproche implementiert ist, ist die logische Programmiersprache PROLOG; für eine praxisorientierte Einführung siehe etwa Clocksin und Mellish [49]. Wir erläutern die Funktionsweise von PROLOG hier nur so weit, wie es nötig ist, um einzusehen, warum diese Sprache für ein Projekt wie Diproche besonders geeignet ist. (Auch das Naproche-System, dessen Aufbau Diproche im Wesentlichen folgt (s.o.), ist in Prolog implementiert.)

Im Gegensatz zu imperativen bzw. objektorientierten Programmiersprachen wie Java, C++ oder Python ist PROLOG eine logische Programmiersprache; Programme bestehen nicht (primär) aus Befehlen, die der Computer ausführen soll, sondern aus einfachen Aussagen und Schlussregeln, mit denen aus diesen Aussagen weitere gefolgert werden können. Ein PROLOG-Programm wird dann "ausgeführt", indem dem Programm "Fragen" über den codierten Gegenstandsbereich gestellt werden; diese Fragen können etwa die nach der Ableitbarkeit einer gewissen Aussage sein oder auch die nach allen Objekten, auf die eine gewisse Aussage zutrifft. PROLOG versucht dann selbstständig, diese Fragen zu beantworten, indem durch ein Backtracking-Verfahren in einer Tiefensuche die Schlussregeln auf die verfügbaren Aussagen angewendet werden, bis die fragliche Aussage abgeleitet wurde oder alle Ableitungsversuche fehlgeschlagen sind. Wenn eine Aussage nicht ableitbar, die Anzahl der möglichen Ableitungsversuche aber sehr groß oder gar unendlich ist, wird das Programm entsprechend erst sehr spät oder gar nicht halten.

So könnte man einem PROLOG-Programm etwa mitteilen, dass Harry ein Zauberer ist, indem man "zauberer(harry)." in den Code schreibt. Die allgemeine Regel, dass alles, was menschlich ist und zaubern kann, ein Zauberer ist, hätte folgende Gestalt:

zauberer(X):- mensch(X), kann_zaubern(X).

Einem Programm, das zusätzlich über die Informationen "mensch(harry)." und "kann_zaubern(harry)." verfügt, könnte man nun etwa die "Frage" "zauberer(harry)." stellen, worauf das Programm mit "true" antworten würde; ebenso könnte man über "zauberer(X)." nach einem Beispiel für etwas fragen, wovon das Programm feststellen kann, dass es sich um einen Zauberer handelt, und würde dann mit "X=harry" ein solches Beispiel erhalten.

Eine sehr nützliche Eigenschaft von PROLOG ist ferner die integrierte Unifikation: So kann man PROLOG etwa durch die Eingabe "[X,+,Y]=[2,+,5]" dazu auffordern, zu prüfen, ob der Term 2+5 sich als Instantiierung des Termes X+Y auffassen läßt, wobei X und Y Variablen sind, d.h. ob es möglich ist, X und Y so durch Terme zu ersetzen, dass sich aus $X + Y$ der Term $2 + 5$ ergibt. Ist das nicht möglich (wie etwa, wenn man [X,+,X]=[2,+5] abfragen würde, so dass man X sowohl mit 2 als auch mit 5 instantiieren müsste), so wird "false" gemeldet; andernfalls (wie in diesem Fall) wird eine Instantiierung mit der gewünschten Eigenschaft ausgegeben, in diesem Fall X=2, Y=5.

Das macht Prolog für die Implementierung automatischer Deduktionsroutinen, wie sie in Diproche etwa im Rahmen des ATP vorkommen, besonders geeignet, ebenso wie für die Verarbeitung von Termen. So erkennt etwa die Unifikationsroutine $(ux - vy)^2 + (uy + vx)^2$ automatisch als eine Instanz von r^2+y^2, so dass sich in folgendem, von der aktuellen Diproche-Version akzeptierten, Beispiel aus $a \cdot b = (ux-vy)^2+(uy+vx)^2$ die Existenz von r und s mit $a \cdot b = r^2+s^2$ folgern läßt:[1]

Es seien a, b, u, v, x, y ganze Zahlen. Es sei $a = u^2 + v^2$ und $b = x^2 + y^2$. Zeige: Es existieren ganze Zahlen r, s so, dass $a * b = r^2 + s^2$.

Beweis: Es ist $a * b = (u^2 + v^2) * (x^2 + y^2) = (u^2 * x^2 - 2 * u * v * x * y + v^2 * y^2) + (u^2 * y^2 + 2 * u * v * x * y + v^2 * x^2) = (u * x - v * y)^2 + (u * y + v * x)^2$. Also gilt $a * b = (u * x - v * y)^2 + (u * y + v * x)^2$. Damit existieren ganze Zahlen r, s so, dass $a * b = r^2 + s^2$. qed.

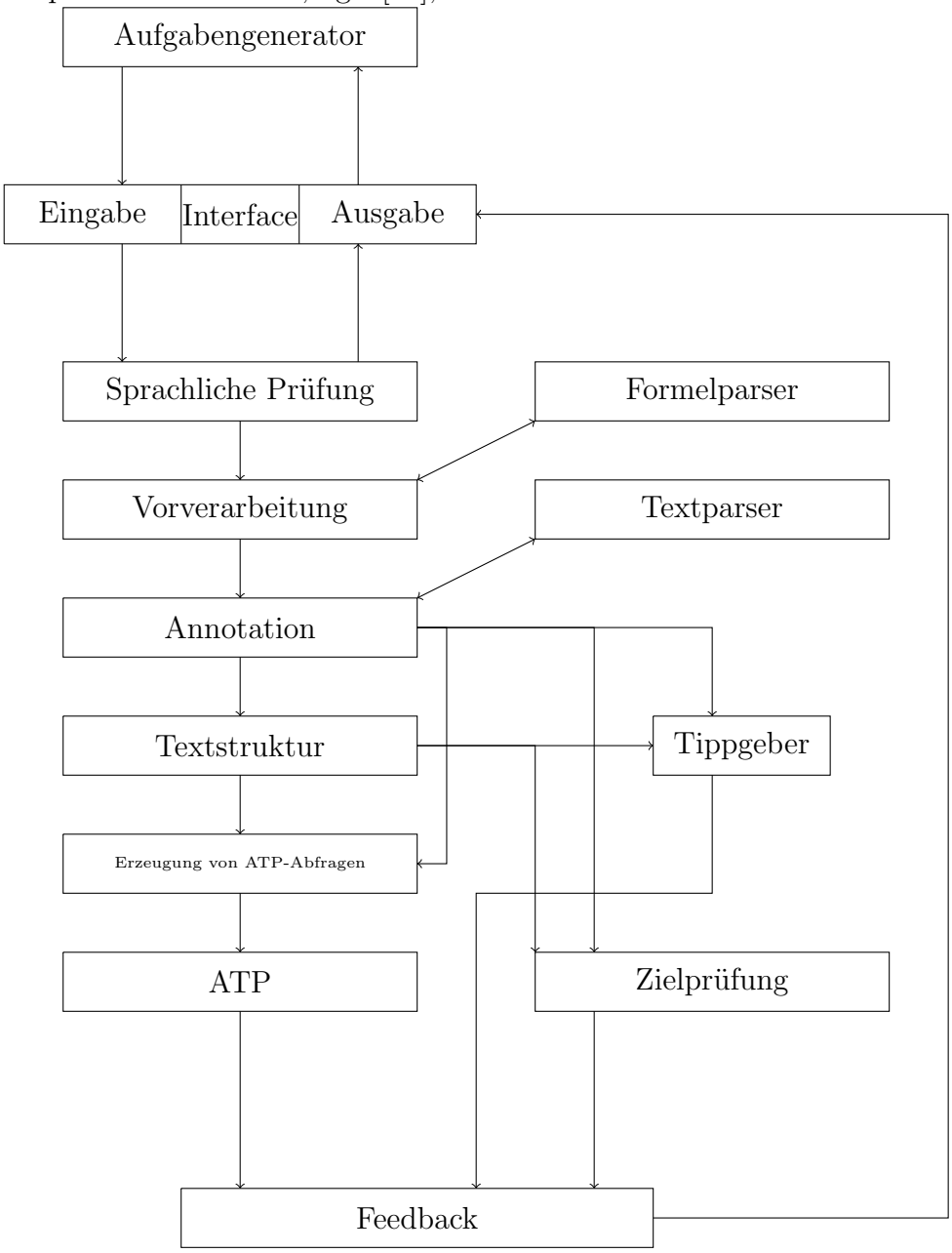
[1] Zusätzlich kommt hier natürlich die automatische Typenermittlung zum Einsatz, mit der aus der anfänglichen Deklaration der auftretenden Variablen als ganze Zahlen geschlossen wird, dass auch $(ux - vy)$ und $(uy + vx)$ ganzzahlig sind.

Zudem verfügt Prolog über eine sehr angenehme Listen- und Stringverarbeitung, die es für linguistische Anwendungen wie das Verfassen von Grammatiken oder rekursiven Semantiken attraktiv macht.

Für die Implementierung von Benutzeroberflächen und Webanwendungen ist Prolog dagegen nur bedingt geeignet; daher wurden diese Systemkomponenten von Thomas Bliem in JavaScript und HTML implementiert. In dieser Arbeit beziehen wir uns allerdings meist auf ein zu Test-, Demonstrations- und Entwicklungszwecken vom Autor entwickeltes Interface,[2] das in der objektorientierten Prolog-Erweiterung XPCE geschrieben ist. Das Interface stellt dem/r BenutzerIn Werkzeuge zum Schreiben eines mathematischen Textes zur Verfügung – etwa eine Menüleiste für Sonderzeichen – und konvertiert die Eingabe anschließend in ein Prolog-Listenformat (s.u.). In dieser Form wird sie dann dem Prolog-Code übergeben, der den Beweis prüft und verschiedene Rückmeldungen (ebenfalls in einem Listenformat) generiert. Diese Rückmeldungen erscheinen dann wiederum in allgemein lesbarer Form im Interface.

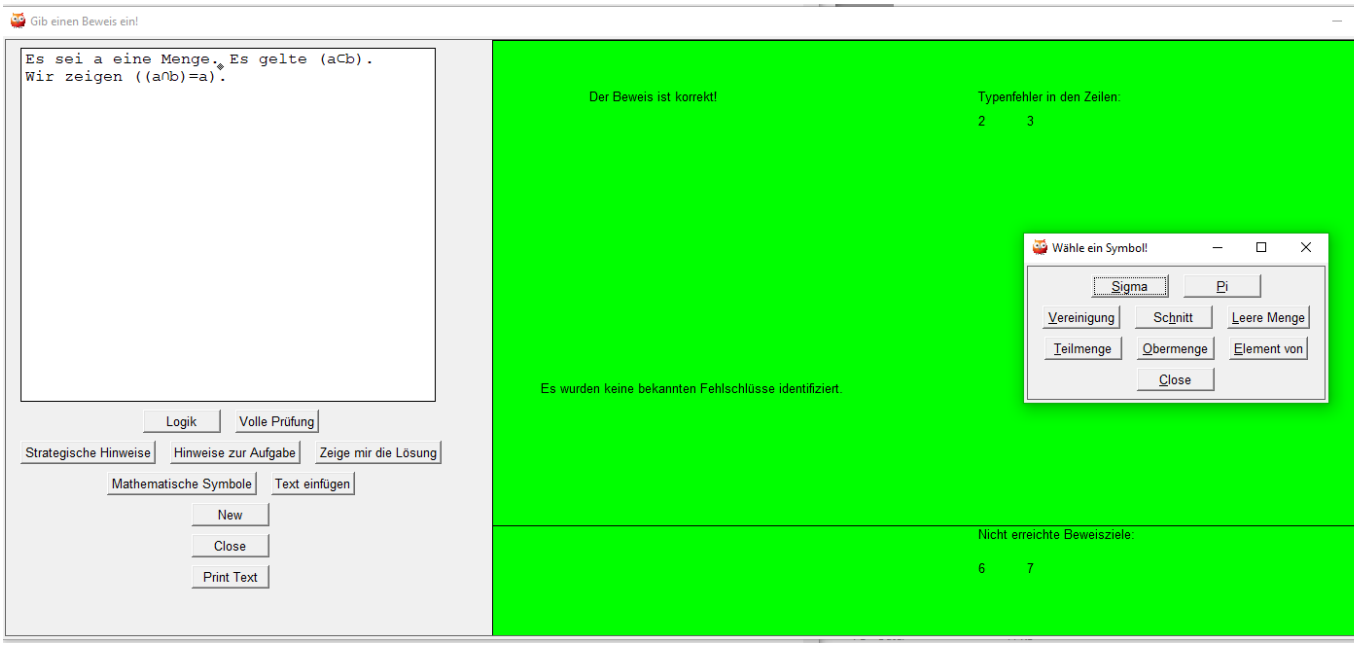
[2]Die meisten Bilder von Diproche-Sitzungen mit Ausnahme derer, die das "Game of Def" betreffen, zeigen das PROLOG-Interface; die Abbildungen zum "Game of Def" zeigen dass von Thomas Bliem entwickelte Webinterface (s.u.).

Diproche besteht derzeit aus folgenden Modulen, die aufeinander wie im Diagramm angegeben Bezug nehmen; die Architektur ist im wesentlichen eine um einige spezifisch didaktische Komponenten wie den Tippgeber ergänzte Variante der Naproche-Architektur, vgl. [47], S. 14:



6.1 Das Interface

Das Interface dient als Schnittstelle zwischen den in Prolog implementierten Routinen, die wir oben erläutert haben, und dem/der BenutzerIn. Zum Interface gehören insbesondere ein Texteingabefenster sowie ein Auswahlmenü für diverse Funktionen und Teilprogramme. Den Studierenden wird Diproche derzeit über ein von Thomas Bliem entwickeltes Webinterface zur Verfügung gestellt; wir erläutern die Funktionen der Hauptkomponente, des Beweisprüfers, kurz anhand des (weitgehend funktionsgleichen) Prolog-Interface, das für Test- und Demonstrationszwecke vom Autor entwickelt wurde. Wir konzentrieren uns auf diejenigen Komponenten, die sich im Web-Interface wiederfinden bzw. deren Einbindung für einen späteren Zeitpunkt geplant ist.



Das XPCE-Interface von Diproche

Das Interface hat links oben ein Textfenster, in das BenutzerInnen freien Text eingeben können. Darunter befindet sich eine Auswahl von Schaltflächen, die mit der Maus bedient werden können. Bei Auswahl von "Volle Prüfung" werden sämtliche oben erläuterten Prüfroutinen (also logische Prüfung, Zielprüfung, Typenprüfung und Fehlschlussdiagnose) angewandt; das Ergebnis wird auf der rechten Seite angezeigt. Über "Strategische Hinweise" und "Hinweise zur Aufgabe" können die oben im Abschnitt "Hilfestellung" als "Direkte Hinweise" bzw. "Allgemeine strategische Hinweise" erläuterten Hilfestellungen abgerufen werden. Über "Zeige mir die Lösung" kann bei im System implementierten Aufgaben eine Musterlösung abgerufen werden, sofern eine hinterlegt wurde; diese Funktion ist natürlich im Hinblick auf den angestrebten Übungseffekt mit Vorsicht einzusetzen;

aktuell ist sie im Webinterface noch nicht verfügbar. Über "mathematische Symbole" schließlich öffnet sich ein Submenü (im Bild oben das graue Fenster rechts), über das diverse mathematische Sonderzeichen in den Text eingefügt werden können.

6.2 Vorverarbeitung

Die Vorverarbeitung dient dazu, einen Eingabetext in das interne Listenformat zu übersetzen, mit dem die weitere Prolog-Verarbeitung erfolgt. Die Hauptarbeit übernimmt hierbei das Modul "plaintextpreprocessing", das auf das Modul "Formelparser" zurückgreift. Zu Beginn der Verarbeitung werden Kommata sowie mehrfache Leerzeichen oder Absätze gelöscht. Der Text wird dann zunächst in eine Liste von Texteinheiten zerlegt, wobei eine Texteinheit entweder ein Satz ist oder eine Annotation (wie "⇒", "Beweis:", "QED" etc.). Das Ende eines Satzes liegt dabei beim nächstfolgenden Punkt, während die Annotationen eine endliche Liste von Sonderfällen bilden.

Aus dem Text

(1) "Angenommen, es gilt A. Nehmen wir ferner an, es gilt A→B. Dann gilt (A&B)."

wird dadurch zunächst die folgende Darstellung gewonnen:

(2) ["Angenommen es gilt A","Nehmen wir ferner an es gilt A→B","Dann gilt (A&B)"]

Im nächsten Schritt wird jeder Satz in eine Folge von Wörtern und Formeln zerlegt. Als "Wort" gilt hierbei jeder String, der ausschließlich aus Buchstaben besteht und mindestens die Länge 2 hat.[3] Worte werden dann in konsequente Kleinschreibung umgewandelt und jeder Satz als Liste von Wörtern und Formeln repräsentiert. So erhält der obige Text folgende Form:

(3)
[[angenommen,es,gilt,'A'],[nehmen,wir,ferner,an,es,gilt,'A→B'],[dann,gilt,'(A&B)']]

Schließlich wird der Formelparser auf jede Formel im Text angewendet, wodurch die Formeln ebenfalls in ein Prolog-internes Listenformat konvertiert

[3] In seltenen Fällen kann es dabei zu Fehlverarbeitungen kommen: So hat etwa die Verwendung des Kleinbuchstaben *v* als logisches Oder die Konsequenz, dass eine ungeklammerte Disjunktion wie "avb" fehlerhaft als (unbekanntes) Wort identifiziert wird, wodurch die Verarbeitung mit einer Fehlermeldung abbricht.

werden (den Formelparser erläutern wir, zusammen mit dem Textparser, im nächsten Abschnitt). Damit erhalten wir die folgende, abschließende Form, die dem Parsing für die weitere Verarbeitung übergeben wird:

(4) [[angenommen, es, gilt, 'A'], [nehmen, wir, ferner, an, es, gilt,[A,->,B]], [dann, gilt, [A, and, B]]].

6.3 Parsing und Formalisierung

Für die Verarbeitung durch einen ATP ist es wesentlich, die natürlichsprachlichen Sätze eines Eingabetextes in ein entsprechendes standardisiertes Format zu übersetzen. Im Fall von Diproche ist das eine Darstellung von Formeln der erststufigen Prädikatenlogik in einem geschachtelten Listenformat. Dabei wird zum einen der Inhalt eines Satzes identifiziert, zum anderen aber auch, um welche Art von Satz (Annahme, Behauptung, Deklaration, Zielankündigung, jeweils mit diversen Untertypen) es sich handelt. Beides wird dann im nächsten Schritt, der Generierung des annotatierten Formates, zusammengeführt.

So würden beispielsweise "Wenn a gilt, dann gilt auch b", "Es gilt $(a \to b)$" und "Es gilt b, sofern a gilt" alle als Behauptung der Aussage $a \to b$ identifiziert.

Diese Formalisierung erfolgt in zwei Schritten: Im ersten Schritt, dem Parsing, werden die Satz- bzw. Formelbestandteile identifiziert und diese Identifikationen in einem Zwischenformat festgehalten. Beim Parsing ergibt sich bereits aus den dabei anwendbaren Regeln der jeweilige Satz- oder Formeltyp. Im zweiten Schritt wird dieses Zwischenformat in die Listendarstellung konvertiert.

Wir erläutern im folgenden zunächst kurz die Grundideen des verwendeten Parsings sowie das Format, in dem die Parsingregeln implementiert sind, die sogenannten "Definite Clause Grammars" (DCGs). Anschließend betrachten wir kurz die konkrete Implementierung des Formel- und Textparsers.

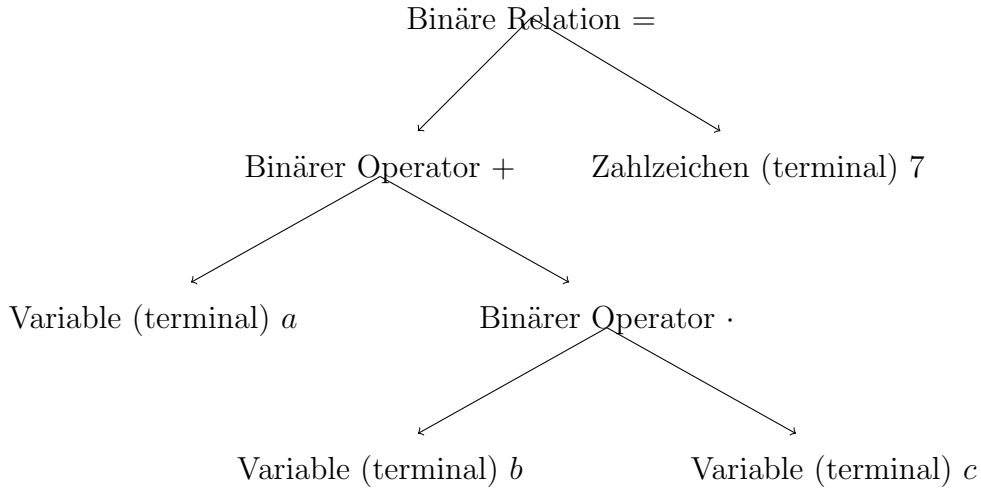
6.3.1 Definite Clause Grammars

Für die Eingabesprache von Diproche ist nur ein sehr begrenzter und überschaubarer Vorrat an Formulierungen erforderlich, weswegen die Komponenten für die linguistische Verarbeitung verhältnismäßig einfach gehalten werden können. Viele Möglichkeiten von Diproche ließen sich bereits mit einer einfache "Template-Grammatik" realisieren, die diverse Formulierungsmöglichkeiten in Form von "Lückentexten" bereit stellt, welche dann entsprechend ausgefüllt werden können. So könnte eine Template-Grammatik etwa Formulierungen wie "Es gelte ...", "Angenommen, es gilt ..." etc. enthalten, so dass ein eingegebener Satz dieser Form als Annahme erkannt

würde, deren Gegenstand dann der Inhalt der Leerstelle ist. Allerdings begrenzt ein solcher Ansatz die Flexibilität im Ausdruck drastisch: Bereits rekursiv aufgebaute Satzmuster wie etwa iterierte Behauptungen ("Es gilt ..., also folgt ... und damit gilt auch ...") lassen sich auf diese Weise nicht mehr abbilden, wenn man keine definitive Obergrenze für die Anzahl der Glieder einführt – und selbst dann wird der Regelsatz aufgrund des Variantenreichtums an Formulierungen für die Übergänge unüberschaubar groß und unübersichtlich. Andererseits ist es für die Verarbeitung in Diproche nicht nötig, etwa in "Angenommen, wir haben" die Worte "angenommen" und "haben" als Verben und "wir" als Subjekt des durch "wir haben" eingeleiteten Satzes zu erkennen. Für Diproche wurde daher ein Mittelweg zwischen einer vollständigen grammatischen Analyse und einer reinen Template-Grammatik eingeschlagen, nämlich eine grammatische Analyse, bei der typische Formulierungsbausteine als atomare Bestandteile fungieren. Diese Analyse verwendet im wesentlichen die gleichen Techniken wie eine vollständige grammatische Analyse; eine wesentliche Rolle spielen dabei formale Grammatiken.

Definite Clause Grammars (DCGs) sind ein Typ von formalen Grammatiken, der für linguistische Anwendungen besonders geeignet ist und von Prolog unterstützt ist. Für DCGs und ihre Verwendung in Prolog siehe z.B. Lehner [134] S. 188f oder Clocksin und Mellish [49], Kapitel 9. Wir geben hier einen kurzen Einblick, um die Erläuterungen zum Formel- und Sprachparser in den nächsten Abschnitten vorzubereiten.

Um einen Ausdruck, wie einen arithmetischen Term, eine Gleichung, eine aussagenlogische Formel oder auch einen natürlichsprachlichen Satz interpretieren zu können, ist es erforderlich, seine Bestandteile zu identifizieren und in ihrer Funktion zu erkennen; dieser Prozess wird als "Parsing" bezeichent. So wäre etwa für die Zeichenkette $(a + (b \cdot c)) = 7$ zunächst zu erkennen, dass hier auf der obersten Ebene = als binäres Relationszeichen eine Relation zwischen den Termen $(a + (b \cdot c))$ und 7 herstellt, wobei 7 als einzelnes Zahlzeichen ein primitives bzw. terminales (nicht weiter zu zerlegendes) Symbol darstellt, während $(a+(b\cdot c))$ weiter dadurch zu verarbeiten wäre, dass + als binärer Operator identifiziert wird, der die Variable a (die dann wiederum ein terminales Symbol darstellt) mit dem Term $(b\cdot c)$ verknüpft, welcher nun seinerseits wieder als Verknüpfung der Variablen b und c durch den binären Operator $\cdot$ aufzufassen wäre. Diese strukturierte Auffassung der Zeichenkette kann in Form eines sogenannten Parsingbaums dargestellt werden, wobei wir die Klammern aus Gründen der Übersichtlichkeit auslassen:



(Parsingbaum der Formel $(a + (b \cdot c)) = 7$)

Intern werden solche Bäume durch ein entsprechendes strukturiertes Format repräsentiert; der Formelparser von Diproche stellt die Formel $(a + (b \cdot c)) = 7$ etwa als

```
af(ob(()),at(ob(()),at(v(l(a))),ao(+),at(ob(()),at(v(l(b))),ao(*),at(v(l(c))),cb())),cb())),
ar(=),nr(c(7)),cb()))
```

dar. Dabei sagt "af", dass der String als arithmetische Formel identifiziert wurde; "ob" und "cb" stehen für "opening bracket" und "closing bracket"; mit "ar(=)" wird = als arithmetische Relation idenfiziert, wobei auf der rechten Seite ein Zahlzeichen ("nr") steht, dass aus einer einzigen Ziffer ("c") besteht, nämlich 7. Der Subterm at(v(l(a))),ao(+),at(ob(()),at(v(l(b))),ao(*),at(v(l(c))),cb())),cb())) auf der anderen Seite ist ein arithmetischer Term ("at"), in dem die aus einem Buchstaben "l" (für "letter") Variable ("v") a durch den arithmetischen Operator ("ao") mit dem arithmetischen Term at(ob(()),at(v(l(b))),ao(*),at(v(l(c))),cb())) verknüpft wird.

Die Erzeugung dieser Darstellung erfolgt durch durch sogenannte Produktionsregeln, wie sie bei der Angabe formaler Grammatiken üblich sind (siehe etwa Erk und Priese [68]); im Rahmen der DCG lassen sich insbesondere kontextfreie Grammatiken (CFGs) ausdrücken, was für die Belange von Diproche genügt.

Die Parsingregeln einer formalen Grammatik legen fest, wie sich Ausdrücke eines gewissen Typs aus anderen Ausdrücken gewisser (nicht notwendigerweise anderer) Typen aufbauen lassen. So ist etwa die Regel

arith_term−− >opening_bracket,arith_term,bin_arith_op,arith_term,closing_bracket

zu lesen als: “Einen arithmetischen Term kann man dadurch bilden, dass man zunächst eine sich öffnende Klammer setzt, dann einen arithmetischen Term, dann einen binären arithmetischen Operator, dann einen weiteren arithmetischen Term und schließlich eine schließende Klammer”.

Eine formale Grammatik ist dann eine Menge solcher Regeln, wobei natürlich für jeden Ausdruck mehrere Bildungsgesetze angegeben werden können. Der in Prolog vorgesehene DCG-Formalismus erlaubt es außerdem, simultan mit der Analyse eines Ausdrucks eine Darstellung des Parsingbaumes zu generieren, wie wir ihn oben angegeben haben. Eine einfache Grammatik für arithmetische Terme hat etwa die folgende Form:

- arith_term(at(OB,AT1,OP,AT2,CB))−− >
  opening_bracket(ob(OB)),arith_term(at(AT1)),bin_arith_op(ao(OP)),
  arith_term(at(AT2)),closing_bracket(cb(CB)).
- arith_term(at(S,AT))−− >sign(s(S)),arith_term(at(AT)).
- arith_term(at(v(a))−− >[a].
- arith_term(at(v(b))−− >[b].
- arith_term(at(v(c))−− >[c].
- bin_arith_op(ao(+))−− >[+].
- bin_arith_op(ao(*))−− >[*].
- opening_bracket(ob(‘(’))−− >[‘(’].
- closing_bracket(cb(‘)’))−− >[‘)’].
- sign(s(-))−− >[-].

Dabei sind alle Regeln außer (1) und (2) terminal, d.h. sie erlauben es, einen Termtyp durch eine Symbolfolge zu realisieren, die selbst nicht weiter analysiert wird. Wie man an den ersten beiden Regeln sieht, können Regeln sich (wie bei CFGs) rekursiv selbst aufrufen. Bei der Implementierung von DCGs ist es jedoch wichtig, linksrekursive Regeln zu vermeiden, das heißt solche, bei denen ein Ausdruckstyp durch eine Regel definiert ist, derzufolge er mit einem Ausdruck desselben Typs anfängt. So wäre etwa die naheliegende “klammerfreie” Variante unserer Regel von oben, nämlich

arith_term−− >arith_term,bin_arith_op,arith_term

linksrekursiv. Durch die Art, wie Prolog DCGs verarbeitet, können linksrekursive Regeln in Endlosschleifen führen (die Regel ruft sich rekursiv immer wieder selbst auf), wodurch die Verarbeitung fehlschlägt. Daher wurde der Formelparser nur für vollständig geklammerte Terme implementiert. Um das in der Praxis übliche Fortlassen von Klammern, besonders von äußeren Klammern, gleichwohl zu ermöglichen, wurde eine vor das Parsing geschaltete Routine implementiert, die fehlende Klammern automatisch ergänzt (s.u.). Entsprechende Maßnahmen zur Vermeidung von Linksrekursivität mussten auch für das Parsing natürlichsprachlicher Ausdrücke ergriffen werden (s.u.).

Um mit diesen Regeln unseren Beispielterm $(a + (b * c))$ zu analysieren, geht Prolog (vereinfacht) wie folgt vor: Zunächst wird versucht, den gegebenen Ausdruck mit der Regel (1) zu analysieren. Dazu wird zunächst nach einem Anfangsstück gesucht, dass sich unter die Termkategorie "opening_bracket" subsumieren läßt. Ein solches wird mit '(' gefunden und als "ob('(')" gespeichert. Anschließend wird der verbleibende String $a + (b * c))$ analysiert, wobei nun nach einem Anfangsstück gesucht wird, das sich als "arith_term" auffassen läßt. Ein solches wird in a gefunden, was als arithmetischer Term vom Typ Variable "at(v(a))" gespeichert wird, so dass nun noch $+(b * c))$ verbleibt, von dem nun + als arithmetischer Operator abgespalten wird. Im nächsten Schritt erfolgt ein rekursiver Aufruf von arith_term, durch den "$(b * c)$" als at(ob((),at(v(l(b))),ao(*),at(v(l(c))),cb())) geparst wird; zum Schluss wird der verbliebene String ')' als "closing_bracket" identifiziert, womit das Parsing erfolgreich abschließt.

6.3.2 Der Formelparser

Der Formelparser hat die Aufgabe, Eingabestrings, die formale Ausdrücke repräsentieren, zu identifizieren (schlägt das Parsing fehl, erfolgt eine entsprechende Rückmeldung, die dann eine entsprechende Bildschirmausgabe zur Folge hat) und in das Listenformat zu übersetzen, mit dem die weiteren Prologroutinen, wie der ATP oder die Typenprüfung, arbeiten. Die grundlegende Funktionsweise wurde im vorigen Abschnitt am Beispiel arithmetischer Formeln erläutert. Dabei wurde insbesondere hervorgehoben, dass der Parser zur Vermeidung von Linksrekursivität nur mit vollständig geklammerten Ausdrücken arbeitet. Nun ist das vollständige Klammern ein in vielen Fällen umständlicher Prozess (so wäre etwa $a \cdot x^2 + b \cdot x + c$ zu schreiben als so etwas wie $(((a \cdot (x^2)) + (b \cdot x)) + c))$, durch den sowohl die Lesbarkeit als auch der Komfort beim Schreiben von Eingabetexten deutliche Einbußen erleiden. Aus diesem Grund ist dem Parser eine Routine vorgeschaltet, die die Klammersetzung, wo nötig, selbstständig vornimmt; dabei wird zunächst gemäß der üblichen Prioritätsregeln verfahren: Terme vor Relationszeichen, also etwa $((a + b) = c)$ statt $(a + (b = c))$,

Exponentiation vor “Punktrechnung”, “Punkt- vor Strichrechnung” etc.; wo sich aus den Prioritätsregeln keine Klammerung ergibt, wird “von links” geklammert, so dass aus $a + b + c + d$ etwa $(((a + b) + c) + d)$ würde.

Anschließend findet ein Parsing statt, wie im letzten Abschnitt beschrieben. Dabei gibt es für die verschiedenen Formeltypen, die im Rahmen von Diproche-Texten auftreten können, separate Grammatiken (die aber z.T. aufeinander Bezug nehmen). Derzeit verarbeitet Diproche folgende Typen formaler Ausdrücke (für eine Erläuterung der verschiedenen Ausdruckstypen siehe das Kapitel zur Sprache von Diproche weiter unten):

- Aussagenlogische Ausdrücke (z.B. $((\neg A \vee B) \to C)$).
- Aussagenlogische Folgerungsketten (z.B. $\neg(A \vee B) \Leftrightarrow (\neg A \wedge \neg B) \Leftrightarrow (\neg B \wedge \neg A) \Rightarrow \neg A$).
- Mengentheoretische Terme (z.B. $(A \cap B) \times (B \cup (C \setminus D))$).
- Mengentheoretische Formeln (z.B. $((A \cap B) \subseteq C) \vee (C = D)$).
- Mengentheoretische “Ungleichungsketten” (z.B. $((A \cap B) \subseteq A \subseteq (A \cup B) = (B \cup A))$).
- Mengentheoretische Folgerungsketten (z.B. $(x \in (A \cap B)) \Rightarrow (x \in A) \Rightarrow (x \in (A \cup B)) \Leftrightarrow (x \in (B \cup A))$).
- Arithmetische Terme (z.B. $(a + 17) \cdot (b - c)$).
- Arithmetische Formeln (z.B. $(3|n!) \vee (a + b) = (c + d)$).
- Arithmetische Ungleichungsketten (z.B. $1 \leq a = (b + c) < (b + c + d) = 5$).
- Arithmetische Folgerungsketten (optional mit “Regieanweisungen” (z.B. $(a = b) \Leftrightarrow (b = a) \Leftrightarrow (*2)(2 * b = 2 * a)$).
- Geometrische Terme (z.B. $l(p, q)$ für die Verbindungsgerade der Punkte p und q).
- Geometrische Formeln (z.B. $l(a, b)||g$).
- Funktionsterme (z.B. $(f * g)(x) = f(g(x))$)
- Gruppentheoretische Terme (z.B. $(a + e) = a$).

6.3.3 Der Textparser

Der Textparser hat, ähnlich wie der Formelparser, die Aufgabe, einen Satz in der Eingabesprache von Diproche als solchen zu erkennen (nicht verarbeitbare Sätze werden wiederum der/dem BenutzerIn als solche gemeldet) und seine Funktion wie auch seine Bedeutung in das Format zu übersetzen, mit dem die weitere Verarbeitung stattfindet.

Derzeit arbeitet Diproche mit folgenden Funktionen und Unterfunktionen:[4]

- Annotationen ("ann"). Hierzu gehören insbesondere Beweisanfangs- ("bam") und Endmarker ("bem"), die weiter nach ihren Typen differenziert sind: Anfangsmarker für Hin- ("hr") und Rückrichtung ("rr") einer Implikation, für Hin- ("shr") und Rückrichtung ("srr") eines Mengengleichheitsbeweises durch zwei Inklusionsbeweise, für Widerspruchs- ("contr"), Kontrapositions- ("cntrpos") und Induktionsbeweise ("ind"), die Einführung von Fällen in Fallunterscheidungen ("fe") etc. Zu den Annotationen gehören ferner die Absätze ("abs") (welche die Gültigkeitsbereiche von Annahmen markieren) sowie die Zieldeklarationen ("goal").

- Annahmen ("ang"), zu denen auch Typendeklarationen ohne ("dkl") und mit ("dklcnt") gehören.

- Behauptungen ("beh"); diese umfassen begründete Behauptungen[5] ("bbh"), multiple Behauptungen[6] ("mltpl"), multiple begründete Behauptungen[7], die unter "bbh" subsumiert, aber in der Darstellung des Inhalts durch ("bbhmult") markiert werden.

- Axiome ("axm"), also Aussagen, die ohne Begründung als für den restlichen Beweis gültig akzeptiert werden. Da dieser Satztyp im Einsatz von Diproche bisher zwar im Parser angelegt, aber noch nicht in die Prüfroutinen implementiert ist (und im derzeitigen Einsatz von Diproche auch keine Rolle spielt), werden wir ihn im folgenden nur noch kursorisch erwähnen.

DCG-Grammatiken für kontrollierte natürliche Sprachen

Trotz der ungleich größeren Komplexität natürlicher Sprache gibt es ausdrucksstarke und natürliche Fragmente der natürlichen Sprache, die für ein

[4] Wir geben die interne Darstellung jeweils in Klammern hinter der Nennung der Kategorie an.

[5] Wie etwa "Wegen a gilt b".

[6] Mehrschrittige Folgerungen in einem Satz, wie "Damit gilt a, also b und damit folgt c".

[7] Kombination der beiden vorhergehenden Typen, wie in "Wegen a gilt b, damit folgt c und folglich haben wir d".

Parsing mit DCGs zugänglich sind; für die Grundlagen des automatischen Parsings natürlicher Sprache siehe etwa [153] oder [134]. In einer formalen Grammatik für natürlichsprachliche Ausdrücke ist ein Satz ein Typ von Ausdruck, der auf verschiedene Arten aus Ausdrücken anderen Typs zusammengesetzt werden kann. So könnten wir etwa einige sehr einfache Sätze mit der folgenden kontextfreien Grammatik analysieren:

- satz−− >nominalphrase,verbalphrase.
- nominalphrase−− >artikel,substantiv.
- verphalphrase−− >adjektiv,verb.
- adjektivphrase−− >adjektiv.
- artikel−− >[der].
- artikel−− >[die].
- artikel−− >[das].
- adjektiv−− >[gruen].
- adjektiv−− >[gross].
- substantiv−− >[haus].
- substantiv−− >[berg].
- verb−− >[ist].

Mit dieser Grammatik lassen sich Sätze wie “Der Berg ist grün.” oder “Das Haus ist groß.” analysieren. Zu beachten ist, dass diese Grammatik auch für grammatisch fehlerhafte Sätze wie “Die Berg ist grün” erfolgreich wäre. Sollen nur grammatisch korrekte Sätze geparst werden können, so muss die Grammatik durch die Berücksichtigung von Kategorien wie Genus, Plural/Singular etc. verfeinert werden. Man beachte weiter, dass der naheliegende Ansatz für eine Regel, die es erlaubt, zwei Sätze durch einen logischen Junktor zu verknüpfen

satz−− >satz,junktor,satz. junktor−− >[und]. junktor−− >[oder].

nicht zur Verfügung steht, weil er linksrekursive Regeln enthält.

Für die Zwecke von Diproche ist, wie wir bereits oben diskutiert haben, eine vollständige grammatische Analyse unnötig, so dass die Grammatik für das Parsing in Diproche sich denn auch eher pragmatisch an den Erfordernissen der

relevanten Ausdruckstypen orientiert. Von den Annotationen bilden lediglich die Zieldeklarationen eine Teilgruppe, die nicht durch die explizite Angabe sämtlicher Elemente ausschöpfbar ist. Die Unterscheidung von Zieldeklarationen, Behauptungen und Annahmen erfolgt einfach durch das Anfangsstück eines Satzes. So enthält die Diproche-Grammatik etwa die Kategorie "assumption initial phrase" ("aip"), die die gängigen Formulierungen für Annahmen wie "Angenommen (dass)", "Nehmen wir an (, dass)", "(Es)gelte (, dass)", "Es sei/seien", "Gesetzt (dass/den Fall/den Fall dass)" umfasst, sowie eine Möglichkeit, gewisse Worte wie "Ausserdem", "Ferner" etc. einer aip voranzustellen, wodurch etwa eine Formulierung wie "Ausserdem gelte a" erlaubt wird. Eine Regel für das Parsing von Annahmen ist dann einfach die folgende:

assumption_parsing(assmpt(assmpt_initial_phrase(A),assmpt_content(B)))⟶
assumption_initial_phrase_parsing(A),content_parsing(B).

Eine Annahme ist also eine Anfangsphrase für Annahmen, gefolgt von einem "Inhalt".[8]

Entsprechend funktionieren die Regeln für das Parsing von Behauptungen (mit Anfangsphrasen wie "Es/Also/Folglich/Damit/Mithin/Ergo/... gilt/folgt/haben wir/erhalten wir/ergibt sich/..."), Zieldeklarationen (mit Anfangsphrasen wie "wir zeigen/zu zeigen/wir wollen zeigen/...") und Axiomen (mit Anfangsphrasen wie "Axiom:"). Bei begründeten Behauptungen, die durch Anfangsphrasen wie "Wegen" oder "Durch" identifiziert werden, unterscheiden die Parsingregeln zwischen dem Begründungsteil und dem begründeten Teil, wobei der Begründungsteil eine Folge von Inhalten ist ("Wegen a, b, c und d...") und der begründete Teil ein einzelner Inhalt. Multiple Behauptungen sind einfach Folgen von Behauptungen, begründete multiple Behauptungen bestehen aus einem Begründungsteil, gefolgt von einer multiplen Behauptung.

An dieser Stelle des Parsings – mit der Identifikation der Anfangsphrasen – ist die Identifikation von Satzfunktion und -unterfunktion bereits dadurch erfolgt, dass die Parsingregeln für Sätze des entsprechenden Typs anwendbar waren (und die für Sätze anderen Typs nicht).

Bei der Unterscheidbarkeit der Satzfunktionen anhand der Anfangsphrasen gibt es allerdings eine Einschränkung: Aus Gründen größerer Natürlichkeit ist es möglich, sowohl einfache Annahmen wie auch Deklarationen von Variablen mit "Es sei/seien" zu beginnen, etwa in Formulierungen wie "Es sei n gerade." oder "Es sei n eine ganze Zahl.". Obwohl diese Sätze in Diproche-Texten deutlich unterschiedliche Funktionen haben (der zweite führt die Variable n ein, der erste

[8]Es gibt noch einige andere Regeln für das Parsing von Annahmen, die insbesondere negierte Annahmen ("Angenommen, a ist falsch") ermöglichen.

nicht; ginge dem ersten keine Deklaration voraus, würde ein Typenfehler gemeldet, beim zweiten nicht) sind die Anfangsphrasen identisch. In diesem Fall ergibt sich erst aus dem Inhalt, ob der Satz als Deklaration oder als einfache Annahme identifiziert wird.

Wir gehen noch kurz auf das Parsing der Inhalte ein. Atomare Inhalte sind zunächst formale Ausdrücke, die Aussagen darstellen (im Gegensatz etwa zu arithmetischen Termen) sowie Eigenschaftszuschreibungen wie "a ist gerade" oder "l ist parallel zu g". Ferner können Inhalte durch natürlichsprachliche Junktoren wie "und", "oder", "wenn ... dann ..." oder "genau dann, wenn" verbunden werden. Dabei wird ein Ausdruck mit mehreren Junktoren stets so interpretiert, dass der erste Junktor Priorität besitzt; so kann auf der linken Seite anstelle eines beliebigen Inhalts stets ein atomarer Inhalt abgespalten werden, womit die Linksrekursivität umgangen wird. (Der Fall der Implikation ist in dieser Hinsicht unproblematisch, da diese natürlichsprachlich nicht als reines Infix wie "oder", "und" oder "ist äquivalent zu" ausgedrückt wird, sondern in Formulierungen wie "Wenn ... dann...", die nicht zu linksrekursiven Regeln führen.)

So können natürlichsprachliche Formulierungen wie "Wenn a oder b gilt, dann gilt auch c oder d" verarbeitet werden.[9]

So wird etwa der Satz

> "Wir zeigen: Wenn a gerade ist, dann sind b und c ungerade."

nach dem Parsing intern wie folgt repräsentiert:

```
goal(gip(gp1(wir), zeigen), claim(impl(iip(wenn),
claim(atomic(property(fmln("fml1"), adj(gerade)))), jvp(ist), iv(dann),
claim(atomic(mult_property(jvp(sind), vap(var("var2"), cop(und),
vap(var("var3"))), adj(ungerade)))))))
```

Auf der obersten Ebene weist diese Darstellung dem Satz den Typ "Zieldeklaration" ("goal") zu, der aus einer Anfangsphrase für Zieldeklarationen ("goal initial phrase", "gip") besteht, nämlich ("wir zeigen") sowie einer Behauptung "Behauptung" ("claim"). Diese Behauptung wird weiter als

[9] Allerdings ist eine solche Schachtelung von Junktoren in voller Stärke derzeit nur verfügbar, wenn die auftretenden atomaren Inhalte Formeln sind, nicht aber, wenn es sich etwa um natürlichsprachliche Eigenschaftszuschreibungen handelt. So kann etwa "Wenn a gerade ist, dann sind b und c ungerade" verarbeitet werden – einschließlich der distributiven Verwendung von "ungerade", die als "b ist ungerade und c ist ungerade" interpretiert wird – nicht aber etwa der Satz "Wenn a ungerade ist und b ungerade ist, dann ist c gerade". Diese Beschränkungen des Parsers sind zwar für den intendierten Einsatzbereich von Diproche nur begrenzt relevant, sollten aber auf lange Sicht behoben werden.

Implikation ("impl") identifiziert und besteht aus der Anfangsphrase "wenn" für Implikationen ("implication initial phrase", "iip"), gefolgt von einer weiteren Behauptung ("claim"), die diesmal atomar ("atomic") ist und in der Eigenschaftszuweisung "property" des Adjektivs ("adj") "gerade" zum formalen Ausdruck ("fmln"), fml1 besteht, der hier das a repräsentiert; dies ist durch "dann" mit einer zweiten Behauptung ("claim") verbunden, welche wiederum atomar ("atomic") ist und aus einer multiplen Eigenschaftszuschreibung ("mult_property") des Adjektivs "ungerade" zu den beiden formalen Ausdrücken fml2 und fml3 besteht.

6.3.4 Formalisierung

Die Formalisierungsroutine schließt den Prozess der linguistischen Verarbeitung ab, indem die aus dem Parsing erhaltenen strukturierten Darstellungen der Eingabestrings als Parsingbäume in das intern von den Prüfroutinen verwendete Listenformat übersetzt werden. Wir erläutern hier kurz dieses Listenformat, das neben erststufigen Formeln u.a. auch Folgerungs-, Umformungs- und Ungleichungsketten repräsentiert.

Eine "Variable" ist ein (großer oder kleiner) lateinischer Buchtstabe, gefolgt einer (möglicherweise leeren) Folge von Ziffern als Index. Variablen werden im Listenformat einfach durch sich selbst repräsentiert.

Aussagenlogische Formeln und Folgerungsketten

- Jede Variable ist eine aussagenlogische Formel (die durch sich selbst repräsentiert wird).
- Werden ϕ und ψ durch A und B dargestellt, so werden Konjunktion, Disjunktion, Implikation und Äquivalenz von ϕ und ψ durch $[A, and, B]$, $[A, or, B]$, $[A, ->, B]$ und $[A, <->, B]$ dargestellt.
- Wird ϕ durch A dargestellt, so wird $\sim \phi$ durch $[neg, A]$ dargestellt.
- Folgerungsketten zwischen aussagenlogischen Formeln werden durch eine Liste dargestellt, in der die Darstellungen der auftretenden Formeln sich mit den jeweiligen Folgerungsszeichen abwechseln, also etwa $(A\&B) \Leftrightarrow (B\&A) \Rightarrow A$ durch $[[A, and, B], <=>, [B, and, A], =>, A]$.

Mengentheoretische Terme, Formeln, Ungleichungs- und Folgerungsketten

- Jede Variable ist ein mengentheoretischer Term, der durch sich selbst dargestellt wird.

- Werden die mengentheoretischen Terme τ_0 und τ_1 durch T_0 und T_1 dargestellt, so werden Schnitt, Vereinigung, das relative Komplement von τ_0 und τ_1 sowie das Komplement von τ_0 dargestellt durch $[T_0, cap, T_1]$, $[T_0, cup, T_1]$, $[T_0, setminus, T_1]$ und $[compl, T_0]$.

- Werden die mengentheoretischen Terme τ_0 und τ_1 durch T_0 und T_1 dargestellt, so werden $\tau_0 \in \tau_1$ durch $[T_0, in, T_1]$, $\tau_0 = \tau_1$ durch $[T_0, =, T_1]$, $\tau_0 \subseteq \tau_1$ durch $[T_0, subseteq, T_1]$ und $\tau_0 \supseteq \tau_1$ durch $[T_0, supseteq, T_1]$ dargestellt.

- Die Regeln für aussagenlogische Verknüpfungen und Folgerungsketten sind die gleichen wie für aussagenlogische Formeln.

- Mengentheoretische Ungleichungsketten werden durch Listen dargestellt, in denen die auftretenden Terme sich mit den Relationszeichen subseteq, supseteq und = abwechseln, also etwa $(\tau_0 \cap \tau_1) \subseteq \tau_0 \subseteq (\tau_0 \cup \tau_1) = (\tau_1 \cup \tau_0)$ durch $[[A, cap, B], subseteq, A, subseteq, [A, cup, B], =, [B, cap, A]]$.

Arithmetische Terme, Formeln, Ungleichungs-, Kongruenz- und Umformungsketten

- Jede Variable ist ein arithmetischer Term, der durch sich selbst dargestellt wird.

- Jede Dezimaldarstellung einer natürlichen Zahl ist ein arithmetischer Term, der durch sich selbst dargestellt wird.

- Werden die arithmetischen Terme τ_0, τ_1, τ_2 durch T_0 bzw. T_1 dargestellt, so werden $-\tau_0$ durch $[-, T_0]$, $(\tau_0 + \tau_1)$ durch $[T_0, +, T_1]$, $(\tau_0 * \tau_1)$ durch $[T_0, *, T_1]$, $(\tau_0 - \tau_1)$ durch $[T_0, -, T_1]$, (τ_0/τ_1) durch $[T_0, /, T_1]$ und $(\tau_0 \hat{} \tau_1)$ (also die Exponentiation) durch $[T_0, exp, T_1]$ dargestellt.

- Werden die arithmetischen Terme τ_0 und τ_1 durch T_0 bzw. T_1 dargestellt, so werden $\tau_0 R \tau_1$ für $R \in \{=, <=, <, >=, >\}$ durch $[T_0, R, T_1]$, $\tau_0 | \tau_1$ durch $[T_0, div, T_1]$, $\tau_0 \sim (\tau_1)\tau_2$ durch $[T_0, [congr, T_1], T_2]$ dargestellt. Ferner stehen [square,T_0] für "τ_0 ist eine Quadratzahl", [cube,T_0] für "τ_0 ist eine Kubikzahl", [even,T_0] für "τ_0 ist gerade" und [odd,T_0] für "τ_0 ist ungerade".

- Die Regeln für aussagenlogische Junktoren sind dieselben wie im Bereich Aussagenlogik; die Regeln für Ungleichungs- und Kongruenzketten sind dieselben wie im Bereich Mengentheorie mit den Relationszeichen $=, >=, >, <=, <$ bzw. $\sim$, $=$ und $|$ anstelle von $=$, $\subseteq$ und $\subseteq$.

- Wird die arithmetische Formel ϕ durch F repräsentiert, so werden $Ax : \phi$ durch $[all, x, F]$ und $Ex : \phi$ durch $[ex, x, F]$ dargestellt.

- Umformungsketten werden ähnlich repräsentiert wie Folgerungsketten in der Aussagenlogik, mit dem Unterschied, dass nach den Folgerungszeichen eine (möglicherweise leere) Liste kommt, die die "Regieanweisung" repräsentiert. So wird etwa $(a = b) \Leftrightarrow (+1)(a + 1 = b + 1) \Leftrightarrow (*2)(2 * a + 2 = 2 * (b + 1)$ repräsentiert durch $[[a, =, b], <=>, [+, 1], [[a, +, 1], =, [b, +, 1]], <=>, [*, 2], [[[2, *, a], +, 2], =, [2, *, [b, +, 1]]]]$.

Bemerkung: Ein Ausdruckstyp "arithmetische Folgerungskette", der etwa Ausdrücke wie "$(4|a) \Rightarrow (2|a) \Leftrightarrow (4|a^2)$" umfassen könnte, ist derzeit nicht vorgesehen und scheint auch in natürlichsprachlichen mathematischen Texten kaum verwendet zu werden.

Geometrische Terme und Formeln

- Jede Variable ist ein geometrischer Term, der durch sich selbst dargestellt wird.

- Sind τ_0, τ_1 geometrische Terme, dargestellt durch T_0 und T_1, so bezeichnet [poi,[T_0,T_1]] ("point of intersection") den Schnittpunkt von τ_0 und $\tau 1$, [line,[T_0, T_1]] die Gerade durch τ_0 und τ_1, [segment,[T_0, T_1]] die Strecke zwischen τ_0 und τ_1, [par,[T_0,T_1]] die Parallele zu τ_0 durch τ_1, [mipt,T_0] den Mittelpunkt von τ_0 und [mittellot,T_0] das Mittellot der Strecke τ_0.

- Sind τ_0, τ_1, ..., τ_n geometrische Terme, dargestellt durch T_0, T_1,..., T_n, so steht [T_0,parallel,T_1] für "τ_0 und τ_1 sind parallel", [T_0,orthogonal,T_1] für "τ_0 ist orthogonal zu τ_1", [collinear,[T_0,...,T_n]] für "τ_0,...,τ_n sind kollinear" sowie [congruent,[$T_0, ..., T_n$]] für "τ_0, ..., τ_n sind kongruent"; ferner gibt es Darstellungen für (echte, rechtwinklige, gleichschenklige und gleichseitige) Dreiecke sowie für die verschiedenen Viereckstypen aus dem "großen Haus der Vierecke" wie Quadrate, Parallelogramme, Rauten, symmetrische Trapeze etc., die wir hier nicht ausführen.

- Die Regeln für aussagenlogische Junktoren sind dieselben wie im Bereich Aussagenlogik.

Bemerkung: Ein Ausdruckstyp "geometrische Umformungs- und Folgerungsketten" ist derzeit nicht vorgesehen.

Der Übersetzungsalgorithmus

Der Übersetzungsalgorithmus für die Konvertierung von Parsingbäumen in das Listenformat funktioniert nun auf die offensichtliche Weise über eine Rekursion.

Beispielsweise lautet die Prolog-Klausel des Prädikates pf_pt_to_list, die in der Übersetzung von Parsingbäumen (parsing trees, "pt") aussagenlogischer Formeln (propositional formula, "pf") den Junktor ("j") der Biimplikation verarbeitet, wie folgt:

```
pf_pt_to_list(pf(ob('('),Left,j(<->),Right,cb(')')),[Left1,<->,Right1]):-
pf_pt_to_list(Left,Left1),!,
pf_pt_to_list(Right,Right1),!.
```

Es werden also unter Auslassung der Klammern der Teil links und rechts vom Junktor separat konvertiert, anschließend werden die Ergebnisse mit einem $<->$ in der Mitte in eine neue Liste geschrieben. Die anderen Junktoren, einschließlich der Negation, funtionieren analog.

Als Veranschaulichung für die Regeln, die bei der Konvertierung natürlicher Sprache zum Einsatz kommen, geben wir eine der Klauseln, mit denen Zuschreibungen von Eigenschaften in Sätzen wie "x ist ungerade" verarbeitet werden:

```
formalize_content_pt(claim(property(var(X),jvp(_),adj(Adj))),[Abbr,[Y]],List):-

abbreviation(Adj,Abbr),!,
member([X,Y],List),!.
```

Für Aussagen ("claim"), vom Typ einer Eigenschaftszuschreibung ("property"), welche einer Variablen ("var") X eine Eigenschaft ("adj") Adj mithilfe einer Verbalphrase ("jvp") zuschreiben, wird also die zum Adjektiv gehörige interne Abkürzung (für "ungerade" ist das z.B. "odd") ermittelt und dann entsprechend ein prädikatenlogischer Ausdruck gebildet (bei "x ist ungerade" ergäbe sich etwa "[odd,[x]]"). (Die dritte Zeile der Klausel dekodiert ein internes Darstellungsformat für Variablen und ist für einen Einblick in die Funktionsweise der Klausel unerheblich.) Entsprechende Regeln existieren für andere Wortreihenfolgen, so dass neben "Damit gilt: x ist ungerade." auch "Damit ist x ungerade.", "Wenn x ungerade ist" etc. verarbeitet werden können.

6.4 Die Annotation

Das Textannotationsmodul verwendet die Parsing- und Formalisierungsroutinen, um den Eingabetext in ein semiformales Zwischenformat zu transformieren, das im wesentlichen wie die im Naproche-System verwendeten "Proof Representation Structures" (PRS, siehe etwa Cramer [53]) funktioniert. Jeder Satz wird dabei als 6-Tupel

[Zeilennummer,Referenten,Zeilenname,Funktion,Unterfunktion,Inhalt]

repräsentiert; Zeilennummer ist die Nummer der Zeile, in der der Satz (oder die Annotation etc.) steht. Referenten ist eine Liste der auftretenden mathematischen Referenten, also Variablen. Zeilenname enthält die Bezeichnung für die Zeile, falls eine eingeführt wurde (siehe den Abschnitt zur Sprache von Diproche weiter unten für die *-Notation für Zeilennamen und ihren Einsatz bei begründeten Behauptungen). Funktion ist die Satzkategorie, die wir oben eingeführt haben ("ang" für "Annahme", "beh" für "Behauptung" etc.), Unterfunktion die jeweilige Unterkategorie ("bem" für "Beweisanfangsmarker", "bbh" für "begründete Behauptung etc.). Die Extraktion von Referenten und Zeilennamen erfolgt durch einfache rekursive Algorithmen, die die Darstellung des Inhalts im Listenformat nach entsprechenden Zeilennamen und Variablen (lateinischen Buchstaben, optional gefolgt von Zahlen) durchsuchen.

So würde etwa der Satz "Angenommen, n ist ungerade." (wenn er am Anfang des Textes steht, also die Zeilennummer 1 erhält) repräsentiert durch:

[1,[n],[],ang,[],[odd,[n]]]

Der gesamte Text wird dann dargestellt als eine Liste von solchen Listen.

Wir verdeutlichen den gesamten Durchlauf durch die Konversionsroutinen noch einmal an einem Beispieltext:

> Es seien a, b und c Aussagen. Wir zeigen: Dann gilt $a \to (b \to a)$.
>
> Beweis: Es gelte a. Dann folgt $b \to a$. qed.

Dieser Beweistext – ein String – wird zunächst in die Listentextdarstellung konvertiert und nimmt damit folgende Gestalt an:

```
[[es,seien,a,b,c,aussagen],
[wir,zeigen,dann,gilt,[a,− >,[b,− >,a]]],
[abs],
[abs],
[beweis],
[es,gelte,a],
[dann,folgt,[b,− >,a]],
[qed]]
```

Die Ausgabe der Textannotation (die Ausgabe des Parsers übergehen wir hier aus Gründen der Übersichtlichkeit) ist dann folgende:

```
[[1,[a,b,c],[],ang,dkl,[[a,is,prop],[b,is,prop],[c,is,prop]]],
[2,[b,a],[],ann,goal,[a,− >,[b,− >,a]]],
[3,[],[],ann,abs,[]],
[4,[],[],ann,abs,[]],
[5,[],[],ann,bam,[]],
[6,[a],[],ang,[],a],
[7,[b,a],[],beh,[],[b,− >,a]],
[8,[],[],ann,bem,[]]]
```

Diese Darstellung eines Beweistextes bezeichnen wir im folgenden auch als “annotiertes Format”.

6.5 Ermittlung der Textstruktur

Das Textstruktur-Modul hat die Funktion, die Abhängigkeitsrelationen oder Zugänglichkeitsrelation zwischen den Zeilen eines Beweistextes zu ermitteln und in Form eines gerichteten Graphen auszugeben. Eine Zeile J ist dabei von einer Zeile I aus zugänglich, falls Zeile I eine Annahme enthält, J größer ist als I und die in Zeile I getroffene Annahme vor Zeile J nicht geschlossen wurde. Der vom Textstrukturmodul produzierte Graph wird dann bei der Erzeugung von Anfragen an den automatischen Beweiser verwendet, um festzustellen, welche Behauptungen und Annahmen für die Ableitung des aktuellen Schrittes zur Verfgung stehen.

Die formalen Regeln für die Zugänglichkeitsrelation müssen dabei so gewählt sein, dass sie einerseits der üblichen Darstellungspraxis möglichst weit entgegen kommen und andererseits leicht und rasch erlernbar sind. In Diproche ist die Zugänglichkeit wie folgt geregelt: Die Zeile I ist von der Zeile J aus zugänglich, wenn I eine Annahme enthält und eine der beiden folgenden Bedingungen gilt:

- I und J stehen im gleichen Absatz und J kommt im Text nach I.
- I steht in einem Absatz, der mit einem Beweisanfangsmarker beginnt und J kommt vor dem Beweisendmarker, der den durch ihn begonnen Beweis beendet.

Die Zugänglichkeitsrelation ist also aus der Darstellung T eines Textes im annotierten Format leicht abzulesen: Zunächst wird für gegebene natürliche Zahlen I und J geprüft, ob $I < J \leq L$, wobei L die Länge der Liste F ist. Falls nicht, ist I von J aus nicht zugänglich. Andernfalls wird geprüft, ob das I-te Element von F die Funktion "ang" hat; falls nicht, ist I von J aus nicht zugänglich. Andernfalls werden die beiden Bedingungen oben geprüft: Befindet sich keine Absatzannotation zwischen I und J, so ist die erste Bedingung erfüllt und I ist von J aus zugänglich. Andernfalls wird geprüft, ob sich zwischen I und der letzten Absatzannotation vor I ein Beweisanfangsmarker befindet; ist das nicht der Fall, ist I von J aus nicht zugänglich. Ist das der Fall, so wird der Beweis nun von diesem I aus bis J durchlaufen, wobei zu einer Variablen mit Startwert 1 für jeden Beweisanfangsmarker 1 addiert und für jeden Beweisendmarker 1 subtrahiert wird. Ist der Wert dieser Variablen bei Zeile J noch ≥ 1, so wurde der relevante Beweisanfangsmarker noch nicht geschlossen und I ist von J aus zugänglich, andernfalls nicht. Der Zugänglichkeitsgraph $\mathcal{G}(T)$ eines Textes T im annotierten Format ist dann die Menge aller Paare (I, J) natürlicher Zahlen, für die I von J aus in T zugänglich ist.

Der auf diese Weise ermittelte Zugänglichkeitsgraph ermöglicht es, zu jeder Zeile die für diese Zeile verfügbaren Annahmen zu identifizieren. Allerdings werden bei einem Beweisschritt natürlich gewöhnlich auch Behauptungen verwendet, die zuvor abgeleitet wurden. Die Ermittlung der verfügbaren Behauptungen verwendet ebenfalls den Zugänglichkeitsgraphen $\mathcal{G}(T)$ mit Eckenmenge $\mathcal{E}(T)$ und Kantenmenge $\mathcal{K}(T)$: In Zeile J sind alle diejenigen Zeilen $I < J$ mit Behauptungen verfügbar, für die gilt, dass $\{k : (i, k) \in \mathcal{G}(T)\} \subseteq \{k : (j, k) \in \mathcal{K}(T)\}$, also alle, von denen aus lediglich solche Annahmen zugänglich sind, die auch von J aus zugänglich sind. Verfügbar sind also an jeder Textstelle alle zuvor gemachten Behauptungen, die unter einer Teilmenge der aktuell verfügbaren Annahmen aufgestellt wurden.

6.6 Erzeugung von Beweiserabfragen

Das Modul "NaturalToTask" ermittelt zu einem Text T im annotierten Format, dem zugehörigen Zugänglichkeitsgraphen $\mathcal{G}(T)$ und einem Zeilenindex I, der maximal der Anzahl der Zeilen von T entspricht, das zugehörige Beweisziel, das

dann im nächsten Schritt an den ATP weitergegeben wird. Das Beweisziel wird repräsentiert als geordnetes Paar

[[Annahmen,Behauptungen,Schritte,Deklarationen],Ziel],

dessen erste Komponente ein 4-Tupel aus den verfügbaren Annahmen, Behauptungen, den bisher erreichten Beweiszielen und den verfügbaren Deklarationen (Deklarationen gelten als Annahmen und werden im Sinne der Zugänglichkeitsrelation entsprechend behandelt) und dessen zweite Komponente die in Zeile I aufgestellte Behauptung ist.

Die Beweiseranfrage ist trivial, wenn I weder eine Behauptung noch eine Deklaration mit inhaltlicher Annahme enthält. In diesem Fall kann die entsprechende Zeile bei der logischen Prüfung übergangen werden.

Enthält die Zeile I eine Behauptung ("beh"), mit Inhalt C, so wird $\mathcal{G}(T)$ verwendet, um die Liste L_A der Zeilen mit verfügbaren Annahmen und die Liste L_B der Zeilen verfügbaren Behauptungen zu ermitteln; L_B ist dann die zweite Komponente des Quadrupels aus der obigen Darstellung des Beweisziels. Bei L_A wird noch zwischen Deklarationen und einfachen Annahmen unterschieden, wobei Deklarationen in die vierte, einfache Annahmen in die erste Komponente eingefügt werden. Bei Deklarationen mit inhaltlicher Annahme wird die inhaltliche Annahme der ersten, die Deklarationen hingegen der vierten Komponente hinzugefügt.[10]

Enthält die Zeile I eine Deklaration mit inhaltlicher Annahme und sind $x_1, ..., x_n$ die deklarierten Variablen mit Typen $T_1, ..., T_n$ und ϕ die Annahme, so ist das Beweisziel $\exists x_1 \in T_1, ..., x_n \in T_n \phi$, während das Quadrupel aus Annahmen, Behauptungen etc. sich wie für Behauptungen ergibt.

Die Liste der bisherigen Beweisschritte schließlich ergibt sich so: Für jede Zeile mit Index $J < I$, die eine Behauptung B enthält, wird die Liste $(A_1, ..., A_n)$ der von J aus zugänglichen Annahmen ermittelt (wobei die Annahmen in der Reihenfolge ihres Auftretens aufgelistet sind). Dann wird der Liste der Beweisschritte die Aussage $(A_1 \rightarrow (A_2 \rightarrow ... \rightarrow (A_n \rightarrow B)...))$ hinzugefügt. Auf diese Weise stehen die durch bisherige Beweisschritte "implizit" erreichten Implikationen für die weitere Argumentation zur Verfügung. Ist etwa unter der Annahme A (aber keiner weiteren) die Aussage B hergeleitet worden, so kann im weiteren Beweis nun $A \rightarrow B$ verwendet werden.

Für den Beweistext

> Es seien a, b ganze Zahlen. Angenommen, a+b ist gerade. Dann ist a+b+1 ungerade. Also ist auch $(a + b + 1)^2$ ungerade.

[10] So würde etwa bei "Es seien a, b ganze Zahlen mit $a|b$." die Deklarationen $[a, int]$ und $[b, int]$ zur Liste der Deklarationen, die Annahme $a|b$ hingegen zur Liste der Annahmen hinzugefügt.

ergibt sich für die letzte Behauptung "Also ist auch $(a+b+1)^2$ ungerade" etwa folgendes Beweisziel:

```
[[[even,[[a,+,b]]]], (Liste der verfügbaren Voraussetzungen)
[[odd,[[[a,+,b],+,1]]]], (Liste der verfügbaren Behauptungen)
[[[even,[[a,+,b]]],→,[odd,[[[a,+,b],+,1]]]]], (Liste der bisherigen Beweisziele)
[[a,is,int],[b,is,int]], (Liste der verfügbaren Deklarationen)
[odd,[[[a,+,b],+,1],exp,2]]] (Beweisziel)
```

6.7 Der automatische Beweiser

Der automatische Beweiser (automated theorem prover, ATP) nimmt die im letzten Schritt erzeugte Beweiserabfrage $[V,B]$ aus einer strukturierten Liste V von verfügbaren Voraussetzungen und einer Behauptung B entgegen und versucht, die Behauptung B mit einer Reihe zur Verfügung stehender elementarer Beweisschritte aus der Voraussetzungsliste V abzuleiten.

Wir erläutern zunächst die Funktionsweise einer Inferenzregel im Prolog-Code, ehe wir auf spezielle Arten von ATP-Regeln eingehen, die in Diproche eingesetzt werden.

Wir betrachten dazu beispielhaft die Prolog-Klausel, die im aussagenlogischen ATP den Modus Ponens repräsentiert:

```
proof_step(mp1,Vss,B,1,_):-
member([A,→,B],Vss),
member(A,X).
```

Im Kopf dieser Regel ist "mp1" die Signatur der Regel; über diese Signatur können Listen von ATP-Regeln für verschiedene Aufgaben zugelassen oder ausgeschlossen werden, wodurch die weiter oben besprochene Konstruktion von "Schwierigkeitsgraden" ermöglichst wird. Vss bezeichnet die Liste der verfügbaren Voraussetzungen (im Bereich Aussagenlogik ist die Unterscheidung nach Annahmen, früheren Behauptungen und zuvor erreichten Beweiszielen unnötig), B die fragliche Behauptung. Die beiden anderen Komponenten sind für das inhaltliche Verständnis der Regel nicht erheblich. Die Klausel sagt nun, dass B mithilfe von mp1 aus Vss gefolgert werden kann, wenn Vss eine Liste der Form [A,→,B] enthält und zugleich deren erstes Element A. Bei der Anwendung der Regel wird die Prolog-interne Unifikation genutzt: Für jedes Element von Vss wird geprüft, ob es eine dreielementige Liste ist, deren zweites Element → ist und deren drittes Element B ist. Ist das der Fall, wird das erste Element an die

Variable A gebunden; im nächsten Schritt wird dann geprüft, ob das nun an die Variable A gebundene Objekt ebenfalls ein Element der Liste Vss ist. Falls ja, ist die Regel erfolgreich. Falls nicht, sorgt das Prolog-eigene Backtracking dafür, dass das nächste Element von Vss ausprobiert wird. Sind alle Elemente von Vss ausprobiert worden ohne dass ein den Bedingungen genügendes gefunden wurde, liefert die Regel den Wert "false" zurück.

Jedes ATP-Modul besteht nun aus einer Liste solcher Regeln. Für eine Beweiserabfrage werden diese nacheinander, in der Reihenfolge ihres Auftretens in der Liste, ausprobiert. Kann eine davon erfolgreich angewendet werden, meldet der ATP "true" zurück und der Beweisschritt gilt als verifiziert. Wurden alle Regeln erfolglos ausprobiert, meldet der ATP "false" zurück und der Beweisschritt gilt als nicht verifiziert.

6.7.1 Aussagenlogische ATP-Regeln

Logische Verknüpfungen wie Negation, Konjunktion, Disjunktion, Implikation und Äquivalenz treten in sämtlichen mathematischen Begründungszusammenhängen aus. Die aussagenlogischen Inferenzregeln, die den Umgang damit ermöglichen, sind daher für alle Themengebiete, die in Diproche behandelt werden, grundlegend.

Diese Inferenzregeln sind im aussagenlogischen Teil des ATP implementiert und fallen in folgende Kategorien:

- Sonderregeln zur Behandlung begründeter, multipler und multipler begründeter Behauptungen.
- Regeln, die die Semantik der Junktoren festlegen, etwa die, dass man aus einer Konjunktion ihre Glieder folgern kann oder dass die Konjunktion $(A \wedge B)$ gefolgert werden kann, wenn ihre beiden Glieder zu den Voraussetzungen gehören, dass eine Disjunktion aus ihren beiden Gliedern gefolgert werden kann etc.
- Grundlegende Eigenschaften der Junktoren wie Kommutativität von $\wedge$, $\vee$ und $\leftrightarrow$ oder Assoziativität von $\wedge$ und $\vee$.
- Regeln, die die Koordination der Junktoren untereinander betreffen, etwa die Distributivität von $\vee$ über die übrigen Junktoren oder die de Morganschen Regeln.
- Regeln, die elementare Beweisstrategien abbilden, wie der Modus Ponens, der Beweis durch Widerspruch (also die Folgerbarkeit von $\neg A$ aus $A \to \bot$), Kontrapositionsbeweise (also die Folgerbarkeit von $A \to B$ aus $\neg B \to \neg A$), Beweise durch Fallunterscheidung (Folgerbarkeit von C aus $A \vee B$, $A \to C$ und $B \to C$ sowie aus $A \to C$ und $\neg A \to C$) etc.

Metaregeln

Der Ansatz eines mit zahlreichen einzeln aufrufbaren Regeln ausgestatteten ATP verfolgt den Zweck, die zulässigen Schritte mit dem Lernfortschritt flexibel anpassen zu können. Allerdings ist nicht jede Art von Fortschritt in dieser Richtung durch eine einfache Schlussregel abbildbar. Ist es etwa bei den ersten Erfahrungen mit logischem Schließen noch sinnvoll, die Glieder einer Konjunktion in einem separaten Schritt abzuspalten, so sollten für fortgeschrittene Aufgaben sämtliche Glieder einer einmal bewiesenen oder vorausgesetzten Konjunktion für weitere Folgerungen zur Verfügung stehen. Um auch solche Fortschritte abbilden zu können, gibt es einige ATP-Regeln – hier als "Metaregeln" bezeichnet -, die die verfügbaren Voraussetzungen und das Beweisziel zunächst vorbereitend verarbeiten – etwa unter gewissen Operationen abschließen – und anschließend die ATP-Regeln, die im vorigen Abschnitt besprochen wurden, auf das so modifzierte Beweisziel anwenden. Zu den derzeit verwendeten Metaregeln gehören insbesondere der eben besprochene Abschluss unter Konjunktionsgliedern sowie der Abschluss unter Modus Ponens, der, wann immer ϕ und $\phi \rightarrow \psi$ zu den Voraussetzungen gehören, diesen auch ψ hinzufügt.

6.7.2 Quantorenlogische Regeln

Die quantorenlogischen Regeln sind, wie die aussagenlogischen Regeln, in allen Themengebieten außer der Aussagenlogik verfügbar. Zu den quantorenlogischen Regeln gehören insbesondere folgende (die zu den typischen Grund- bzw. abgeleiteten Regeln für den Umgang mit Quantoren in logischen Kalkülen wie etwa dem Hilbert-Kalkül[11] gehören):

- Die Instantiierung universell quantifizierter Aussagen, also die Folgerung von $\phi(a_1, ..., a_n)$ aus $\forall x_1, ..., x_n \phi(x_1, ..., x_n)$; bei der Anwendung dieser Regel auf typenbeschränkte Allquantoren (wie "für alle reellen Zahlen n" oder "es gibt eine Menge x"), also der Ableitung von $\phi(a_1, ..., a_n)$ aus $\forall x_1 \in T_1, ..., x_n \in T_n \phi(x_1, ..., x_n)$, wird zusätzlich geprüft, ob a_i für alle $i \in \{1, 2, ..., n\}$ vom Typ T_i ist.

- Die Ableitung allquantifizierter Aussagen $\forall x_1, ..., x_n \phi(x_1, ..., x_n)$ aus der Aussage $\phi(x_1, ..., x_n)$, sofern $x_1, ..., x_n$ nicht frei in den bei der Ableitung von ϕ verfügbaren Voraussetzungen vorkommen (mit einer entsprechenden Variante für typenbeschränkte Allquantoren).

- Die Ableitung der existenzquantifizierten Aussage $\exists x_1, ..., x_n \phi(x_1, ..., x_n)$

[11] Siehe etwa Fitting [70], S. 146f.

aus der Instanz $\phi(a_1, ..., a_n)$ (mit einer entsprechenden Variante für typenbeschränkte Existenzquantoren).

- Die Ableitung von $\phi(a_1, ..., a_n)$ aus $\exists x_1, ..., x_n \phi(x_1, ..., x_n)$, sofern $a_1, ..., a_n$ Variablen sind, die in den verfügbaren Voraussetzungen nicht vorkommen (also "neue" Variablen sind), d.h. die Möglichkeit, Beispiele zu wählen und zu benennen.

- Regeln für die Negation von Quantoren, etwa die Äquivalenz von $\neg\forall x\phi$ und $\exists x\neg\phi$.

6.7.3 Gebietsspezifische ATP-Regeln

Wie bereits oben besprochen arbeitet jedes Themengebiet mathematischer Übungen mit jeweils spezifischen Inferenzregeln, die in Submodulen des ATP implementiert sind. Das sind derzeit die Themengebiete "Gruppentheorie", "Funktionen und Relationen", "elementare Zahlentheorie" (mit weiteren Submodulen zu Teilbarkeit und Kongruenzen), "Geometrie", sowie "Boolesche Mengenlehre". Wir beschränken uns in der Darstellung auf die letzten drei (die auch bereits in der Lehre eingesetzt wurden) und besprechen hier kurz das auf Boolesche Mengenlehre spezialisierte ATP-Modul; die ATP-Module für elementare Zahlentheorie und Geometrie werden in der Behandlung der entsprechenden "Spielwiesen" weiter unten behandelt.

Die Inferenzregeln des ATP für Boolesche Mengenlehre lassen sich wie folgt einteilen:

- Inferenzregeln, die die Semantik der mengentheoretischen Operatoren festlegen, etwa die Folgerbarkeit von $x \in (A \cap B)$ aus $x \in A$ und $x \in B$.

- Grundlegende Eigenschaften der mengentheoretischen Operatoren wie Kommutativität und Assoziativität der Vereinigung und des Schnittes, die Distributivität von $\cup$ über $\cap$ etc.

- Inferenzregeln, die die Semantik der mengentheoretischen Relationen $=$, $\subseteq$ und $\supseteq$ festlegen, insbesondere das Extensionalitätsgesetz (also die Folgerbarkeit von $A = B$ aus $A \subseteq B$ und $B \subseteq A$).

- Grundlegende Eigenschaften der mengentheoretischen Relationen, etwa Transitivität und Reflexivität von $=$ und $\subseteq$ sowie die Symmetrie der Gleichheit.

- Inferenzregeln, die die Semantik der leeren Menge $\emptyset$ festlegen und Regeln, die die Verknüpfung der leeren Menge durch mengentheoretische Operatoren betreffen (z.B. $A \cap \emptyset = \emptyset$).

- Inferenzregeln zur Fixierung der Semantik des kartesischen Produktes.

- Regeln für die Umformung von Mengentermen, wie etwa die de Morganschen Regeln.

- Regeln, die die Substituierbarkeit von mengentheoretischen Teiltermen durch ihnen koextensionale Terme garantieren.

6.7.4 Verifikation von Term- und Gleichungsumformungen

Obwohl deduktive Schritte, die gewöhnlich als Termumformungen bezeichnet werden, letztlich als geraffte Darstellungsform von deduktiven Schrittfolgen angesehen werden können, bilden sie hinsichtlich ihrer üblichen Darstellungsweise eine Sonderform des mathematischen Argumentierens, die durch den Ansatz eines (kontrollierten) ATP nur schwer abzubilden ist. Zwar geht jede gültige (d.h. werterhaltende) Umformung arithmetischer Terme auf eine begrenzte Anzahl von Techniken – wie etwa das Ausklammern, das Ausmultiplizieren, das Kürzen, das Erweitern, das Fortlassen von Summanden mit gleichem Betrag, aber verschiedenen Vorzeichen, das Ausrechnen numerischer Subterme etc. – zurück, so dass es im Prinzip möglich wäre, einen Vorrat an Regeln zu formulieren und dann Schrittfolgen darauf zu überprüfen, ob jeder Schritt durch eine davon gerechtfertigt werden kann. Die übliche Praxis im Umgang mit arithmetischen Termumformungen ist es indes, Schrittfolgen, die sich durch die Anwendung solcher Standardtechniken rechtfertigen lassen, in einen Schritt zusammenzufassen. So mag es zwar im Hinblick auf die Nachvollziehbarkeit wünschenswert sein, in einer Gleichung wie $\frac{ab+a^2+ba+b^2}{2b+a-b} = a + b$ noch einige Zwischenschritte einzuzufügen, zwingend erforderlich ist es sicherlich nicht. Wollte man darauf bestehen, in jedem Schritt jeweils nur eine elementare Umformungsregel anzuwenden, dehnt sich so eine Gleichung unnötig aus, etwa zu $\frac{ab+a^2+ba+b^2}{2b+a-b} = \frac{ab+a^2+ba+b^2}{b+a} = \frac{ab+a^2+ba+b^2}{a+b} = \frac{ab+a^2+ab+b^2}{a+b} = \frac{a^2+2ba+b^2}{a+b} = \frac{(a+b)^2}{a+b} = a + b$; erlaubt man hingegen die gleichzeitige Anwendung verschiedener Umformungsregeln auf verschiedene Teilterme oder auch die Anwendung mehrerer Umformungsregeln nacheinander, wird es für den/die BenutzerIn schnell unübersichtlich, was als Umformungsschritt akzeptabel ist. Nun sind Termumformungen zwar Bestandteil zahlreicher Aufgaben, für deren Einsatz Diproche konzipiert wurde – insbesondere in der elementaren Zahlentheorie und bei Aufgaben zur vollständigen Induktion – stehen jedoch selbst nicht im Mittelpunkt der Betrachtung. Eine gewisse Vertrautheit mit arithmetischen Umformungen – die ja bereits zum Stoff der Mittelstufe gehören – wird auf Seiten der/des BenutzerIn, die/der Beweisaufgaben in Angriff nimmt,

vorausgesetzt.[12] Es scheint daher wenig didaktischer Wert darin zu liegen, auf kleinschrittigen und explizierten Termumformungen zu bestehen. Für Diproche wurde daher die Entscheidung getroffen, jede korrekte Gleichheit von Termen, die sich unabhängig von Zusatzvoraussetzungen bezüglich der auftretenden Größen gilt, als Umformungsschritt zu akzeptieren.

In Diproche wird daher zwischen zwei Arten von Umformungen bzw. den sich daraus ergebenden Gleichheiten $T_0 = T_1$ arithmetischer Terme unterschieden, nämlich (i) solchen, die stets zu einer wahren Aussage führen, wenn man die in ihnen auftretenden Variablen mit reellen Zahlen instantiiert und (ii) solchen, die sich durch die Anwendung expliziter Annahmen über diese Variablen ergeben.

Schritte der Form (i) werden stets akzeptiert, wenn die zugehörige Termgleichung korrekt ist. Das Problem, die Korrektheit von Termgleichungen zwischen zwei Polynomen mit mehreren Variablen zu entscheiden, ist in der Komplexitätstheorie als "polynomial identity testing" (PIT) bekannt[13]; zur Lösung von PIT ist bislang kein effizienter (polynomialer) Algorithmus bekannt. Allerdings gehört PIT zu einer als "bounded-error probabilistic polynomial time" (BPP)[14] bekannten Klasse von Entscheidungsproblemen, die mit begrenzter Fehlerwahrscheinlichkeit in polynomialer Zeit probabilistisch entschieden werden können: Dazu werden einfach alle in den Termen auftretenden Variablen mit Zufallszahlen instantiiert und die sich ergebenden Werte verglichen.[15] Da zwei Polynome entweder gleich sind oder nur auf einer Nullmenge übereinstimmen, ergibt sich hieraus ein schnelles und theoretisch sehr zuverlässiges Verfahren zur Überprüfung von Termgleichheiten. Eben dieses Verfahren wird auch in Diproche verwendet. Hier ist allerdings zu beachten, dass einerseits in Diproche-Termen auch exponentielle Ausdrücke auftreten können und andererseits die Maschinenarithmetik nur begrenzt genau ist, so dass zwar Darstellungen verschiedener Funktionen noch immer zuverlässig unterschieden werden können, kumulierte Rundungsfehler aber dazu führen können, dass zwei verschiedene Darstellungen derselben Funktion zu unterschiedlichen Werten führen. Diproche wählt daher die einzusetzenden Werte aus einem begrenzten Bereich und vergleicht nur die ersten 4 Stellen der Ergebnisse. Um die sich daraus ergebende größere Unsicherheit zu kompensieren, werden mehrere probabilistische Tests hintereinander ausgeführt (was in der Komplexitätstheorie eine unter dem Namen "BPP-Algorithmus"[16] bekannte und gängige Technik ist, um die

[12] Diese Voraussetzung muss indes darum nicht bei allen Studierenden vorliegen. Die Förderung bzw. der Aufbau dieser durchaus wichtigen Kompetenz ist aber nicht Gegenstand dieser Arbeit.

[13] Siehe etwa [1], Example 16.1.

[14] Siehe etwa[1], Definition 7.1.

[15] Dieser Algorithmus ist als "Schwartz-Zippel-Algorithmus" bekannt, siehe etwa [1], Lemma A.25.

[16] Siehe etwa [1], Theorem 7.10.

Fehlerwahrscheinlichkeit probabilistischer Entscheidungsverfahren zu erhöhen). In der Praxis hat sich eine Anzahl von 10 Wiederholungen als guter Kompromiss zwischen einer raschen Ausführung und einer ausreichenden Sicherheit ergeben.

Zu den Schritten der Form (ii) gehören insbesondere Substitutionen, also Ersetzungen von Variablen – allgemeiner (Teil-)termen – durch Terme, die ihnen nach den verfügbaren Voraussetzung gleich sind. Diese sind in Diproche-Texten von den Umformungen vom Typ (i) strikt zu trennen. So wird unter den Voraussetzungen $a = 1$ und $b = 2$ zwar $a + b = 1 + 2 = 3$ akzeptiert, nicht aber $a + b = 3$. Hierin liegt gewiß ein Zugeständnis an die technischen Möglichkeiten auf Kosten der Natürlichkeit, allerdings eines mit verhältnismäßig leicht durchschaubaren Folgen für die akzeptierten Texte, deren Beherzigung überdies zur Lesbarkeit der akzeptierten Texte beitragen dürfte. Wie das Beispiel bereits zeigt, können mehrere Substitutionen an verschiedenen Termen simultan in einem Schritt durchgeführt werden, allerdings keine Iterationen solcher Substitutionen: Unter den Voraussetzungen $a + b = c$, $c = 3$ würde also etwa die Gleichungskette $a + b = c = 3$ akzeptiert, nicht aber $a + b = 3$.

Für die Umformung mengentheoretischer Terme steht einerseits kein ähnlich angenehmer probabilistischer Mechanismus zur Verfügung, andererseits greifen hier auch die didaktischen Argumente weniger, die im Fall der arithmetischen Terme für eine "liberale" Behandlung sprachen: Umformungen mengentheoretischer Terme sind aus der Schule kaum bekannt und die Vertrautheit mit ihnen kann bei den Studierenden nicht vorausgesetzt werden; überdies bilden sie in dem Bereich, in dem sie in Diproche auftreten – nämlich der Booleschen Mengenlehre – gerade den zentralen Gegenstand. Aus diesem Grund erfolgt die Verifikation von mengentheoretischen Termumformungen regelbasiert: In jedem Glied einer Kette von Termgleichheiten und Teilmengenbeziehungen wird das Mengenlehre-Modul des ATP herangezogen, das versucht, den Schritt als Instanz einer von zahlreichen Umformungsregeln aufzufassen. Diese Regeln können auf Teilterme angewendet werden, auch simultan auf mehrere in einem Schritt, so dass etwa die Gleichung $((a \cap a) \cup (b \cup b)) = (a \cup b)$ akzeptiert würde, da der ATP die Idempotenz des Schnittoperators $X \cap X = X$ als Schlussregel enthält. Iterationen solcher Regelanwendungen – also die Anwendung von Umformungsregeln auf Teilterme, die in diesem Schritt bereits umgeformte Teilterme enthalten – in einem Schritt zusammenzufassen gilt hingegen nicht als elementarer Schritt. Unserer Erwartung nach wird hierdurch ein didaktisch sinnvoller Standard für die schrittweise Umformung mengentheoretischer Terme gesetzt. Sollte die Lehrpraxis die Erfordernis zeigen, wäre die derzeitige Verfahrensweise natürlich leicht durch eine – angesichts der in den relevanten Anwendungen zu erwartenden geringen Zahl von Variablen und begrenzten Länge der Ausdrücke – ausreichend effiziente Verifikationsroutine für mengentheoretische

Termumformungen ersetzbar, die, analog zur oben beschriebenen Behandlung arithmetischer Terme, alle korrekten Gleichheiten akzeptiert, welche sich ohne Zusatzvoraussetzungen an die auftretenden Mengenvariablen ergeben.[17]

Verifikation von Implikationsketten

Diproche erlaubt in begrenztem Umfang die Darstellung mehrerer aufeinanderfolgender Schlüsse als formale "Schlusskette" (zur Syntax von Schlussketten in der Diproche-Syntax siehe den Abschnitt zur Sprache von Diproche weiter unten). So erlaubt (und verifiziert) Diproche etwa folgende Eingaben in den Bereichen Aussagenlogik bzw. Boolesche Mengenlehre:

> Es gilt $(a\&b) \Leftrightarrow (b\&a) \Rightarrow\sim (\sim b \vee \sim a) \Leftrightarrow (b\&a) \Rightarrow b$.

> Es gilt $((a \cap b) = (a \cup b)) \Leftrightarrow ((a \cup b) = (a \cap b)) \Rightarrow ((a \cup b) \subseteq (a \cap b))$.

Die Verifikation erfolgt hier einfach dadurch, dass für jede Implikation bzw. Äquivalenz separat geprüft wird, ob die Herleitung der jeweiligen Aussage aus verfügbaren Voraussetzungen und der jeweils anderen Aussage vom ATP durchgeführt werden kann.

Wiederum stellen hier die arithmetischen Implikationsketten einen Sonderfall dar. Bei diesen ist es zusätzlich möglich, den Implikations- bzw. Äquivalenzpfeilen "Regieanweisungen" hinzuzufügen, die den Umformungsschritt angeben, durch den die Äquivalenz bzw. Implikation erreicht wird. So wird etwa folgende Implikationskette von Diproche als formal korrekt akzeptiert und auch inhaltlich verifiziert:

> Es gilt $(a = b) \Leftrightarrow (+1)(a + 1 = 1 + b) \Rightarrow (^2)(a^2 + 2 * a + 1 = 1 + b^2 + 2 * b)$.[a]
>
> [a] Man beachte, dass das Quadrieren keine Äquivalenzumformung ist; ersetzt man $\Rightarrow$ in der zweiten Folgerung also durch $\Leftrightarrow$, wird die Umformungskette nicht mehr verifiziert.

Die Verifikation solcher "kommentierter Umformungen" erfolgt dadurch, dass (i) geprüft wird, ob die angegebene Umformung eine Implikation in die entsprechende Richtung zu folgern erlaubt (das ist bei Addition und Subtraktion stets der Fall; Multiplikationen erlauben zunächst nur eine Implikation von links

[17] Dazu übersetzt man die mengentheoretische Formel in eine logisch äquivalente aussagenlogische Formel und prüft diese mithilfe einer Wahrheitstafel. Tatsächlich wird eine solche Technik bei den mengentheoretischen Mathediktaten und Vereinfachungsaufgaben bereits eingesetzt.

nach rechts und führen nur dann zu einer Äquivalenz, wenn der Faktor von 0 verschieden ist, bei der Division ist die Operation nur zulässig, wenn bekannt ist, dass der Divisor von 0 verschieden ist, während etwa beim Quadrieren ohne weitere Voraussetzungen stets nur eine Implikation gefolgert werden kann) und (ii) ob die Seiten der sich ergebenden (Un-)gleichung sich durch die entsprechenden Operationen aus den Seiten der jeweils anderen (Un-)gleichung ergeben. Um $(a = b) \Leftrightarrow (+1)(a + 1 = 1 + b)$ zu verifizieren, wird also zunächst festgestellt, dass die Addition von 1 eine Äquivalenzumformung ist; anschließend wird das oben beschriebene probabilistische Verfahren zur Verifikation von Termumformungen verwendet, um zu prüfen, ob $a + 1 = a + 1$ und $b + 1 = 1 + b$ ist. Auf diese Weise müssen also Umformungen nicht zunächst rein syntaktisch angewendet werden, etwa dadurch, dass auf beiden Seiten rechts ein "+1" ergänzt wird, sondern lassen sich mit verschiedenen Termumformungen kombinieren, was insbesondere bei Schritten wie dem Streichen gleicher Summanden auf beiden Seiten oder dem Multiplizieren mit einem gemeinsamen Hauptnenner bei Brüchen einen deutlichen Zugewinn an Komfort und Natürlichkeit bedeutet.

In begrenztem Umfang können diese Regieanweisungen in fortgeschrittenen Schwierigkeitsgraden auch fortgelassen werden, dann nämlich, wenn auf beiden Seiten eine lineare Operation, also eine der Form $T \mapsto Ta + b$ mit $a, b \in \mathbb{R}$, vorgenommen wird. Solche Umformungen werden durch ein spezielles Modul identifiziert, wobei wiederum die probabilistische Methode zum Einsatz kommt: Um zu prüfen, ob $S_1 = T_1$ mithilfe einer linearen Operation aus $S_0 = T_0$ folgt, werden zunächst die Variablen von S_0 und S_1 eine gewisse endliche Anzahl n von Malen (für die Praxis hat sich $n = 10$ als sinnvoller Wert erwiesen) mit Zufallszahlen instantiiert, wodurch sich zwei n-stellige Wertvektoren $\vec{s_0} = (s_0^1, ..., s_0^{10})$ und $\vec{s_1} = (s_1^1, ..., s_1^{10})$ ergeben. Falls diese durch eine lineare Transformation $\vec{x} \mapsto \vec{x}a + b$ ineinander überführbar sind, sind a und b eindeutig bestimmt und können durch Auflösen des Gleichungssystems $as_0^1 + b = s_1^1$, $as_0^2 + b = s_1^2$ ermittelt werden. Anschließend werden zwei ebenfalls zufallsgenerierte 10-stellige Wertvektoren für T_0 und T_1 erzeugt und es wird geprüft, ob dieselbe Operation auch diese ineinander überführt. Durch diese Methode kann etwa der Schritt

> Es gilt $(2 * a + 3 = 2 * b + 1) \Leftrightarrow (a + 1 = b)$.

auch ohne explizite Angabe einer Umformung verifiziert werden.

6.8 Hinweise für BenutzerInnen

Wie im letzten Kapitel beschrieben, gibt es drei Arten von Hinweisen, die das System BenutzerInnen geben kann: Diejenigen (i), die von der Lehrperson mit der Aufgabe eingegeben werden und auf Anfrage ausgegeben werden, sind vom technischen Standpunkt aus trivial. Die automatisch erzeugten Zwischenschritte (iii) sind derzeit noch in der Entwicklung. Wir beschränken uns daher auf eine nähere Erläuterung der strategischen Hinweise (ii).

Wie bereits weiter oben erklärt, ermittelt der Tippgeber zunächst die aktuelle Beweisaufgabe, d.h. eine Liste Vss der verfügbaren Voraussetzungen und bereits bewiesenen Sätze und eine Liste Goals von Aussagen, die das aktuelle Beweisziel darstellen (zur Erklärung, warum hier eine Liste von Aussagen statt einer einzigen betrachtet wird siehe den entsprechenden Abschnitt im vorigen Kapitel).

Der Tippgeber besteht nun aus derzeit 33 Klauseln, die zu einem Paar (Vss,Goals) einen Hinweis in Textform generieren, wobei zu einem Paar i.A. mehrere Hinweise erzeugt werden. Ein typisches Beispiel ist etwa folgende Klausel:

```
hint(_,_PartialProof,_,[A,↔,B],Assmpts,[zeige,[B,→,A]]):-
member([A,→,B],Assmpts).
```

Diese Klausel generiert für das Beweisziel $A \leftrightarrow B$ den Hinweis, die Implikation $B \rightarrow A$ zu zeigen, sofern $A \rightarrow B$ sich bereits unter den verfügbaren Aussagen befindet.

Die Klauseln des Tippgebers gliedern sich in "zielbezogene" und "annahmenbasierte" Hinweise, je nachdem, ob sie auf Strategien abzielen, das Beweisziel zu erreichen oder darauf, die gegebenen Annahmen nutzbar zu machen. Sind sowohl zielbezogene als auch annahmenbasierte Hinweise verfügbar, wird der/dem BenutzerIn stets einer der zielbezogenen angezeigt. Wir betrachten beide Arten von Hinweisen separat.

Zu den zielbezogene Hinweisen gehören:

- Hinweise zur Quantorenlogik, wie etwa der, bei einem Ziel der Form $\forall x \phi(x)$ zunächst ein beliebiges x zu fixieren und dann für dieses zu zeigen, dass $\phi(x)$ gilt oder den, $\exists x \phi(x)$ durch einen Widerspruchsbeweis mit Annahme $\forall x \neg \phi(x)$ anzugehen.

- Hinweise zur Aussagenlogik, insbesondere: (i) Die Möglichkeit der Aufteilung einer Konjunktion in zwei Teilaussagen, (ii) für Implikationen $\phi \rightarrow \psi$ die Möglichkeiten, ϕ anzunehmen und ϕ zu folgern, mit Kontraposition zu arbeiten oder aus $\phi \wedge \neg \psi$ einen Widerspruch abzuleiten, (iii) für

Disjunktionen ($\phi \vee \psi$) eines der Disjunktionsglieder zu beweisen oder aus der Negation beider einen Widerspruch abzuleiten, eine Äquivalenz in zwei Implikationen aufzuspalten oder eine Negation $\neg\phi$ dadurch zu beweisen, dass aus der Annahme ϕ auf einen Widerspruch hingearbeitet wird.

- Generelle Hinweise, wie etwa der, das Gegenteil der fraglichen Aussage anzunehmen und auf einen Widerspruch hinzuarbeiten.

Zu den annahmenbasierte Hinweisen gehören:

- Der Hinweis auf die Abspaltung der Glieder einer Konjunktion $\phi \wedge \psi$.
- Der Hinweis, dass sowohl ϕ als auch $\phi \rightarrow \psi$ zu den verfügbaren Aussagen gehören, sofern ψ noch nicht dazu gehört.
- Der Hinweis, bei einer Disjunktion ($\phi \vee \psi$) unter den verfügbaren Aussagen eine Fallunterscheidung danach zu machen, welche der beiden Aussagen gilt.
- Der Hinweis, Voraussetzungen so abzuändern, dass die obigen Regeln anwendbar werden, etwa eine negierte Disjunktion oder Implikation durch Anwendung der de Morganschen Regeln in eine Konjunktion umzuformen und dann aufzuspalten, eine Implikation als Disjunktion zu schreiben und eine Fallunterscheidung anzuwenden etc.

6.9 Rückmeldungen

Das Feedback-Modul generiert Rückmeldungen zu einem gegebenen Beweistext. Dabei wird gemeldet:

1. Ob alle verwendeten Zeichen zum Vorrat der Zeichen gehört, die Diproche verarbeiten kann (insbesondere sind derzeit weder Umlaute noch ein ß in Diproche-Texten zulässig). (Falls nicht, wird die Verarbeitung an dieser Stelle abgebrochen und es findet keine weitere Rückmeldung statt außer einer Liste der unbekannten Zeichen).
2. Ob alle verwendeten Wörter zu den im Lexikon von Diproche enthaltenen Schlüsselwörtern gehören. (Falls nicht, wird die Verarbeitung an dieser Stelle abgebrochen und es findet keine weitere Rückmeldung statt außer einer Liste der unbekannten Wörter).

3. Ob alle auftretenden Formeln korrekt gebildet wurden, d.h. vom Formelparser (s.o.) verarbeitet und in das interne Listenformat konvertiert werden konnten. (Falls nicht, wird die Verarbeitung an dieser Stelle abgebrochen und es findet keine weitere Rückmeldung statt außer einer Liste der nicht korrekt gebildeten Formeln).

4. Ob alle Sätze syntaktisch korrekt gebildet wurden, so dass das annotierte Format generiert werden konnte. (Falls nicht, wird die Verarbeitung an dieser Stelle abgebrochen und es findet keine weitere Rückmeldung außer der Anzeige des ersten nicht verarbeitbaren Satzes.)

Sind die Schritte (1)-(4) ohne Fehlermeldung durchgeführt worden, erfolgt die Rückmeldung "Der Text konnte sprachlich verarbeitet werden". Wenn die Verarbeitung für einen Satz mehr als 10 Sekunden dauert, wird die Verarbeitung mit einer entsprechenden Meldung abgebrochen.

5. Ob alle in einer Zeile auftretenden Referenten (z.B. Variablen) zuvor eingeführt wurden, wobei "zuvor" heißt, dass die fragliche Zeile von der Zeile aus, in der die Einführung stattfand, zugänglich sein muss und ob die erzeugten Sätze logisch sinnvoll sind, d.h. ob der Inhalt einer Annahme oder Behauptung überhaupt eine Aussage ist und ob bei Funktionen alle Argumente vom richtigen Typ sind (im Gegensatz etwa zu einer Addition von Aussagen oder einer Disjunktion von Zahlen); das erfordert ein "Type-checking", also eine Verfolgung der Datentypen durch den Beweisfluss. Falls nicht, werden die entsprechenden Beweisteile unter der Meldung"In folgenden Zeilen konnte die Typenverwendung nicht nachvollzogen werden:" zurückgemeldet, die Verarbeitung aber fortgesetzt. Gibt es keine Typenfehler, wird "Es wurden keine Typenfehler erkannt!" gemeldet.

6. Ob alle Schritte logisch verifiziert werden konnten. Falls nicht, werden die entsprechenden Aussagen unter der Meldung "Der Beweis wurde logisch nicht verifiziert. Folgende Textstellen konnten nicht nachvollzogen werden:" ausgegeben; andernfalls wird "Alle Folgerungsschritte konnten nachvollzogen werden!" gemeldet.

7. Ob unter den nicht verifizierten Beweiszeilen solche waren, die mithilfe der im "Anti-ATP" enthaltenen Fehlschlussmuster reproduziert werden konnten. Falls das der Fall ist, werden die entsprechenden Textteile unter der Meldung "Fehlschlüsse werden für folgende Zeilen vermutet:" ausgegeben; andernfalls wird "Es wurden keine bekannten Fehlschlüsse identifiziert." gemeldet.

8. Ob alle deklarierten Beweisziele beim entsprechenden Beweisendmarker erreicht wurden (diese Information liefert das Modul zur Zielverfolgung). Falls nicht, werden die nicht erreichten Beweisziele zusammen mit den Indizes der Zeilen, in denen der betreffende Teilbeweis durch einen Beweisendmarker abgeschlossen wird, unter der Meldung "Von den erkannten Beweiszielen wurden Anz nicht erreicht! Naemlich folgende:" ausgegeben, wobei Anz die Anzahl der nicht erreichten Ziele ist. Wurden alle Beweisziele erreicht, wird "Alle erkannten Beweisziele wurden erreicht!" gemeldet.

6.10 Automatische Erzeugung von Beweisaufgaben

Die Wirksamkeit eines Systems wie Diproche hängt – neben der Motivation seiner BenutzerInnen – erheblich von der Qualität und dem Umfang des zur Verfügung gestellten Übungsmaterials ab. Es ist geplant, im Rahmen der Spielwiesen zahlreiche instruktive Übungsaufgaben zur Verfügung zu stellen. Dennoch ist es sicherlich vorteilhaft, möglichst viele Typen von Beweisaufgaben nach Belieben weiter zu vertiefen und an neuen Aufgaben erproben zu können. Ein/e gute/r menschliche/r TutorIn könnte auf Nachfrage weitere Übungsaufgaben zu einer gewissen Technik oder einem gewissen Themenbereich erfinden. Auch diese Funktion ist in begrenztem Rahmen automatisierbar und in Diproche durch die automatische Aufgabenerzeugung auch implementiert. Derzeit können Aufgaben aus den Bereichen Aussagenlogik, Boolesche Mengenlehre, elementare Zahlentheorie und Induktion automatisch generiert werden.

6.10.1 Automatische Erzeugung von Beweisaufgaben zur Aussagenlogik

Das Modul zur Aufgabenerzeugung soll zwei Arten von Aufgaben zur Aussagenlogik generieren: Einerseits Beweisaufgaben, in denen zu zeigen ist, dass eine gewisse aussagenlogische Formel eine Tautologie ist; andererseits Entscheidungsaufgaben, in denen zu entscheiden ist, ob eine gewisse aussagenlogische Formel eine Tautologie ist. Solche Aufgaben können mithilfe eines Zufallsgenerators erzeugt werden (s.u.); hierzu ist es aber erforderlich, klare und implementierbare Kriterien dafür zu entwickeln, wann eine Aufgabe "sinnvoll" ist.

Einige Kriterien für eine "sinnvolle" Beweisaufgabe in der Aussagenlogik sind folgende:[18]

[18]Selbstverständlich kann der jeweilige didaktische Kontext es erfordern, bei der Konstruktion einer Übungsaufgabe von einem, mehreren oder sogar allen der folgenden Kriterien abzuweichen. So könnte etwa die insgesamt schwer verständliche, abschreckend lange, aber letztlich triviale

- Die fragliche Aussage sollte einen verständlichen "Sinn" bzw. eine erkennbare "Struktur" haben. Um ein rein mechanisches "Abarbeiten" etwa einer aussagenlogischen Formel zu vermeiden, das letztlich womöglich ein bloßes auf ein "Ausformulieren" einer Wahrheitstafel hinausläuft, sollte es für die Bearbeitung von Vorteil sein, diesen Sinn bzw. diese Struktur zu erfassen. So läßt "$(A \wedge ((A \to B) \wedge ((B \to C) \wedge (C \to D)))) \to D$" sich als Formalisierung eines Kettenschlusses auffassen, wohingegen die in der letzten Fußnote gezeigte Formel keine (offensichtliche und) vergleichbar eingängige Interpretation besitzt.

- Der Beweis der fraglichen Aussage sollte übersichtlich sein und bezüglich Länge und Verständlichkeit klare Vorteile gegenüber Wahrheitstafel erkennen lassen.

- Im Idealfall sollte die Aussage nicht auf den ersten Blick offensichtlich sein, um ein Beweisbedürfnis zu wecken und das Beweisen nicht als ritualhaftes Untermauern von Offensichtlichkeiten erscheinen zu lassen.

Nun ist (1) durch eine Automatisierung nur schwer zu erfassen (und zudem auch davon abhängig, ob der oder die Einzelne im vorliegenden Fall den Sinn bzw. die Struktur erfasst); immerhin kann man sich diesem Ziel annähern, indem man Aussagen erzeugt, die zwei überschaubare und etwa gleichlange aussagenlogische Ausdrücke durch einen "Hauptjunktor" miteinander verbinden. In der Implementierung wurde zudem versucht, den Kriterien (2) und (3) zu genügen, indem Aussagen erzeugt werden, die absolut betrachtet nicht zu kurz sind (also viele Teilformeln haben), so dass das Aufstellen einer Wahrheitstafel zumindest mühsam würde.

Die automatische Aufgabenerzeugung für Beweisaufgaben im Bereich Aussagenlogik verfährt hierzu wie folgt:

1. Der primäre logische Junktor J wird zufällig und gleichverteilt aus der Menge $\{\wedge, \vee, \to, \leftrightarrow\}$ gewählt. Damit liegt fest, dass die zu beweisende Aussage die Form $(\phi J \psi)$ haben wird.

2. Es werden alle aussagenlogischen Formeln generiert, die drei Vorkommen aussagenlogischer Variablen enthalten.

3. Es werden alle Wahrheitswertverteilungen aller in Schritt (2) generierten Aussagen erzeugt, die für ϕ infrage kommen. (Bei $\wedge$ wäre das nur diejenige,

aussagenlogische Formel $(A \wedge \neg A) \to ((A \vee (D \to ((\neg E \vee A) \to C))) \to ((E \vee \neg A) \wedge (B \vee \neg D)))$ gut darauf dazu dienen, die Funktion der materialen Implikation einzuüben und insbesondere von einem sturen "Wahrheitstafel"-Ansatz wegzuleiten. Die folgenden Kriterien sind also nicht als endgültiger von von jedem Kontext unabhängiger Maßstab zu sehen.

die in jedem Fall den Wahrheitswert 1 zuordnet.) Von diesen wird eine, V, zufällig ausgewählt. Ebenso wird zufällig eine der Aussagen, ϕ, gewählt, die diese Wahrheitswertverteilung aufweisen.

4. Nun werden alle Wahrheitswertverteilungen aller in Schritt (2) generierten Aussagen erzeugt, die für ψ infrage kommen, wenn ϕ bereits gegeben ist. Ebenso wird zufällig eine der zugehörigen Aussagen ψ gewählt.

5. Schließlich wird die Aussage $(\phi J \psi)$ als Beweisziel ausgegeben.

Zur Erzeugung von Entscheidungsaufgaben kann ähnlich vorgegangen werden, wobei nun die Verteilungen in Schritt (3) und (4) beliebig erzeugt werden können. Um die Chance, eine Tautologie zu erzeugen, auf einem geeigneten Niveau zu halten, wird im ersten Schritt zufällig entschieden, ob das obige Verfahren zur Erzeugung einer Tautologie angewendet wird oder ob eine rein zufällige Aussage ausgegeben wird.

Dass die auf diese Weise erzeugten Tautologien tatsächlich im oben genannten Sinn geeignete Aufgaben sind, ist letztlich eine Frage der Erfahrung im Einsatz. Umfangreiches Erproben hat gezeigt, dass die auf diese Weise generierten Aussagen im Allgemeinen sinnvoll lösbare Beweisaufgaben darstellen.[19] Zur Illustration führen wir hier einige Beispiele für durch die Aufgabenerzeugung generierte Tautologien auf (es handelt sich um die ersten fünf, die das Programm auf Abfrage erzeugt hat). Für beliebige Aussagen a, b, c ist zu zeigen, dass:

1. $((\neg(a \leftrightarrow \neg b) \wedge c) \rightarrow \neg(\neg(b \leftrightarrow a) \leftrightarrow c))$
2. $(\neg(\neg(c \rightarrow c) \wedge \neg a) \wedge (b \vee (b \rightarrow b)))$
3. $(\neg((c \wedge b) \vee \neg a) \rightarrow \neg(\neg(b \rightarrow a) \wedge b))$
4. $((\neg a \leftrightarrow \neg(b \vee \neg c)) \leftrightarrow \neg(\neg a \leftrightarrow \neg(c \wedge \neg b)))$
5. $((\neg(a \wedge \neg a) \vee \neg a) \wedge \neg(b \leftrightarrow (\neg b \wedge \neg b)))$

6.10.2 Automatische Erzeugung von Beweisaufgaben zur Booleschen Mengenlehre

Beweisaufgaben zur Booleschen Mengenlehre ergeben sich auf natürliche Weise als Übersetzung von aussagenlogischen Tautologien in die Sprache der Mengenlehre; dabei werden Implikationen zu Teilmengenbeziehungen und Äquivalenzen zu

[19] Im anderen Fall steht es dem/der BenutzerIn natürlich jederzeit frei, die Aufgabe zu verwerfen und eine neue generieren zu lassen.

Mengengleichheiten. Ferner lassen sich aussagenlogische Junktoren auf die offensichtliche Weise in Boolesche Operatoren übersetzen, so etwa $\wedge$ in $\cap$, $\vee$ in $\cup$ und Negation in Komplementierung. Eine Übersetzung von $\rightarrow$ und $\leftrightarrow$ in Boolesche Operatoren ist ebenfalls möglich, (indem man als Zwischenschritt $(A \rightarrow B)$ in $(\neg A \vee B)$ bzw. $(A \leftrightarrow B)$ in $((A \rightarrow B) \wedge (B \rightarrow A))$ und dies dann weiter übersetzt), führt aber bei zufällig generierten Aussagen oft zu einer unangenehmen Häufung von Komplementierungen bzw. zu unangenehm langen Termen. Das Verfahren zur Erzeugung von Beweisaufgaben zur Booleschen Mengenlehre ist daher folgendes:

1. Der Junktor J wird aus der Menge $\rightarrow$ und $\leftrightarrow$ zufällig gewählt.
2. Wie Schritt (2) im Verfahren für aussagenlogische Tautologien, aber für Formeln, die nur die Junktoren $\wedge$, $\vee$ und $\neg$ verwenden.
3. Wie in den Schritten (3)-(5) im Verfahren für aussagenlogische Tautologien werden die Aussagen ϕ und ψ erzeugt.
4. Schließlich werden ϕ und ψ wie oben erklärt in Boolesche Terme T_ϕ und T_ψ übersetzt und für J="$\rightarrow$" wird die Aussage $T_\phi \subseteq T_\psi$ sowie für J="$\leftrightarrow$" die Aussage $T_\phi = T_\psi$ ausgegeben.

Auch hier führen wir exemplarisch einige der auf diese Weise generierten Aufgaben auf. Für beliebige Mengen a, b, c ist zu zeigen, dass:

1. $((b \cup (\bar{c} \cap \bar{a})) \subseteq \overline{(c \cap \bar{(}\bar{a} \cup b)))}$
2. $((\bar{c} \cup \overline{(\bar{c} \cup b)}) = (\bar{c} \cup (c \cap \bar{b})))$
3. $((b \cap (\bar{a} \cup c)) \subseteq \overline{((\bar{c} \cap a) \cap \bar{c})})$
4. $((\bar{a} \cup (c \cup \bar{b})) \subseteq ((c \cup \bar{b}) \cup \bar{a}))$

6.10.3 Automatische Erzeugung von Beweisaufgaben zur vollständigen Induktion

Zu den typischen Einsatzgebieten für das Verfahren der vollständigen Induktion im Anfängerbereich gehören die folgenden:

1. Explizite Ausdrücke für indizierte Summen wie $\Sigma_{i=1}^{n} \frac{n(n+1)}{2}$, $\Sigma_{i=1}^{n} q^i = \frac{q^{n+1}-1}{q-1}$ oder $\Sigma_{i=1}^{n} \frac{1}{i(i+1)} = \frac{n}{n+1}$.
2. Teilbarkeitsaussagen wie $3|4^n - 1$.
3. Ungleichungen wie $2^n > n^2$ für $n \geq 5$.

Aufgaben vom Typ 1 sind insofern besonders geeignet, als die auf diese Weise bewiesenen Formeln häufig ein breites Spektrum an Anwendungen und damit ein über den Übungseffekt hinausgehendes Interesse haben. Andererseits scheint die Anzahl der einfachen Induktionen zugänglichen Formeln dieser Art begrenzt zu sein – im wesentlichen handelt es sich um Summen von Polynomen, Potenzen oder Summen von reziproken Produkten. Für ein automatische Entdecken weiterer interessanter Aufgaben dieser Art fehlt uns derzeit ein Ansatz. Denkbar wäre es, Linearkombinationen aus den genannten Typen und ggf. weiteren zu erzeugen und etwa nach dem Wert der Summe $\Sigma_{i=1}^{n}(i + q^i + 1)$ zu fragen; aber in diesen Fällen ergibt sich zum einen im Allgemeinen keine interessante Gestalt für das Ergebnis, zum anderen wäre es für den/die BenutzerIn sicherlich die bessere Strategie, die Zerlegung $\Sigma_{i=1}^{n}(i + q^i + 1) = \Sigma_{i=1}^{n} i + \Sigma_{i=1}^{n} q^i + \Sigma_{i=1}^{n} 1$ zu "sehen" und die bekannten Formeln einzeln anzuwenden statt erneut mit Induktion anzusetzen; insofern wäre bei Aufgaben dieser Art die Motivation gering, Induktion einzusetzen und die Verwendung dieser Beweisstrategie würde zu einem lästigen technischen Zusatzaufwand statt eine strategisch überlegte Erleichterung darzustellen. Diese Präsentation einer Beweistechnik sollte sicherlich vermieden werden.

Anders sieht es dagegen bei Aussagen vom Typ 2, also Teilbarkeitsaussagen aus. Indem man für diverse Primzahlen p die Ordnung von p in verschiedenen natürlichen Zahlen n ausrechnet, geeignete Kombinationen herstellt, die entstehenden Kombinationen für verschiedene Primzahlen erneut kombiniert und den entstehenden Term anschließend noch etwas modifiziert, erhält man zahlreiche Teilbarkeitsaussagen, die für den Einsatz des Induktionsverfahrens gut geeignet sind (die Aussagen sind damit letztlich alle von der Form $a|c_1 k^{\mathrm{ord}_k(a)\cdot n} + c_2$, wobei $\mathrm{ord}_k(a)$ die Ordnung von k in $\mathbb{Z}/a\mathbb{Z}$ ist und $c_1 \equiv c_2$ modulo a). Einige Beispielaufgaben, die auf diese Weise erzeugt werden, sind folgende: Es ist zu zeigen, dass für alle natürlichen Zahlen n folgendes gilt:

1. $4|2 * 3^{2*n} + 2$
2. $12|2 * 11^{2*n} - 14$
3. $13|4 * 11^{12*n} - 17$
4. $3|(-1) * 2^{2*n} + 1$
5. $13|25 * 7^{12*n} - 25$

Auch Aufgaben vom Typ 3, also induktiv zu beweisende Ungleichungen, sind leicht zu erzeugen. Wir betrachten hier Ungleichungen vom Typ $c_1 a^n + c_2 < c_3 b^n$, wobei $c_1, c_2, c_3, a, b \in \mathbb{N}$ und $a < b$ (alle Aussagen dieser Form sind für

alle natürlichen Zahlen n ab einer gewissen Grenze N wahr). Die betreffenden Konstanten werden aus einem begrenzten Zahlbereich gewählt, um die Aufgabe überschaubar zu halten; anschließend wird die Grenze N durch eine Abschätzung ermittelt. Auf diese Weise erhält man Aussagen wie die folgenden:

1. Für $n > 3$ gilt $2 \cdot 7^n + 2 < 14^n$
2. Für $n > 4$ gilt $2 \cdot 3^n + 4 < 2 \cdot 4^n$
3. Für $n > 6$ gilt $2 \cdot 12^n + 5 < 3 \cdot 14^n$
4. Für $n > 2$ gilt $2 \cdot 4^n + 2 < 2 \cdot 7^n$
5. Für $n > 1$ gilt $3 \cdot 12^n + 2 < 5 \cdot 13^n$

6.11 Einige "Spielwiesen"

Wie schon oben diskutiert, ist es in vielen Hinsichten vom Kontext abhängig, was als "natürliche" Sprache der Mathematik gelten kann. Insbesondere hängt der Vorrat an zur Verfügung stehenden Begriffen, Notationen, elementaren Annahmen und Schlußweisen vom jeweiligen mathematischen Teilgebiet und dem intendierten Zielpublikum ab, dem ein Beweis kommuniziert werden soll. Diproche berücksichtigt diesen Umstand insofern, als über die "Spielwiesen" eine Reihe von Themenfeldern mitsamt einem begrifflichen, notationellen, axiomatischen und deduktiven Hintergrund zur Verfügung steht.

6.11.1 Zur Methodik der Implementierung von Spielwiesen und ihrer Validierung

Das Ziel von Diproche ist es, für das Üben von Beweisaufgaben eine standardisierte Umgebung zur Verfügung zu stellen, die der üblichen Beweis- und Darstellungspraxis des jeweiligen Gebietes möglichst nahe kommt; die Standardisierung sollte für den/die BenutzerIn so wenig wie möglich sichtbar werden oder nur in dem Umfang, in dem sie Eigenschaften der mathematischen Methodik und Sprache modelliert. Was in einer schriftlichen Bearbeitung einer Übungsaufgabe als korrektes und korrekt dargestelltes Argument gelten würde, sollte also nach Möglichkeit auch von Diproche akzeptiert werden. Umgekehrt sollte Diproche nicht – etwa aufgrund eines zu starken ATPs – solche Argumente akzeptieren, von denen beim Lernstand der intendierten BearbeiterInnen gesagt werden müsste, dass sie wesentliche Lücken enthalten.

Das Ziel der Entwicklung einer "Spielwiese" ist also ein empirisches: Für einen intendierten Korpus von Aufgaben und Beispiellösungen sollte Diproche möglichst

genau solche Stellen monieren, die auch bei einer Korrektur der Lösungen durch eine/n (idealisierte/n) menschliche/n KorrektorIn als fehlerhaft angesehen würden. Um sich diesem Ziel anzunähern, liegt folgende Vorgehensweise nahe, die bei der Entwicklung der Spielwiesen nicht völlig strikt, aber doch im wesentlichen befolgt wurde:

1. Wahl eines geeigneten Gebietes (wie "Aussagenlogik", "Boolesche Mengenlehre" oder "Zahlentheorie"; für eine Diskussion der Kriterien für "geeignet", siehe den Abschnitt am Anfang vom Kapitel "Entwicklungsmöglichkeiten" weiter unten).

2. Auswahl geeigneter Aufgabentypen zu diesem Gebiet (etwa "Gleichheitsbeweise für Mengenterme" oder "einfache Teilbarkeitsbeweise".

3. Aufbau eines Korpus von Aufgaben- und Lösungstexten der entsprechenden Typen; Auswahl eines Teils dieses Korpus für die Entwicklung, während der andere Teil für die Testung zurückbehalten wird.

4. Analyse der auftretenden Symbole, formalen Terme, Begriffe, Formulierungen und Schlussweisen.

5. Implementierung des in (4) gefundenen in gesonderten Modulen des Formel- und Textparsers sowie des ATP.

6. Test des so entwickelten Systems mit Aufgaben aus dem dafür zurückbehaltenen Teil des Korpus. Dabei gilt eine Aufgabe als mit dem System erfolgreich bearbeitet, wenn sich Lösungstexte dafür im System textnah abbilden lassen. Die Lösungstexte sollten dabei aus externen Quellen stammen, um zu vermeiden, dass Lösungen (ggf. unbewusst) "auf das System zugeschrieben" werden.

7. Sofern erforderlich, Ergänzung fehlender Symbole, Formulierungen, Schlussweisen etc.

8. Wiederholung der Schritte (6)-(7), bis der größte Teil der Aufgaben des ausgewählten Typen sich im System wiedergeben läßt.

Die folgenden "Spielwiesen" sind derzeit realisiert.[20]

[20]Es ist natürlich jederzeit möglich, weitere hinzuzufügen. Für einige Vorschläge in dieser Hinsicht siehe den ersten Abschnitt im Kapitel "Entwicklungsmöglichkeiten" weiter unten.

6.11.2 Aussagenlogik

Die Aussagenlogik ist dadurch ausgezeichnet, dass es in ihr nur einen einzigen Objekttyp gibt, nämlich den der "Aussage", der innerhalb von Diproche als "prop" repräsentiert ist. Spezielle Aussagen sind die Aussagen "verum" und "falsum", die die tautologische und die kontradiktorische Aussage bezeichnen. Als Funktionen stehen ausschließlich die (in Infixnotation geschriebenen) logischen Junktoren $\wedge$, $\vee$, $\rightarrow$, $\leftrightarrow$ sowie der Negationsoperator $\neg$ zur Verfügung. Relationen gibt es in der Aussagenlogik keine. Die Aussagenlogik ist also in ontologischer Hinsicht eine äußerst simple Theorie. Zugleich ermöglicht sie es, besondere Phänomene des mathematischen Sprachgebrauchs, etwa die inklusive Lesart der Disjunktion oder die Semantik der materialen Implikation, in voller Allgemeinheit darzustellen und grundlegende Beweisstrategien wie direkte Beweise, Kontrapositionsbeweise, Widerspruchsbeweise oder Fallunterscheidungen zu üben. Da sich aus der Definition der Junktoren bisweilen überraschende bis absurd anmutende, aber dennoch tautologische Aussagen ergeben – etwa $(A \rightarrow B) \vee (B \rightarrow A)$, obwohl der "intuitive" Folgerungsoperator sicherlich nicht zwischen zwei beliebigen Aussagen in mindestens einer Richtung zu einer wahren Aussage führt – läßt sich auch durchaus ein Beweisbedürfnis wecken. Aufgaben sind stets von der Art, zu zeigen, dass ein gewisser aussagenlogischer Ausdruck tautologisch ist, wobei die Aufgabenstellung auch so formuliert sein kann, dass ein gewisser Ausdruck dieser Art sich unter gewissen Bedingungen herleiten läßt.

Ein typischer Diproche-Beweis in der Aussagenlogik ist etwa folgender:

> Es seien A, B Aussagen. Zeige: Dann gilt $(A \rightarrow B) \vee (B \rightarrow A)$.
> Beweis:
> Fall 1: Es gelte A. Dann gilt $(B \rightarrow A)$. Also gilt $(A \rightarrow B) \vee (B \rightarrow A)$. qed.
> Fall 2: Es gelte $\neg A$. Dann gilt $(A \rightarrow B)$. Also gilt $(A \rightarrow B) \vee (B \rightarrow A)$. qed.
> Damit haben wir $(A \rightarrow B) \vee (B \rightarrow A)$. qed.

Ein zusätzlicher Vorteil an der Aussagenlogik ist, dass sie mit einem recht begrenzten Vorrat an Formulierungen auskommt und sich so gut zum Einstieg in die kontrollierte natürliche Sprache von Diproche eignet. Deklarationen treten zwar auf, aber nur, um den verwendeten Variablen den Typ "Aussage" zuzuweisen; diese lassen sich überdies zumeist in die Aufgabenstellung integrieren, so dass sie von dem/der BenutzerIn nicht mehr vorgenommen werden müssen. Diproche-Beweise in der Aussagenlogik benötigen damit lediglich Formulierungen für Annahmen und Folgerungen – diese werden im Kapitel über die Sprache von Diproche weiter unten näher erläutert – sowie Annotationen (darunter insbesondere Beweisanfangs- und Endmarker) sowie ggf. die Ankündigung von

Zwischenzielen.

Die Formelsyntax ist dieselbe wie die bei den aussagenlogischen Mathediktaten, die wir oben beschrieben haben. Zusätzlich stehen innerhalb der Spielwiese "Aussagenlogik" Boolesche Umformungs- und Implikationsketten zur Verfügung (siehe den Abschnitt zur Formelsyntax im nächsten Kapitel), die zusätzlich zu den aussagenlogischen Junktoren noch die Folgerungs- und Äquivalenzbeziehungen $\Rightarrow$ und $\Leftrightarrow$ verwenden. Innerhalb der Spielwiese "Aussagenlogik" stehen ausschließlich aussagenlogische Deduktionsregeln zur Verfügung, also insbesondere keine Quantorenregeln.

Derzeit stehen im aussagenlogischen ATP nur zwei Schwierigkeitsgrade zur Verfügung: Bei "full" wird die volle Stärke des ATP genutzt. Im Schwierigkeitsgrad "nonrec" hingegen stehen nur solche Regeln zur Verfügung, die nicht ihrerseits auf andere Regeln zugreifen. So erlaubt "full" bereits einen recht weitreichenden Umgang mit Negationen in einem Schritt, so dass etwa folgende Eingabe verifiziert wird:

Es gilt $(\neg(a \vee \neg a)\&b) \Leftrightarrow ((\neg a\&a)\&b)$.

Bei einer kleinschrittigen Arbeitsweise wären hier mindestens vier Zwischenschritte erforderlich: Zunächst die Abspaltung der Disjunktionsglieder, dann die Anwendung der de Morganschen Regel auf den ersten Teil, schließlich die Auflösung der doppelten Negation und das erneute Zusammensetzen zu einer Konjunktion.

6.11.3 Boolesche Mengenlehre

Die Boolesche Mengenlehre arbeitet mit drei Objekttypen: Objekten, Mengen und Aussagen. Wie üblich wird – u.a. um den Paradoxien der naiven Mengenlehre zu entgehen, aber auch, um sicherzustellen, dass Komplemente von Mengen wiederum Mengen sind und nicht etwa echte Klassen – stillschweigend die Existenz einer universellen Menge U (intern in Diproche repräsentiert als "univset") angenommen, von der alle auftretenden Objekte Elemente und alle auftretenden Mengen Teilmengen sind.

In Darstellungen zur Booleschen Mengenlehre wird die universelle Menge gewöhnlich nicht explizit erwähnt. Das hat zur Folge, dass Elemente kein expliziter Typ zugewiesen wird. Um dieser Praxis entgegen zu kommen, werden Ausdrücke der Form $x \in A$ als hinsichtlich der Typenverwendung korrekt betrachtet, sobald A als Menge deklariert ist.

Innerhalb der axiomatischen Mengenlehre gibt es die zweistelligen Funktionen $\cap$ und $\cup$ für den Schnitt und die Vereinigung von Mengen, die durch /

repräsentierte mengentheoretischer Differenz[21] sowie die einstellige Funktion zur Bildung des Komplementes einer Menge X, notiert als $(-X)$. Ferner gibt es die zweistelligen Relationen $\subseteq$, $\supseteq$ und $\in$; die ersten vier sind, wie üblich, Relationen zwischen Mengen, die letzte ist eine Relation zwischen Ojekten und Mengen. Sind a und b Mengen, so wird also $(a \in b)$ vom System nicht als falsch betrachtet, sondern als Typenfehler, ebenso wie etwa $(a \subseteq b)$, wenn a und b Objekte sind. Die zweistellige Relation $=$ ist sowohl auf Objekten als auch auf Mengen erklärt. Ansonsten ist die Syntax für Terme und Formeln die gleiche wie die im Abschnitt zu den mengentheoretischen Mathediktaten weiter oben erläuterte; zusätzlich stehen als Ausdrucksmittel noch mengentheoretische (Un-)gleichungsketten zur Verfügung, also Ausdrücke wie $T_0 = T_1 \subseteq T_2 \subseteq T_3 = T_4$, wobei T_1, T_2, T_3 und T_4 mengentheoretische Terme sind. Die in der Praxis durchaus üblichen Abkürzungen für negierte Beziehungen wie $x \notin X$ oder $x \neq y$ stehen derzeit nicht zur Verfügung und müssen als $\neg x \in X$ bzw. $\neg x = y$ ausgedrückt werden.

Die Boolesche Mengenlehre ist eine Theorie der erststufigen Logik; daher stehen – jeweils gemäß dem aktuellen Schwierigkeitsgrad – auch quantorenlogische Regeln zur Verfügung. Darüber hinaus existiert ein Submodul des ATP, das übliche Schlussweisen der Booleschen Mengenlehre enthält, wie etwa folgende:

- Die Folgerbarkeit von $x \in B$ aus $x \in A$ und $A \subseteq B$.
- Die Folgerbarkeit von $x \in (-A)$ aus $\neg x \in A$.
- Die Folgerbarkeit von $x \in A$ bzw. von $x \in B$ aus $x \in (A \cap B)$.
- Wurde (ohne weitere Annahmen) aus der Annahme $x \in A$ gefolgert, dass $x \in B$, so kann auf $A \subseteq B$ geschlossen werden. Das gleiche gilt, wenn die Annahme $\forall x((x \in A) \rightarrow (x \in B))$ zur Verfügung steht. Die Teilbeweise, die zu den beiden Aussage $A \subseteq B$ und $B \subseteq A$ bzw. $A \supseteq B$ gehören, können durch die Annotationen $\subseteq$ und $\supseteq$ analog zu den Implikationsrichtungen $\Rightarrow$ und $\Leftarrow$ bei einem Äquivalenzbeweis strukturiert werden.
- Aus $A \subseteq B$ und $B \subseteq A$ oder $A \subseteq B$ und $A \supseteq B$ kann auf $A = B$ geschlossen werden.
- Wird aus der Annahme $x \in A$ ein Widerspruch hergeleitet, so gilt $A = \emptyset$ als bewiesen.

Zusätzlich sorgen Sonderklauseln der Zielverfolgung dafür, dass Annotationen wie "$\subseteq$" als Anzeige für eine Richtung einer Mengengleichheit korrekt interpretiert werden.

[21] Die Abweichung von der üblichen Schreibweise $\setminus$ hat technische Gründe und wird in späteren Versionen durch diese ersetzt werden.

Wir betrachten hier beispielhaft zwei Diproche-Lösungen zu einer Übungsaufgabe aus der Algebra 1-Vorlesung von Hinrich Lorenzen im Herbstsemester 2020/2021. Die Aufgabe lautete:

> Es seien A, B, C Mengen. Zeige: Dann gilt $(A/(B \cap C)) = ((A/B) \cup (A/C))$.

Diese Aufgabe läßt sich nun einerseits mit einem Inklusionsargument in beide Richtungen lösen, wobei die Möglichkeit, Beweise zu "schachteln", gleich mehrfach – einerseits in der Aufspaltung in zwei Richtungen, andererseits innerhalb dieser Teilbeweise durch die Verwendung von Fallunterscheidungen – benutzt wird:

> Beweis:
>
> $\subseteq$ Es sei $x \in (A/(B \cap C))$. Dann ist $x \in A$ und $\neg x \in (B \cap C)$. Also ist $\neg x \in B$ oder $\neg x \in C$.
>
> Fall 1: Es sei $\neg x \in B$. Dann ist $x \in (A/B)$. Also ist $x \in ((A/B) \cup (A/C))$.
> qed.
>
> Fall 2: Es sei $\neg x \in C$. Dann ist $x \in (A/C)$. Also ist $x \in ((A/B) \cup (A/C))$.
> qed.
>
> Damit haben wir $x \in ((A/B) \cup (A/C))$.
> qed.
>
> $\supseteq$ Nun sei $x \in ((A/B) \cup (A/C))$. Dann ist $x \in (A/B)$ oder $x \in (A/C)$.
>
> Fall 1: Es sei $x \in (A/B)$. Dann ist $x \in A$ und $\neg x \in B$. Also ist $\neg x \in (B \cap C)$. Damit ist $x \in (A/(B \cap C))$. qed.
>
> Fall 2: Nun sei $x \in (A/C)$. Dann ist $x \in A$ und $\neg x \in C$. Also ist $\neg x \in (B \cap C)$. Damit ist $x \in (A/(B \cap C))$. qed.
>
> Also ist $x \in (A/(B \cap C))$.
> qed.
>
> Damit folgt $(A/(B \cap C)) = ((A/B) \cup (A/C))$.
> qed.

Ebenfalls möglich, und deutlich kürzer, ist die Lösung mithilfe einer Gleichungskette:

Beweis: Es ist $(A/(B \cap C)) = (A \cap (-(B \cap C))) = (A \cap ((-B) \cup (-C))) = ((A \cap (-B)) \cup (A \cap (-C))) = ((A/B) \cup (A/C))$. Also ist $(A/(B \cap C)) = ((A/B) \cup (A/C))$. qed.

Ein dritter Weg wäre, den Mengenterm $A/(B \cap C)$ zunächst in den Klassenterm $\{x : x \in A \wedge \neg x \in (B \cap C)\}$ und weiter in umzuwandeln und die verbleibenden mengentheoretischen Operatoren (hier den Schnittoperator) durch ihre Definitionen zu ersetzen, um zu $\{x : x \in A \wedge \neg(x \in B \wedge x \in C)\}$ zu gelangen, und dann die Formel $x \in A \wedge \neg(x \in B \wedge x \in C)$ aussagenlogisch zu $((x \in A \wedge \neg x \in B) \vee (x \in A \wedge \neg x \in C))$ umzuformen.

Die Verwendung von Klassentermen gehört zum grundlegenden mathematischen Handwerkszeug; sicher wäre es daher wünschenswert, eine entsprechende Umgebung auch in Diproche zur Verfügung stellen zu können. Allerdings führt ein naiver Gebrauch von Klassentermen rasch in die Russellsche Antinomie [175], läßt sich doch über $\{x : \neg x \in x\}$ ein Klassenterm angeben, der, wenn er eine Menge R bezeichnete, zum Widerspruch $R \in R \leftrightarrow R \notin R$ führen würde. Im Rahmen eines Systems wie Diproche ist diese Möglichkeit sicherlich nicht akzeptabel, ließe sich doch, würde sie zugelassen, jede Aussage ϕ dadurch beweisen, dass man erst die Russellsche Antinomie herleitet und dann "ex falso quodlibet" anwendet. Dieses Problem wird in der Lehrpraxis meist 'pragmatisch' geregelt: Einerseits ist nicht zu erwarten, dass StudienanfängerInnen, die gerade die Grundzüge des Umgangs mit Mengen erlernen, auf diesen Gedanken verfallen, auf den selbst Frege erst durch Russell aufmerksam wurde. Wo eine derartige Lösung eingereicht würde, wäre sie wohl eher als Scherz eines mengentheoretische bereits versierten Studierenden zu verstehen, was von einer/m menschlichen KorrektorIn auch entsprechend erkannt und bemerkt werden könnte. Diese Leistung kann leider von einem automatischen System kaum erwartet werden: Die Kategorie "Russell-artiges Argument" automatisch zu erkennen, geht jedenfalls weit über die derzeitigen Möglichkeiten von Diproche hinaus. Sich andererseits darauf zu verlassen, dass niemand auf derartige Lösungen verfällt, würde das System mit erheblichen logischen Unsicherheiten belasten. Die letzte Möglichkeit, nämlich statt naiver Mengenlehre explizite axiomatisch formulierte Mengenbildungsprinzipien – etwa die Axiome von ZFC – zu unterrichten, scheidet aus didaktischen Gründen aus: Es ist ein grundlegendes Verständnis der Mengenlehre und ihrer Symbole erforderlich, um die ZFC-Axiome zu verstehen. Diese Axiome können also nicht dazu dienen, dieses Verständnis erst zu vermitteln.

Die Integration von Klassentermen scheint einen also vor die nicht eben attraktive Wahl zwischen einem technisch kaum möglichen, einem inkonsistenten und einem didaktisch unsinnigen System zu stellen. Aus diesem Grund sind

Klassenterme in Diproche derzeit nicht verfügbar.[22]

Funktionen und Relationen

Die Spielwiese "Funktionen und Relationen" erweitert die Spielwiese "Boolesche Mengenlehre" um die die Konzepte der Funktion und der Relation. Insbesondere enthält das Vokabular Worte wie "injektiv" oder "transitiv"; als Notation wurde etwa die Schreibweise $f : A \to B$ für "f ist eine Funktion von der Menge A in die Menge B" ergänzt. So können insbesondere Aufgaben zur Interaktion der mengentheoretischen Operatoren mit Funktionen sowie zu Eigenschaften von Funktionen gestellt werden. Ein Beispieltext aus der Spielwiese "Funktionen und Relationen" ist folgender Diproche-Beweis für die Aussage, dass die Verkettung $(g \circ f)$ zweier injektiver Funktionen nur dann injektiv ist, wenn f injektiv ist:

> Es sei $f : A \to B$. Es sei $g : B \to C$. Dann ist $(g * f) : A \to C$. Angenommen, $(g * f)$ ist injektiv. Wir zeigen: Dann ist auch f injektiv.
>
> Beweis:
>
> Es seien $x \in A$ und $y \in A$. Es gelte $f(x) = f(y)$. Dann folgt $(g * f)(x) = g(f(x)) = g(f(y)) = (g * f)(y)$. Also gilt $x = y$.
>
> Folglich ist f injektiv.
> qed.

6.11.4 Gruppentheorie

Die Spielwiese zur Gruppentheorie ist derzeit nur in Grundzügen implementiert. Wir geben daher nur einen knappen Überblick.

In der Gruppentheorie gibt es zwei Objekttypen: Gruppenelemente und Aussagen. Als spezielles Gruppenelement ist das neutrale Element 0 ausgezeichnet. Ferner gibt es die zweistellige Funktion $+$, die die Gruppenverknüpfung repräsentiert, sowie die Funktion, die zu einem gegebenen Gruppenelement a das Inverse Element liefert, notiert als $(-a)$. Als abkürzende Schreibweise wird wie üblich $(a - b)$ für $(a + (-b))$ akzeptiert.

Als besondere Darstellungsformen stehen wiederum Implikations- und Umformungsketten zur Verfügung. Außerdem stehen als elementare ATP-Regeln in einem separaten Gruppentheorie-Modul u.a. die folgenden zur Verfügung, die sich aus den Gruppenaxiomen ergeben:

[22] Ein möglicher Lösungsansatz könnte darin bestehen, Klassenterme nur in einem sehr beschränkten Rahmen zuzulassen, etwa nur mit Teilmengen der natürlichen Zahlen zu arbeiten, so dass ein Ausdruck wie $x \in x$ schon im Hinblick auf die Typenverwendung unsinnig wird.

- $(a + (b + c))$ und $((a + b) + c)$ können durcheinander ersetzt werden.
- $a + 0$ und $0 + a$ können durch a ersetzt werden.
- $a + (-a)$, $(-a) + a$ und $a - a$ können durch 0 ersetzt werden.
- Ist es Teil der Annahmen, dass die Gruppe abelsch ist, so können $(a+b)$ und $(b + a)$ durcheinander ersetzt werden. In den höheren Schwierigkeitsgraden können überdies aufgrund des Assoziativitätsgesetzes die Klammern aus einer endlichen Summe fortgelassen werden und in einer abelschen Gruppe können überdies beliebige Umordnungen vorgenommen werden.

6.11.5 Elementare Zahlentheorie[23]

Die elementare Zahlentheorie ist als Einstiegsgebiet zum Beweisenlernen aus verschiedenen Gründen besonders geeignet: Ihr Gegenstandsbereich, die Menge der natürlichen Zahlen, ist bis zu einem gewissen Grad jedem und jeder vertraut, seine Konstruktion ist logisch und ontologisch "harmlos" (insbesondere erfordert sie im Gegensatz etwa zu den reellen Zahlen keine aktualen Unendlichkeiten in Form von Grenzwerten etc.), sie weckt erfahrungsgemäß auch bei mathematisch mäßig Interessierten eine gewisse innere Beteiligung und bietet einen reichen Vorrat an kurzen und eleganten Beweisen, die mit einem überschaubaren Vorwissen zugänglich sind. Zudem hilft ein gutes Verständnis der Zahlentheorie sehr als Zugang zu weiteren Gebieten der Mathematik, wie etwa der Algebra, wo viele Strukturen als Verallgemeinerungen bzw. Abstraktionen der natürlichen Zahlen verstanden werden können. Ferner ist sie ein natürliches Einsatzgebiet für eine der wichtigsten elementaren Beweismethoden, die vollständige Induktion.

Gerade die Eigenschaft, dass eine große Vielfalt an Aufgaben mithilfe einer klar umgrenzbaren Menge an Grundaussagen, Schlussweisen und Methoden lösbar ist, macht die elementare Zahlentheorie auch für den Einsatz von Diproche besonders geeignet, da der auf diese Weise umgrenzte Bereich gültiger Schlüsse als Grundlage für den ATP verwendet werden kann.

In Diproche wird derzeit insbesondere die Teilbarkeitslehre unterstützt. Dazu steht das Symbol | für Teilbarkeit zur Verfügung und ferner die Schreibweise $a \equiv_m b$ für "a und b lassen bei Division durch m denselben Rest", d.h. $m|(a - b)$.

Als besondere Figur stehen ferner Teilbarkeits- und Kongruenzketten zur Verfügung, d.h. Ausdrücke der Form

$$a = b \mid c \mid d = e$$

[23]Die Ausführungen dieses und des folgenden Abschnitts sind in weiten Teilen deutsche Fassungen von Teilen des Artikels "'Number Theory and Axiomatic Geometry in the Diproche System", [32].

und andererseits der Form

$$a = b \equiv_m c = d \equiv_n e.$$

Als Ergebnis einer Teilbarkeitskette gilt wie üblich die stärkste Aussage, die sich bezüglich Anfangs- und Endglied daraus ergibt; falls die Kette ausschließlich Gleichheiten enthält, ist das die Gleichheit, ist eine Teilbarkeitsrelation dabei, die Teilbarkeit; im obigen Beispiel ergäbe sich $a|e$. Bei Kongruenzketten ist die Sache etwas komplizierter: Die stärkste sich ergebende Aussage wäre eine Kongruenz modulo des größten gemeinsamen Teilers aller in der Kette auftretenden Moduli. Da dieser Wert i.A. nicht zur Verfügung steht (insbesondere, wenn mit variablen Moduli gearbeitet wird) und der entsprechende Schluss überdies bereits ein Maß an zahlentheoretischer Einsicht verlangt, der das im Rahmen von Diproche als "elementar" anzusehende überschreitet, werden derzeit nur Kongruenzketten als auswertbar angesehen, in denen alle auftretenden Moduli paarweise gleich sind oder konkrete Werte haben. Somit würde $a = b \equiv_4 c = d \equiv_6 e$ zum Ergebnis $a \equiv_2 e$ führen und $a = b \equiv_m c = d \equiv_m e$ zu $a \equiv_m e$, während $a = b \equiv_m c \equiv_n d$ zwar geprüft würde, aber für den weiteren Beweis keine Information liefert (und insofern überflüssig ist).

Wir betrachten hier exemplarisch einen Beweistext zur Zahlentheorie, der von der aktuellen Diproche-Version als korrekt akzeptiert wird.

Es sei n eine natuerliche Zahl. Angenommen n^2 ist gerade. Wir zeigen: Dann ist 4 ein Teiler von n^2.

Beweis:

Angenommen, n ist ungerade. Dann ist auch n^2 ungerade. Widerspruch.

Also ist n gerade. Folglich existiert eine natuerliche Zahl k mit $n = 2 * k$. Es sei k eine natuerliche Zahl mit $n = 2 * k$. Dann folgt $n^2 = (2 * k)^2 = 4 * k^2$. Also ist 4 ein Teiler von n^2. qed.

6.11.6 Die Implementierung der Zahlentheorie-Spielwiese

Grundlage für die Implementierung der Spielwiese zur elementaren Zahlentheorie war eine Reihe von Übungsaufgaben für die einführende Vorlesung zur Algebra an der Europa-Universität Flensburg, zusammen mit den zugehörigen Musterlösungen. Die kontrollierte Sprache erfasst die Begriffe der Parität ("gerade" und "ungerade"), der Teilbarkeit, Restklassen, Quadrat- und Kubikzahlen sowie Gleichheit und Ungleichheit. In diesem begrenzten begrifflichen Rahmen läßt sich bereits eine größere Menge an Übungsaufgaben formulieren

und lösen. Ausgespart sind damit vorläufig z.B. Begriffe wie der der Primzahl und der Primfaktorzerlegung und damit zentrale Begriffe der multiplikativen Zahlentheorie wie Teilerfremdheit, kleinstes gemeinsames Vielfaches, größter gemeinsamer Teiler etc. Ein Grund hierfür ist, dass etwa Primfaktorzerlegungen sich nicht mehr auf natürliche Weise in eine einsortige Domäne einfügen lassen; sie sind keine ganzen Zahlen, sondern endliche Folgen ganzer Zahlen, wobei aber dieser Folgencharakter in zahlentheoretischen Texten kaum je expliziert wird. Ihre Integration würde es daher erfordern, die implizit gelassene Behandlung von Folgen zu automatisieren. Angesichts des enormen damit verbundenen Aufwandes einerseits und der andererseits bereits recht großen Übungsmöglichkeiten, die sich im Rahmen einer rein auf ganzen Zahlen basierenden Ontologie realisieren lassen, haben wir vorerst darauf verzichtet.

Im Textparser wurden Regelsätze ergänzt, um mit unären und binären Prädikaten wie "gerade/ungerade", "Quadratzahl/Kubikzahl", "ist ein Teiler von" etc. umgehen zu können. Dabei sind auch kollektive Verwendungen wie "Es seien a, b, c ungerade" zulässig. Die Formalisierungsroutine wurde entsprechend ergänzt, um mit diesen Begriffen umgehen zu können; kollektive Verwendungen werden dabei als Konjunktionen formalisiert, so dass "es seien a, b, c ungerade" intern als [[odd,[a]],and,[[odd,[b]],and,[odd,[c]]]] repräsentiert würde.[24] Entsprechend wurden im Formelparser symbolische Ausdrücke für Teilbarkeit ($a|b$ für "a teilt b") und Kongruenzen ($a \sim (m)b$) für "a ist modulo m kongruent zu b" hinzugefügt. Die mit Abstand umfangreichste Ergänzung betrifft den ATP; das auf Zahlentheorie spezialisierte Submodul des ATP besteht seinerseits aus 5 Teilmodulen, wobei die für allgemeine Aussagenlogik und Quantorenlogik zuständigen Module, die bei der Verifikation zahlentheoretischer Beweisschritte natürlich ebenfalls zum Einsatz kommen,[25] nicht mitgezählt sind:[26]

- Ein Modul mit allgemeinen Regeln zur Zahlentheorie, das 158 Schlussregeln umfasst.
- Ein Modul mit 40 Inferenzregeln zur Teilbarkeit.

[24] Werden zweistellige Relationen kollektiv verwendet, so wird der Ausdruck als Konjunktion der Behauptungen so interpretiert, dass die fragliche Relation zwischen je zwei verschiedenen der fraglichen Objekte gilt. Die einzige Relation, für die diese Verwendung derzeit vorgesehen ist, ist "paarweise verschieden", wobei "es seien a, b, c paarweise verschieden" entsprechend interpretiert wird als [[neg,[a,=,b]],and,[[neg,[a,=,c]],and,[neg,[b,=,c]]]].

[25] Eine für den Autor überraschende Beobachtung sowohl in der Spielwiese zur Zahlentheorie wie auch in der zur axiomatischen Geometrie war indes, dass von diesen allgemein logischen Regeln kaum je mehr als eine handvoll wirklich benutzt wurde. Die meisten Beweisschritte in den hier betrachteten Beweisen scheinen spezifisch-inhaltlicher Natur zu sein statt formallogischer.

[26] Die unten angegebenen Regelzahlen beziehen sich auf den Zeitpunkt der Abfassung dieses Abschnitts; mit der Weiterentwicklung der Spielwiese werden sie sich verändern (verändert haben), was zumeist bedeutet, dass weitere Regeln hinzukommen.

- Ein Modul mit 58 Inferenzregeln zu Kongruenzen.
- Ein Modul mit 52 Inferenzregeln, die speziell natürliche Zahlen betreffen (im Gegensatz zu ganzen Zahlen).
- Ferner das auch von anderen Spielwiesen genutzte Termumformungsmodul mit speziellen Regeln für Termumformungen, Gleichungs- und Ungleichungsketten, Ketten von Äquivalenzumformungen etc.

Das so erhaltene System wurde sodann "getestet" durch die probeweise Anwendung auf die Beispiele und Übungsaufgaben in Kapitel 3 von Chartrand et al. [51], in dem verschiedene grundlegende Beweisaufgaben durch Beweisaufgaben zu geraden und ungeraden Zahlen erläutert werden. Dieses Buch ist eines der beiden Lehrbücher, die im Rahmen US-amerikanischer Lehrveranstaltungen zum Beweisenlernen am häufigsten als Textgrundlage verwendet (siehe dazu die systematische Studie zu solchen Kursen von David und Zazkis [61], S. 393); somit sind die dort behandelten Beispiele als im Sinne der Ziele von Diproche einschlägig zu betrachten. Im Einzelnen wurden betrachtet die Beispiele 3.4, 3.5, 3.6, 3.8, 3.10, 3.11, 3.12, 3.13, 3.14, 3.15, deren Beweise im Buch selbst vorgestellt werden und die in die Diproche-Syntax (insbesondere also vom englischen Original in die deutsche Sprache) übertragen wurden, sowie die Übungsaufgaben 3.8, 3.9, 3.10, 3.16, 3.17, 3.18, 3.19, 3.20, 3.21, 3.26, 3.27, 3.28, 3.29, zu denen wir die Lösungen selbst verfasst haben. Auf diese Weise konnte sowohl erprobt werden, wie nah sich im Rahmen des Systems unabhängig vom System geschriebene und damit als "natürlich" anzusehende Beweistexte abbilden lassen als auch, wie flüssig die Bearbeitung von Übungsaufgaben funktioniert. Einige Übungsaufgaben und Beispiele wurden trotz ihrer Zugehörigkeit zum Thema "gerade und ungerade Zahlen" aus der Untersuchung ausgeschlossen, da sie entweder Brüche verwendeten oder Aufgaben über explizit gegebene endliche Mengen betrafen, also von der Art "Für alle x aus $\{1, 2, 3\}$ gilt...", was beides von der aktuellen Implementierung des Zahlentheorie-Moduls nicht unterstützt wird.

Das Ergebnis war erfreulich: Für fast alle Beispiele in diesem Abschnitt von der fraglichen Art ließen sich Diproche-Texten schreiben, die einer deutschen Übersetzung der englischen Beweistexte in [51] sehr nahe kamen und zugleich vom System akzeptiert wurden.[27] Eine beispielhafte Gegenüberstellung eines Beweistextes aus Chartrand et al. mit einer Reformulierung in Diproche findet

[27] Gleichwohl ist hier relativierend anzumerken, dass die Übertragung ins Deutsche vom Autor vorgenommen wurde, also auf der Basis einer weitreichenden Kenntnis der Ausdrucks- und Verarbeitungsmöglichkeiten im Rahmen von Diproche. Es wäre sicher vorzuziehen, dieselbe Aufgabe mit Versuchspersonen durchzuführen, die lediglich eine kurze Einführung in das System erhalten, es aber nicht entwickelt haben. Hierzu liegen bisher nur anekdotische Erfahrungen vor. Hier liegt sicher eine Frage für eine noch durchzuführende Untersuchung.

sich auf der nächsten Seite. Von besonderem Interesse sind die Beispiele 3.19 und 3.20, in denen dem/der LeserIn absichtlich falsche Beweise zur Prüfung vorgelegt wurden: In beiden Fällen konnte der Text ohne Schwierigkeiten in einen Text übersetzt werden, der von Diproche verarbeitet werden konnte und in dem Diproche den Fehler dann auch korrekt identifizieren konnte (in 3.19 musste allerdings die Tatsache, dass 1 eine ungerade Zahl ist, explizit erwähnt werden damit der in [51] vorgesehene problematische Schritt der *einzige* war, der von Diproche als nicht verifizierbar hervorgehoben wurde).

Ein Beispielbeweis von Chartrand et al.

If n is an odd integer, then $4n^3 + 2n - 1$ is odd.

Proof: Assume that n is odd. Then $n = 2y+1$ for some integer y. Therefore, $4n^3 + 2n - 1 =$
$4(2y + 1)^3 + 2(2y + 1) - 1$
$= 4(8y^3 + 12y^2 + 6y + 1) + 4y + 2 - 1 =$
$32y^3 + 48y^2 + 28y + 5 =$
$2(16y^3 + 24y^2 + 14y + 2) + 1.$
Since $16y^3 + 24y^2 + 14y + 2$ is an integer, $4n^3 + 2n - 1$ is odd. ■[a]

[a] Chartrand et al., S. 83, Result 3.6.

Eine Diproche-Formulierung dieses Beweises

Es sei n eine ganze Zahl. Zeige: Wenn n ungerade ist, dann ist 4*nˆ3+2*n-1 ungerade.

Beweis: Angenommen, n ist ungerade. Es sei y eine ganze Zahl mit n=2*y+1. Also gilt
4*nˆ3+2*n-1
=4*(2*y+1)ˆ3+2*(2*y+1)-1=
4*(8*yˆ3+12*yˆ2+6*y+1)+4*y+2-1
=32*yˆ3+48*yˆ2+28*y+5
=2*(16*yˆ3+24*yˆ2+14*y+2)+1
=2*(16*yˆ3+24*yˆ2+14*y+2)+1.
Also ist 4*nˆ3+2*n-1 ungerade.
qed.

Lediglich die Beispiele 3.16 und 3.17 erwiesen sich als schwieriger: 3.17 erforderte an einigen Stellen die Ersetzung von Wendungen wie ‘n ist gerade” oder “n ist ungerade” durch formale Ausdrücke wie “$2|n$” bzw. “$\sim 2|n$”. Grund hierfür ist die Schwierigkeit, innerhalb natürlichsprachlicher Formulierungen die “intendierte” Priorität der logischen Junktoren zu ermitteln: Beispielsweise interpretiert Diproche die Formulierung “nicht A oder B” als “$(\neg A) \vee B$, während es hier als “$\neg(A \vee B)$ zu lesen ist. Solche Ambiguitäten sind in mathematischen Texten nicht selten; welche Bedeutung die intendierte ist, wird sich im Allgemeinen aus pragmatischen Erwägungen ergeben: Zu verwenden ist diejenige Lesart, mit der der Beweis funktioniert. (Obwohl es technisch natürlich möglich ist, diesen Vorgang dadurch zu imitieren, dass man etwa sämtliche Interpretationen generiert und danach prüft, ob eine davon zu einem funktionierenden Beweis führt, ist eine solche “Pragmatik-Simulation” in Diproche derzeit nicht vorgesehen. Grund

dafür ist zum einen das Ziel, die Verarbeitungszeiten möglichst gering zu halten, während das Erzeugen und Prüfen zahlreicher semantischer Varianten leicht zu einer kombinatorischen Explosion führen kann; zum zweiten, und wichtiger, aber die Zielsetzung, BenutzerInnen einerseits eine möglichst einfache und klare Semantik anzubieten, anhand derer sich die Beurteilungen des Systems klar nachvollziehen lassen – und eben dadurch BenutzerInnen zu einem möglichst präzisen Sprachgebrauch zu bewegen.) Da es derzeit in Diproche nicht vorgesehen ist, natürlichsprachliche Texte durch Klammern zu gliedern, andererseits aber "gerade" und "ungerade" keine direkten formalsprachlichen Entsprechungen haben, mussten die betreffenden Aussagen vollständig formalisiert werden zu Ausdrücken wie "$\sim (2|n \vee 2|m)$", was zu einer deutlicheren Abweichung vom Originaltext führte.

In Beispiel 3.16, in dem eine Schachtelung von Beweisstrategien vorgenommen wird (es handelt sich um einen Äquivalenzbeweis, wobei der erste Teil seinerseits durch eine Fallunterscheidung erfolgt) entstanden Schwierigkeiten durch die Verwendung von Formulierungen wie "von der gleichen Parität" oder "von verschiedenen Paritäten", die in der aktuellen Diproche-Version nicht implementiert sind.[28] Hier waren deutliche Änderungen des gegebenen Argumentes erforderlich, insbesondere auch die Ergänzung von Argumentationsschritten, die ein wenig vom gegebenen Beweisziel abführten.

Aus den genannten Gründen betrachten wir die Erfahrungen mit den Beispielen 3.16 und 3.17 als Fehlschläge für das System. Damit wurde das System erfolgreich getestet in 22 von 24 Testfällen (23 von 25, wenn man die zwei unterschiedlichen Lösungen, die in [51] für Beispiel 3.14, die beide in Diproche formuliert werden konnten, als zwei Beispiele zählt), was einer Erfolgsquote von etwa 92 Prozent entspricht. Auf unserem Bürorechner an der EUF lag die durchschnittliche Laufzeit für diese Testfälle bei etwa 7 Sekunden, mit einer maximalen Laufzeit von etwa 20 Sekunden.

Zusätzlich wurden die ersten 5 von 8 Beispielen in Kapitel 4.1 in [51] zur Teilbarkeit erfolgreich in Diproche reproduziert.

Wir betrachten das als ein sehr erfreuliches Ergebnis, insbesondere, da es ja – gemäß unserer Zielsetzung – nicht das Ziel des Diproche-Systems ist, als allgemeiner automatischer Prüfer für beliebige Beweisübungen zu fungieren, sondern lediglich, deutlich bescheidener, ein Mittel zum Einüben des Beweisens anhand didaktisch geeigneter Aufgaben zur Verfügung zu stellen. Es ist insofern kein Nachteil, dass das System nur für bestimmte Typen von Übung funktioniert, solange es eine ausreichende Menge von didaktisch geeignetem Material gibt, das unter die realisierten Typen fällt. In dieser Hinsicht ist die Fähigkeit des

[28] Es wäre natürlich nicht schwierig, diese Formulierungen einzubauen; das ist für spätere Versionen auch vorgesehen.

Systems, einen erheblichen Anteil an Übungsaufgaben eines etablierten Lehrbuchs abzubilden, durchaus ermutigend.[29]

Vollständige Induktion

Hinsichtlich der Ausdrucksmöglichkeiten stellt die Spielwiese zur vollständigen Induktion eine Erweiterung der Spielwiese zur elementaren Zahlentheorie dar, wobei zusätzlich indizierte Summen und Produkte verwendet werden können. Als zusätzliche Beweisanfangsmarker stehen die Formulierungen "Induktionsanfang" und "Induktionsschritt" zur Verfügung; soll eine Schlussfolgerung mithilfe der vollständigen Induktionsregel gezogen werden, muss das durch eine gesonderte Formulierung wie "Mit Induktion", "Induktiv" etc. eingeleitet werden.

6.12 Axiomatische Geometrie[30]

Die axiomatische Geometrie spielt in der universitären Lehre derzeit eine eher untergeordnete Rolle. Es gibt nur wenige Universitäten, an denen überhaupt Lehrveranstaltungen zu diesem Thema angeboten werden. An der Europa-Universität Flensburg ist sie Teil des regulären Curriculums und wird typischerweise von Studierenden des zweiten Semesters belegt. Für den Einsatz im Rahmen dieser Veranstaltung wurde eine "Spielwiese" zur axiomatischen Geometrie implementiert. Aus Gründen, die im nächsten Abschnitt erläutert werden, sind wir inzwischen davon überzeugt, dass die axiomatischen Geometrie beim Erwerb erster Beweiskompetenzen und als Teil der mathematischen Grundausbildung eine wichtige Rolle spielen kann.

6.12.1 Didaktische Vorzüge der axiomatischen Geometrie

Die axiomatische Geometrie eignet sich für den Anfängerbereich aus zunächst ähnlichen Gründen wie die elementare Zahlentheorie: Die betrachteten Objekte – Punkte, Geraden, Kreise, Dreiecke, ebene Figuren etc. – sind Studierenden zumeist aus der alltäglichen Anschauung wie auch aus dem Schulunterricht wohlbekannt

[29] Michael Schmitz von der Universität Flensburg hat im Aufgabenarchiv der deutschen Mathematikolympiade einige Beweisaufgaben gefunden, die erfolgreich im Rahmen von Diproche gelöst werden konnten, nämlich MO090833 sowie MO520833. Diese Bestätigung ist indes rein anekdotisch: Insbesondere ist Diproche nicht mit dem Ziel entworfen, im Training für Mathematikolympiaden eingesetzt zu werden. Darüber hinaus haben wir eine Reihe von einfachen Beweisen zur Teilbarkeit aus verschiedenen Skripten, Online-Darstellungen und Lehrvideos ohne größere Schwierigkeiten in Diproche übertragen können.

[30] Die Ausführungen dieses Abschnitts sind im wesentlichen eine deutsche Version von Teilen des Artikels "Number Theory and Axiomatic Geometry in the Diproche System", [32].

und das Gebiet ist reich an einfachen, aber überraschenden Aussagen, die geeignet sind, Interesse zu wecken und ein Beweisbedürfnis hervorzurufen. Ferner ist es ein wesentlicher Bestandteil der meisten geometrischen Untersuchungen, Skizzen anzufertigen und anhand der Betrachtung von Figuren zu arbeiten. Diese Interaktion zwischen graphischer Darstellung und verbal formuliertem Argument, die in der Geometrie in besonderem Maße zutage tritt, ist ein erheblicher didaktischer Vorzug der Geometrie: Hier muss eine je besondere Figur "allgemein angeschaut" werden, was einen exemplarischen Fall für die mathematische Anschauung im allgemeinen darstellt. Es gibt wohl kein Gebiet in der Mathematik, in dem es nicht hilfreich wäre, in irgend einer Form eine solche Anschauung einzusetzen. Weiter sind aus geometrischen Beweisen auch in heuristischer Hinsicht wertvolle Lehren zu ziehen: Zum einen werden Fortschritte oft dadurch erzielt, dass bezüglich einer gegebenen Konfiguration die "Perspektive" oder "Sichtweise" geändert wird, so etwa, wenn drei Linien, die im Verlauf einer Konstruktion "nebenbei" entstanden sind, nun als ein Dreieck bildend angesehen und in das Zentrum der Aufmerksamkeit gerückt werden; zum anderen ist es häufig erforderlich, zur Erzielung eines weiteren Fortschrittes "Hilfsobjekte" einzuführen (im einfachsten Fall etwa dadurch, dass ein Schnittpunkte zweier Geraden als solcher herausgestellt und benannt wird; dann dadurch, dass zu zwei gegebenen Punkten die durch sie verlaufende Gerade oder in einem gegebenen Dreieck die mit ihm assoziierten Objekte (Seitenhalbierenden, Höhen, Mittelsenkrechten, Um- und Inkreis etc.) eingeführt werden; schließlich dadurch, dass ganze Konfigurationen verschoben, gedreht, gespiegelt etc. werden, um dadurch entstandene Symmetrien auszunutzen), um die Voraussetzung für die Anwendung eines Hilfssatzes oder einer Konstruktionsmethode zu schaffen. Der axiomatische Aspekt schließlich verstärkt die Spannung und damit zugleich die Interaktion zwischen Bild und Text, da er es ermöglicht und erfordert, die auf anschaulichem Weg gefundene Lösung letztlich in Form eines Textes darzustellen, der ohne Bezug auf die Anschauung "für sich steht".[31]

Schließlich ist die Geometrie, neben der Zahlentheorie, ein Bereich der Mathematik, aus dem viele der modernen Bereiche, etwa die Algebra oder die Analysis, historisch hervorgegangen sind. Die Vertrautheit mit der Geometrie kann damit die Grundlage für einen genetischen Zugang zur Lehre der modernen Mathematik darstellen, wie in etwa Toeplitz in [194] vertritt.[32]

Ein typischer Beweistext zur axiomatischen Mengenlehre, der von Diproche akzeptiert wird, ist etwa folgender:

[31]Die Art, wie Bilder im Rahmen und als Teil von geometrischen Beweisen funktionieren ist ein faszinierendes Gebiet; vgl. etwa die Arbeiten von J. Mumma [152] über die Verwendung von Diagrammen in Euklids "Elementen".

[32]Dieser Vorteil wird aber voraussichtlich nur dann zutage treten, wenn die Lehre in diesen anderen Bereichen entsprechende Bezüge explizit herstellt.

Es seien a, d, c, d_1 Punkte. Es sei $d(a,c,d)$ gleichschenklig. Ferner sei $d(a,c,d)$ rechtwinklig. Es sei l der Mittelpunkt von $s(d,d_1)$. Angenommen, l liegt auf $l(a,c)$. Es sei $l(d,d_1)$ orthogonal zu $l(a,c)$.

Wir zeigen: Dann ist $v(a,d,c,d_1)$ ein Quadrat.

Beweis: Wir haben $s(a,d) \sim s(d,c)$. Es ist $l(a,d)$ orthogonal zu $l(c,d)$. Es gilt l liegt auf $l(d,d1)$. Also gilt $l(l,d) = l(d,d_1)$. Damit ist $l(l,d)$ orthogonal zu $l(a,c)$. Nach der Mittellotregel folgt $s(a,l) \sim s(l,c)$. Also ist l der Mittelpunkt von $s(a,c)$. Damit ist $v(a,d,c,d_1)$ ein Parallelogramm. Wegen $s(a,d) \sim s(d,c)$ ist $v(a,d,c,d_1)$ sogar eine Raute. Also ist $v(a,d,c,d_1)$ ein Quadrat. qed.

Wir geben weiterhin das folgende Beispiel, eine Version des Lehrsatzes von Thales, der eine Diproche-Version des Beweises in Lorenzen [139] darstellt, als Fall eines Äquivalenzbeweises in der Geometrie:

Es sei $d(a,b,c)$ ein echtes Dreieck. Es sei m der Mittelpunkt von $s(a,b)$.
Wir zeigen: Dann ist $d(a,b,c)$ rechtwinklig gdw $s(m,a) \sim s(m,c)$.

Beweis: Es sei $l := m(s(a,c))$. Dann folgt $l(m,l)||l(b,c)$.

$\Rightarrow$ Es sei d(a,b,c) rechtwinklig. Dann ist $l(a,c)$ orthogonal zu $l(b,c)$. Also ist $l(m,l)$ orthogonal zu $l(a,c)$. Damit folgt $s(m,a) \sim s(m,c)$. qed.
$\Leftarrow$ Nun gelte $s(m,a) \sim s(m,c)$. Dann ist $l(m,l)$ senkrecht zu $l(a,c)$. Also ist $l(b,c)$ orthogonal zu $l(a,c)$. Damit ist $d(a,b,c)$ rechtwinklig. qed.
Also ist $d(a,b,c)$ rechtwinklig gdw $s(m,a) \sim s(m,c)$.
qed.

Die Implementierung der Spielwiese zur axiomatischen Geometrie

Im Fall der axiomatischen Geometrie lagen die Dinge etwas komplizierter als bei der elementaren Zahlentheorie. Ein Grund war, dass das Gebiet, trotz seiner didaktischen Vorzüge, nur wenig gelehrt wird und entsprechend wenig Material zur Verfügung stand, um das System daran zu entwickeln. Ein Teil der Übungsaufgaben zur Vorlesung von Hinrich Lorenzen zur axiomatischen Geometrie im Frühjahrssemester 2020 wurde zur Entwicklung der Spielwiese, insbesondere also zum Aufbau einer geeigneten kontrollierten natürlichen Sprache sowie eines Vorrats von Schlussregeln, verwendet, ein anderer Teil dann zum "Testen" des Systems verwendet, zusammen mit Aussagen und Beweisen aus dem zugehörigen (unveröffentlichten) Vorlesungsskript.

Der axiomatische Aufbau der Vorlesung, der einerseits die textnahe

Formalisierung und Automatisierung ermöglicht, hat andererseits zur Folge, dass Terminologie und verfügbare Methoden selbst im Bezug auf die grundlegendsten Begriffe wie "Punkt" oder "Gerade" im Verlauf der Vorlesung permanent entwickelt werden. Dadurch ist es schwierig, passende "Schwierigkeitsgrade" – also Mengen von Inferenzregeln – zu isolieren, die für eine nennenswerte Anzahl von Übungsaufgaben angemessen sind. War ein Testfall einmal erfolgreich verarbeitet, war es häufig erforderlich, dem ATP-Modul eine Beweisregel hinzuzufügen, durch die diese Aufgabe trivialisiert wurde, um einen sinnvollen methodischen Rahmen für die Bearbeitung der nachfolgenden Aufgaben zu schaffen. Es ist natürlich in der Geometrie wie auch in den anderen "Spielwiesen" möglich, die volle Stärke des ATP zuzulassen und alle implementierten Beweisregeln zur Verfügung zu stellen. Doch das hätte in einer Lehrveranstaltung, in der die Methoden permanent inkrementell aufgebaut werden, eben zur Folge, dass das System für den begleitenden Einsatz im laufenden Lehrbetrieb untauglich würde.

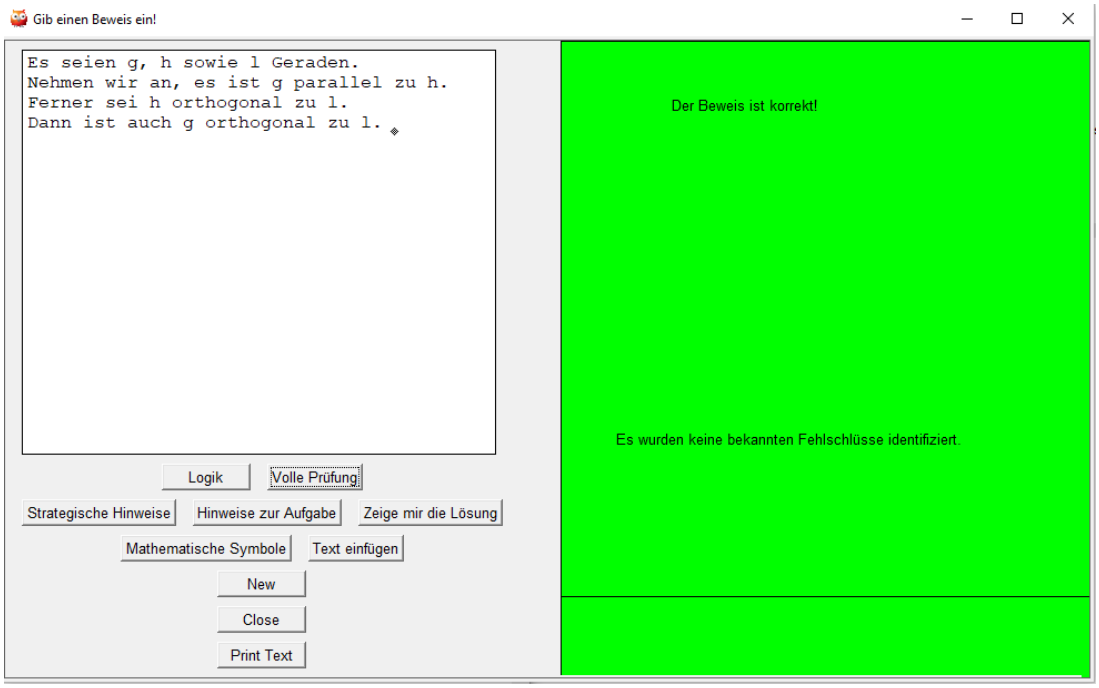
Es ist eine noch offene Herausforderung, im Bereich der axiomatischen Geometrie brauchbare Schwierigkeitsgrade zu identifizieren. Falls das fehlschlägt, bliebe die Möglichkeit, das Kriterium der Tauglichkeit für einen größeren Bereich (oder einen ganzen Typ) von Aufgaben fallen zu lassen und die verfügbaren Inferenzregeln für jede Übungsaufgabe separat zu spezifizieren. Obschon sicherlich möglich, ist das zweifellos weder für BenutzerInnen noch für die Implementierung eine sonderlich attraktive Lösung.

Ein weiteres, gegenüber den anderen Spielwiesen neues, Merkmal der Geometrie ist die mehrsortige Domäne, die mit Punkten, Linien, Strecken, Kreise etc. verschiedene Grundtypen von Objekten umfasst, die in der Inferenz unterschiedlich behandelt werden müssen (beispielsweise kann die Existenz einer Geraden g durch den Punkt p, die h nicht schneidet, aus der Voraussetzung, dass p nicht auf h liegt, gefolgert werden, wenn h eine Gerade ist, nicht aber, wenn h ein Kreis ist). Des Weiteren werden neue Objekte in diesem Bereich häufig nicht durch eine explizite Deklaration eingeführt ("Es sei p ein Punkt") sondern implizit über eine Konstruktion, die den Typ determiniert ("Definiere p als den Schnittpunkt von l mit g"). Das machte es erforderlich, die Trennung zwischen Typenprüfung und logischer Prüfung zum Teil fallen zu lassen und Typenberechnungen nun auch vom Geometrie-Modul des ATP anstellen zu lassen.

In die Entwicklung des Geometrie-ATPs flossen im wesentlichen drei Ansätze ein: Zum einen wurden Inferenzregeln implementiert, die die Benutzung von Axiomen oder grundlegenden Lemmata aus dem – weitgehend an Lorenzen [139] orientierten - Vorlesungsskript reflektierten. Weiter wurden diejenigen Inferenzregeln integriert, die in [139] – etwa als "Axiome des Schließens" – explizit erwähnt werden. Schließlich wurde das so entstandene vorläufige System mit zahlreichen einfachen Beweisen aus dem Vorlesungsskript und Lösungen zu

Übungsaufgaben getestet und nötigenfalls um weitere Regeln ergänzt. Zur Zeit der Niederschrift dieses Abschnitts enthielt der Geometrie-ATP 544 themenspezifische Inferenzregeln, von denen einige auf weitere, in der Zählung nicht enthaltene, Subregeln Bezug nehmen.

Table 6.1: Eingabe eines Beispieltextes zur axiomatischen Geometrie über das Diproche-Interface



Die ersten zehn Übungsblätter zur Vorlesung zur axiomatischen Geometrie im FS 2020 wurden anschließend als Testfälle verwendet, wobei die Trennung zwischen den für die Entwicklung verwendeten und den Testfällen aufgrund des geringen Umfangs an verfügbarem Material nicht strikt aufrecht erhalten wurde. Diese 10 Übungsblätter enthielten insgesamt 47 Aufgaben mit Abgabepflicht; die vereinzelt vorkommenden Aufgaben ohne Abgabepflicht wurden nicht unter die Testfälle aufgenommen (und hätten sich schon aufgrund der Art der Aufgabenstellung auch nicht dafür geeignet). Viele der Aufgaben gliederten sich wiederum in Teilaufgaben; auch diese sind in der Zählung nicht enthalten; eine Aufgabe mit 5 Aufgabenteilen zählt hier also nur als eine. Von diesen 47 Aufgaben waren nur 12 für eine Behandlung mit Diproche insofern geeignet, als die Aufgabenstellung sich in der vom Diproche-System aktuell zur Verfügung gestellten natürlichen Sprache für die Geometrie ausdrücken ließen. Dabei wurde eine Aufgabe mit mehreren Aufgabenteilen als "geeignet" gezählt, wenn mindestens eine der Teilaufgaben geeignet war. Die als "ungeeignet" aussortierten Übungsaufgaben wiesen eines

der folgenden Merkmale auf:

- Sie waren "Meta-Aufgaben", die nicht unmittelbar die Gegenstände der axiomatischen Geometrie (wie Punkte, Geraden, Kreise...) betrafen, sondern Modelle gewisser axiomatischer Theorien, etwa von der Art "Finde eine affine Ebene so, dass..." oder "Zeige, dass eine affine Ebene der Ordnung n genau n^2 Punkte enthält". Solche Aufgaben sind der Formulierung in einer rein geometrischen Sprache nicht zugänglich (jedenfalls nicht in einer natürlichen, textnahen Art und Weise), sondern würden eine Sprache erfordern, zu deren Domäne Begriffe wie "Struktur" oder "Theorie" gehören, also zumindest Fragmente aus einer Sprache der Modelltheorie.

- Sie bezogen sich auf bestimmte endliche Modelle gewisse Axiome, wie "Prüfe, ob der der Satz des Thales in folgendem Modell M gilt." (zumeist auf das sogenannte 9-Punkte-Modell, siehe [139]). Die aktuelle Version von Diproche unterstützt die Arbeit in einem bestimmten Modell nicht. Obwohl es problemlos möglich wäre, ein solches Modell in dem Sinne zu implementieren, dass formale geometrische Aussagen darin ausgewertet werden können, wäre das für die in der meisten Übungsaufgaben etwa zum 9-Punkte-Modell kaum hilfreich. Diese erfordern zumeist nicht axiomatisch-geometrische, sondern eher kombinatorische Argumente, die typischerweise in einer Meta-Sprache formuliert sind und in erheblichem Umfang Gebrauch von Formulierungen wie "ohne Beschränkung der Allgemeinheit" oder Symmetrieüberlegungen machen. Obwohl solche Argumente einer automatisierten Verifikation der hier beschriebenen Art nicht prinzipiell unzugänglich sein mögen – man könnte etwa im Falle von Symmetrieargumenten die Fälle, die sich "symmetrisch' ergeben sollen, automatisch explizit generieren und dann diese Explikationen prüfen – werden sie von der aktuellen Diproche-Version nicht unterstützt.

- Sie enthielten Begriffe aus der dynamischen Geometrie, insbesondere Spiegelungen. Obwohl solche geometrischen Operationen – wie Spiegelungen, Drehungen, Verschiebungen, Streckungen etc. – sich selbstverständlich auch in Diproche implementieren lassen (wie man erkennt, wenn man die vorhandenen Systeme zum automatischen Beweisprüfen in der Geometrie betrachtet), sind sie aktuell nicht vorgesehen. Der Grund hierfür ist einfach der, dass das aktuelle Interface keine ausreichend bequeme und natürliche Eingabe hierfür erlaubt. Diese Möglichkeiten werden mit dem Neudesign und der Weiterentwicklung des Interface allmählich ergänzt werden.

- Es handelte sich gar nicht um Beweisaufgaben im von Diproche betrachteten Sinn, sondern um Aufgaben, in denen gewisse Objekte zu zählen, Beispiele

für gewisse Objekte zu zeichnen, eine gewisse Konstruktion auszuführen oder ein "Beweispuzzle" zu lösen waren.

Einige der 12 oben erwähnten Übungsaufgaben enthielten Teilaufgaben, die ihrerseits für eine Behandlung in Diproche ungeeignet waren. Durch Aussondern der geeigneten Teilaufgaben der geeigneten Übungsaufgaben erhielten wir einen Bestand von 23 geeigneten Teilaufgaben. In der Behandlung dieser im Rahmen des aktuellen Systems immerhin formulierbaren Aufgaben traten weitere Schwierigkeiten zutage:

- Obwohl die Aufgabenstellung sich inhaltlich nur auf Objekte aus der Domäne der axiomatischen Geometrie bezog, enthielten sie "hochstufige" und damit schwer übersetzbare Formulierungen, wie etwa "In jeder affinen Ebene gehen durch jeden Punkt mindestens drei Geraden". Dies läßt sich zwar durch eine Formulierung wie "Es sei p ein Punkt. Dann existieren Geraden l, g, h so, dass $\sim g = h$ und $\sim g = l$ und $\sim h = l$ und $p \in g$ und $p \in h$ und $p \in l$." in die Diproche-Sprache übersetzen, die dann vom System auch verarbeitet werden kann. Allerdings sind diese Formulierungen schon für die Aufgabenstellung selbst recht umständlich; umso mühsamer wird es, eine Lösung in dieser Sprache zu schreiben, insbesondere dann, wenn (wie es die Zielverfolgung verlangt) der Beweistext mit der zu beweisenden Aussage enden muss. In solchen Fällen wurde daher der Anspruch, eine von der Zielprüfung als solche erkannte Lösung schreiben zu lassen, fallen gelassen und die Rückmeldung der Zielverfolgung ignoriert; der/die BenutzerIn sollte die Aufgabe als gelöst ansehen, sobald drei Geraden definiert wurden und von diesen gezeigt wurde, dass sie paarweise verschieden sind und den Punkt p enthalten.

- Triviale Sonderfälle: Typische Beispiele entarteter Sonderfälle in der Geometrie sind Dreiecke, in denen zwei oder alle drei Ecken zusammenfallen oder in denen die Ecken kollinear sind. Häufig gelten Allaussagen über Dreiecke auch in solchen Fällen, erfordern dann aber ein modifiziertes Argument (das nicht selten eine Trivialisierung des Arguments für die "typischen Fälle" darstellt). Obschon es in Diproche möglich ist, solche Argumente in Diproche etwa im Rahmen einer Fallunterscheidung zu formulieren, ist das doch oft unnötig umständlich. Aus diesem Grund wurden in den Aufgaben bisweilen Annahmen hinzugefügt, die triviale Spezialfälle explizit ausschlossen.

Aus diesen Gründen wurden beim Testen zum einen die Zieldeklarationen fortgelassen und die Übungsaufgabe als erfolgreich gelöst angesehen, sobald alle "Bestandteile" der angestrebten Schlussfolgerung erreicht sind (was dem üblichen Vorgehen bei der Korrektur von fortgeschritteneren Übungsaufgaben

durch menschliche KorrektorInnen entspricht) und zum anderen wurden Übungsaufgaben durch Zusatzannahmen vereinfacht und umformuliert, die entartete Spezialfälle ausschließen.

Mit diesen Veränderungen konnten 13 Teilaufgaben mit der aktuellen Diproche-Version erfolgreich bearbeitet werden, also etwas mehr als 50 Prozent. Bei den meisten Übungsaufgaben war dabei noch ein Eingriff in den Quellcode erforderlich; in den meisten Fällen handelte es sich um Korrekturen einfacher Implementierungsfehler (etwa die Korrektur einer falsch benannten Variable im ATP), einige erforderten es, weitere ATP-Regeln oder neue Varianten bereits vorhandener ATP-Regeln hinzuzufügen. In einem Fall wurden neue Formulierungen ins Vokabular aufgenommen. Für diese 13 Übungsaufgaben war die Verarbeitungszeit angemessen niedrig (im Durchschnitt weniger als 5 Sekunden auf unserem Bürorechner an der Europa-Universität Flensburg). Bei den übrigen Aufgaben, die als Misserfolg klassifiziert wurden, traten eine oder mehrere der folgenden Schwierigkeiten auf:

1. Obwohl die Aufgabe selbst innerhalb des derzeitigen begrifflichen Rahmens formuliert werden konnte, machte die intendierte Lösung Gebrauch von Begriffen oder Inferenzregeln, die in der aktuellen Version nicht unterstützt werden (z.B. solchen aus der dynamischen Geometrie, s.o.).

2. Die automatische Prüfung war "zu präzise". Beispielsweise fiel während der Eingabe einer Lösung auf, dass zusätzliche Argumente erforderlich waren, um etwa zu zeigen, dass zwei Geraden verschieden sind, obwohl das "aufgrund der Zeichnung offensichtlich" war. Das kann einerseits als ein positiver Effekt der Verwendung automatischer Prüfmethoden angesehen werden, die eben, ohne Bezug zur Anschauung, lediglich den vorliegenden Text verarbeiten und ist bei Aufgaben, die gegenüber dem geometrischen eher das axiomatische Denken betonen, sogar begrüßenswert; sobald aber der "kreative" Aspekt einer Aufgabe den "logischen" überwiegt, wird man so zu (unnötig) langen und umständlichen Texten gezwungen, um eine richtige und gute Beweisidee darzustellen.

Während (1) ein rein technisches Problem ist, das durch eine Erweiterung der bestehenden Systemkomponenten gelöst werden kann, ist (2) ein deutlich mehr prinzipielles – oder auch "soziologisches" – Problem, an dem sich konkret die im zweiten Kapitel herausgearbeiteten Beschränkungen der Möglichkeiten automatischen Beweisprüfens zeigen: Mit der zunehmenden inhaltlichen Entwicklung in einer Vorlesung werden gewisse Formen von Ungenauigkeiten in Beweisen akzeptabel, unter anderem deswegen, weil die Ansprüche an formale Strenge und Präzision der Darstellung zugunsten einer größeren Bequemlichkeit,

mit der "wesentliche Ideen" formuliert werden können, abgesenkt werden. In Lehrbüchern wird dies bisweilen durch Bemerkungen expliziert, die ankündigen, dass "solche Fälle in Zukunft nicht mehr erwähnt" werden etc.; häufig sind das extreme Spezialfälle, die etwa entstehen, wenn eine gewisse Menge leer, eine gewisse Zahl gleich 0 ist etc. Es ist eine prinzipielle und derzeit noch offene Entscheidung, ob Diproche auch nur versuchen sollte, diesen Teil der mathematischen Praxis abzubilden: Auf der einen Seite würde die Prüfung dadurch weniger zuverlässig und sich stärker an die durch eine/n menschliche/n KorrektorIn annähern, die oder der es entweder nicht bemerken oder absichtlich übergehen mag, dass gewisse "uninteressante" Spezialfälle nicht behandelt wurden. Um hier zu einer qualifizierten Empfehlung zu gelangen, wäre es wichtig, zu untersuchen, wie das vom Rechner zur Verfügung gestellte Feedback im Vergleich zu einem von menschlichen KorrektorInnen erzeugten Feedback von Studierenden verstanden und behandelt wird, was eine Frage für das Gebiet der "Mensch-Maschine-Interaktion" (siehe z.B. Boy [26]) ist. Diese Schwierigkeiten sind jedenfalls teilweise darauf zurückzuführen, dass "echte" Beweistexte, wie sie für die Kommunikation von Beweisen zwischen Menschen verwendet werden, keine rein logischen, sondern eben auch soziale Objekte sind, für deren Akzeptabilität der soziale Kontext eine wesentliche Rolle spielt (s.o.) und dass solche Kontexte schwierig, wenn nicht unmöglich in den klaren Grenzen logischer Systeme einzufangen sind. An diesem Punkt führt die Verwendung automatischer Beweisprüfer in der mathematischen Lehre in eine diffizile Verschlingung von Logik, Soziologie und Psychologie.

Für die axiomatische Geometrie zeigt sich also ein deutlich weniger klares Bild als für die zuvor behandelte elementare Zahlentheorie. Es sollte allerdings beachtet werden, dass die für die Evaluierung der "Spielwiese" zur elementaren Zahlentheorie herangezogenen Übungsaufgaben durchweg zu einem recht klar begrenzten Typ von Aufgabe gehörten, während die betrachteten Geometrie-Aufgaben von Übungsblättern stammten, die einen zweieinhalb Monate langen Vorlesungsverlauf begleiteten und bezüglich Inhalt, verwendetem Vokabular, Schwierigkeit und erforderlichen (heuristischem wie faktischem) Hintergrundwissen deutlich vielfältiger waren. Vor diesem Hintergrund ziehen wir aus den oben besprochenen Erfahrungen folgende Schlussfolgerungen:

1. Wie kaum überraschen kann, ist von Diproche nicht zu erwarten, dass es gut auf beliebige Übungsaufgaben zur axiomatischen Geomtrie anwendbar ist. Obwohl das in einigen Fällen durchaus funktioniert, müssen Übungsaufgaben, die für den Einsatz von Diproche vorgesehen sind, sorgfältig ausgewählt und formuliert werden.

2. Für sorgfältig ausgewählte Übungsaufgaben akzeptiert Diproche Beweistexte, die sprachlich den Musterlösungen recht ähnlich sind, erfordert

aber eine zusätzliche “Schicht” an formaler Präzision und inhaltlicher Ausführlichkeit; für Aufgaben, die eher den axiomatischen Charakter der axiomatischen Geometrie betonen, ist das durchaus von Vorteil.

Kapitel 7

Die (Eingabe)sprache von Diproche

Sicherlich ist die Verwendbarkeit eines Systems wie Diproche entscheidend von der akzeptierten Eingabesprache abhängig; diese soll nun dargelegt werden. Dazu stellen wir in diesem Abschnitt zunächst einige Vorüberlegungen zur Wahl einer geeigneten Eingabesprache für das System Diproche an. Auf diesen aufbauend wird die Sprache von Diproche im Anschluss beschrieben. Zentrale Punkte der Diproche-CNL, wie etwa die Einteilung von Sätze in Annahmen und Behauptungen, die wiederum durch gewisse Anfangsphrasen unterschieden werden, sind parallel zu der etwa in Cramer [53] vorgestellten Naproche-CNL; abgesehen davon, dass es sich bei der Naproche-CNL um ein kontrolliertes Fragment der englischen Sprache, bei der Diproche-CNL hingegen um ein kontrolliertes Deutsch handelt, ergeben sich aus der didaktischen Zielsetzung von Diproche auch auf der Strukturebene diverse Unterschiede (beispielsweise im Umgang mit Deklarationen), auf die wir an den entsprechenden Stellen eingehen.

7.1 Vorüberlegungen zur Konstruktion einer geeigneten CNL für Diproche

Die Eingabesprache von Diproche soll der natürlichen Sprache, in der Mathematik üblicherweise ausgedrückt wird, zumindest so ähnlich sein, dass ihre Verwendung für StudienanfängerInnen über den zum Studienbeginn stets zu leistenden Erwerb der mathematischen Fachsprache hinaus keine wesentliche Schwierigkeit darstellt; Ziel der Verwendung des Systems ist es, Mathematik zu lernen, nicht die Handhabung eines spezifischen (Quasi-)Formalismus. Andererseits sollte die durch die Benutzung von Diproche erlernte Sprache zur Formulierung von Beweisen auch in Kontexten tauglich sein, in denen nicht mehr mit dem System kommuniziert wird oder doch zumindest beim Erwerb solcher Ausdrucksfähigkeiten ein sinnvoller erster Schritt sein.

Damit liegt das Ziel nahe, die Eingabesprache von Diproche möglichst weit an die aus schriftlichen mathematischen Diskursen – Musterlösungen, Lehrbüchern, Forschungsaufsätzen etc. – anzunähern. Die Zielsetzung, die natürliche mathematische Sprache maschinenlesbar zu machen, ist bereits verschiedentlich verfolgt worden; wir haben bereits oben das Naproche-System erwähnt, ebenso wie SAD, das auf der Eingabesprache ForThel basiert. Eine weitreichende linguistische Analyse der natürlichen mathematischen Sprache mit dem Ziel der Automatisierbarkeit hat Ganesalingasm [83] vorgelegt; das Ergebnis ist ein Parser für mathematisches Englisch, der einen nennenswerten Anteil von Sätzen, die in mathematischen Texten vorgefunden werden können, syntaktisch verarbeiten und ihre Semantik extrahieren kann (wozu insbesondere die Auflösung einer erstaunlichen Vielzahl von Ambiguitäten gehört, die sich im Verlauf von Ganesalingams Arbeit in der mathematischen Sprache gezeigt haben, siehe insbesondere [83], Kapitel 4).

Auch angesichts dieser technischen Möglichkeiten gibt es indes Gründe, die Eingabesprache von Diproche gezielt kleiner zu halten. Dazu gehört zum einen, dass der intendierte Einsatzbereich – Übungsaufgaben für StudienanfängerInnen – gegenüber der Gesamtheit mathematischer Texte deutlich eingeschränkt ist. Darüber hinaus stellt die logisch-mathematische Fachsprache, in der Beweistexte geschrieben werden sollten, gegenüber der natürlichen Alltagssprache geradezu eine Fremdsprache dar. Nicht nur ist das Vokabular zum Teil unvertraut ("genau dann, wenn"), es werden auch bekannte Begriffe in einem technisch präzisierten Sinn gebraucht, der erst erlernt werden muss (so etwa der Unterschied zwischen "Es gilt" als Einführung einer Behauptung und "Es gelte" als Einführung einer Annahme, der gerade zu Beginn häufig zu Verwechselungen Anlass gibt). Diese sprachlichen Eigenarten der Mathematik stellen für StudienanfängerInnen eine wesentliche Hürde dar, deren Nichtbewältigung den gesamten weiteren Studienverlauf beeinträchtigt.[1] Dennoch wird diese "Sprache der Mathematik" nur selten explizit unterrichtet.[2] Da der Einsatz von Diproche stets auch eine Vermittlung der akzeptierten Eingabesprache erfordert, gibt er zugleich zu einer

[1] Dieses Problem potenziert sich, wenn die Lehre nicht in der Muttersprache des oder der Studierenden erfolgt, siehe etwa Yushua und Bokhari, [213].

[2] Vgl. etwa Jamison [123], p. 46: "For the most part, this structure can be traced back to the Greeks, who in their writing explicitly described these structures. Unfortunately, this structure is often taught today by a kind of osmosis. Fragmented examples are presented in lectures and elementary texts. Over a number of years, talented students may finally unconsciously piece it all together and go on to graduate school. But the majority of students give up in despair and conclude that mathematics is just mystical gibberish". Jamison plädiert entsprechend dafür, die mathematische Sprache zum expliziten Lehrgegenstand zu machen und gibt didaktische Anregungen für die Umsetzung dieses Ansatzes. Als automatisches Korrekturinstrument kann Diproche auch als Beitrag zur Förderung dieses mathematischen "Spracherwerbs" gesehen werden.

solchen Explikation Anlaß, in der Formulierungsweisen mit ihrer Bedeutung und ihren Nuancen stärker bewußt gemacht werden. Dabei ist es sicherlich sinnvoll, mit einer möglichst einfachen Sprache zu beginnen und die Ausdruckskraft erst allmählich zu erhöhen, etwa mit Formulierungen für Annahmen und Folgerungen (wie sie für die Aussagenlogik ausreichen) zu beginnen und Annotationen, begründete und multiple Behauptungen, Absatzstruktur, Deklarationen mit inhaltlicher Annahme etc. dann sukzessive einzuführen. Die Sprache von Diproche stellt ein für ihren Bereich hinreichend ausdrucksstarkes Fragment der mathematischen Fachsprache dar, das mit einer überschaubaren – und daher (hoffentlich) leicht zu erlernenden – Anzahl an Formulierungen auskommt. Wenn es aber didaktisch sinnvoll ist, die mathematische Fachsprache in der Lehre explizit und sukzessive aufzubauen, sollten die Beschränkungen der Eingabesprache von Diproche zu Beginn eher einen hilfreichen Rahmen für Lernende wie für Lehrende als eine Schwierigkeit darstellen.

Als Gegenpol zum Ziel eines möglichst "natürlichen" und "ungezwungenen" Formulierens im Rahmen des Systems ergibt sich aus der didaktischen Zielsetzung der Sensibilisierung für die Besonderheiten der mathematischen Fachsprache das Ziel einer Normierung im Hinblick auf logische Präzision. Die Formulierungen, die im Rahmen von Diproche verwendet werden können, sollten – nicht nur aus technischer, sondern eben auch aus didaktischer Sicht – eine klare und eindeutige logische Bedeutung haben. Neben der angezielten Förderung des mathematischen Sprachverständnisses hat dies einen zweiten Grund darin, die Interpretation, die das System von einem Text generiert, für den/die BenutzerIn so durchschaubar wie möglich zu halten. Insbesondere sollten daher logisch unterschiedliche Funktionen nicht durch identische Formulierungen ausgedrückt werden. Ambiguitäten der mathematischen Fachsprache, die von kompetenten LeserInnen auf verschiedene Weisen – etwa durch Kontextbetrachtungen oder pragmatische Erwägungen – aufgelöst werden, sollten folglich in der Diproche-CNL nach Möglichkeit vermieden werden. Dadurch erfolgt eine zusätzliche Normierung der natürlichen mathematischen Sprachpraxis. Ein Beispiel hierfür ist die strikte Unterscheidung zwischen Existenzaussagen und Variableneinführungen, der weiter unten noch einmal gesondert diskutiert wird.

Als Anforderungen an die Diproche-CNL ergeben sich damit folgende Punkte:

1. Nähe zur für einfache Übungsaufgaben im jeweiligen Themenbereich üblichen Sprachpraxis. Dazu gehören insbesondere typische Figuren wie etwa "$\rightarrow$" und $\leftarrow$ in Äquivalenzbeweisen oder "$\subseteq$" und "$\supseteq$" in Beweisen von Mengengleichheit; diese sollten zur Verfügung stehen.

2. Leichte und rasche Erlernbarkeit.

3. Logische Präzision und Eindeutigkeit.

Wir betonen an dieser Stelle mit Bezug auf Punkt (1), dass die Diproche-CNL ausschließlich der Darstellung der deduktiven Schrittfolge in Beweisen dient. Mathematische Texte enthalten – glücklicherweise – daneben noch eine Reihe weiterer Informationen: Strategische Vorüberlegungen etwa, die den Beweis einerseits vorbereiten und andererseits dem oder der LeserIn helfen, dem Beweistext auch heuristische Lehren zu entnehmen, oder Passagen, die der Veranschaulichung des Gedankenganges dienen und oft von Bildern begleitet sind. Solche Informationen können von Diproche natürlich weder gelesen noch verarbeitet werden und entsprechend ist die Diproche-CNL auch nicht darauf ausgelegt, sie ausdrücken zu können.[3] In Diproche ließen sich solche für den Beweisverlauf im engeren Sinne nicht erforderlichen Textteile durch eine Kommentarfunktion realisieren, durch die gewisse Textteile als von der Verarbeitung zu übergehende ausgewiesen werden können. Für die derzeit vorgesehene Einsatzweise von Diproche erscheint das aber kaum erforderlich, erfordern die elementaren Beweise, für die das System derzeit ausgelegt ist, doch weder strategische Vorüberlegungen noch gesonderte Anstrengungen zur Veranschaulichung.

Für die Entwicklung der Diproche-CNL wurde im wesentlichen der schon oben in den Abschnitten zum Design von Spielwiesen erläuterte Weg beschritten. Als Vorbild für die Diproche-CNL wurden Musterlösungen zu Übungsaufgaben aus verschiedenen Jahrgängen der Vorlesung "Algebra 1" an der Europa-Universität Flensburg herangezogen. Diese wurden in einem ersten Arbeitsschritt logisch disambiguiert, wobei darauf geachtet wurde, die Textsubstanz möglichst wenig zu verändern. Aus den so gewonnenen Texten wurden sodann wesentliche Argumentationsfiguren, Formulierungen und Notationen extrahiert. Der so erhaltene Vorrat an Formulierungsmöglichkeiten wurde dann durch Sichtung weiterer Bearbeitungen ergänzt. Damit ergab sich eine erste Version der Diproche-CNL. Die Weiterentwicklung erfolgte dann dadurch, dass eine Reihe von Aufgaben mithilfe des Systems bearbeitet wurde, sowohl durch den Autor als auch durch andere MitarbeiterInnen des Abteilung für Mathematik und ihre Didaktik an der EUF. Auf diese Weise traten immer wieder wünschenswerte, aber noch nicht verfügbare Formulierungen und Figuren zutage, die dann auf ihre Verträglichkeit mit der logischen Gesamtkonzeption der Diproche-CNL geprüft und ggf. in die CNL integriert wurden.

[3]Gerade für Hinweise zur Veranschaulichung erscheint die Angabe einer passenden CNL angesichts der Vielfalt der möglichen Bilder, die – wie etwa die bekannte Domino-Metapher zur Erläuterung von Induktionsbeweisen – beliebigen Bereichen des Alltags entstammen können, auch ein hoffnungsloses Unterfangen zu sein.

7.2 Die Diproche-CNL

Wir wollen nun die in Diproche verwendete kontrollierte natürliche Sprache vorstellen. Gegenstand der automatischen Prüfung durch Diproche sind Diproche-Texte. Ein Diproche-Text ist dabei eine Folge von einzelnen funktional selbstständigen Bestandteilen, die wir hier – in einem gegenüber dem üblichen Sprachgebrauch etwas erweiterten Sinn – "Sätze" nennen. Sätze sind dabei sowohl natürlichsprachliche Sätze im üblichen Sinn als auch Annotationen wie "Beweis:", "qed.", "Fall 1:", "⇒" oder auch ein Absatz (d.h. eine Leerzeile). In Diproche gibt es zunächst fünf Grundtypen von Sätzen, nämlich Annahmen, Deklarationen, Definitionen, Axiome, Behauptungen und Annotationen, wobei diese Grundtypen z.T. weiter in Untertypen ausdifferenziert sind. Für jeden dieser Satztypen gibt es spezifische Formulierungen, die sein Vorliegen anzeigen; ferner hat jeder eine bestimmte logische Funktion. Wir werden diese Satztypen weiter unten der Reihe nach behandeln.

7.2.1 Aussagen

Zunächst betrachten wir aber die Syntax der Aussagen selbst, denen durch ihren Status als Annahme, Axiom, Behauptung etc. eine logische Funktion innerhalb eines Beweistextes zugewiesen wird. (So betreffen etwa die Sätze "Angenommen, a ist gerade." und "Also ist a gerade." beide die Aussage "a ist gerade", der durch die einleitenden Formulierungen dann unterschiedliche Funktionen zugewiesen werden, denen dann auch verschiedene Verarbeitungsweisen durch das System entsprechen.)

Eine Aussage ist entweder atomar oder aus atomaren Aussagen zusammengesetzt. Atomare Aussagen sind von zwei möglichen Arten:

- Beliebige formale Ausdrücke, die Aussagen repräsentieren, wie $2|a$, $\forall x \exists y x^2 = x$ oder $(A \vee B) \to \neg C$.[4]
- Ein- oder mehrstellige Relationsaussagen in natürlicher Sprache, wie etwa

[4]Man beachte, dass damit nicht-atomare Formeln als atomare Aussagen gelten. In Diproche-Texten fungieren also formale Ausdrücke beliebiger Komplexität als primitive Objekte. Dies ist durchaus im Einklang mit der mathematischen Sprachpraxis, was man etwa daran erkennt, dass man im natürlichsprachlichen Text nicht auf ihre Bestandteile referieren kann, ohne sie vorher explizit einzuführen. So ist "Also ist a gerade oder b eine Quadratzahl. Da ersteres falsch ist, ist letzteres bewiesen." verständlich, während "Also $2|a \vee \exists y \in \mathbb{Z} x = y^2$. Da ersteres falsch ist, ist letztere bewiesen." zumindest irritierend wirkt und jedenfalls kaum als vorbildlicher mathematischer Sprachgebrauch gelten kann. Aus ähnlichen Gründen wird im Naproche-System zwischen mathematischen und natürlichsprachlichen Referenten unterschieden, siehe etwa Cramer [53], S. 4f sowie S. 197f.

"a ist gerade", "a ist ein Teiler von b", "g ist parallel zu h" oder "g ist die Parallele zu h durch den Punkt p".

Diese atomaren Aussagen können nun durch natürlichsprachliche Junktoren modifiziert bzw. verbunden werden:

- Durch Disjunktion, in Formulierungen wie "a ist gerade oder b ist eine Quadratzahl".
- Durch Konjunktion, in Formulierungen wie "a ist gerade und b ist eine Quadratzahl" oder "sowohl ist a gerade als auch b eine Quadratzahl".
- Durch Implikation, in Formulierungen wie "Wenn a gerade ist, dann ist b eine Quadratzahl" oder "$2|a$ impliziert $3|b$".
- Durch Biimplikation, in Formulierungen wie "a ist gerade gdw $a-1$ ungerade ist", "a ist gerade genau dann, wenn $a-1$ ungerade ist", "$2|a$ ist aequivalent zu $3|b$".
- Durch Negation; hier steht einerseits das "nicht" zur Verfügung für Formulierungen wie "a ist nicht gerade", was bei Eigenschaftszuschreibungen wie "a ist eine Quadratzahl" statt des etwas sperrigen "a ist nicht eine Quadratzahl" auch als "a ist keine Quadratzahl" ausgedrückt werden kann; andererseits kann eine Aussage auch dadurch negiert werden, dass explizit ihre Falschheit behauptet wird, in Formulierungen wie "$3|a$ ist falsch".

Im Gegensatz zu den rekursiven Bildungsregeln für formale Aussagen ist die Schachtelung natürlichsprachlicher Junktoren nur bedingt sinnvoll. Zum einen werden die so gebildeten Formulierungen schnell unübersichtlich; wichtiger sind die begrenzten Möglichkeiten, Klammerungen auszudrücken. Schon bei einer Verwendung von zwei Junktoren in einer Aussage wie "Also gilt A oder B und C." kann zwischen den beiden Interpretationen $(A \vee (B \wedge C))$ und $((A \vee B) \wedge C)$ anhand der schriftlichen Darstellung des Satzes nicht unterschieden werden.[5] Auch wenn es im Prinzip möglich wäre, hier über die Prioriätsregeln zwischen den Junktoren eine Lesart auszuzeichnen, würde man sich hiermit wohl recht weit von einer natürlichen und intuitiven Sprachpraxis entfernen: Auch wenn man BenutzerInnen dann jeweils leicht erklären könnte, worin der Fehler besteht, würden solche in didaktischer Hinsicht nicht sonderlich lehrreichen Fehler wohl immer wieder gemacht und so die Bedienbarkeit des Systems verringern. Aus diesem Grund wurde die Schachtelung von natürlichsprachlichen Junktoren in Diproche eng

[5] In der gesprochenen Sprache kann beides noch durch lautliche Marker wie etwa Variationen in der Sprechgeschwindigkeit ausgedrückt werden.

begrenzt: Die Negation ist mit allen anderen Junktoren beliebig kombinierbar; ansonsten sind Kombinationen von Implikation und Disjunktion bzw. Konjunktion in Formulierungen wie "Wenn a nicht gerade ist, dann ist b gerade oder c ist ungerade." zwar in sehr begrenztem Umfang möglich, funktionieren aber nicht stabil und sollten aus den eben genannten Gründen vermieden werden. Die bisherige Erfahrung zeigt, dass sich mit diesen begrenzten Mitteln das Ziel, eine größere Menge an Übungsaufgaben auf natürliche Weise bearbeiten zu können, bereits erreichen läßt. Wo komplexere Kombinationen von Junktoren erforderlich sind, besteht die Möglichkeit, diese durch einen formalen Ausdruck wiederzugeben.

Auch Quantoren können in der Diproche-CNL natürlichsprachlich ausgedrückt werden. Für den Allquantor stehen hierzu folgende Formulierungen zur Verfügung:

- Fuer alle x gilt: ...
- Fuer jedes a gilt: ...

Allerdings führt die Verwendung einer quantifizierten Variablen ohne Angabe eines Typs dazu, dass ein Typenfehler gemeldet wird. Die Beschränkung des Allquantors auf einen Typ von Objekt kann durch Formulierungen wie "Fuer alle ganzen Zahlen x gilt: ..." bzw. "Fuer jede ganze Zahl x gilt: ..." vorgenommen werden. Ferner kann durch Wendungen wie "Fuer alle ganzen Zahlen x, y und z gilt ..." über mehrere Variablen zugleich quantifiziert werden, durch Aufzählung ("Fuer alle ganzen Zahlen x, y, alle reellen Zahlen z und alle rationalen Zahlen q gilt...") auch simultan über Variablen verschiedenen Typs. Auch das häufig anzutreffende "Nachklappen" des Allquantors in Wendungen wie "Es gilt $x = x$ fuer jede reelle Zahl x" wird von Diproche akzeptiert.

Existenzquantoren werden über die Formulierung "Es existiert/existieren" bzw. "Es gibt" eingeführt. Auch diese können mit Typenbeschränkungen kombiniert oder zur simultanen Quantifikation über mehrere Variablen verwendet werden. Beispielhafte Verwendungsweisen von Existenzquantoren in Diproche sind folgende:

- Es existieren ganze Zahlen x, y mit $x = y + 3$.
- Es gibt eine reelle Zahl q so, dass $q^2 = q$.

Quantorenwechsel lassen sich im Prinzip durch Iterationen dieser Konstrukte bilden. So ließe sich etwa die Formel

$$\forall x \in \mathbb{Z} \exists y \in \mathbb{Z} \forall z \in \mathbb{Z} x \cdot y < z$$

ausdrücken durch

Fuer alle ganzen Zahlen x gilt: Es existiert eine ganze Zahl y so, dass gilt: Fuer alle ganzen Zahlen z gilt: $x * y < z$. (*)

Für den häufigsten Fall eines einzelnen Quantorenwechsels ist diese Ausdrucksweise eher unüblich; Diproche erlaubt daher auch die folgenden, gängigeren Formulierungen:

- Fuer jede ganze Zahl x gibt es eine reelle Zahl y mit $x = y$.
- Es existiert eine reelle Zahl x so, dass $x = y$ fuer jede ganze Zahl y.

Die Trennung zwischen akzeptierten Formulierungen und der Vielfalt von "freien" Wendungen, die ihnen ähnlich sind, aber dem System derzeit nicht bekannt sind, erfordert hier einige Übung. Für den ersten Umgang mit dem System ist es daher, trotz der Vielzahl der sprachlichen Gestaltungsmöglichkeiten, sinnvoll, sich an das in (*) demonstrierten Darstellungsmuster zu halten.

Nachdem wir die Darstellung der Bildung natürlichsprachlicher Aussagen mit den sprachlichen Mitteln von Diproche damit abgeschlossen haben, kommen wir nun zu den in Diproche vorgesehen logischen Funktionen und den entsprechenden Formulierungsmöglichkeiten. Ein Diproche-Text ist eine Folge von Elementen der folgenden Typen:

7.2.2 Annahmen

Durch eine Annahme wird für den weiteren Beweisgang – bis die Annahme zurückgezogen wird – eine zusätzliche Aussage eingeführt, die bei der Herleitung der zwischen der Einführung und dem Zurückziehen der Annahme auftretenden Behauptungen verwendet werden kann. Allerdings gelten auch sämtliche im Geltungsbereich der Annahme auftretenden Behauptungen nur bedingt unter der jeweiligen Annahme.

In der aktuellen Version der Diproche-CNL werden Annahmen mit einer der folgenden Formulierungen eingeführt:

- Angenommen, es gilt...
- Angenommen...
- Gelte...
- Es gelte...
- Nehmen wir an, dass...
- Nehmen wir an, es gilt...

- Gesetzt, dass...

Die Hauptfunktion von Annahmen ist der Zusammenhang von Inferenzen mit der Implikation: Wenn eine Behauptung B korrekt aus einer Annahme A gefolgert wird, so gilt die Aussage $A \to B$ als bewiesen. Eine Annahme steht im Beweis für weitere Beweisschritte zur Verfügung, bis sie geschlossen wird. Für den Geltungsbereich von Annahmen in einem Beweistext gelten folgende Regeln:

- Eine Annahme gilt generell von der Stelle ihrer Einführung bis zum Ende des Absatzes, in dem sie eingeführt wurde; an dieser Stelle endet ihre Gültigkeit, mit folgender Ausnahme:
- Annahmen, die direkt nach einer Annotation (s.u.) eingeführt werden, die einen Beweisanfangsmarker einführt, gelten bis zum entsprechenden Beweisendmarker (s.u.). Beweisanfangsmarker sind u.a. "Beweis", "=>" und "<=" sowie Falleinführungen im Rahmen einer Fallunterscheidung.

Beispiel: Betrachten wir folgenden Diproche-Text (bei dem die einzelnen Beweiszeilen der besseren Beschreibbarkeit halber nummeriert wurden, wobei diese Nummerierung aber kein Teil der Diproche-Syntax ist):

(1) Angenommen, es gilt a.
(2) [Absatz]
(3) Fallunterscheidung.
(4) Fall 1. (5) Nehmen wir an, es gilt $\neg b$.
(6) Dann folgt $(\neg b \vee a)$. (7) qed.

(8) Fall 2. (9) Nehmen wir an, es gilt b.
(10) Dann folgt $(a \to b)$. (11) qed.

(12) Also gilt c.

Hier gilt (1) wegen des gleich darauf folgenden Absatzes (2) für keine der folgenden Zeilen. Dagegen gilt (5) trotz des Absatzes für (6), weil es direkt nach einem Beweisanfangsmarker eingeführt wurde, aber nicht mehr für (8) oder die folgenden Zeilen, weil zwischendurch der Beweisendmarker "qed" steht; und entsprechend gilt (9) für die Zeile (10), aber keine weiteren Zeilen.

7.2.3 Deklarationen

Ein logisch korrekter Beweis erfordert es inbesondere, die verwendeten Variablen einzuführen und anzugeben, zu welchem Bereich die durch sie bezeichneten

Objekte gehören. Sätze, die den Typ eines durch eine Variable zu bezeichnenden Objektes angeben, heißen – in Übereinstimmung mit der in der Informatik üblichen Terminologie – Deklarationen. Deklarationen werden durch folgende Formulierungen eingeführt, die wir als "Deklarationsinitialphrasen" (DIP) bezeichnen:

- Es sei/seien...
- Betrachte...
- Wähle...
- Fixiere...

Eine Deklaration hat dann im einfachsten Fall eine der Formen DIP+Var+TP bzw. DIP+TP+Var, wobei Var eine Variable und TP die jeweilige Typenbezeichnung ist; Beispiele für Deklarationen wären etwa "Es sei x eine reelle Zahl" oder "Betrachte eine ganze Zahl z". Es ist ebenfalls möglich, mehreren Variablen zugleich denselben Typ zuzuweisen und mehrere Deklarationen in einem Satz vorzunehmen. Außerdem können komplexere Typen wie Mengen und Funktionen in natürlicher Sprache beschrieben werden.

Einige Beispiele für in Diproche akzeptable Deklarationen:

- Es sei n eine natuerliche Zahl.
- Es sei f eine Funktion von den Mengen von natuerlichen Zahlen in die Funktionen von den reellen Zahlen in die reellen Zahlen.
- Es seien m, n und k sowie p und q reelle Zahlen.
- Es seien x und y reelle Zahlen, n eine rationale Zahl sowie m und n Mengen.

Besonders in Fällen, in denen die Existenz eines Objektes mit gewissen Eigenschaften gezeigt wurde und nun ein Name für ein solches Objekt eingeführt werden soll, ist es bequem, Deklaration und inhaltliche Annahme zu kombinieren. Das ist in Diproche mit Formulierungen wie "Es sei/en ... so, dass ..." möglich, wie in folgenden Beispielen:

- Es sei x eine Menge mit $x \notin x$.
- Fixiere reelle Zahlen a und b derart, dass $a < b$.
- Waehle eine ganze Zahl k so, dass $x = 2 * k$.

Zu beachten ist, dass durch Deklarationen mit inhaltlicher Annahme, im Gegensatz zu Annahmen, die Existenz eines Objektes der fraglichen Art präsupponiert wird. Entsprechend wird bei Deklarationen mit inhaltlichen Annahmen zunächst geprüft, ob die Existenz so eines Objektes bereits bewiesen wurde oder aus den bisher erreichten Beweiszielen hervorgeht. Z.B. wäre die Satzfolge "Es sei k eine ganze Zahl. Angenommen, es gilt $x = 2 * k$." eine reine Folge von Annahmen, die keine weitere logische Prüfung erfordert; dagegen wird bei "Es sei k eine ganze Zahl so, dass $x = 2 * k$." zunächst überprüft, ob die Existenz eines solchen k bereits gezeigt wurde; ohne weiteren Kontext würde hier also eine nicht verifizierbare Aussage gemeldet. Dagegen würde "Es sei x eine ganze Zahl. Angenommen, x ist gerade. Es sei k eine ganze Zahl so, dass $x = 2 * k$." akzeptiert, weil hier die Existenz eines solchen k aus der Annahme folgt, dass x gerade ist.

Deklarationen und Existenzbehauptungen

Der Unterschied zwischen Existenzbehauptungen ("Es gibt eine ganze Zahl x so, dass ϕ") und Deklarationen ("Es sei x eine ganze Zahl so, dass ϕ") besteht darin, dass letztere den Referenten "x" in den Diskurs einführen, so dass in weiteren Sätzen darauf Bezug genommen werden kann, erstere hingegen nicht. So würde in

> Es existiert eine ganze Zahl i mit $i > 1$. Dann ist $i > 1$.

der zweite Satz sowohl als logisch nicht verifizierbar wie auch als hinsichtlich der Typenverwendung inkorrekt gemeldet, weil das i an der Stelle, wo es verwendet wird, nicht mehr zur Verfügung steht und also als nicht eingeführte Variable betrachtet wird. Dagegen würde der zweite Satz in

> Es sei i eine ganze Zahl so, dass $i > 1$. Dann ist $i > 1$.

sowohl logisch als auch hinsichtlich der Typenverwendung akzeptiert.

Diese Trennung zwischen Existenzbehauptung und expliziter Einführung wird im mathematischen Sprachgebrauch häufig verwischt und mag daher unnötig pedantisch erscheinen. Es ist allerdings zu beachten, dass die Regel, Existenzbehauptungen generell so zu lesen, dass sie den betreffenden Referenten einführen, rasch zu Inkonsistenzen führt. Betrachten wir dazu folgendes Beispiel:

> Es existiert eine ganze Zahl i mit $i = 0$. Es existiert eine ganze Zahl i mit $i = 1$. Also ist $0 = i = 1$.

Hier sind sowohl der erste als auch der zweite Satz logisch und bezüglich der Typenverwendung korrekte Existenzaussagen; würden aber beide so gelesen, dass sie den Referenten i einführen, wäre die Konsequenz $0 = 1$ unausweichlich; man hätte also aus zwei korrekten Aussagen einen Widerspruch gefolgert. Dieses Problem wird in Diproche dadurch vermieden, dass Existenzaussagen keine Referenten in den Diskus einführen. Damit wäre $0 = i = 1$ eine Aussage über ein unbekanntes neues Objekt i, die als logisch nicht verifizierbar und als fehlerhaft in Bezug auf die Typenverwendung gemeldet würde.

Nun läßt sich das Beispiel unter Verwendung von Deklarationen anstelle von Existenzaussagen umformulieren:

> Es sei i eine ganze Zahl mit $i = 0$. Es sei i eine ganze Zahl mit $i = 1$. Dann ist $0 = i = 1$.

Hier erkennt Diproche sowohl den ersten als auch den zweiten Satz als Deklarationen, die i in den Diskurs einführen, so dass die Folgerung $0 = i = 1$ logisch korrekt ist. Allerdings wird der zweite Satz als in Bezug auf die Typenverwendung fehlerhaft gemeldet, weil an dieser Stelle der bereits eingeführte Referent i erneut deklariert wird. Auf diese Weise führt die mehrfache Einführung eines Referenten zu Fehlermeldungen, weswegen die Herleitung des Widerspruchs blockiert ist. Man beachte aber, dass die Typenkorrektur hier einen wesentlichen Beitrag zur logischen Korrektheit leistet und Typenfehler insofern nicht als rein "kosmetische" Fehler betrachtet werden sollten, die durch das Hinzufügen von Deklarationen behoben werden können; vielmehr können Typenfehler auf wesentliche logische Fehler hinweisen.

Diproche unterscheidet also klar zwischen reinen Existenzaussagen und Deklarationen. Ein Argument wie "Angenommen, x ist gerade. Dann existiert eine ganze Zahl i mit $x = 2i$. Also ist $x^2 = (2i)^2 = 4i^2$." wird also von Diproche als fehlerhaft bewertet, obwohl ein solches Vorgehen in Darstellungen von Lösungen für einfache Beweisaufgaben durchaus üblich sind. (Auch die Erfahrung im Einsatz zeigt, dass selbst DozentInnen sich an diesen Punkt erst gewöhnen müssen.) Um die Intention des obigen Argumentes in einer von Diproche akzeptierten Weise zu formulieren, müsste man sagen: "Angenommen, x ist gerade. Dann existiert eine ganze Zahl i mit $x = 2i$. Es sei i eine ganze Zahl mit $x = 2i$ Also ist $x^2 = (2i)^2 = 4i^2$.", also auf die Existenzbehauptung eine explizite Deklaration folgen lassen. Hierin liegt also eine Beschränkung der Natürlichkeit der von Diproche akzeptierten Lösungen.

Diese Beschränkung scheint uns allerdings zum einen in didaktischer Hinsicht vorteilhaft zu sein, wird hierdurch doch ein durchaus wichtiger und subtiler Punkt des mathematischen Sprachgebrauchs betont. Das Vorgehen von Diproche steht also im Einklang mit dem oben formulierten Ziel, die mathematische Sprache

mit ihren Eigenarten zum expliziten Gegenstand der Lehre zu machen. Das in der mathematischen Darstellungspraxis übliche "Verwischen" des Unterschiedes zwischen Existenzbehauptung und Deklaration ist dann unproblematisch, wenn dieser Unterschied grundsätzlich klar ist und AutorInnen von Texten wie deren LeserInnen z.B. aufgrund pragmatischer Erwägungen wissen, wie der Text präzise zu interpretieren ist, etwa durch Heuristiken wie "Betrachte Existenzquantoren als einführend, solange dadurch kein Konflikt auftritt; andernfalls löse den Konflikt so auf, dass der Text möglichst sinnvoll und der Beweis erfolgreich ist". Weder derartige Heuristiken noch die für die präzise Interpretation erforderlichen Unterscheidungen können aber auf Seiten der/des Anfängers/In vorausgesetzt werden. Indem Diproche die ausdrückliche Unterscheidung in der Darstellung erfordert, trägt es dazu bei, die Kompetenzen auszubilden, die später einen nachlässigeren Sprachgebrauch ermöglichen.

Zum anderen werden die etwas uneleganten Verdopplungen von Aussagen, die zuerst in Existenzbehauptungen und dann in Deklarationen auftreten, in Diproche dadurch vermieden, dass in den fortgeschritteneren Schwierigkeitsgraden die Deklaration auch ohne vorherige Existenzbehauptung verwendet werden kann, sofern die damit einhergehende implizite Existenzbehauptung an der entsprechenden Textstelle verifizierbar ist. So würde auch die Formulierung "Angenommen, x ist gerade. Es sei i eine ganze Zahl mit $x = 2i$. Also ist $x^2 = (2i)^2 = 4i^2$." akzeptiert.

Wir wollen diese Vorgehensweise kurz mit der deutlich anderen kontrastieren, die in Naproche benutzt wird und dafür sorgt, dass Naproche auch in dieser Hinsicht "nachlässig" formulierte Texte akzeptiert.[6] Naproche verwendet zur Verarbeitung der Variablenbindung eine Variante eines als "Dynamische Prädikatenlogik" (DPL) bekannten Formalismus, in dem es möglich ist, dass ein Referent sein Bezugsobjekt ändert. In der Naproche-Verarbeitung eines Textes erhält jeder Referent eine interne Kennung, die bei mehreren Vorkommen desselben Referenten zunächst unterschiedlich ist. Erst durch eine weitere Verarbeitung wird festgestellt, welche Vorkommen eines Referenten sich auf dasselbe Objekt beziehen. Nun hat die Einführung eines bereits eingeführten Referenten den Effekt, dass die ursprüngliche Bindung des Referenten aufgehoben wird und die Referenz sich ändert. Verdeutlichen wir Vorkommen desselben Referenten mit verschiedenem Bezug durch eine Indizierung, so wird unser Beispiel von oben durch Naproche also wie folgt interpretiert:

[6]Für die Behandlung der Variablenbindung durch Existenzquantoren in Naproche siehe Cramer [53], S. 87f. Ferner danken wir Marcos Cramer für seine Erläuterungen dazu im persönlichen Gespräch.

Es existiert eine ganze Zahl i_0 mit $i_0 = 0$. Es existiert eine ganze Zahl i_1 mit $i_1 = 1$. Also ist $0 = i_1 = 1$.

Hier ist durch die erneute Einführung des Referenten i in der zweiten Aussage die Beziehung von i auf i_0 also aufgehoben und i ab diesem Punkt – bis auf weiteres – als i_1 interpretiert. Damit wird im dritten Satz $0 = i_1$ schlicht als eine unbegründete Aussage erkannt (verfügbar ist ja nur, dass $i_0 = 0$) und als fehlerhaft gemeldet.

Diese Vorgehensweise ist sicherlich elegant und kommt den oben vorgeschlagenen "pragmatischen Regeln" deutlich näher als die Verarbeitungsweise von Diproche. Im Hinblick auf unsere didaktische Zielsetzung ist sie aber nur bedingt geeignet: Um das Verhalten des Systems richtig zu interpretieren, muss ein Studienanfänger zunächst dafür sensibilisiert werden, dass Naproche intern mit einer Tiefenstruktur arbeitet, die zwischen gleich erscheinenden Referenten unterscheidet; dadurch entsteht weiter zusätzlich zur logischen Zugänglichkeitsrelation (s.o.) noch die Möglichkeit, dass eine Behauptung zwar noch verfügbar ist, nicht aber die durch sie eingeführten Referenzen. Zudem erschwert es der in Naproche gewählt Weg der Verarbeitung gerade dadurch, dass er in dieser Hinsicht "entgegenkommend" ist, für den zum Verständnis der mathematischen Sprache relevanten Unterschied zwischen Existenzaussagen und Deklarationen zu sensibilisieren: Was für den/die fortgeschrittenen BenutzerIn ein Vorteil ist, ist im Anfängerbereich potenziell ein didaktischer Nachteil. Die Vorgehensweise von Diproche ist also kein Zurückfallen hinter den Entwicklungsstand von Naproche, sondern wurde auf der Basis dieser Erwägungen didaktisch motiviert gewählt.

7.2.4 Definitionen

In mathematischen Texten, unter den von Diproche derzeit behandelten Themen besonders in der axiomatischen Geometrie, werden syntaktisch komplexe Terme häufig durch eine einzige Variable abgekürzt. In Diproche kann das einerseits durch das explizite Definitionszeichen := geschehen, wenn ein Objekt durch einen formalen Term gegeben ist. Andererseits kann eine Definition auch natürlichsprachlich formuliert werden. Die folgenden Formulierungen für Definitionen werden von Diproche akzeptiert:

- Es sei $s := s(a, b)$.
- Definiere m als Mittelpunkt von $s(a, b)$.

Auch hier gilt, wie für Deklarationen mit inhaltlicher Annahme: Definitionen präsupponieren eine Existenzbehauptung, die an der jeweiligen Stelle im Beweis beweisbar sein muss.

7.2.5 Axiome

Als "Axiome" werden im Sinne von Diproche Aussagen bezeichnet, die als wahr angenommen werden (also nicht verifiziert werden müssen), aber im Unterschied zu explizite Annahmen vom System nicht als logische Voraussetzungen der aus ihnen gezogenen Folgerungen gelten und insofern "ohne weiteres verwendet" werden können. Das ist z.B. dann nützlich, wenn – etwa bei einer bereits fortgeschrittenen Lehrveranstaltung – bei einer Aufgabe eine größere Anzahl von Vorannahmen über die jeweilige Domäne verfügbar sein sollen als der ATP mit vertretbarem Zeitaufwand verarbeiten kann.

Axiome werden über eine der folgenden Formulierungen eingeführt (diese Liste kann sich mit der weiteren Entwicklung des Systems ändern):

- Laut Vorlesung gilt...
- Bekanntlich gilt...
- Axiom:
- Wir wissen, dass...

Hinsichtlich der Zugänglichkeitsrelation funktionieren Axiome wie Annahmen, die nicht geschlossen werden können. Sie stehen also in allen auf ihre Einführung folgenden Schritten zur Verfügung.

Im Rahmen des aktuellen Einsatzes des Systems ist die Verwendung von Axiomen nicht erforderlich, birgt aber andererseits das Risiko, dass die Möglichkeit, sich beliebige Voraussetzungen zu verschaffen, dazu benutzt wird, unvollständige Beweise zu führen, die vom System dennoch als korrekt angesehen werden. Aus diesem Grund stehen Axiome in der aktuell eingesetzten Version von Diproche nicht zur Verfügung.

7.2.6 Behauptungen

Behauptungen sind Aussagen, die an der Stelle, an der sie stehen, aus dem bisher Erarbeiteten mithilfe eines elementaren Schrittes folgerbar sein sollten. Aussagen werden u.a. über folgende Formulierungen eingeführt:

- Damit erhalten wir...

- Es gilt...
- Also haben wir...
- Folglich gilt...
- Damit folgt...
- Dann...
- Widerspruch. (Damit wird behauptet, dass an dieser Stelle ein Widerspruch abgeleitet werden kann.)
- "Falls A, so gilt B" "Wenn A, dann B"

Neben den erwähnten funktionieren noch weitere Varianten wie "Nun gilt", "Jetzt folgt" etc.; es ist angestrebt, die Standardformulierungen zur Einführung von Behauptungen in Beweisen möglichst erschöpfend abzudecken.

Zur Herleitung einer Behauptung stehen alle Annahmen zur Verfügung, die bisher noch nicht geschlossen wurden, sowie alle bisher angeführten Axiome (s.o.), alle Behauptungen, die bisher aus den noch offenen Annahmen gefolgert wurden, sowie alle Folgerungsbeziehungen zwischen bisher getroffenen Annahmen (ob inzwischen geschlossen oder nicht) und den daraus abgeleiteten Folgerungen.

7.2.7 Multiple Behauptungen

Die unter StudienanfängerInnen verbreitete Sitte, nahezu beliebige Ausdrücke – Terme mit Termen, Aussagen mit Aussagen, aber bisweilen auch Terme mit Aussagen – durch Implikationspfeile zu verbinden, wird von Diproche ausdrücklich nicht unterstützt. Angesichts der sehr variablen und nicht selten unklaren Verwendung des Pfeiles (wenn etwa am einen Ende eine Zahl und am anderen eine Aussage steht) darf bezweifelt werden, dass der auf diese Weise angezeigte Zusammenhang im Allgemeinen klar ist; explizites Nachfragen zeigt des Öfteren, dass hierdurch wenig mehr als eine eher locker motivierte Assoziation oder eine rein zeitliche Sukzession – "ich habe das nach dem geschrieben" – gemeint ist.[7] Es sollte nach Möglichkeit vermieden werden, dass der zwar beschränkt, aber

[7] In vielen Fällen läßt sich die Absicht hinter solchen merkwürdigen Verwendungen von Pfeilen allerdings durchaus rekonstruieren: Ein Pfeil von einer Aussage zu einer Zahl etwa kann als "und folglich ist das Ergebnis 1" gelesen werden, eine Sequenz wie "$(x_i : i \in \mathbb{N})$ ist eine beschränkte Folge reeller Zahlen"$\rightarrow$ Bolzano-Weierstraß $\rightarrow$"konvergente Teilfolge" als "Hier liegt es nahe, den Satz von Bolzano-Weierstraß zu verwenden und damit folgt, dass eine konvergente Teilfolge existiert". Die "gute Absicht" hinter solchen Formulierungen erkennen und entsprechende Rückmeldungen geben zu können ist eine für menschliche KorrektorInnen sehr nützliche Fähigkeit. Von der technischen Herausforderung abgesehen, die der Versuch

doch immerhin als solcher im Hintergrund arbeitende Beweiser solche Ausdrücke "überinterpretiert" und korrekte logische Schritte "sieht", wo im Grunde nur eine Assoziation aufgeschrieben wurde.

Dennoch ist die "Ein Satz pro Schritt"-Form zum Teil unnötig streng. Es gehört durchaus zur korrekten mathematischen Darstellungspraxis, "Kettenschlüsse" verbal zu formulieren und dadurch in sehr komprimierter Form einen Teilbeweis zu führen, wie etwa im folgenden Satz:

"Dann gilt a, also gilt $(a \vee a)$, und damit folgt $\neg(\neg a \wedge \neg a)$."

Solche multiplen Behauptungen werden daher von Diproche akzeptiert. Sie bestehen formal aus mehreren Behauptungen, die statt als einzelne Sätze formuliert zu sein als Nebensätze aneinandergereiht sind und werden in der Beweisverifikation behandelt wie mehrere einzelne Sätze. Als Konsequenz eines solchen Satzes wird allerdings nur die jeweils letzte Aussage (in diesem Fall also $\neg(\neg a \wedge \neg a)$) behalten, während die übrigen als bloße Zwischenschritte betrachtet werden, die im weiteren Beweis nicht mehr zur Verfügung stehen.

7.2.8 Multiple bedingte Behauptungen

Als weitere Variante der "multiplen Behauptungen" sind ferner multiple Folgerungen aus einer Voraussetzung zulässig. Ein Beispiel für einen solchen Satz ist etwa

"Falls a und $(a \to b)$, so folgt b, also gilt $(a \wedge b)$."

Als Konsequenz eines solchen Kettensatzes wird nur die Implikation vom ersten zum letzten Glied behalten, hier also $(a \wedge (a \to b)) \to (a \wedge b)$. Auf diese Weise ist es insbesondere möglich, auf explizite Fallunterscheidungen zu verzichten, wo sämtliche Fälle knapp abgehandelt werden können. Diese Formulierungspraxis findet etwa in folgendem Beweistext Verwendung:

> Sei n eine ganze Zahl. Ist n gerade, so ist n^2 gerade, also ist $n^2 + n$ gerade. Ist n ungerade, so ist n^2 ungerade, also ist $n^2 + n$ gerade. Also ist $n^2 + n$ gerade.

Solche Formulierungen werden daher von Diproche akzeptiert.

bedeuten würde, solche "wohlwollenden" Lesarten automatisiert zu raten, stünde eine solche Vorgehensweise dem Ansatz von Diproche, Studierende auf ein präzises Formulieren hinzuleiten, jedoch offenkundig entgegen.

7.2.9 Begründete Behauptungen

Eine Schwierigkeit beim Lesen und auch beim automatischen Prüfen natürlichsprachlicher Beweise ist die, bei einer behaupteten Folgerung die relevanten Zwischenergebnisse zu ermitteln, aus denen die Behauptung folgen soll. Diese Schwierigkeit kann schon ab einer recht überschaubaren Anzahl von Beweiszeilen auftreten, wie das folgende (freilich eigens zu diesem Zweck konstruierte) Beispiel zeigt, in dem der/die LeserIn versuchen mag, zu sehen, aus welchen anderen Zeilen die Aussagen ab der dritten folgen:

1. Annahme: $(A \to (B \to C)) \to D$
2. Annahme: $(A \vee \neg D) \wedge (\neg A \vee \neg D) \wedge (\neg A \vee D \vee E)$
3. Folgerung: $\neg D$
4. Folgerung: $\neg(A \to (B \to C))$
5. Folgerung: A
6. Folgerung: $\neg(B \to C)$
7. Folgerung: B
8. Folgerung: $\neg C$
9. Folgerung: $\neg(\neg A \vee D)$
10. Folgerung: E

Diese Schwierigkeit menschlicher LeserInnen findet ihre Entsprechung bei Systemen zum automatischen Beweisprüfen in einer mit wachsender Textlänge rasch anwachsenden Laufzeit:[8] Muss jede von n verfügbaren Beweisregeln daraufhin geprüft werden, ob sie es erlaubt, die Behauptung ϕ aus den aktuell verfügbaren Annahmen und den bisher erreichten Folgerungen $\{\psi_1, ..., \psi_k\}$ abzuleiten, so muss – wenn wir davon ausgehen, dass Beweisregeln sich auf jeweils zwei Voraussetzungen beziehen (deren Reihenfolge relevant ist)[9] – für

[8] Eine Formulierung der im obigen Beispiel dargestellten Schlusskette in der Diproche-Sprache benötigt zu ihrer Verifikation über die Webversion von Diproche immerhin schon knapp 3 Sekunden.

[9] Diese Voraussetzung ist natürlich eine Vereinfachung. Einige Regeln, etwa die Ableitung von ϕ aus $\neg\neg\phi$ oder aus $\phi \wedge \psi$, haben nur eine Voraussetzung, einige der "höherstufigen", besonders der gebietsspezifischen, haben mehr ("Ist ABC ein echtes Dreieck, M der Mittelpunkt der Strecke AB und liegt bei C ein rechter Winkel, so sind die Strecken MA und MC kongruent" hat etwa drei Voraussetzungen). Da letztere deutlich häufiger sind als erstere wird die Komplexität mit der Festsetzung auf 2 Annahmen noch deutlich unterschätzt.

jedes geordnete Paar $(i, j) \in \{1, 2, ..., k\} \times \{1, 2, ..., k\}$ geprüft werden, ob die fragliche Regel es erlaubt, ϕ aus ψ_i und ψ_j abzuleiten, was also zu k^2 versuchten Anwendungen je Regel damit zu $n \cdot k^2$ Ableitungsversuchen pro Beweiszeile führt, die eine Behauptung enthält. Da das im allgemeinen die meisten Beweiszeilen sein werden, kommen wir für einen Text der Länge l auf wenigstens $\Sigma_{k=1}^{l} n \times k^2 = n \cdot \frac{l \cdot (l+1) \cdot (2l+1)}{6}$ Schritte, also ein kubisches Wachstum, wobei die Konstante n typischerweise im dreistelligen Bereich liegen wird (siehe dazu die detaillierte Besprechung der "Spielwiesen" zur Zahlentheorie und zur axiomatischen Geometrie in dieser Arbeit). Bei längeren Beweistexten können sich auf diese Weise erhebliche Verzögerungen in der Verarbeitung ergeben. Diese Schwierigkeit ist in der Entwicklung von Systemen wie Naproche früh bemerkt worden, wo ihr mit der Entwicklung von Heuristiken für die Prämissenauswahl begegnet wurde, siehe Kühlwein et al. [46] (und die Diskussion zur Verfügbarkeit von Annnahmen in dieser Arbeit). Diese Heuristiken haben sich im Hinblick auf die Reduktion von Verarbeitungszeiten als äußerst erfolgreich erwiesen. Ihr Einsatz in Diproche erscheint aber angesichts der didaktischen Zielsetzung als nicht ratsam: Zum einen beeinflussen sie in einer für den oder die BenutzerIn nicht nachvollziehbaren Weise den Erfolg der Textverifikation, wenn etwa die logisch, aber eben nicht heuristisch gleichgültige Wiederholung einer bereits bewiesenen Behauptung an einer Stelle im Beweis den Unterschied zwischen Erfolg und Misserfolg ausmacht; zum anderen ist es Teil der Zielsetzung von Diproche, BenutzerInnen zu einer möglichst strukturierten und übersichtlichen Darstellung hinzuführen und sie dazu zu bringen, sich möglichst viele Aspekte eines Beweises bewußt zu machen und sie zu explizieren.

Bis zu einem gewissen Grad kann dieser Schwierigkeit durch einen gut strukturierten und gegliederten Beweisaufbau begegnet werden, etwa durch die explizite Verwendung von Subbehauptungen, Fallunterscheidungen und Teilbeweisen. Diese Strukturen werden daher von Diproche auch unterstützt. Jedoch bietet sich nicht immer eine Zerlegung in natürliche Zwischenergebnisse an: Bisweilen ist es nötig, in späteren Teilen des Beweises auf zu Beginn erreichte Einsichten zurückzugreifen. In der Praxis wird das dadurch gelöst, dass das fragliche Zwischenergebnis entweder explizit wiederholt, oder, wo dies aufgrund der Länge des Ergebnisses oder der Anzahl seiner Wiederholungen unbequem wäre, benannt wird und über seine Benennung später wieder abgerufen werden kann; so könnte es in Zeile 10 des obigen Beweises statt "Wegen $(A \vee \neg D) \wedge (\neg A \vee \neg D) \wedge (\neg A \vee D \vee E)$ und $\neg(\neg A \vee D)$ folgt E" etwa heißen "Wegen (2) und (9) folgt E.".

Auch diese Art der Bezugnahme auf frühere Resultate ist in Diproche möglich. Als Namen für Beweiszeilen wird dabei die Zeichenkombination **l, gefolgt von einer natürlichen Zahl, verwendet, so dass etwa **l1 oder **l32 mögliche Zeilennamen wären. Um eine Beweiszeile, die eine Behauptung oder eine Annahme

enthält, zu benennen, stellt man ihr den gewünschten Namen einfach voran. Später kann im Rahmen einer begründeten Behauptung darauf zurück zu kommen, verwendet man eine der Formulierungen “Wegen **l...” oder “Da **l...”. Es ist dabei auch möglich, auf mehrere Zeilen zugleich zu verweisen und dabei benannte Ergebnisse mit explizit wiederholten zu kombinieren, wie im folgenden Beispiel:

- **l1 Es gelte a.
- **l2 Ferner gelte $(a \to b)$.
- Wegen a und **l2 folgt b.

Formal sind diese “begründeten Behauptungen” in Diproche ein Subtyp des Typs “Behauptung”

Damit eine begründete Behauptung als korrekt gilt, müssen alle als Begründung (oben vor dem “gilt”) angeführten Aussagen an dieser Stelle zur Verfügung stehen (im oben erklärten Sinn) und das Ziel in einem Schritt aus diesen folgerbar sein. Steht direkt vor einer begründeten Behauptung eine Annahme oder Behauptung, so steht auch diese noch zur Verfügung.

```
Es gilt a. Angenommen, es gilt b. Da auch a gilt, gilt nun (a&b).
```

Zur Verifikation der letzten Behauptung (a&b) stehen also das explizit erwähnte a zur Verfügung und ferner b, weil b direkt vor diesem Satz angenommen wird.

7.2.10 Annotationen

Annotationen dienen generell dazu, den Beweis zu gliedern und zu steuern, insbesondere zur Aktualisierung des Beweiszieles und der zur Verfügung stehenden Annahmen. Die in Diproche verfügbaren Annotationen orientieren sich an den Erfordernissen und Gepflogenheiten bei der Darstellung von Lösungen zu Übungsaufgaben im Anfängerbereich. Dazu gehören etwa, wie schon erwähnt, gängige Kürzel wie “$\subseteq$” oder “$\Leftarrow$”, nicht jedoch etwa höhere Textstrukturmarker wie “Theorem:”, “Definition”, “Lemma” etc., wie sie in für die Verarbeitung längerer mathematischer Texte vorgesehenen Systemen wie Naproche verfügbar sind (siehe etwa Cramer, [53], S. 182). In Diproche gibt es derzeit folgende Annotationen:

- “Beweis” ist ein Beweisanfangsmarker, der zeigt, dass nun ein neuer (Teil-)Beweis beginnt. Als Beweisanfangsmarker fungieren ebenso die Formulierungen “Induktionsanfang” und “Induktionsschritt”.

- "Wir zeigen, dass X/Zu zeigen: X/Wir zeigen X" deklariert X als aktuelles Beweisziel
- Absätze schließen alle Annahmen des vorhergenden Absatzes (d.h. des Textabschnittes nach dem letzten Absatz), sofern keine der oben erwähnten Sonderregeln für Annahmen greift, die direkt auf Beweisanfangsmarker folgen.
- "=>" und "<=" gelten als Beweisanfangsmarker; ist das aktuelle Beweisziel eine Biimplikation ($A \leftrightarrow B$), so haben sie zur Folge, dass die entsprechende Richtung der Biimplikation als neues aktuelles Beweisziel gilt.
- Analog fungieren im Kontext der Booleschen Mengenlehre "$\subseteq$" und "$\supseteq$" als Beweisanfangsmarker. Ist das Beweisziel eine Mengengleichheit, so haben sie den Effekt, dass die entsprechende Teilmengenrelation zum aktuellen Beweisziel wird.
- Falleinführungen: "Fall X": gelten als Beweisanfangsmarker bezüglich Annahmen. Im Rahmen von Fallunterscheidungen kann damit ein Fall eingeführt werden.
- Beweisendmarker: qed. Mit qed wird das letzte Beweisziel als erreicht angesehen und das zuvor aktuelle gilt als neues aktuelles Beweisziel

7.3 Formelsyntax

Neben natürlichsprachlichen Ausdrücken sind Formeln ein wesentlicher Bestandteil der mathematischen Sprache. Für formalsprachliche Ausdrücke stehen, im Gegensatz zu natürlichsprachlichen Formulierungen, formale (rekursive) Definitionen zur Verfügung. Allerdings sind diese für gewöhnlich unnötig streng und führen zu unübersichtlichen Ausdrücken, weswegen sie in der Praxis durch faktisch permissivere Syntaxen abgelöst werden; so ist es z.B. im Allgemeinen zulässig, in einem Term wie $(a + (b + c))$ die Klammern fortzulassen. Derartige Konventionen zu berücksichtigen ist für die Bereitstellung einer quasi-natürlichen Eingabesprache für mathematische Beweistexte unerlässlich. Eine detaillierte Erläuterung der verschiedenen Typen formaler Ausdrücke in Diproche mit ihrer Syntax, einschließlich der diversen Abweichungen von der "strikten" Formelsyntax, die von Diproche unterstützt werden, findet sich im Anhang. Hier beschränken wir uns auf eine exemplarische Erläuterung der auftretenden Ausdruckstypen:

- Aussagenlogische Terme wie $((A \wedge B) \rightarrow (C \vee D))$; äußere Klammern können hier fortgelassen werden; dagegen sind Prioritätsregeln zwischen

den Junktoren aus Gründen, die im Anhang A erläutert sind, derzeit nicht implementiert.

- Prädikatenlogische Formeln wie $\forall x : \exists y : (x \leq y \vee x > y)$.
- Boolesche Mengenterme wie $((-A) \cup B) \cap C$.
- Arithmetische Terme wie $(a + b) \cdot (b^2 - a^2)$.
- Aussagenlogische Folgerungsketten wie $(A \vee B) \Leftrightarrow \neg(\neg A \wedge \neg B) \Rightarrow \neg(\neg B \wedge \neg A)$.
- Boolesche "Ungleichungsketten" wie $(A \cap B) = (B \cap A) \subseteq A \subseteq (A \cup C)$.
- Arithmetische Ungleichungsketten wie $a^2 - 2 \cdot a \cdot b + b^2 = (a - b)^2 \geq 0 > (-1)$.
- Implikationsketten in der Booleschen Mengenlehre, wie $(A = B) \Leftrightarrow ((A \subseteq B) \wedge (B \subseteq A)) \Rightarrow (A \subseteq B) \Leftrightarrow (B = (A \cup B))$.
- Arithmetische Umformungsketten, optional mit expliziter Angabe von Umformungsschritten, wie $(a = b - 1) \Leftrightarrow (+1)(a + 1 = b) \Leftrightarrow (\cdot 2)(2 \cdot a + 2 = 2 \cdot b)$.

7.3.1 Sonstiges

Eine Anzahl von "Füllwörtern", die zur Steuerung des Leseflusses und dem eigenen Verständnis des Beweises dienen mögen, aber für die Arbeitsweise des Systems (derzeit) unerheblich sind, können frei verwendet werden und werden vom System ignoriert; derzeit sind das u.a.: nun, jetzt, ferner, ausserdem, weiter, aber, auch, insbesondere, andererseits, sicherlich, jedenfalls, schon, ebenfalls, endlich, schliesslich.

7.4 Die Logik von Diproche

Wesentlich zum Verständnis eines Beweistext ist die Erfassung seiner logischen Struktur. Dazu gehört insbesondere die Klarheit darüber, welcher logische Akt an jeder Stelle des Beweises vollzogen wird und worauf dabei jeweils Bezug genommen wird. In mathematischen Beweisen unterscheiden wir, wie bereits oben erwähnt, drei Typen von Akten, nämlich das Annehmen, das Behaupten und das Annotieren.[10] Dabei gilt eine Behauptung dann als korrekter Beweisschritt, wenn

[10] In natürlichen Beweistexten gibt es sicher weitere Akttypen; so etwa Bemerkungen zur Veranschaulichung oder heuristische bzw. strategische Hinweise darauf, warum ein gewisser

sie aus den Annahmen ableitbar ist, die an der Stelle zur Verfügung stehen, an der die Behauptung gemacht wird. Da in einem Beweis oft unter einer Vielzahl verschiedener Mengen von Annahmen gearbeitet werden muss, liegt mit einem Beweis immer auch eine logische Textstruktur vor, die die Zugänglichkeit von Annahmen an den einzelnen Textstellen beschreibt. Wir betrachten hierzu ein Beispiel:

1. Es sei n eine natürliche Zahl.
2. Dann ist n gerade oder n ist ungerade.
3. Angenommen, n ist gerade.
4. Dann ist auch n^2 gerade.
5. Also ist $n^2 - n$ gerade.
6. Angenommen nun, n ist ungerade.
7. Dann ist auch n^2 ungerade.
8. Also ist $n^2 - n$ gerade.
9. Also gilt: Für jede natürliche Zahl n ist $n^2 - n$ gerade.

Die intendierte Lesart dieses Textes ist offenbar folgende: Die Deklaration (Annahme) (1) steht in allen folgenden Zeilen (2)-(9) zur Verfügung, ebenso die Behauptung (2) in den Zeilen (3)-(9). Die Annahme (3) steht dagegen nur in den Zeilen (4) und (5) zur Verfügung. Die Behauptung (4) steht nur für die Ableitung von Behauptung (5) zur Verfügung. Implizit ist an dieser Stelle nun gezeigt: Ist n eine gerade (3) natürliche (1) Zahl, so ist $n^2 - n$ gerade (*). Entsprechend steht die Annahme (6) genau für die Behauptungen (7) und (8) zur Verfügung und die Behauptung (7) noch für die Behauptung (8). Nach (8) ist – wiederum implizit – gezeigt: Ist n eine ungerade (6) natürliche (1) Zahl, so ist $n^2 - n$ gerade (**). In (9) stehen nun neben (1) und (2) nur noch die beiden impliziten Folgerungen (*) und (**) zur Verfügung, woraus sich nun (9) ergibt.

Die Rekonstruktion dieser "Verfügbarkeiten" bzw. "Zugänglichkeiten" ist offenbar für das korrekte Verständnis – und damit die Prüfung – des Beweistextes wesentlich: Wer nicht versteht, dass (2) im gesamten Text zur Verfügung steht, dem oder der fehlt in (9) die nötige Voraussetzung für eine Fallunterscheidung.

Schritt naheliegt, Erläuterungen von Beweisplänen etc. Da diese aber zum einen für die logische Korrektheit eines Beweises – jedenfalls im Sinne automatischer Verifizierbarkeit – unerheblich sind und zum anderen in Bearbeitungen von Übungsaufgaben keine große Rolle spielen, werden sie im Rahmen von Diproche übergangen.

Wer andererseits nicht bemerkt, dass die Annahme (3) in (6) wieder aufgehoben ist, arbeitet ab da unter den widersprüchlichen Voraussetzungen "n ist gerade" und "n ist ungerade", womit zwar alle weiteren Schritte nach "ex falso quodlibet" folgen, zum Schluss aber nur gezeigt ist, dass "Für jede natürliche Zahl n ist $n^2 - n$ gerade" aus einem Widerspruch ableitbar ist und nicht, dass es wahr ist.

Die Fähigkeit, einem mathematisch argumentierende Text den intendierten Geltungsbereich von Annahmen, Deklarationen etc. zu entnehmen, ist also Voraussetzung für dessen Beurteilung schon im Hinblick auf die logische Korrektheit. Entsprechend liegt hierin eine Herausforderung für automatische Systeme, die eine solche Beurteilung vornehmen sollen. Jedes der oben besprochenen Systeme zum automatischen Verstehen natürlichsprachlicher mathematischer Texte (wie Naproche oder SAD) hat sich dieser Aufgabe denn auch zwangsläufig auf die eine oder andere Weise angenommen: So hat etwa in Naproche das Wort "Thus" am Beginn eines Satzes den Effekt, das Ende des Geltungsbereiches für die zuletzt gemachte Annahme zu markieren. Wie wenig trivial diese Herausforderung ist, zeigt sich u.a. daran, dass hierfür nicht allein (und z.T., wie etwa im Beispiel oben, nicht einmal primär) explizite Textmarker herangezogen werden, sondern in erheblichem Maß auch pragmatische Überlegungen angestellt werden: So ist es für den/die mathematische kompetente/n LeserIn offensichtlich, dass im Textbeispiel oben der Geltungsbereich der Annahme in Zeile (3) nur die Schritte (4) und (5) umfasst, aber nicht mehr Schritt (7), u.a. deshalb, weil bei der Lektüre implizit von einem strategisch sinnvollen Vorgehen der dargestellten Argumentation ausgegangen wird und es in dem vorliegenden Kontext unsinnig wäre, unter explizit widersprüchlichen Annahmen (hier: n ist gerade und n ist ungerade) zu arbeiten. Auf diese Weise kann im Rahmen einer kompetenten Lektüre eine Lesart, bei der etwa (3) bis zum Rest des Beweises in Geltung bleibt, ausgeschlossen werden. Werden Annahmen im Geltungsbereich von Annahmen gemacht, so ergeben sich auf diese Weise rasch sehr viele "im Prinzip" mögliche Lesarten bezüglich der Geltungsbereiche der Annahmen; aus diesen "die" oder zumindest "eine" richtige auszuwählen, erfordert auf Seiten der/des Lesers/In Kompetenzen, die über ein bloßes "Satz für Satz"-Verstehen hinausgehen, so z.B., den groben Beweisplan aus initialen Andeutungen zu "erraten" (im Beispiel oben: Beweis der Behauptung durch Fallunterscheidung nach der Parität von n) und die Geltungsbereiche der Annahmen diesem Plan gemäß zu bestimmen.[11] Dadurch wird das Nachvollziehen eines Beweises neben dessen Auffindung selbst zu einer Aufgabe, die Heuristiken erfordert (etwa: "Wenn am Ende die Behauptung folgen soll, müssen zuvor alle Annahmen geschlossen sein" oder "Es ist nicht

[11] Hierin mag ein Grund liegen, warum ein Beweis sich bisweilen durch mehrmalige Lektüre besser erschließt als durch eine noch so gründliche einmalige.

sinnvoll, unter explizit widersprüchlichen Annahmen zu arbeiten"). Einige Ansätze, diese Art von Kompetenz zu automatisieren und für das automatische Verstehen mathematische Texte zu nutzen sind z.B. von Zinn [216] durch die Verwendung von "Proof Plans" gemacht worden; im Rahmen des Diproche-Projektes ist dergleichen aber weder praktikabel noch auch nur sinnvoll: Die Verwendung von aus pragmatischen und strategischen Erwägungen erschlossenen Informationen unterläuft das Ziel, eine klare und für den/die BenutzerIn durchschaubare Textsemantik zur Verfügung zu stellen. Damit wird es umgekehrt schwieriger, aus Rückmeldungen des Systems auf logische Fehler im Beweis zu schließen: Liegt "tatsächlich" ein Fehler vor oder wurde der Beweis lediglich vom System nicht "verstanden", weil die gegebenen Marker nicht ausreichten, um den Beweisplan zu identifizieren oder der Beweisplan dem System nicht bekannt war? Hier geraten wir also erneut in das Spannungsfeld zwischen der Simulation menschlicher Lesekompetenz und den didaktischen Anforderungen an eine Korrektur, die für Studierende hilfreich ist, die erste Beweiskompetenzen erst erwerben. Ferner ist es gerade im Anfängerbereich sinnvoll, logische Strukturmerkmale eines Textes wie die Geltungsbereiche von Annahmen klar zu explizieren und entsprechend in Diproche eine Darstellungsform zu verwenden, die eine solche Explikation erfordert. Zugleich ist der Forderung nach größtmöglicher Natürlichkeit der Darstellung Rechnung zu tragen. Da explizite Endmarker für die Geltungsbereiche von Annahmen in mathematischen Texten so gut wie nie vorkommen, ist deren Einführung entsprechend zu vermeiden. In Diproche wird versucht, diese Geltungsbereiche durch die Verwendung von Absätzen als Anzeigen zusammenhängender Sinnabschnitte auf zugleich natürliche als auch explizite Weise zu bestimmen: Das Ende eines Absatzes markiert zugleich das Ende des Geltungsbereiches aller Annahmen, die in diesem Absatz gemacht wurden. Da ein Beweis es erfordern kann, mehrere Teilargumente unter einer globalen Annahme durchzuführen (etwa bei einer Fallunterscheidung innerhalb des Geltungsbereiches einer Annahme) sind Annahmen, die im Absatz nach einem Beweisanfangsmarker (z.B. "Beweis:") kommen, davon ausgenommen: Diese bleiben bis in Geltung, bis der (Teil-)Beweis, der durch den Beweisanfangsmarker begonnen wird, beendet ist.

Zusammenfassend gelten für die Geltungsbereiche von Annahmen in Diproche also folgende Regeln:

- Eine Annahme, die in einem Absatz steht, der durch einen Beweisanfangsmarker begonnen wird, gilt bis zur Beendigung des entsprechenden Teilbeweises durch einen Beweisendmarker (wie "qed").
- Alle anderen Annahmen gelten bis zum Ende des Absatzes, in dem sie gemacht wurden.

Der obigen Beweis sähe damit in Diproche so aus, wobei die Ergänzungen durch

Fettdruck markiert sind:

1. **Beweis:**
2. Es sei n eine natürliche Zahl.
3. Dann ist n gerade oder n ist ungerade.
4. **(Absatz)**
5. Angenommen, n ist gerade.
6. Dann ist auch n^2 gerade.
7. Also ist $n^2 - n$ gerade.
8. **(Absatz)**
9. Angenommen nun, n ist ungerade.
10. Dann ist auch n^2 ungerade.
11. Also ist $n^2 - n$ gerade.
12. **(Absatz)**
13. Also gilt: Für jede natürliche Zahl n ist $n^2 - n$ gerade.
14. **QED**

7.4.1 Implizite Geltungsbereiche von Deklarationen mit inhaltlicher Annahme

Außer für Annahmen gelten Bemerkungen vom Beginn des vorherigen Abschnittes auch für Deklarationen ("Es sei x eine ganze Zahl") und Deklarationen mit inhaltlichen Annahmen ("Es sei k eine ganze Zahl mit $x = 2 * k$"), einschließlich lokaler Definitionen von Notationen ("Definiere s als Mittelsenkrechte von $\overline{AB}$"). Für "reine" Deklarationen – also solche, die lediglich einer Variablen einen Typ zuweisen, aber keine weitere inhaltliche Annahmen einführen – sind die oben angegebenen Regeln auch ohne weiteres anwendbar. Bei Deklarationen mit inhaltlichen Annahmen sowie lokalen Definitionen ergibt sich jedoch eine zusätzliche Schwierigkeit. Zur Verdeutlichung betrachten wir folgenden Beispieltext:

1. Es sei x eine ganze Zahl.

2. Angenommen, x ist gerade.

3. Es sei k eine ganze Zahl mit $x = 2 * k$.

4. Dann gilt $x^2 = (2 * k)^2 = 4 * k^2$.

5. Damit ist 4 ein Teiler von x^2.

Hier sind die Deklaration (1) sowie die Annahme (2) offenbar in allen folgenden Zeilen verfügbar. Ebenso steht die Deklaration von k als ganzer Zahl in (3) in allen folgenden Zeilen zur Verfügung. Nun liefert (3) aber noch die Zusatzinformation über k, dass $x = 2 * k$. Auch diese Information steht in den folgenden Zeilen (4) und (5) zur Verfügung. An dieser Stelle entsteht nun mit Hinblick auf die logischen Abhängigkeiten ein Problem: Die Information, dass $x = 2 * k$, ist in Zeile (5) zwar verfügbar; da (3) aber lediglich die Wahl der Variablen k betrifft und k in (5) nicht vorkommt, ist (5) von (3) logisch unabhängig: (5) folgt allein aus (2), wohingegen (4), in dem die Variable k auftritt, inhaltlich auf (3) zurückgreift. Würden die Regeln aus dem vorherigen Abschnitt verwendet, um zu ermitteln, auf Basis welcher Annahmen eine Folgerung gezogen wurde, so würde (5) als aus (2) und (3) gefolgert erkannt, womit das Argument unzureichend wäre, um zu zeigen, dass die Quadrate gerader Zahlen stets durch 4 teilbar sind. Ohne (3) wiederum ist der Beweis nicht zu führen, weil hier die Definition von "gerade" eingeht. Damit würden zahlreiche elementare Argumentationen, die auf der Einführung von Hilfsvariablen beruhen, in Diproche unmöglich.

Dasselbe Phänomen tritt auf, wenn etwa in der Geometrie Hilflinien oder andere Hilfsobjekte eingeführt werden, die aber in der abschließenden Folgerung nicht mehr vorkommen: Auch hier darf eine Folgerung, die diese Hilfsobjekte nicht betrifft, offenbar nicht als logisch abhängig von den definierenden Eigenschaften dieser Objekte angesehen werden.

Dabei ist zu beachten, dass eine solche Abhängigkeit einer Aussage von einer inhaltlichen Deklaration oder Definition nicht allein dann vorliegt, wenn die Aussage die dort deklarierten Variablen explizit betrifft; ebenso kann die Abhängigkeit indirekt über eine Kette von Deklarationen/Definitionen entstehen; in "Es seien a, b, c Punkte. Es gelte $a \neq b$. Definiere m als Mittelsenkrechte von $s(a, b)$. Definiere p als Parallele zu m durch c. Dann ist p orthogonal zu $l(a, b)$." etwa ist die letzte Behauptung weiterhin von der Definition von m als der Mittelsenkrechten von $s(a, b)$ logisch abhängig, obwohl m in dieser Behauptung nicht mehr auftritt, wohl aber eben das in Abhängigkeit von m definierte p.

Die Zugänglichkeitsregeln aus dem vorherigen Abschnitt sind also zu ergänzen um Regeln für die logische Abhängigkeit von Annahmen, die die definierenden Eigenschaften von Objekten betreffen, welche durch Deklarationen mit inhaltlicher

Annahme, Wahlen oder lokale Definitionen eingeführt werden. Wir ergänzen daher die oben angeführten Regeln um solche der "logischen Abhängigkeit".

Zu einer Formel ϕ sei FV_ϕ die Menge der in ϕ frei vorkommenden Variablen; für einen Diproche-Satz Z, der eine Deklaration mit inhaltlicher Annahme oder eine Definition enthält sei ferner $D(Z)$ die Menge der durch Z deklarierten Variablen, $\phi(Z)$ die inhaltliche Annahme bzw. Definition in Z und $\mathrm{FV}_Z :=\mathrm{FV}_{\phi(Z)}$ die Menge der in dieser Annahme bzw. Definition frei vorkommenden Variablen; enthält Z lediglich eine Annahme oder eine Behauptung, so ist $\phi(Z)$ diese Annahme bzw. Behauptung.

Für zwei Diproche-Sätze Z, Z' aus einem Diproche-Text T (also einer endlichen Folge von Diproche-Sätzen) nennen wir Z' 'in T unmittelbar logisch abhängig' von Z, falls Z in T von Z' aus zugänglich ist (im Sinne der im letzten Abschnitt definierten Regeln), Z eine Deklaration mit inhaltlicher Annahme oder eine Definition ist und $\mathrm{D}(Z)\cap\mathrm{FV}_{Z'} \neq \emptyset$ gilt, also mindestens eine der in Z deklarierten Variablen in Z' vorkommt.

Für zwei Diproche-Sätze Z, Z' in einem Diproche-Text T nennen wir Z' 'in T logisch abhängig' von Z, falls Z eine Deklaration mit inhaltlicher Annahme oder eine Definition ist und eine endliche Folge $(Z_1, Z_2, ..., Z_n)$ von Zeilen von T so existiert, dass $Z_1 = Z$, $Z_n = Z'$ und für jedes $i \in \{1, 2, ..., (n-1)\}$ gilt, dass Z_{i+1} von Z_i in T unmittelbar logisch abhängig ist.

Ist Z' in T nicht von Z logisch abhängig, so nennen wir Z' 'in T logisch unabhängig' von Z.

Liegt nun ein Diproche-Text T vor, bei dem in einer gewissen Zeile Z' eine Behauptung ϕ steht, so wird diese Zeile von Diproche wie folgt behandelt:

- Sind $Z_1, ..., Z_n$ die von Z' aus zugänglichen Zeilen und $\phi_1, ..., \phi_n$ die in diesen Zeilen gemachten inhaltlichen Annahmen bzw. Definitionen, so wird versucht, $\phi(Z')$ aus $\{\phi_1, ..., \phi_n\}$ logisch herzuleiten. Wenn das gelingt, so gilt die Zeile als erfolgreich verifiziert. Andernfalls gilt die Zeile als logisch nicht verifiziert und es wird eine entsprechende Rückmeldung erzeugt.

- Als die mit der Zeile Z' erreichte Aussage steht für alle folgenden Schritte ferner $\phi_1 \to (\phi_2 \to (\phi_3 \to ...(\phi_n \to \phi)))$ zur Verfügung, wobei $\phi_1, ..., \phi_n$ nun diejenigen Zeilen von T sind, von denen Z' logisch abhängig ist.

Auf diese Weise stehen für die Ableitung von (5) im oben angegebenen Textbeispiel also die Inhalte der Zeilen (1), (2) und (3) zur Verfügung, als erreichte Implikation wird aber lediglich die von (2) nach (5) gespeichert.[12]

[12]Die Deklaration (1) wird in der aktuellen Behandlung durch Diproche ebenfalls fortgelassen. Hierdurch können Fehler entstehen, wenn dieselbe Variable an verschiedenen Stellen im Beweis als zu unterschiedlichen Typen gehörig deklariert wird. Zwar ist dieses Problem technisch durchaus

7.5 Beispieltexte

Um einen Eindruck von den sprachlichen wie auch logischen Möglichkeiten von Diproche zu verschaffen, geben wir eine Reihe von Lösungen zu Übungsaufgaben an, die von Diproche akzeptiert werden. Die Aufgaben stammen überwiegend von den Übungsblättern zur Vorlesung "Algebra 1" von Hinrich Lorenzen an der Universität Flensburg (diverse Jahrgänge).

7.5.1 Beispieltexte zur Aussagenlogik

Ein sehr einfacher Äquivalenzbveweis.

Es seien A, B Aussagen. Zeige: Dann gilt $A\&B$ gdw $B\&A$ gilt.
Beweis:

$\Rightarrow$ Es gelte $A\&B$. Dann gilt A. Ferner gilt B. Also folgt $B\&A$. qed.

$\Leftarrow$ Nun gelte $B\&A$. Dann haben wir B. Ausserdem ergibt sich A. Folglich erhalten wir $A\&B$. qed.

Damit ist $A\&B$ aequivalent zu $B\&A$. qed.

Eine einführende Aufgabe zur Interaktion von Junktoren.

Es seien A, B, C Aussagen. Es gelte $(A \vee B) \to C$. Zeige: Dann gilt auch $A \to C$ und $B \to C$.

Beweis:

Es gelte A. Dann folgt $(A \vee B)$. Wegen $(A \vee B) \to C$ folgt C.
Also gilt $A \to C$.

Nun gelte B. Dann folgt wiederum $(A \vee B)$. Also gilt C.
Damit haben wir $B \to C$.

Folglich gilt $A \to C$ und $B \to C$. qed.

Eine doppelte Anwendung der Kontraposition unter Verwendung begründeter Behauptungen.

---

zu beheben; die derzeitigen Anwendungen indes werden allenfalls in seltenen Fällen zu so etwas führen; entsprechend genügt unseres Erachtens vorerst der Hinweise an den/die BenutzerIn, dergleichen als stilistische Unsauberkeit zu unterlassen.

Es seien A, B, C Aussagen. Es gelte $A \to B$. Ferner gelte $B \to C$. Ausserdem gelte $\neg C$. Zeige: Dann gilt $\neg A$.

Beweis: Wegen $\neg C$ und $B \to C$ folgt $\neg B$. Wegen $\neg B$ und $A \to B$ folgt $\neg A$. qed.

Ein "Kettenschluss", der sich gut dafür eignet, die Vorteile argumentativer Lösungen gegenüber Wahrheitstafeln zu demonstrieren. Hier wurde mehrfach von multiplen Behauptungen Gebrauch gemacht.

Es seien A, B, C, D, E, F Aussagen. Zeige: Dann gilt $((A\&((A \to B)\&(B \to C)\&(C \to D)\&(D \to E)\&(E \to F))) \to F)$.

Beweis: Es gelte $(A\&((A \to B)\&((B \to C)\&((C \to D)\&((D \to E)\&(E \to F))))))$. Dann gilt A. Ferner gilt $A \to B$, also gilt B. Ausserdem gilt $B \to C$, damit ergibt sich C. Weiter haben wir $C \to D$, also folgt D. Wir haben $D \to E$, also auch E. Schliesslich gilt auch $E \to F$. Damit folgt nun endlich F.
qed.

7.5.2 Boolesche Mengenlehre

Eine Teilmengenbeziehung:[13]

Es seien A, B, C Mengen. Wir zeigen $A \cap (B \cup C) \subseteq ((A \cap B) \cup (A \cap C))$.

Beweis: Es gelte $x \in A \cap (B \cup C)$. Dann folgt $x \in A$ und $x \in (B \cup C)$. Damit folgt $x \in B$ oder $x \in C$.

Fall 1: Es sei $x \in B$. Dann folgt $x \in (A \cap B)$. Also gilt auch $x \in ((A \cap B) \cup (A \cap C))$. qed.

Fall 2: Es sei $x \in C$. Dann folgt $x \in (A \cap C)$. Damit haben wir wiederum $x \in ((A \cap B) \cup (A \cap C))$. qed.

Insgesamt erhalten wir nun $x \in ((A \cap B) \cup (A \cap C))$. qed.

Eine Mengengleichheit, gezeigt durch zwei separate Teilmengenbeziehungen:

[13] Ein Teil von Blatt 9, Aufgabe 3.3(b)

Es seien A, B Mengen. Wir zeigen $(A \cap (A \cup B)) = A$.

Beweis:

$\subseteq$: Es sei $x \in (A \cap (A \cup B))$. Dann folgt $x \in A$. qed.

$\supseteq$: Nun gelte $x \in A$. Dann ist auch $x \in (A \cup B)$. Folglich erhalten wir $x \in (A \cap (A \cup B))$. qed.
Damit folgt nun $(A \cap (A \cup B)) = A$. qed.

Eine Mengengleichheit, gezeigt durch eine Umformungskette (Blatt 9, Aufgabe $4(a)$):

Es seien A, B Mengen. Wir zeigen: Dann gilt $(A \cap (B/A)) = \emptyset$.

Beweis: Wir haben $(A \cap (B/A)) = (A \cap (B \cap (-A))) = (A \cap ((-A) \cap B)) = ((A \cap (-A)) \cap B) = (\emptyset \cap B) = \emptyset$. Also ist $(A \cap (B/A)) = \emptyset$. qed.

Ein Beispiel zur Verwendung kartesischer Produkte (Blatt 9, Aufgabe $5(b)$):

Es seien A, B, C, D Mengen. Zeige: Dann gilt $((A \times B) \cap (C \times D)) = ((A \cap C) \times (B \cap D))$.

Beweis:

$\subset$ Es sei $(x, y) \in ((A \times B) \cap (C \times D))$. Also ist $(x, y) \in (A \times B)$ und $(x, y) \in (C \times D)$. Also ist $x \in A$ und $y \in B$. Ferner ist $x \in C$ und $y \in D$. Also ist $x \in (A \cap C)$. Weiter ist $y \in (B \cap D)$. Also ist $(x, y) \in ((A \cap C) \times (B \cap D))$. qed.

$\subset$ Es sei $(x, y) \in ((A \cap C) \times (B \cap D))$. Also ist $x \in (A \cap C)$ und $y \in (B \cap D)$. Damit ist $x \in A$ und $x \in C$. Weiter ist $y \in B$ und $y \in D$. Damit ist $(x, y) \in (A \times B)$. Ausserdem ist $(x, y) \in (C \times D)$. Also ist $(x, y) \in ((A \times B) \cap (C \times D))$. qed.

Also ist $((A \times B) \cap (C \times D)) = ((A \cap C) \times (B \cap D))$. qed.

7.5.3 Gerade und ungerade Zahlen

Ein direkter Beweis (Blatt 8, Aufgabe $1(b)$):

Es sei x eine ganze Zahl. Zeige: Wenn x ungerade ist, dann ist $(x-3)*(x+3)$ gerade.

Beweis: Angenommen, x ist ungerade. Es sei k eine ganze Zahl mit $x = 2 * k - 1$. Dann ist $(x - 3) * (x + 3) = ((2 * k - 1) - 3) * (x + 3) = (2 * k - 4) * (x + 3) = 2 * ((k - 2) * (x + 3))$. Also ist $(x - 3) * (x + 3)$ gerade. qed.

Ein einfacher Kontrapositionsbeweis (Blatt 10, Aufgabe $1(d)$):

Es sei x eine ganze Zahl. Wir zeigen: Wenn $2 - x$ ungerade ist, dann ist x ungerade.

Beweis: Angenommen, x ist nicht ungerade. Dann ist x gerade. Es sei k eine ganze Zahl mit $x = 2 * k$. Dann ist $2 - x = 2 - (2 * k) = 2 * (1 - k)$. Also ist $2 - x$ gerade. Also ist $2 - x$ nicht ungerade. qed.

Ein Beweis mit Fallunterscheidungen (Blatt 9, Aufgabe $1(b)$):

Es sei n eine ganze Zahl. Zeige: Dann ist $4 * n + 5$ ungerade.

Beweis:

Fall 1: Es sei n gerade. Es sei k eine ganze Zahl mit $n = 2 * k$. Dann ist $4 * n + 5 = 4 * (2 * k) + 5 = 2 * (4 * k + 2) + 1$. Also ist $4 * n + 5$ ungerade. qed.

Fall 2: Es sei n ungerade. Es sei k eine ganze Zahl mit $n = 2 * k - 1$. Dann ist $4 * n + 5 = 4 * (2 * k - 1) + 5 = 2 * (4 * k) + 1$. Damit ist $4 * n + 5$ wiederum ungerade. qed.

Also ist $4 * n + 5$ ungerade. qed.

7.5.4 Teilbarkeit

Ein einfacher Beweis unter Verwendung der Definition der Teilbarkeit; Blatt 10, Aufgabe $3(a)$

Es seien a, b ganze zahlen. es gelte $3|a$. Zeige: Gilt $3|a+b$, so gilt $3|b$.

Beweis: Es gelte $3|a+b$. Es sei q eine ganze Zahl mit $a = 3*q$. Es sei k eine ganze Zahl mit $a+b = 3*k$. Dann ist $b = (a+b)-a = (3*k)-(3*q) = 3*(k-q)$. Also gilt $3|b$. qed.

Ein Beweis einer allgemeineren Teilbarkeitseigenschaft unter Verwendung einer Äquivalenzumformung; Blatt 10, Aufgabe $3(b)$

Es seien a, b, c ganze Zahlen. Es gelte $\sim (c = 0)$ und $a*c|b*c$. Zeige: Dann gilt $a|b$.

Beweis: Es sei q eine ganze Zahl mit $(a*c)*q = b*c$. Es gilt $((a*c)*q = b*c) \Leftrightarrow (/c)(a*q = b)$. Also gilt $a|b$. qed.

Ein Teilbarkeitsbeweis mithilfe einer Fallunterscheidung (Blatt 10, Aufgabe $1(c)$):

Es sei n eine ganze Zahl. Wir zeigen: 8 teilt $(2*n-1)^2-1$.

Beweis: Es ist $(2*n-1)^2-1 = (2*n-2)*2*n = 4*(n-1)*n$.

Fall 1: Angenommen, n ist gerade. Es sei k eine ganze Zahl mit $n = 2*k$. Dann ist $4*(n-1)*n = 4*(n-1)*(2*k) = 8*((n-1)*k)$. Damit folgt $8|4*(n-1)*n = (2*n-1)^2-1$. Also gilt $8|(2*n-1)^2-1$. qed.

Fall 2: Nun sei n ungerade. Es sei k eine ganze Zahl mit $n = 2*k-1$. Dann ist $4*(n-1)*n = 4*((2*k-1)-1)*n = 8*((k-1)*n)$. Also gilt $8|4*(n-1)*n = (2*n-1)^2-1$. Damit haben wir wiederum $8|(2*n-1)^2-1$. qed.

Also gilt $8|(2*n-1)^2-1$. qed.

Kapitel 8

Die Einbindung des Systems in die Lehre

In diesem Kapitel wollen wir eruieren, wie der Einsatz von Diproche in der universitären Lehre aussehen kann und wie er sich auswirkt.

Diproche wurde im Herbstsemester 2020/2021 probeweise an der Europa-Universität Flensburg eingesetzt, und zwar im Rahmen der Vorlesung "Algebra" von Prof. Hinrich Lorenzen, in deren Rahmen Anfängerstudierende des ersten Semesters in grundlegende Inhalte und Methoden der Mathematik eingeführt werden. Zu den behandelten Themen gehören Aussagenlogik und Grundlagen der Mengenlehre ebenso wie der Umgang mit quantorenlogischen Formeln, das Formalisieren von Aussagen sowie grundlegende Beweisstrategien wie direkte Beweise, Widerspruchs- und Kontrapositionsbeweise oder Beweise mit vollständiger Induktion. Die Vorlesung passt daher inhaltlich sehr gut zu den von Diproche unterstützten Anwendungsfeldern; tatsächlich wurde Diproche z.T. entlang dieser Lehrveranstaltung entwickelt.

Wir werden nun zunächst schildern, wie Diproche konkret in die Lehrveranstaltung eingebunden wurde. Anschließend evaluieren wir den Einsatz von Diproche (i) anhand der Bearbeitungshäufigkeiten und -erfolge der Diproche-Übungsaufgaben, (ii) anhand einer Befragung der TeilnehmerInnen der Lehrveranstaltung und (iii) durch eine statistische Untersuchung des Zusammenhangs zwischen Bearbeitungshäufigkeit und -erfolg bei den Diproche-Übungsaufgaben und der Bearbeitung einer Beweisaufgabe zur Mengenlehre, die im Rahmen der Abschlussklausur gestellt wurde.

Im vorliegenden Abschnitt soll es nun darum gehen, wie der Einsatz von Diproche von den Beteiligten erlebt wurde wie auch darum, seine Wirkung abzuschätzen. Konkret werden wir uns folgenden Fragen widmen:

- Das Ausmaß und die Art der Nutzung: Wie viele TeilnehmerInnen der Vorlesung haben das System verwendet? Wie viele der Übungsaufgaben wurden bearbeitet? Wie viel Zeit wurde dafür aufgewendet?

- Die Bedienbarkeit des Systems: Wie leicht fällt es Studierenden, Texte zu formulieren, die – unabhängig von ihrer inhaltlichen Korrektheit – von Diproche verarbeitet werden können? Wie schwierig ist die Eingabe über das Interface? Wie sind die zusätzlichen Herausforderungen zu gewichten, die durch die trotz einiger Vereinfachungen noch immer recht strikten Formelsyntax entstehen? Wie gut können Studierende die Rückmeldungen des Systems interpretieren, auf ihren Text beziehen und als Ausgangspunkt für Verbesserungen nutzen?

- Die Auswirkungen des Systems auf die Motivation der Studierenden: Trägt das System zur vermehrten und intensivierten Befassung mit Beweisen bei? Wie gerne gehen Studierende mit dem System um? Trägt die unmittelbare Rückmeldung dazu bei, länger an Beweistexten zu arbeiten? Oder wirken die Fehlermeldungen demotivierend?

- Die Auswirkungen des Systems auf die Kompetenzen der Studierenden: Trägt die Verwendung des Systems dazu bei, dass Beweise besser verstanden werden? Hat das System eine Auswirkung auf die Kompetenz, Beweistexte korrekt zu formulieren und zu strukturieren? Vermindert es die Anzahl bekannter Fehlumformungen und Fehlschlüsse?

8.1 Beschreibung des Einsatzes von Diproche im Rahmen der Algebra-Vorlesung an der Europa-Universität Flensburg im Herbstsemester 2020/2021

Um Diproche in der Lehre einsetzen zu können, ist es erforderlich, das System an den bestehenden Lehrbetrieb anzubinden. Bezüglich der Lehrveranstaltung "Algebra 1" lassen sich hierzu folgende Komponenten identifizieren:

1. Die Vorlesungen, die jede Woche in zwei Sitzungen mit je 4 Stunden stattfand.

2. Die Übungsblätter, die wöchentlich an die Studierenden ausgegeben wurden und von diesen zu bearbeiten sind.

3. Die Korrektur der Übungsblätter durch studentische Hilfskräfte.

4. Die vorbereitenden Übungen, in denen die Studierenden in Anwesenheit einer Lehrperson in kleinen Gruppen an den Übungsaufgaben arbeiten und Fragen dazu stellen können.

5. Die Übungsgruppen, in denen jeweils die Aufgaben der Vorwoche besprochen werden.

Der ursprüngliche Plan zur Erprobung des Diproche-Systems in der Lehre sah vor, eine oder zwei gesonderte vorbereitende Übungen und Übungsgruppen einzurichten, die vom Entwickler des Diproche-Systems geleitet werden sollten und in denen unter dessen Anleitung mit dem System gearbeitet worden wäre. Der Vorteil hätte hier darin bestanden, dass bei der Arbeit mit dem System ständig eine Lehrperson präsent gewesen wäre, die das System in allen Einzelheiten kennt und dessen Funktionen wie auch die Motivation dahinter erläutern kann. Die Wirkung des Systems sollte dann durch Vergleich der Klausurergebnisse der TeilnehmerInnen dieser Übungen mit denen einer Kontrollgruppe aus TeilnehmerInnen anderen Übungsgruppen überprüft werden. Dieser Plan wurde jedoch aus ethischen Erwägungen fallen gelassen: Da eine förderliche Wirkung vom System erwartet wurde, wäre es eine ungerechtfertigte Benachteiligung der Studierenden aus der Kontrollgruppe gewesen, keinen Zugang dazu zu haben. Aus diesem Grund wurde beschlossen, dass das System von sämtlichen VorlesungsteilnehmerInnen verwendet werden sollte.

Auf diese Weise waren weitaus umfangreichere Maßnahmen erforderlich, um die Einbindung des Systems zu ermöglichen: Zunächst wurden in gemeinsamen Sitzungen von Entwickler und Dozent Stellen im Vorlesungsskript identifiziert, die sich für Demonstrationen des Systems besonders anboten; die Funktionsweise wie auch die Eingabesprache von Diproche sollten so durch die Vorführung von Beispielen vorlesungsbegleitend eingeführt werden. Zur Vorbereitung der MitarbeiterInnen wurde ein Workshop abgehalten, in dem das System vorgestellt und anhand von Beispielen erläutert wurde. Die Studierenden sollten dann anhand von zusätzlichen Aufgaben auf den Übungsblättern, die explizit als Diproche-Aufgaben gekennzeichnet waren, mit dem System arbeiten. Um die studentischen Hilfskräfte, die mit der Korrektur der Übungsaufgaben betraut waren, nicht zu überlasten, wurde entschieden, dass die Diproche-Aufgaben nicht zusätzlich von Hand korrigiert werden sollten, sondern die Rückmeldungen des Systems ausreichen sollten.[1] Auf diese Weise war eine zusätzliche Schulung der KorrektorInnen unnötig und entfiel. Die ursprüngliche Planung sah ferner vor, die Diproche-Aufgaben mit einer Woche Vorlauf zu stellen und von den MitarbeiterInnen testen zu lassen, so dass verschiedene Lösungsansätze ausprobiert und bei Schwierigkeiten das System ggf. korrigiert oder ergänzt oder umgekehrt

[1] Wir bemerken an dieser Stelle, dass Carter und Monks in der Evaluation von Lurch (s.o.) sehr positive Effekte einer zusätzlichen menschlichen Korrektur der mit dem System bearbeiteten Beweisaufgaben festgestellt haben; diese führten in einigen Fällen zu einer Steigerung des Vertrauens der Studierenden in die Rückmeldungen des Systems, siehe Carter und Monks [42], S. 9.

Hinweise zur Bedienung an die Studierenden gegeben werden konnten.

Bedingt durch die Schließung der Europa-Universität Flensburg im Herbstsemester 2020/2021 aufgrund der Corona-Pandemie und die Umstellung von Präsenzveranstaltungen auf digitale Lehre konnte dieser Plan nur bedingt umgesetzt werden. Der veränderte Veranstaltungsmodus führte zu einer insgesamt erhöhten Arbeitsbelastung und erschwerte die Planung, so dass eine frühzeitige Ausarbeitung der Übungsaufgaben sich als unrealistisch erwies; die Aufgaben wurden so nicht mit einer Woche Vorlauf gestellt, sondern wenige Stunden, ehe die Übungsblätter ausgegeben wurden. Auf diese Weise unterblieb die intendierte einwöchige Testphase; traten Probleme mit dem System auf, mussten diese kurzfristig behoben werden, während die Studierenden bereits mit den Aufgaben beschäftigt waren; auch blieb kaum Zeit, diese Änderungen am System rechtzeitig gründlich zu testen und ggf. noch einmal zu überarbeiten. Schwierigkeiten mit der Inbetriebnahme des Webservers führten dazu, dass das System nicht mit Beginn der Vorlesungszeit einsatzbereit war, sondern erst in der fünften Vorlesungswoche, so dass die Studierenden erst deutlich später als geplant mit dem System in Kontakt kamen, für die ursprünglich intendierte systematische Einführung keine Zeit blieb und die Erläuterungen zum System sich primär auf die Demonstration von Beispielbeweisen beschränken musste. Auch nach der fünften Vorlesungswoche konnte das System nur durch Ausweichen auf einen virtuellen Server verfügbar gemacht werden, was die Zugriffsmöglichkeiten einschränkte, wodurch Änderungen am System erst mit deutlichen Verzögerungen wirksam wurden. Aus Kostengründen konnte der virtuelle Server nur mit eingeschränkten Ressourcen betrieben werden, wodurch das System sich verlangsamte und zu Spitzenzeiten, etwa bei simultanem Zugriff durch mehrere Übungsgruppen, bisweilen überlastet war. Der verringerte kollegiale Kontakt erschwerte den Austausch über das System, wodurch es wiederum schwieriger wurde, in den Übungsgruppen hilfreiche Erklärungen zum System abzugeben. Einige MitarbeiterInnen kamen erst nach dem einführenden Workshop zur Abteilung und konnten so das System nur anhand diverser Beispieltexte kennenlernen. Der Ausfall sämtlicher Präsenzveranstaltungen verringerte die Interaktion zwischen Lehrkräften und Studierenden, wodurch die Studierenden deutlich weniger Hilfestellung im Umgang mit dem System erhielten als ursprünglich geplant.[2] Auch ist zu berücksichtigen, dass der Einsatz des Systems bei Studierenden des ersten Semesters erfolgte, die infolge der Umstellung auf digitale Lehrformate weder zueinander noch zu den Lehrkräften im Laufe des Semesters persönlichen

[2]So fanden etwa die vorbereitenden Übungen über Webex statt, wo die Studierenden in kleinen separaten Gruppenräumen an den Aufgaben arbeiteten und über einen Button Hilfe anfordern konnten. Auf diese Weise wurden deutlich weniger Fragen gestellt; ferner entfiel die in der Präsenzlehre gegebene Möglichkeit, durch das Umhergehen im Raum auf Schwierigkeiten aufmerksam zu werden und aktiv helfend einzugreifen.

Kontakt hatten; dies dürfte die Hemmschwelle, sich bei Schwierigkeiten in einer Übungsgruppe oder auch per Mail an die DozentInnen zu wenden, zusätzlich erhöht haben. Die erschwerte Planbarkeit führte dazu, dass Diproche-Aufgaben primär für die Wiederholung von bereits in den Vorwochen behandeltem Stoff eingesetzt wurde. Insbesondere waren die Arten von Beweisen, die in Diproche zu führen waren, vorher bereits in Vorlesung und Übungen in anderer Form dargestellt worden, wodurch das Vorhaben, die Diproche-Syntax als ersten Zugang zur Fachsprache zu vermitteln, durchkreuzt wurde. So erschien die Diproche-Syntax als nachträgliche Einschränkung einer bereits teilweise erlernten Ausdrucksweise, die eine zusätzliche Irritation darstellte. Die angestrebte "Unsichtbarkeit der Standardisierung" (s.o.) war somit nur ebenfalls bedingt gegeben.[3]

Diese Abweichungen der tatsächlichen Praxis von der ursprüngliche geplanten Weise des Einsatzes hatten zur Folge, dass die Studierenden nicht im intendierten Umfang in das System eingeführt wurden. So musste z.B. aus Zeitgründen darauf verzichtet werden, die Absatzstruktur für die Markierung der Geltungsbereiche von Annahmen zu erläutern und zu üben, wodurch eine Gliederung in Teilbeweise nur durch explizite Marker wie "Fall 1:", "$\Rightarrow$:" oder "$\subseteq$:" möglich war, was in einigen der ausgewerteten Abgaben deutlich als Fehlerursache in Erscheinung trat. Dies sollte bei den Erläuterungen zur Einbindung des Systems in den Vorlesungs- und Übungsbetrieb wie auch bei der Auswertung der Ergebnisse berücksichtigt werden.

Im Hinblick auf die Entwicklung des Systems erwies sich dieser erste Testlauf dennoch als durchaus fruchtbar. Zum einen gab es vonseiten der DozentInnen immer wieder Hinweise auf wünschenswerte Formulierungen, die im System bisher nicht implementiert waren; in den meisten Fällen wurden diese ergänzt und trugen dazu bei, die Diproche-CNL der natürlichen mathematischen Sprache anzunähern und so die Bedienbarkeit des Systems zu verbessern.[4] Zum anderen wurden vereinzelt auch Inferenzschritte angemerkt, die vom System nicht akzeptiert wurden, aber mit einiger Berechtigung als "hinreichend kleinschrittig" angesehen werden konnten, um als korrekt angesehen zu werden. In solchen Fällen waren die bereits oben in den Ausführungen zum ATP erläuterten Abwägungen zu treffen; in der Folge wurden die fraglichen Regeln in einigen Fällen hinzugefügt, in anderen

[3]Diese Schwierigkeiten ergaben sich also einerseits aus der speziellen Situation im Herbstsemester 2020/2021, andererseits aus den Verzögerungen bei der Inbetriebnahme des Webservers. Die Einarbeitung in das System und die Betreuung der Studierenden im Hinblick auf den Umgang damit im Rahmen der vorbereitenden Übungen sowie der Übungsgruppen stellte für die ohnehin stärker als sonst belasteten KollegInnen der Abteilung für Mathematik und ihre Didaktik einen deutlichen zusätzlichen Arbeitsaufwand dar. Ich möchte ihnen an dieser Stelle für die engagierte und gute Zusammenarbeit danken wie auch für zahlreiche Hinweise, die bei der Verbesserung des Systems geholfen haben.

[4]Zu den ergänzten Formulierungsmöglichkeiten gehörten etwa die Wendungen "Wähle/Fixiere/Betrachte..." für die Deklaration von Variablen.

nicht.[5] Ferner tauchten vereinzelt auch Fehler auf, die in der Mehrzahl der Fälle rasch behoben werden konnten.

8.1.1 Einbindung in den Übungsbetrieb

Die Einbindung von Diproche erfolgte vor allem dadurch, dass auf den Übungsblättern Aufgaben gestellt wurden, die mit verschiedenen Komponenten des Diproche-Systems zu bearbeiten waren. Wir geben hier einen Überblick über die Aufgaben sowie die Erfahrungen, die daraus erwuchsen. Die Aufgaben wurden von Hinrich Lorenzen, Michael Schmitz und dem Autor gemeinsam erarbeitet.

Auf den Blättern 1 und 4 wurden noch keine Diproche-Aufgaben gestellt. Das lag zum einen daran, dass zu diesem Zeitpunkt der Server noch nicht vom Netz aus zu erreichen war, zum anderen auch daran, dass die Vorlesungen noch nicht ausreichend fortgeschritten war, um Aufgaben zu stellen, die mit dem System zu bearbeiten gewesen wären.

Die Diproche-Aufgaben wurden in das System eingepflegt; auf den Übungsblättern erschien dann letztlich nicht der Aufgabentext, sondern lediglich Links, unter denen die jeweilige Aufgabe im System zu erreichen war; auf diese Weise sollte die "Hürde" zur Verwendung des Systems herabgesetzt und vermieden werden, dass die Aufgaben ohne Verwendung des Systems bearbeitet wurden. Vereinzelt wurde auch bei den übrigen Übungsaufgaben darauf hingewiesen, dass diese mit dem System bearbeitet werden konnten und ein entsprechender Link zur Verfügung gestellt.

Studierende waren ausdrücklich aufgefordert, sich bei Schwierigkeiten im Umgang mit dem System zu melden. Solche Anfragen kamen nur sehr vereinzelt und betrafen dann stets Rückmeldungen des Systems; die Erreichbarkeit des Systems scheint also zu keinem Zeitpunkt ein Problem gewesen zu sein.

Um die Studierenden angesichts der begrenzten Möglichkeiten zur direkten Hilfestellung nicht mit einer zu großen Vielzahl an Übungs- und Eingabeformaten zu überfordern, wurden nur einige der oben beschriebenen Diproche-Komponenten tatsächlich eingesetzt. Die gestellten Übungsaufgaben waren von folgenden Typen:

- Aufgaben zu den aussagenlogischen "Mathediktaten", bei denen natürlichsprachliche Aussagen in die Sprache der Aussagenlogik zu übersetzen waren.

- Aufgaben zu den mengentheoretischen "Mathediktaten", bei denen

[5] Hinzugefügt wurde etwa die Regel, dass die arithmetische Termgleichheit $T_0' = T_1'$ gefolgert werden kann, wenn $T_0 = T_1$ zu den Voraussetzungen gehört und reelle Zahlen a, b so existieren, dass $aT_0 + b = T_0'$ und $aT_1 + b = T_1'$, wodurch z.B. folgender Text akzeptiert wird: "Es seien a, b ganze Zahlen. Es gelte $4 * a + 2 = 2 * b - 4$. Dann ist $(a + a) = (b - 1) - 2$.".

natürlichsprachliche Aussagen in die Sprache der Mengenlehre zu übersetzen waren.

- Aussagenlogische "Vereinfachen"-Aufgaben, bei denen zu einer gegebenen aussagenlogischen Formel eine möglichst einfache äquivalente aussagenlogische Formel anzugeben war.
- Mengentheoretische "Vereinfachen"-Aufgaben, bei denen zu einer gegebenen aussagenlogischen Formel eine möglichst einfache äquivalente mengentheoretische Formel anzugeben war.
- "Reformulate"-Aufgaben, bei denen eine gegebene aussagenlogische Formel auf möglichst einfache Weise mit einem fixierten Vorrat an aussagenlogischen Junktoren auszudrücken war, wozu insbesondere auch noch unvertraute Junktoren wie die exklusive Disjunktion gehörten.
- "Game of Def"-Aufgaben, bei denen eine in einem Bild gegebene Menge von Quadraten durch eine Formel in der erststufigen Prädikatenlogik zu beschreiben war.
- Beweisaufgaben, bei denen ein Beweisziel gegeben war und die Aufgabe darin bestand, einen Beweistext zu schreiben, der von Diproche als Beweis des Zieles akzeptiert wird.

Zunächst in den Besprechungen von Hinrich Lorenzen, Michael Schmitz und dem Entwickler erwogen, aber aufgrund der auf die oben genannten Ursachen zurückzuführende geringeren Zeit für die Befassung mit dem System im Herbstsemester 2020/2021 nicht umgesetzt wurden folgende Arten von Aufgaben:

- Umformulierungsaufgaben, bei denen eine gegebene Beweisskizze zu einem vollständigen, von Diproche akzeptierten Beweis auszuformulieren ist.
- Beurteilungsaufgaben, bei denen zu einem gegebenen Beweistext die zugehörige Diproche-Ausgabe vorherzusagen und zu erklären ist.

8.1.2 Erfahrungen mit Diproche-Übungsaufgaben

Die Übungsblätter zur Algebra 1-Vorlesung wurden wöchentlich ausgegeben und waren mit einer Woche Bearbeitungszeit abzugeben, wobei die Studierenden dazu angehalten waren, in Zweiergruppen zu arbeiten, um einerseits den Anreiz zur Kooperation zu erhöhen und andererseits den Arbeitsaufwand für die mit der Korrektur betrauten studentischen Hilfskräfte zu verringern. Im Herbstsemester 2020/2021 nahmen 228 Studierende am Übungsbetrieb teil, was also etwa 114

wöchentlichen Abgaben entspricht.[6] Für die Untersuchung zu den Diproche-Aufgaben wurde eine Stichprobe von 28 Abgabeteams ausgewählt, was also 56 beteiligten Studierenden entspricht, etwa einem Viertel der TeilnehmerInnenzahl. Da erst ab dem fünften Übungsblatt Diproche-Aufgaben gestellt wurden und diese erst ab Blatt 6 abzugeben waren, umfasst die Auswertung das fünfte Blatt noch nicht, auf dem die ersten Aufgaben mit dem System zu bearbeiten waren.

Wir werden nun die sieben Übungsblätter, auf denen Diproche-Aufgaben vorkamen, separat betrachten. Dabei bezeichnen wir eine Aufgabe als "bearbeitet", wenn die Abgabe einen Lösungsversuch zu dieser Aufgabe enthielt.[7]

Blatt 5

Auf Blatt 5 wurden, im Rahmen der Einführung in die Aussagenlogik, zehn Formalisierungsaufgaben gestellt, die mit den Diproche-Modulen "Mathediktat Aussagenlogik" und "Mathediktat Mengenlehre" zu bearbeiten waren. Hier wurde darauf geachtet, natürlichsprachliche Formulierungen zu geben, die die zugehörigen logischen Junktoren nicht sofort offensichtlich machten, um die Studierenden zur vermehrten und vertieften Reflexion über die logische Struktur der jeweiligen Aussage anzuregen. Für diese Aufgaben bestand keine Abgabepflicht. Sie wurden als die letzten beiden Aufgaben auf dem Blatt gestellt.

In der Vorlesung wurden einige Beispielaufgaben der gleichen Art vorgestellt und mit den Studierenden besprochen. Die Studierenden – die zu diesem Zeitpunkt noch keinen Zugriff auf das System hatten – konnten dem Dozenten dabei schriftlich über die Chat-Funktion des für die Vorlesung verwendeten Online-Konferenzdienstes Lösungsvorschläge zukommen lassen, die dann in die Eingabezeile des Systems kopiert wurden, so dass die jeweiligen Rückmeldungen des Systems betrachtet werden konnten. Dabei traten erwartungsgemäß – die Syntax der Aussagenlogik und Mengenlehre war zu diesem Zeitpunkt noch nicht lange eingeführt – sowohl inhaltliche als auch formale Fehler auf, die dann besprochen wurden.

[6] Die genaue Zahl der Abgaben variiert von Woche zu Woche: Einerseits stiegen vereinzelte Studierende noch verspätet in den Übungsbetrieb ein, während andere die Teilnahme abbrachen; andererseits gab es bedingt durch Schwierigkeiten bei der Bildung von Abgabeteams auch Einzelabgaben.

[7] Damit soll ausdrücklich nicht ausgeschlossen werden, dass auch bei Studierenden, die keine Lösung abgegeben haben, eine Auseinandersetzung mit diesen Aufgaben stattgefunden hat, möglicherweise auch unter Einsatz der Diproche-Korrektur. Die z.T. stark voneinander abweichenden Bearbeitungszahlen zwischen verschiedenen Teilaufgaben einiger Diproche-Aufgaben legen in Verbindung mit den häufig nahe bei 100% liegenden Anteilen korrekter Bearbeitungen im Gegenteil den Schluss nahe, dass Studierende, die aufgrund der automatischen Korrektur bereits um die Fehlerhaftigkeit ihres Lösungsversuches wussten, darauf verzichtet haben, ihn abzugeben.

Da für die Diproche-Aufgaben von Blatt 5 noch keine Abgabepflicht bestand, fehlt die Grundlage für eine statistische Auswertung sowohl der Bearbeitungshäufigkeit wie auch des Erfolges. In den vorbereitenden Übungen wurden diese Aufgaben von den Studierenden kaum thematisiert, vermutlich, weil sie nicht abzugeben waren und auf dem Übungsblatt als letzte Aufgaben platziert wurden. Ein genereller Eindruck aus den wenigen Nachfragen in den vorbereitenden Übungen, der von mehreren DozentInnen geteilt wurde, war der, dass die Schwierigkeiten bei den mengentheoretischen Mathediktaten deutlich größer waren als bei den aussagenlogischen; insbesondere wurden hier häufig syntaktische Fehler gemacht, die zur Meldung führten, dass die Eingabe nicht verarbeitet werden konnte, etwa die Verwechselung logischer Junktoren mit mengentheoretischen Operatoren. Die Besprechung in den Übungsgruppen der folgenden Woche legte nahe, dass die Schwierigkeiten nicht primär im Umgang mit dem System bestanden, sondern mehr in der Übersetzung natürlicher Sprache in die Sprache der Mengenlehre. Mithilfe von Nachfragen wie "Wie bilden wir aus der Menge H der Häuser und der Menge G der grünen Dinge die Menge der grünen Häuser?" vonseiten der Lehrkraft konnten die Aufgaben jedoch letztlich überwiegend gelöst werden. Vereinzelt gab es auch Nachfragen dazu, wie gewisse Sonderzeichen wie etwa das "&" über die Tastatur einzugeben seien, die aber rasch geklärt werden konnten.

Blatt 6

Auf Blatt 6 wurden vier Vereinfachungsaufgaben zur Aussagenlogik gestellt, um den Umgang mit aussagenlogischen Umformungsregeln zu üben, der bereits eine erste, einfache Form des Beweisens darstellt.[8]

Ferner wurden einige weitere, einfachere aussagenlogische und mengentheoretische Mathediktate gestellt; Grund hierfür war die Rückmeldung von einigen ÜbungsleiterInnen über Schwierigkeiten, die einige Studierende mit der Bedienung des Systems hatten. Die Aufgaben dienten somit primär dem Ziel, den Umgang mit dem System zu üben und Berührungsängste abzubauen. Die ursprünglich für diese Woche geplanten "Game of Def"-Aufgaben wurden aus demselben Grund in die nächste Woche verschoben.

Um den Anreiz für eine ernsthafte Beschäftigung mit den Aufgaben zu erhöhen, bestand für diese Aufgaben eine Abgabepflicht; ferner wurden sie als erste

[8] Allerdings ist das "Vereinfachen"-Modul, wie oben erwähnt, kein Programm zur Überprüfung solcher Umformungsfolgen; diese ließe sich zwar im Rahmen der aussagenlogischen Spielwiese von Diproche durchführen, worauf aber aus Gründen der Einfachheit zunächst zugunsten einer reinen Ergebniskontrolle verzichtet wurde.

Aufgaben auf dem Übungsblatt gestellt, um zu einer vermehrten Befassung mit den Diproche-Aufgaben im Rahmen der vorbereitenden Übungen anzuregen.

Die Aufgaben waren folgende:

(i) **Mathediktate Aussagenlogik und Mengenlehre:**

1. Es bezeichne L die Aussage "Im Zoo gibt es Loewen.". Druecke die Aussage"'Im Zoo gibt es keine Loewen" in der Sprache der Aussagenlogik aus.

2. Es sei L die Aussage "Mario laeuft" und S die Aussage "Mario springt". Druecke die Aussage "Mario laeuft und springt" in der Sprache der Aussagenlogik aus.

3. Es sei S die Aussage "Ich schalte das Licht an." und H die Aussage "Es wird hell.". Druecke die Aussage "Es wird hell, wenn ich das Licht anschalte." in der Sprache der Aussagenlogik aus.

4. Es bezeichne Z die Menge der Zoos und L die Menge der Orte, an denen es Loewen gibt. Druecke "Jeder Zoo hat Loewen" in der Sprache der Mengenlehre aus.

5. Es sei S die Menge aller springenden Menschen und L die Menge aller laufenden Menschen. Druecke die Aussage "Alle Menschen, die springen, laufen auch." in der Sprache der Mengenlehre aus.

6. Es sei W die Menge der Wale, S die Menge der Saeugetiere, M die Menge der Meeresbewohner. Druecke "Wale sind Meeressaeuger" in der Sprache der Mengenlehre aus.

Inhaltlich war in den ersten drei Aufgabenteilen also nur ein einzelner aussagenlogischer Junktor korrekt anzuwenden, in (4) und (5) entsprechend nur eine einzelne mengentheoretische Relation; lediglich (6) erforderte die gleichzeitige Verwendung mengentheoretischer Relationen und Operatoren.

(ii) **"Vereinfachen"-Aufgaben**

Die zu vereinfachenden aussagenlogischen Ausdrücke waren:

1. $\neg(B \to \neg C) \wedge ((A \wedge C) \to (A \vee C))$

2. $(\neg(A \wedge \neg B) \wedge A) \vee (\neg(A \wedge \neg B) \wedge \neg A)$

3. $((A \to B) \to \neg B) \to B$

4. $\neg(B \to (B \to B)) \leftrightarrow ((B \to B) \to B)$

Eine von verschiedenen Studierenden berichtete und auch in den vorbereitenden Übungen beobachtete Erfahrung mit den "Vereinfachen"-Aufgaben war, dass eine Eingabe zu der Rückmeldung führte, die eingegebene Formel sei zwar zur Ausgangsformel logisch äquivalent, aber noch nicht von minimaler Länge; wurde dann keine Idee für eine weitere Vereinfachung gefunden, wurde der Lösungsprozess trotzdem abgebrochen. Hier hatte die unmittelbare Rückmeldung also nur bedingt eine verstärkte Beschäftigung mit der Aufgabe zur Folge. Eine Lehre, die sich aus dieser Beobachtung bereits ziehen läßt, ist die, dass eine interaktive Fehlerkorrektur nur dann zu einem erhöhten Übungseffekt führt, wenn die BenutzerInnen im Fall einer Fehlermeldung Ansätze haben, um ihre Lösung zu verbessern. Dagegen wurden die aussagenlogischen Mathediktate überwiegend als unproblematisch gemeldet; die meisten Studierenden konnten diese Aufgaben korrekt lösen und ihre Lösungen auch über das System so eingeben, dass sie als korrekt gemeldet wurden. Nachfragen bezogen sich überwiegend darauf, wie gewisse Sonderzeichen, wie etwa die Negation ($\sim$), über die Tastatur einzugeben seien. Auch hierin liegt eine zwar grundlegende, aber eben darum wesentliche Voraussetzung für die Bedienbarkeit des Systems, die bei einem Einsatz des Systems in der Lehre rechtzeitig thematisiert werden sollte.

Zu den Übungsaufgaben zu Blatt 6 gab es 25 Abgaben. Bei den "Mathediktat"-Aufgaben wurde, entsprechend der vom System zur Verfügung gestellten Rückmeldung, zwischen Abgaben unterschieden, die (i) als korrekt gemeldet wurden und (ii) syntaktisch korrekt waren, aber nicht zum jeweiligen natürlichsprachlichen Satz äquivalent, also inhaltlich falsch waren. Nicht vom System, aber für die folgende Auswertung händisch erfasst wurde weiter die Anzahl der inhaltlich korrekten Abgaben mit syntaktischen Fehlern sowie der sowohl inhaltlich als auch syntaktisch falschen Abgaben. Da das System syntaktisch fehlerhafte Eingaben nicht inhaltlich verarbeiten kann, wurden beide vom System einfach als "syntaktisch falsch" gemeldet. Interessant für die Evaluation des Systems ist hier besonders die Anzahl der inhaltlich korrekten, aber syntaktisch fehlerhaften Abgaben; bei diesen hätte das System einen inhaltlich korrekten Gedanken aufgrund der syntaktischen Normierung nicht als solchen erkannt und gemeldet. Tatsächlich trat dieser Fall aber nicht auf.

Die folgende Tabelle fasst die Abgabehäufigkeiten sowie die Erfolge bei den jeweiligen Teilaufgaben zusammen. Die Prozentahlen in der zweiten Spalte geben dabei den Anteil der Bearbeitungen an der Gesamtzahl der Abgaben an; die übrigen Prozentzahlen beziehen sich auf die in der zweiten Spalte angegebene Anzahl der Bearbeitungen der jeweiligen Teilaufgabe, nicht auf die Gesamtzahl aller Abgaben.

Table 8.1: Bearbeitung und Bearbeitungserfolg der “Mathediktate” auf Blatt 6

Aufgabe	Bearbeitet	korrekt	syntaktisch korrekt, aber inhaltlich falsch	inhaltlich korrekt, aber syntaktisch falsch	inhaltlich und syntaktisch falsch
1	24 (96%)	24 (100%)	0	0	0
2	24 (96%)	24 (100%)	0	0	0
3	24 (96%)	24 (100%)	0	0	0
4	24 (96%)	23 (96%)	1 (4%)	0	0
5	24 (96%)	24 (100%)	0	0	0
6	23 (92%)	22 (96%)	1 (4%)	0	0

Table 8.2: Bearbeitung und Bearbeitungserfolg der “Vereinfachen”-Aufgaben auf Blatt 6

Aufgabe	Bearbeitet	korrekt und optimal	korrekt, aber nicht optimal	nicht korrekt
1	19 (76%)	15 (79%)	1 (5%)	3 (16%)
2	19 (76%)	16 (84%)	2 (11%)	1 (5%)
3	20 (80%)	18 (90%)	0	2 (10%)
4	19 (76%)	17 (89%)	0	2 (11%)

Bei 11 der 25 Abgaben, also bei 44% der Abgaben, wurden alle 10 Diproche-Aufgaben korrekt bearbeitet.

Wie die Daten zeigen, hatten die Studierenden mit den “Mathediktat”-Aufgaben wenig Schwierigkeiten; die Aufgabe wurde in allen Aufgabenteilen überwiegend bearbeitet und überwiegend richtig. Erwartungsgemäß war (6) die schwierigste Teilaufgabe, doch auch diese wurde überwiegend richtig gelöst, einige wenige Male auch dadurch, dass zwei Teilmengenaussagen durch eine Konjunktion verbunden wurden anstatt den Schnittoperator zu verwenden. In einigen Fällen kam es bei den mengentheoretischen Mathediktaten zu Verwechselungen von Teilmengen- und Elementrelation. Da solche Verwechselungen aus der Lehrerfahrung der vorangegangenen Semester bekannt waren, war diese Möglichkeit im System explizit vorgesehen, so dass die Rückmeldung “Die $\in$-Relation kann in dieser Aufgabe nicht vorkommen” ausgegeben wurde. Dass trotzdem die fehlerhaften Lösungen abgegeben wurden, legt nahe, dass die betreffenden Studierenden (i) ihre Lösung trotz der Rückmeldung des Systems für richtig hielten, (ii) die Rückmeldung ernst genommen haben, aber keinen Ansatz oder kein Interesse hatten, ihre Lösung zu verbessern oder (iii) sie das System nicht zur Prüfung ihrer Lösung verwendet haben. Gegen (iii) spricht, dass die meisten dieser Abgaben sich an die Eingabesyntax gehalten haben, die von der in der Vorlesung eingeführten leicht abwich (z.B. musste die Negation durch eine Tilde $\sim$ statt als $\neg$ notiert werden). Ob die Ursache in (i) oder (ii) liegt, kann auf Basis der Daten nicht entschieden werden.

Auch die “Vereinfachen”-Aufgaben wurden von der Mehrheit der Studierenden bearbeitet, nämlich, je nach Aufgabenteil, auf 76% bzw. 80% der Abgaben. Hier fanden sich bereits häufiger korrekte, aber nicht optimale oder nicht korrekte

Abgaben; in der Mehrzahl der Fälle wurden diese Aufgaben aber korrekt gelöst.

Blatt 7

Auf Blatt 7 wurden zur Wiederholung und Vertiefung des Themas "Aussagenlogik" sechs Aufgaben aus dem Modul "Reformulate" gestellt, bei denen gegebene aussagenlogische Formeln mit einem begrenzten Vorrat an Junktoren auszudrücken waren.

In der dem Blatt vorangehenden Vorlesungswoche wurden erstmals Quantoren eingeführt. Zur einführenden Übung der Verwendung mit Quantoren eignen sich vor allem die erststufigen Mathediktate und das "Game of Def". Da zu diesem Zeitpunkt der Vorlesung noch keine hinreichende Vertrautheit mit dem thematischen Bereich "reelle Funktionen" vorausgesetzt werden konnte, der für den Einsatz der erststufigen Mathediktate Voraussetzung ist, wurden 10 Aufgaben zum "Game of Def" gestellt, von denen die ersten 8 zur Einführung dienten und nur die letzten beiden Quantoren verwendeten.[9] Das waren einmal die Menge aller Quadrate, die sich sowohl rechts von als auch über dem mittleren Feld befanden sowie die Menge der Nachbarn von Nachbarn des mittleren Feldes. In beiden Fällen muss für eine erfolgreiche "Bildbeschreibung" durch eine existenzquantifizierte Variable eine "Vermittlung" zwischen dem mittleren Quadrat u und den zu beschreibenden Quadraten hergestellt werden. Die beiden Aufgaben funktionieren weitgehend analog zueinander, so dass in den vorbereitenden Übungen die erste gemeinsam bearbeitet werden konnte, während die andere dann als Anwendung des in der ersten erworbenen Prinzips zu dessen eigenständiger Erprobung Anlass gab. Um den Einstieg in diese neue Arbeitsumgebung einfach zu gestalten, wurden zunächst nur die Prädikate "rechts", "links", "ueber", "unter" sowie "nachbar" eingeführt, die Distanzmaße also vorläufig zurückgestellt.

Zudem bestand Aufgabe 8 darin, mengentheoretische Formeln möglichst weit zu vereinfachen. Die Aufgabe war nicht als Diproche-Aufgabe gestellt, die Studierenden wurden aber darauf hingewiesen, dass sie ihre Lösungen über das mengentheoretische Submodul von "Vereinfachen" überprüfen konnten.

Erwartungsgemäß gab es beim "Game of Def" anfänglich Schwierigkeiten mit der Eingabesyntax, insbesondere mit den strikten Konventionen bezüglich der Klammerung: so können, im Gegensatz zur in Vorlesung und Übungen verwendeten Notation, im "Game of Def" etwa in mehrgliedrigen Disjunktionen oder Konjunktionen die Klammern nicht wie üblich fortgelassen werden. Ähnliches ist über Aufgabe 8 zu sagen, da die Eingabesyntax für mengentheoretische Vereinfachungsaufgaben es – wiederum in Abweichung von den in der Vorlesung verwendeten Konventionen – erfordert, bei der Verwendung mengentheoretischer

[9] Die vollständige Liste der Aufgaben findet sich in Anhang B.

Operatoren grundsätzlich Klammern zu setzen.

Die Abgaben erfolgten überwiegend handschriftlich; nur selten wurde die Diproche-Eingabe in die abgegebene Datei kopiert. Bei der Auswertung wurden daher Lösungen als korrekt angesehen, wenn die Abweichung von einer korrekten Lösung geringfügig und plausibel auf einen Übertragungsfehler zurückzuführen war, wie etwa eine fehlende schließende Klammer am Ende eines komplexeren Terms oder die Verwendung des Umlauts in "über" (statt, wie es die Diproche-Syntax erfordert, "ueber"). Die "Game of Def"-Aufgaben wurden z.T. in größeren Online-Gruppen kooperativ bearbeitet, wodurch es viele identische Abgaben gab; andererseits fanden sich auch deutlich voneinander und den Musterlösungen abweichende Lösungen, die z.T. recht originelle gedankliche Ansätze erkennen ließen und damit zeigten, dass eine eigenständige Auseinandersetzung mit der jeweiligen Aufgabe stattgefunden hatte.

Insgesamt gab es 27 Abgaben zu den Aufgaben von Blatt 7. Die Bearbeitungshäufigkeiten und -erfolge der Diproche-Aufgaben sind in den folgenden Tabellen erfasst. Die Prozentzahlen in der Spalte "bearbeitet" geben dabei den Anteil an der Gesamtzahl (27) der Abgaben an, während die übrigen Prozentzahlen sich auf die Anzahl der Bearbeitungen der jeweiligen Teilaufgabe, nicht auf die Gesamtzahl der Abgaben, beziehen. Bei den "Reformulate"-Aufgaben wurde unterschieden zwischen "richtigen" Abgaben, die das System als korrekt meldete, sowie syntaktisch falschen und syntaktisch korrekten, aber inhaltlich falschen. Bei syntaktischen Fehlern wurde nicht überprüft, ob die angegebene Formel bei "wohlwollender" Interpretation inhaltlich richtig wäre; jede Abgabe fiel also in genau eine Kategorie.

Table 8.3: Ergebnisse der "Reformulate"-Aufgaben auf Blatt 7

Aufgabe	Bearbeitet	richtig	syntaktisch falsch	inhaltlich falsch
1	23 (85%)	23 (100%)	0	0
2	22 (81%)	21 (95%)	0	1 (5%)
3	21 (78%)	21 (100%)	0	0
4	12 (44%)	12 (100%)	0	0
5	14 (52%)	13 (93%)	0	1 (7%)
6	16 (59%)	16 (100%)	0	0

Die Aufgabenteile (1)-(3) der "Game of Def"-Aufgaben dienten zur Übung des Verständnisses der Grundrelationen; ab Aufgabenteil (4) war zur korrekten Bearbeitung die Verwendung aussagenlogischer Junktoren erforderlich, wobei (4) und (5) jeweils nur einen Junktor (die Konjunktion bzw. die Disjunktion) erforderten, (6) eine Kombination mehrerer Disjunktionen und (7) und (8) (Allmenge bzw. leere Menge) jeweils eine Überlegung, um eine Formel zu finden,

die auf alle bzw. keines der Quadrate zutrifft; die Aufgabenteile (9) und (10) erforderten die Verwendung von Quantoren.

Die Bearbeitungen der Aufgaben enthielten vereinzelt syntaktische Fehler, die aber angesichts ihrer geringen Anzahl nicht separat gezählt, sondern als "nicht korrekt" erfasst wurden.

Table 8.4: Ergebnisse der "Game of Def"-Aufgaben auf Blatt 7

Aufgabe	Bearbeitet	richtig
1	21 (78%)	21 (100%)
2	21 (78%)	21 (100%)
3	21 (78%)	21 (100%)
4	19 (70%)	19 (100%)
5	16 (59%)	16 (100%)
6	16 (59%)	14 (88%)
7	19 (70%)	19 (100%)
8	17 (63%)	16 (95%)
9	16 (59%)	14 (88%)
10	17 (63%)	17 (100%)

Insgesamt enthielten zehn der Abgaben, also 37%, korrekte Lösungen zu allen Diproche-Aufgaben, während bei drei Abgaben, also 11%, keine der Diproche-Aufgaben bearbeitet wurde. Die überwiegende Mehrheit der Studierenden hat also mindestens eine der Teilaufgaben mit dem Diproche-System korrekt bearbeitet. Der Umstand, dass es nur wenig fehlerhafte Abgaben gab, aber die Bearbeitungszahlen deutlich mit der Schwierigkeit der Aufgaben variieren, läßt vermuten, dass die seltener bearbeiteten Teilaufgaben durchaus versucht, aber nach Fehlermeldungen durch das System nicht abgegeben wurden. Die äußerst geringe Anzahl von syntaktischen Fehlern zeigt jedenfalls, dass die Eingabesyntax für die Studierenden kaum ein Hindernis darstellte und Schwierigkeiten bei der Bearbeitung der Aufgaben eher inhaltlicher Natur waren, die Nutzung des Systems also keine wesentlichen zusätzlichen Schwierigkeiten bereitete.

Blatt 8

In der siebten und achten Vorlesungswoche wurden in der Vorlesung das mathematische Beweisen sowie grundlegende Beweisstrategien wie der direkten Beweis und der Kontrapositionsbeweis thematisiert. Auf Blatt 8 wurden daher erstmals Aufgaben gestellt, die mit der Hauptkomponente des Systems, dem natürlichsprachlichen Beweisprüfer, zu bearbeiten waren. Dazu erfolgte zunächst

eine Einführung in das System in der Vorlesung, in deren Rahmen Beispielbeweise in der Diproche-Syntax für folgende Aussagen aus dem Bereich "gerade/ungerade Zahlen" vorgestellt wurden:

- Es sei x eine ganze Zahl. Zeige: Wenn x ungerade ist, dann ist $9 * x + 5$ gerade.
- Es sei x eine ganze Zahl. Zeige: Wenn x gerade ist, dann ist $5 * x - 3$ ungerade.
- Es sei x eine ganze Zahl. Zeige: Wenn $3 * x$ gerade ist, dann ist x gerade.

Die ersten beiden Aufgaben wurden durch direkte Beweise gelöst, die dritte durch einen Kontrapositionsbeweis. In jedem Fall wurden die Rückmeldungen des Systems für verschiedene Varianten der Eingabe gezeigt, was einerseits die sprachliche Flexibilität des Systems, andererseits aber auch deren Grenzen aufzeigte. Weiter wurde auf Abweichungen des Systems von der in der Vorlesung gängigen Darstellungspraxis hingewiesen, insbesondere darauf, dass (i) Existenzbehauptungen in Diproche-Texten nicht zur Einführung von Variablen verwendet werden können und (ii) Kontrapositionsbeweise für eine Implikation $A \to B$ mit der Annahme der Negation A beginnen und mit der Folgerung der Negation von B enden müssen, um als erfolgreich zu gelten, und nicht mit Aussagen, die bereits logische Umformungen dieser Negationen darstellen; so muss im dritten oben aufgeführten Beispiel ein Kontrapositionsbeweis etwa mit der Annahme "x ist nicht gerade" beginnen, woraus erst im nächsten Schritt gefolgert werden kann, dass x ungerade ist.[10] Die Studierenden wurden aufgefordert, die vorgeführten Beweise als Vorbild für die Diproche-Übungsaufgaben zu verwenden.

Auf dem achten Übungsblatt wurden 9 weitere "Game of Def"-Aufgaben gestellt, die nun neben den bereits auf dem siebten Blatt eingeführten Relationen der Nachbarschaf sowie "links/rechts/ueber/unter" auch Aufgaben unter Verwendung der "Taxi-Metrik"[11] enthielten. Ferner wurden erstmals Aufgaben gestellt, die die Verwendung mehrerer Quantoren erforderten, etwa die

[10] Hier erscheint, trotz dieser klaren Abweichung von der üblichen Sprachpraxis, eine Anpassung nicht ratsam; um das System insgesamt kohärent zu halten, müsste ein Kontrapositionsbeweis der Implikation $A \to B$ als ein Beweis einer "offensichtlich zur Negation von A äquivalenten" Aussage auf Basis der Annahme einer "offensichtlich zur Negation von B äquivalenten Aussage" verstanden werden. Den unterschiedlichen Vorstellungen bezüglich "offensichtlicher" Äquivalenz in einer konsistenten und formal stringenten Weise gerecht zu werden, die überdies noch den Lernfortschritt berücksichtigt, scheint jedoch aussichtslos.

[11] Die Taxi-Metrik misst den Abstand zweier Gitterquadrate mit den Koordinaten (x_1, y_1) und (x_2, y_2) durch die Summe der Beträge der Differenzen der Koordinaten, also als $|x_1 - x_2| + |y_1 - y_2|$; siehe z.B. Gardner [85].

Beschreibung der Menge der Nachbarn von Nachbarn des mittleren Feldes u. Die "Game of Def"-Aufgaben sind in Anhang B aufgeführt.

Zur eigentlichen Prüfkomponente wurden vier Beweisaufgaben aus dem Bereich "gerade/ungerade Zahlen" gestellt, die den in der Vorlesung vorgestellten Beispielen folgten:

- Es sei x eine ganze Zahl. Zeige: Wenn x gerade ist, dann ist $2-3*x$ gerade.
- Es sei x eine ganze Zahl. Zeige: Wenn x ungerade ist, dann ist $(x-3)*(x+3)$ gerade.
- Es sei x eine ganze Zahl. Zeige: Wenn $17*x+15$ ungerade ist, dann ist x gerade.
- Es sei x eine ganze Zahl. Zeige: Wenn x^2+2 ungerade ist, dann ist x ungerade.

Davon boten sich die beiden ersten Aufgaben für einen direkten Beweis, die dritte und die vierte hingegen für die Behandlung durch einen Kontrapositionsbeweis an. Um die Strategiewahl zu erleichtern, wurden die Studierenden in der Aufgabenstellung explizit darauf hingewiesen, dass die beiden letzten Aufgaben durch Kontraposition zu beweisen waren.

Eine Schwierigkeit, die bei der Bearbeitung bisweilen auftrat, bestand darin, dass die zunächst auf ausdrückliche Nachfrage zur Steigerung der sprachlichen Flexibilität als Alternative zu "mit" eingeführte Wendung "so, dass" im Rahmen von Deklarationen mit inhaltlicher Annahme, wie sie etwa in Sätzen wie "Es sei k eine ganze Zahl so, dass $n = 2k$" auftritt, einige Studierende dazu verleitete, an die Gleichung noch ein "ist" anzuschließen und Sätze wie "Es sei k eine ganze Zahl so, dass $n = 2k$ ist." zu bilden, die sich bei Verwendung von "mit" statt "so, dass" kaum nahegelegt hätten. Da solche Formulierungen im Parser nicht vorgesehen waren, führten diese Eingaben zu einer Fehlermeldung. An diesem kleinen Beispiel läßt sich eine Erwartung bestätigen, die bereits weiter oben bei den Erwägungen zur Konstruktion einer geeigneten kontrollierten Eingabesprache formuliert worden war: Erweiterungen der Sprache legen neue Formulierungen nahe, deren Integration zusätzliche Erweiterungen erfordern würde, so dass man potenziell ins Uferlose gelangt; aus diesem Grund wird die Bedienbarkeit durch eine starke sprachliche Normierung letztlich nicht erschwert, sondern erleichtert. Im konkreten Fall könnte man versucht sein, die Fehlermeldung für angemessen zu halten: Liest man das Gleichheitszeichen als "ist gleich", so ergibt sich durch das angehängte "ist" eine Dopplung. Um der üblichen Sprachpraxis entgegen zu kommen, wurde diese Formulierungen jedoch letztlich rasch in den Parser

integriert, so dass sie bei der Bearbeitung von Blatt 8 noch verwendet werden konnten.

Zu Blatt 8 wurden 26 Bearbeitungen abgebeben. Da diese z.T. handschriftlich erfolgten, wurden die Bearbeitungen von Hand in das Diproche-Eingabefenster übertragen. Fehler in der Eingabesyntax, die plausibel als bloße Übertragungsfehler zu erklären waren (etwa ein fehlender Punkt am Satzende, das Fortlassen des Multiplikationszeichens in einer Formulierung wie "Damit gilt $n = 2 * k$" oder die Darstellung einer Potenz als x^2 statt als $x\hat{}2$), wurden dabei korrigiert. In der unten stehenden Auswertung sind die Bearbeitungen der Beweisaufgaben in jeweils genau eine von drei Kategorien eingeteilt worden:

- "Richtig" bedeutet, dass das System den Eingabetext als in jeder Hinsicht korrekten Beweis der fraglichen Behauptung akzeptierte; insbesondere also, dass (i) alle Beweisziele als erreicht galten, (ii) sämtliche Folgerungsschritte verifiziert werden konnten und (iii) keine Typenfehler vorlagen.

- "Fast" wurde verwendet, wenn bei einer korrekten und vollständigen Lösung Fehlermeldungen auftraten, die auf die (sprachliche oder logische) Normierung durch Diproche zurückzuführen waren. So wurde etwa ein ansonsten korrekter Kontrapositionsbeweis für (3), der mit der Annahme "x ist ungerade" statt mit der Annahme "x ist nicht gerade" begann oder mit "$17 * x + 5$ ist gerade" statt mit "$17 * x + 5$ ist nicht ungerade" endete (was zur Meldung eines nicht erreichten Beweiszieles führte), als "fast richtig" gewertet.

- Lagen dagegen inhaltliche Fehler oder Lücken vor, wozu etwa auch nicht eingeführte Variablen gezählt wurden, wurde die Bearbeitung in die Kategorie Fehler/Lücken eingeordnet.

Für die Beweisaufgaben ergaben sich damit folgende Ergebnisse (wobei sich die Prozentzahlen in der Spalte "Bearbeitet" auf die Gesamtzahl der 26 Abgaben beziehen, die in den übrigen Spalten hingegen auf die Gesamtzahl der Bearbeitungen der jeweiligen Teilaufgabe, nicht auf die Gesamtzahl der Abgaben):

Wie die Tabelle zeigt, wurden alle Teilaufgaben von der Mehrheit der Studierenden bearbeitet; die geringste Bearbeitungshäufigkeit von 70% lag bei der vierten Teilaufgabe vor. Die Zahl "fast richtiger" Bearbeitungen lag bei den mit Kontraposition zu bearbeitenden Teilaufgaben deutlich höher (28% bzw. 32% vs. 5% bzw. 10%); der Grund hierfür war die bereits oben besprochene strikte Normierung in Bezug auf die Syntax von Kontrapositionsbeweisen. Eine deutliche Mehrheit der Bearbeitungen zur ersten Teilaufgabe (85%) enthielt einen vom System als korrekt gemeldeten Beweistext. Dagegen traten in der zweiten

Table 8.5: Bearbeitungen der Diproche-Beweisaufgaben zur Zahlentheorie auf Blatt 8

Aufgabe	Bearbeitet	Richtig	fast richtig	inhaltliche Fehler/Lücken
1	20(77%)	17 (85%)	1 (5%)	2 (10%)
2	20 (77%)	13 (65%)	2 (10%)	5 (25%)
3	19 (73%)	11 (58%)	6 (32%)	7 (37%)
4	18 (70%)	9 (50%)	5 (28%)	4 (22%)

Teilaufgabe vermehrt Fehler auf. Die häufigsten Fehler waren (neben kleineren Syntaxfehlern wie etwa einzelnen fehlenden Klammern) folgende:

- Es wurden Existenzaussagen als Variableneinführung verwendet oder es wurde in einer Existenzaussage kein Bereich für den Quantor angegeben, was zur Meldung von Typenfehlern führte.
- Am Anfang begründeter Aussagen (wie etwa "Da x gerade ist, existiert eine ganze Zahl k mit $x = 2 * k$") wurde statt "Da" das Wort "wenn" verwendet, was dazu führte, dass die jeweilige Aussage als Implikation interpretiert wurde. Diese Implikation ist dann zwar richtig, allerdings gilt durch "Wenn x gerade ist, existiert eine ganze Zahl k mit $x = 2 * k$" die Aussage "Es existiert eine ganze Zahl k mit $x = 2 * k$." noch nicht als bewiesen, woraus sich dann im weiteren Verlauf der Bearbeitung Fehler ergaben (etwa, wenn im nächsten Schritt ein solches k fixiert werden sollte).
- Es wurden statt Beweistexten lediglich Formelketten ohne Verwendung natürlicher Sprache abgegeben.

Anteilig an allen Abgaben haben damit 65% der Studierenden einen in jeder Hinsicht korrekten Lösungstext zu Aufgabe 1 abgegeben.

Bei den "Game of Def"-Aufgaben ergab sich folgendes Bild:

Auch hier zeigt sich eine mit steigendem Schwierigkeitsgrad deutlich abnehmende Bearbeitungshäufigkeit, während der Anteil korrekter Bearbeitungen mit Ausnahme der letzten Zeile konstant hoch liegt. Auch das dürfte darauf zurückzuführen sein, dass bereits durch das System als falsch bekannte Lösungen nicht abgegeben wurden.

Blatt 9

Die Diproche-Beweisaufgaben zur Zahlentheorie auf Blatt 9 dienten der Wiederholung und Festigung von Beweisen durch Fallunterscheidungen. Ferner

Table 8.6: Ergebnisse der "Game of Def"-Aufgaben auf Blatt 8

Aufgabe	Bearbeitet	richtig
1	17 (65%)	15 (88%)
2	16 (62%)	16 (100%)
3	15 (58%)	15 (100%)
4	14 (54%)	14 (100%)
5	13 (50%)	13 (100%)
6	13 (50%)	12 (92%)
7	8 (31%)	8 (100%)
8	9 (35%)	8 (89%)
9	4 (15%)	2 (50%)

wurde erstmals Aufgaben zur Spielwiese "Boolesche Mengenlehre" gestellt. Zur Vorbereitung wurden in der Vorlesung einerseits ein Fallunterscheidungsbeweis der Aussage "Es sei n eine ganze Zahl. Zeige: Dann ist $n^2 + n$ gerade." mit Diproche vorgeführt, andererseits ein einfacher Mengengleichheitsbeweis unter Verwendung zweier Inklusionsbeweise vorgestellt.

Die vier Aufgaben zu Fallunterscheidungen waren folgende:

- Es sei x eine ganze Zahl. Zeige: Dann ist $x^2 - x$ gerade.
- Es sei x eine ganze Zahl. Zeige: Dann ist $x^3 + x$ gerade.
- Es sei z eine ganze Zahl. Zeige: Dann ist $z^2 + z + 1$ ungerade.
- Es sei z eine ganze Zahl. Zeige: Dann ist $4 * z^5 + 2 * z^3 + 5 * z^2 - z$ gerade.

Außerdem wurden folgende drei Aufgaben zur Booleschen Mengenlehre gestellt:

- Es seien A, B Mengen. Es gelte $(A \cap B) = (A \cup B)$. Zeige: Dann ist $A = B$.
- Es seien A, B Mengen. Es gelte $A \subseteq B$. Zeige: Dann folgt $(A \cup B) = B$.
- Es seien A, B, C Mengen. Es gelte $A \subseteq B$. Ferner sei $B \subseteq C$. Zeige: Dann folgt $(A \cap C) = A$.

Zu Blatt 9 gab es insgesamt 21 Abgaben. Die Bearbeitungshäufigkeit und -erfolge der Diproche-Beweisaufgaben sind in den folgenden Tabellen aufgeführt. Wiederum beziehen die Prozentangaben in der zweiten Spalte sich auf die Gesamtzahl (21) der Abgaben, die in den übrigen Spalten hingegen beziehen sich auf die Anzahl abgegebener Lösungen zur jeweiligen Teilaufgabe, nicht auf die Gesamtzahl der Abgaben. In einigen wenigen Fällen wurden zwar Lösungen zu

den Aufgaben abgegeben, aber offenbar kein Versuch unternommen, sie in der Diproche-Syntax zu formulieren; diese Abgaben wurden unter "nicht bearbeitet" gezählt.

Table 8.7: Bearbeitung der Diproche-Beweisaufgaben zur Zahlentheorie auf Blatt 9

Aufgabe	Bearbeitet	richtig	fast richtig	Fehler/Lücken
1	12 (57%)	6 (50%)	0	6 (50%)
2	12 (57%)	5 (42%)	0	7 (58%)
3	11 (52%)	6 (55%)	0	5 (45%)
4	11 (52%)	1 (9%)	1 (9%)	7 (64%)

Wie aus der Auflistung hervorgeht, wurden die ersten drei Aufgaben jeweils in etwa der Hälfte der Fälle, in denen sie überhaupt bearbeitet wurden, auch richtig bearbeitet. Auffällig ist der geringe Anteil an "fast richtigen" Bearbeitungen, was darauf hinweist, dass die Normierung durch das System jedenfalls bei den Studierenden, die sich näher mit der Aufgabe beschäftigt haben, keine wesentliche zusätzliche Hürde darstellte: Mit einer Ausnahme wurden die Lösungen entweder vom System als korrekt beurteilt oder sie enthielten Fehler, die auch eine menschliche Korrektur als solche hätte anmerken müssen. Die hauptsächliche Fehlerquelle waren fehlerhafte Termumformungen. Dazu gehörten u.a. die fehlerhafte Verwendung von Potenzgesetzen, so dass etwa $(2k)^5$ zu $2k^5$ oder $(2k)^2$ zu $2k^2$ umgeformt wurden, sowie Vorzeichenfehler beim Ausmultiplizieren und die distributive Verwendung der Exponentiation über die Addition in Schritten wie $(2k + 1)^2 = 4k^2 + 1$. Diese Fehler traten vermehrt bei dem vierten Aufgabenteil zutage, wo von niemand der Weg gewählt wurde, die Terme mit geraden Koeffizienten unverändert zu lassen, sondern stets im Fall, dass z ungerade ist, die Darstellung $(2 * k - 1)$ überall für z eingesetzt und der so entstehende Term ausgerechnet wurde; bei dieser dann sehr aufwändigen Umformungsarbeit kam es häufig zu einem oder mehreren der genannten Fehler, so dass hier die Erfolgsquote deutlich niedriger liegt als bei den drei anderen Teilaufgaben. Die fehlerhaften Umformungsschritte wurden vom System stets als solche erkannt und gemeldet; die Fehlschlussdiagnose für Termumformungen meldete hingegen nur selten einen bekannten Fehlertyp, obwohl insbesondere für die distributive Verwendung der Exponentiation durchaus Erkennungsregeln implementiert waren. Grund hierfür war, dass fehlerhafte Umformungsschritte häufig zusammen mit mehreren anderen Umformungen vorgenommen wurden, so dass die in der jeweiligen Regel vorgesehene Fehlerform nicht mehr vorlag. Nach dieser Erfahrung wurde die Fehlschlussdiagnose für Termumformungen so modifiziert, dass der

Fehler "distributive Verwendung der Exponentiation" immer dann erkannt wird, wenn eine fehlerhafte Gleichung der Form $(T_1 + T_2)^n = T_3 + T_4$ vorliegt, wobei T_1, T_2, T_3, T_4 beliebige Terme sind, die $T_1^n = T_3$ und $T_2^n = T_4$ erfüllen.

Das Vorliegen zahlreicher fehlerhafter Abgaben, die auch vom System als solche gemeldet wurden, weist darauf hin, dass die Studierenden Schwierigkeiten haben, Rückmeldungen des Systems so auf ihren Lösungstext zu beziehen, dass auf dieser Basis eine Verbesserung des Textes möglich ist. Diese Vermutung wird auch durch die Befragung der Studierenden bestätigt (s.u.). Hier zeigt sich wiederum, dass eine Meldung fehlerhafter Schritte allein nicht unbedingt zu einem Lerneffekt führen muss: Liegt keine Strategie vor, die Fehlermeldung zur Verbesserung der Lösung zu verwenden, wird die Fehlermeldung zwar bemerkt, hat aber womöglich keinen weiteren Lerneffekt zur Folge. Es ist zu hoffen, dass dieser Schwierigkeit (i) durch eine von ausführlicheren Erläuterungen begleitete automatische Fehlschlussdiagnose und (ii) durch eine vermehrte Interaktion zwischen TutorInnen und Studierenden, in der die Fehlermeldungen erläutert werden, begegnet werden kann.

Table 8.8: Bearbeitungen der Diproche-Beweisaufgaben zur Mengenlehre auf Blatt 9

Aufgabe	Bearbeitet	richtig	fast richtig	Fehler/Lücken
1	7 (33%)	3 (43%)	4 (57%)	0
2	7 (33%)	0	1 (14%)	6 (86%)
3	3 (14%)	1 (33%)	0	2 (67%)

Bei den Mengenlehre-Beweisaufgaben lagen sowohl Bearbeitungshäufigkeit als auch Bearbeitungserfolg deutlich niedriger als bei den Aufgaben zur elementaren Zahlentheorie. Auch lag hier der Anteil "fast richtiger" Lösungen deutlich höher. Eine Hauptschwierigkeit bei den "fast richtigen" Abgaben waren Meldungen nicht erreichter Beweisziele, die sich daraus ergaben, dass zwei in Diproche vorgesehene Ansätze zum Beweis von Teilmengenaussagen vermischt wurden: Eine Aussage der Form $A \subseteq B$ wird von der Zielverfolgung einerseits dann als auf Basis der Annahmen ϕ_1, ϕ_2,..., ϕ_n als bewiesen angesehen, wenn $A \subseteq B$ an einer Textstelle abgeleitet wird, an der alle offenen Annahmen zu $\{\phi_1, ..., \phi_n\}$ gehören, andererseits aber auch dann, wenn für eine freie Variable x die Aussage $x \in B$ an einer Textstelle abgeleitet wird, an der alle offenen Annahmen zu $\{\phi_1, ..., \phi_n, x \in A\}$ gehören. Wird dagegen $A \subseteq B$ noch im Gültigkeitsbereich der zusätzlichen Annahme $x \in A$ abgeleitet, gilt das Beweisziel nicht erreicht. Um diese Schwierigkeit zu vermeiden, hätte es einer systematischen Einführung in die Funktionsweise der Zielverfolgung sowie die Gültigkeitsbereiche von Annahmen

bedurft, die aus den eingangs genannten Gründen unterblieb. In zahlreichen Fällen wurde somit bei einer Lösung ein nicht erreichtes Beweisziel gemeldet, bei der es zur Korrektur nur einer Löschung der Endformel "$A \subseteq B$" bedurft hätte.

Table 8.9: Gegenüberstellung: (i) und (ii) werden als korrekt gemeldet, bei (iii) wird ein nicht erreichts Beweisziel gemeldet.

(i)	(ii)	(iii)
Beweis:	Beweis: Es gelte $x \in A$.	Beweis: Es gelte $x \in A$.
	...	...
...	...	...
Also gilt $A \subseteq B$. qed.	Also gilt $x \in B$. qed.	Also gilt $x \in B$. Damit folgt $A \subseteq B$. qed.

Ferner gab es hier auch vermehrt Abgaben, die Fehler oder Lücken aufwiesen. Hier waren die Hauptgründe zu große Deduktionsschritte sowie in der zweiten Teilaufgabe der Versuch, aus $x \in (A \cup B)$ ohne Verwendung der Zusatzannahme $A \subseteq B$ zu folgern, dass $x \in B$ ist.

Blatt 10 Das Aufgabenblatt 10 war über die Weihnachtspause zu bearbeiten, so dass eine deutlich längere Bearbeitsungszeit zur Verfügung stand als für die übrigen Blätter. Es wurden sechs Diproche-Beweisaufgaben zur elementaren Zahlentheorie gestellt sowie drei zur Booleschen Mengenlehre. Von den Aufgaben zur Zahlentheorie waren jeweils zwei mit einem direkten Beweis, einem Kontrapositionsbeweis und einer Fallunterscheidung zu lösen, wobei diesmal keine Hilfestellung zur Strategiewahl gegeben wurde. Ferner betrafen die dritte, fünfte und sechste Teilaufgabe erstmals den Begriff der Teilbarkeit. Diese Aufgaben sind insofern von besonderem Interesse, als den Studierenden zuvor keine Teilbarkeitsbeweise mit Diproche vorgeführt worden waren. Die Studierenden kannten also weder Diproche-spezifische Formulierungen zur Behandlung von Teilbarkeitsaussagen noch die entsprechende Formelsyntax. Wie gut diese Aufgaben bewältigt wurden, ist damit ein Indikator für die "Natürlichkeit" der Diproche-Sprache und der Diproche-Logik. Können auch Beweisaufgaben zu neuen Begriffen im Rahmen des Systems erfolgreich bearbeitet werden, ohne dass dabei entsprechende Beispieltexte zur Nachahmung zur Verfügung stehen, spricht das dafür, dass die Eingabesprache natürlich und damit intuitiv gewählt ist.

Um den Studierenden bei der Bedienung des Systems zusätzliche Unterstützung zu geben, wurde Anfang Januar eine zusätzliche, etwa einstündige abendliche Lehreinheit abgehalten, die für die Studierenden freiwillig war. Es beteiligten sich 114 Studierende, also etwa die Hälfte der GesamtteilnehmerInnenzahl der Vorlesung.

Die Beweisaufgaben zur elementaren Zahlentheorie waren folgende:

1. Es sei x eine ganze Zahl. Wir zeigen: Wenn $13 - 3 * x$ gerade ist, dann ist x ungerade.

2. Es sei n eine ganze Zahl. Zeige: Dann ist $4 * n + 5$ ungerade.

3. Es sei n eine ganze Zahl. Wir zeigen: 8 teilt $(2 * n - 1)^2 - 1$.

4. Es sei x eine ganze Zahl. Wir zeigen Wenn $2 - x$ ungerade ist, dann ist x ungerade.

5. Es sei a eine ganze Zahl. Zeige: Wenn a gerade ist, dann ist $3 * a^2$ durch 4 teilbar.

6. Es sei n eine ganze Zahl. Zeige: Wenn $3|(n - 1)$ dann folgt $3|(n^2 - 1)$.

Die Aufgaben zum Bereich "Boolesche Mengenlehre" waren folgende, wobei erstmals auch Aussagen zum kartesischen Produkt zu zeigen waren:

1. Es seien A, B, C Mengen. Wir zeigen: $((A \cap B) \times C) \subseteq (A \times (B \cup C))$.

2. Es seien A, B, C, D, E Mengen. Es gelte $A \subseteq D$. Ferner gelte $C \subseteq E$. Zeige: Dann gilt $(A \times (B \cap C)) \subseteq (D \times E)$.

3. Es seien A, B, C, D Mengen. Es gelte $A \subseteq B$. Ferner gelte $C \subseteq D$. Wir zeigen $(A \cap C) \subseteq (B \cap D)$.

Zu Blatt 10 wurden 27 Lösungen abgegeben. Die Bearbeitungshäufigkeit und -erfolge können folgenden Tabellen entnommen werden. Wiederum beziehen sich die Prozentzahlen in der Spalte "bearbeitet" auf die Gesamtzahl (27) der Abgaben, während die Prozentzahlen in den übrigen Spalten sich auf die Anzahl der Bearbeitungen der jeweiligen Teilaufgabe beziehen.

Mit Ausnahme von Teilaufgabe (6) lag die Bearbeitungshäufigkeit damit recht konstant bei etwa 40%. Bei den Aufgabenteilen (1) und (4) lag die Verwendung eines Kontrapositionsbeweises, bei den Aufgabenteile (2) und (3) ein Beweis durch Fallunterscheidung und bei den Aufgabenteilen (5) und (6) eine Behandlung durch direkte Beweise nahe. Ein Haupgrund für Fehler waren wiederum fehlerhafte Termumformungen wie Vorzeichenfehler oder fehlerhaftes Ausmultiplizieren; am häufigsten erfolgreich bearbeitet wurde Teilaufgabe (2), bei der solche Fehler am leichtesten zu vermeiden waren. Bei "fast richtigen" Bearbeitungen waren die Hauptgründe wiederum die schon oben besprochenen Formulierungsdetails bei Kontrapositionsbeweisen sowie ferner bei Fallunterscheidungsbeweisen, dass (i) ein Fall in dem Moment als abgeschlossen betrachtet wurde, in dem das Erreichen des Beweisziels "offensichtlich" war, ohne es noch einmal explizit zu erwähnen oder

Table 8.10: Bearbeitungen der Diproche-Beweisaufgaben zur Zahlentheorie auf Blatt 10

Aufgabe	Bearbeitet	richtig	fast richtig	Fehler/Lücken
1	11 (41%)	7 (64%)	3 (27%)	1 (9%)
2	11 (41%)	10 (91%)	1 (9%)	0
3	10 (37%)	5 (50%)	2 (20%)	3 (30%)
4	12 (44%)	6 (50%)	4 (33%)	2 (17%)
5	12 (44%)	8 (67%)	1 (8%)	3 (25%)
6	5 (19%)	0	3 (60%)	2 (40%)

(ii), dass nach Behandlung aller Fälle die jeweilige Behauptung nicht noch einmal außerhalb der Fallumgebungen als nun erreichtes Beweisziel erwähnt wurde. In beiden Fällen wurden nicht erreichte Beweisziele gemeldet. Ein dritter Grund war der Umgang mit Äquivalenzumformungen von Gleichungen in Aufgabenteil (6), auf den wir gleich noch näher eingehen werden.

Die Aufgabenteile (3) und (5) zur Teilbarkeit wurden, obwohl hierfür weder das systemspezifische Vokabular noch die entsprechende Notation eingeführt worden waren, ähnlich häufig und auch ähnlich häufig korrekt bearbeitet wie die übrigen Aufgabenteile; insbesondere ergab sich hier keine gegenüber den übrigen Aufgaben erhöhte Anzahl "fast richtiger" – also vom System aus Normierungsgründen als fehlerhaft gemeldeter – Lösungen, was dafür spricht, dass die CNL für diesen Beweistyp intuitiv konstruiert wurde. Ein anderes Bild ergibt sich bei Aufgabenteil (6), der zum einen deutlich seltener bearbeitet wurde als die übrigen Aufgabenteile, zum anderen nie mit einer vom System als korrekt akzeptierten Lösung und zum dritten mit dem größten Anteil "fast richtiger" Lösungen. Grund hierfür war, dass diese Aufgabe nicht nur Termumformungen, sondern Äquivalenzumformungen von Gleichungen erforderte, und die hierfür erforderliche Syntax zuvor nicht eingeführt worden war. So war ein naheliegender Lösungsansatz, aufgrund der Voraussetzung $3|(n-1)$ zunächst eine ganze Zahl k mit $(n-1) = 3k$ einzuführen und dann auf $n = 3k+1$ zu schließen. Dieser Schluss wurde aber von Diproche nur akzeptiert, wenn die zugehörige Gleichungsumformung $(n-1) = 3k \Leftrightarrow n = 3k+1$ zuvor explizit gemacht wurde. Da weder diese logische Normierung noch die Syntax für Äquivalenzumformungen von Gleichungen zuvor eingeführt worden waren, wurde dieser Weg aber nicht beschritten. Als Konsequenz wurde beschlossen, dem System die in den "fast richtigen" Lösungen verwendete Schlussregel hinzuzufügen, dass nämlich die arithmetische Termgleichheit $A = B$ ohne weitere Erläuterungen gefolgert werden kann, wenn $C = D$ zu den Voraussetzungen gehört und $A = B$ sich durch eine lineare Umformung (also eine der Form $\cdot r + s$ mit $r, s \in \mathbb{R}$)

ergibt. Von dieser modifizierten Variante wurden diese "fast richtigen" Lösungen als korrekt akzeptiert.

Table 8.11: Bearbeitungen der Diproche-Beweisaufgaben zur Mengenlehre auf Blatt 10

Aufgabe	Bearbeitet	richtig	fast richtig	Fehler/Lücken
1	12 (44%)	8 (67%)	2 (17%)	2 (17%)
2	12 (44%)	11 (92%)	1 (8%)	0
3	11 (41%)	9 (82%)	0	2 (18%)

Bei den Aufgaben zur Booleschen Mengenlehre lag die Bearbeitungshäufigkeit über alle drei Aufgabenteile nahezu konstant bei etwas über 40%, wie auch bei den meisten Beweisaufgaben zur elementaren Zahlentheorie. Alle Aufgabenteile, insbesondere auch diejenigen, die das kartesische Produkt als neuen Mengenoperator verwendeten, wurden überwiegend korrekt bearbeitet.

Blatt 11 In den Besprechungen zu Blatt 10 zeigte sich, dass viele Studierende noch z.T. erhebliche Schwierigkeiten im Umgang mit Mengen hatten. Aus diesem Grund wurden auf Blatt 11 statt der ursprünglich avisierten Vermittlung von Beweisstrukturen, die bisher nicht in Diproche-Beweisaufgaben aufgetreten waren, wie etwa Widerspruchsbeweisen, weitere vier Beweisaufgaben zur Mengenlehre gestellt.

1. Es seien A, B, C Mengen. Zeige: Dann gilt $(A/(B/C)) = ((A \cap C) \cup (A/B))$.
2. Es seien A, B, C Mengen. Es sei $A \subseteq B$. Wir zeigen: Dann gilt $(A \cap C) \subseteq ((A \cap B)/(-C))$.
3. Es seien A, B, C Mengen. Zeige: Dann gilt $(A \times (B \cap C)) = ((A \times B) \cap (A \times C))$.
4. Es seien A, B, C Mengen. Zeige: Dann gilt $(A \times (B \cup C)) = ((A \times B) \cup (A \times C))$.

Zu Blatt 11 gab es 28 Abgaben. Die Bearbeitungshäufigkeiten und -erfolge bei den Diproche-Beweisaufgaben ergeben sich aus der folgenden Tabelle; wiederum beziehen die Prozentangaben in der Spalte "bearbeitet" sich auf die Gesamtzahl von 28 Abgaben, während die übrigen Prozentzahlen sich auf die Anzahl der Bearbeitungen der jeweiligen Teilaufgabe beziehen.

Wie sich zeigt, lag die Bearbeitungshäufigkeit mit etwa 30% etwas niedriger als bei den Diproche-Beweisaufgaben auf Blatt 10. Ferner waren auch bei den

Table 8.12: Bearbeitungen zu den Diproche-Beweisaufgaben zur Mengenlehre auf Blatt 11

Aufgabe	Bearbeitet	richtig	fast richtig	Fehler/Lücken
1	9 (32%)	4 (44%)	2 (22%)	3 (33%)
2	10 (36%)	4 (40%)	3 (30%)	3 (30%)
3	7 (25%)	2 (29%)	1 (14%)	4 (57%)
4	8 (29%)	2 (25%)	0	6 (75%)

abgegebenen Bearbeitungen die Anteile korrekter Bearbeitungen geringer. Die Hauptursache für "fast richtige" Lösungen lag wiederum in Schwierigkeiten mit der Zielverfolgung. So wurde Aufgabenteil 1 meist mit einer Umformungskette von Mengentermen gelöst, also einem Ausdruck der Form $(A \setminus (B \setminus C)) = T_0 = T_1 = ... = T_n = ((A \cap C) \cup (A \setminus B))$, wobei der Boolesche Mengenterm T_{i+1} jeweils aus T_i durch eine elementare Umformungsregel hervorging. Damit das Beweisziel als erreicht gilt, war es indes erforderlich, das Endresultat der Gleichungskette, also $(A \setminus (B \setminus C)) = ((A \cap C) \cup (A \setminus B))$ noch einmal explizit zu erwähnen, was zuweilen unterlassen wurde. Lücken ergaben sich in den Teilen (3) und (4) unter anderem daraus, dass nur eine der beiden erforderlichen Inklusionen bewiesen wurde.

Überblick und Fazit

Wir erhalten damit die in der folgende Tabelle aufgeführten Gesamtzahlen bezüglich Bearbeitungshäufigkeit und -erfolg bei den verschiedenen Aufgabentypen; hierbei beziehen sich die Prozentangaben in der fünften Spalte auf die Gesamtzahl der Bearbeitungen des jeweiligen Aufgabentyps, während sich die in der sechsten Spalte auf die Gesamtzahl der 28 Abgabeteams beziehen.

Table 8.13: Bearbeitungen der Diproche-Aufgaben nach Aufgabentypen

Aufgabentyp	Aufgaben	Bearbeitet	Bearbeitungen pro Aufgabe	richtige Bearbeitungen	richtige Bearbeitungen pro Aufgabe
Mathediktate	6	143	23,8	141 (99%)	23,5 (84%)
Game of Def	19	292	15,4	281 (96%)	14,79 (53%)
Reformulate	6	108	18	106 (98%)	17,67 (63%)
Vereinfachen	4	77	19,25	66 (86%)	16,5 (59%)
Beweisaufgaben Zahlentheorie	14	184	13,14	104 (57%)	7,43 (27%)
Beweisaufgaben Mengenlehre	10	86	8,6	44 (51%)	4,4 (16%)

Bei einer Maximalzahl von 28 Abgaben (der Anzahl der Abgabeteams) wurden damit die Mathediktate im Schnitt am häufigsten bearbeitet, gefolgt von den "Vereinfachen"-Aufgaben, "Reformulate", dem "Game of Def" und den Beweisaufgaben zur Zahlentheorie, die durchschnittlich von etwa der Hälfte der

Abgabeteams bearbeitet wurden. Die Beweisaufgaben zur Mengenlehre bilden dabei deutlich das Schlusslicht in puncto Bearbeitungshäufigkeit: Hier wurde eine Aufgabe im Durchschnitt nur bei einem Drittel der Abgaben bearbeitet. Nimmt man den Anteil richtiger Bearbeitungen an der Gesamtzahl der abgegebenen Lösungen als Maß für den Schwierigkeitsgrad, fielen die Formalisierungsübungen – die Mathediktate sowie die "Reformulate"- und "Game of Def"-Aufgaben mit jeweils deutlich über 90% am leichtesten, während die Beweisaufgaben zur Mengenlehre mit knapp über 50% demgegenüber abfallen. Noch deutlicher werden die Ergebnisse, wenn man – die zu vermutende Tendenz, vom System als fehlerhaft gemeldete Lösungen nicht abzugeben berücksichtigend – die durchschnittliche Zahl vom System als richtig gemeldeter Lösungen pro Aufgabe eines bestimmten Typs betrachtet (Spalte 6): Hier fallen die Beweisaufgaben gegenüber allen übrigen Aufgabentypen deutlich ab: die durchschnittliche Beweisaufgabe zur Zahlentheorie wurde nur von etwas mehr als einem Viertel der Abgabeteams korrekt bearbeitet, während es bei den Beweisaufgaben zur Mengenlehre sogar nur knapp über einem Siebtel waren.

Neben konkreten technischen Änderungen wie der Integration zusätzlichen Vokabulars und weiterer Beweisregeln, die überwiegend noch im Verlauf des Semesters vorgenommen wurden, ergeben sich aus der Erfassung der Abgaben folgende Lehren in Bezug auf das System: Die Aufgaben zu den Systemkomponenten, die keine Freitexteingabe erfordern, wurden insgesamt von einer Mehrheit der Abgabeteams korrekt bearbeitet. Hier zeigten sich keine deutlichen Zusatzschwierigkeiten durch die syntaktische Normierung. Demgegenüber fielen die Beweisaufgaben, die mit der Prüfkomponente des Systems zu bearbeiten waren, deutlich schwerer. Eine Hauptursache für inhaltlich richtige, aber vom System als fehlerhaft gemeldete Abgaben war die starke implizite Normierung, die in der Zielverfolgung enthalten ist, insbesondere bei Kontrapositions- und Fallunterscheidungsbeweisen. Zugleich zeigen die vom System als korrekt bewerteten Abgaben, dass Studierende im Anfängerbereich grundsätzlich in der Lage sind, ohne eine aufwändigere Einführung Lösungstexte zu verfassen, die vom System als richtig akzeptiert werden. In dieser Hinsicht wurde gegenüber den weiter oben besprochenen Versuchen, automatische Beweisassistenten direkt in der Lehre einzusetzen, sicherlich ein Fortschritt erzielt: Es ist kaum anzunehmen, dass die Studierenden mit einer vergleichbar umfänglichen Einführung etwa in der Lage gewesen wären, Beweise in Coq- oder Mizar-Syntax zu verfassen. Die zahlreichen Abgaben, bei denen das System eine konkrete Textstelle als fehlerhaft markierte – vor allem, wenn es sich um Termumformungen handelte – weisen andererseits darauf hin, dass die Studierenden Schwierigkeiten damit haben, Fehlermeldungen zur Fehlerkorrektur zu nutzen.

Daher wäre beim Einsatz des Systems eine stärkere Interaktion zwischen Studierenden und Lehrkräften wünschenswert, um (i) stärker als bisher auf die Vermittlung der Eingabesyntax eingehen zu können, (ii) Hilfestellung beim Umgang mit Fehlermeldungen zu leisten und insbesondere (iii) deutlicher zu machen, welche Fehlermeldungen durch eine Anpassung der Eingabesyntax an die Normierung des Systems zu beheben sind und welche auf inhaltliche Fehler hinweisen, die auch von menschlichen KorrektorInnen moniert würden, und so (iv) ein stärkeres Vertrauen in die Relevanz der Rückmeldungen des Systems zu bewirken. Dazu dürfte es insbesondere hilfreich sein, Diproche-Aufgaben zumindest vereinzelt auch von Hand zu korrigiere, sowie in Lehrveranstaltungen gemeinsam mit den Studierenden mit dem System zu arbeiten. Schließlich sollten Studierende noch stärker als bisher dazu ermutigt werden, auch vom System als fehlerhaft gemeldete Lösungen abzugeben.

8.2 Zur Förderung von Kernkompetenzen

Als Vorüberlegung zu einer Untersuchung, wie der Einsatz von Diproche sich auf die Ausbildung mathematischer Kompetenzen auswirkt, betrachten wir hier zunächst, für welche mathematischen Kompetenzen eine förderliche Wirkung zu erwarten ist.

Hierzu orientieren wir uns an den offiziellen Vorgaben bezüglich der Bildungsziele des mathematischen Unterrichts, um zunächst einen Überblick über die relevanten mathematischen Kompetenzen zu gewinnen. Anschließend überlegen wir, welche davon durch Diproche voraussichtlich zu fördern sind.

Die von der Kultusministerkonferenz herausgegebenen "Bildungsstandards im Fach Mathematik für die Allgemeine Hochschulreife" [122] nennen auf S. 8 die folgenden mathematischen Kompetenzen als Grundlage der allgemeinen Hochschulreife:

- *K1: Mathematisch argumentieren*
- *K2: Probleme mathematisch lösen*
- *K3: Mathematisch modellieren*
- *K4: Mathematische Darstellungen verwenden*
- *K5: Mit Mathematik symbolisch/formal/technisch umgehen*
- *K6: Mathematisch kommunizieren*

[[122], S. 8]

In den vom Schleswig-Holsteinschen Ministerium für Schule und Berufsbildung herausgegebenen Fachanforderungen für das Fach Mathematik [78] untergliedern sich diese jeweils weiter in drei Tätigkeitsfelder. Ehe wir uns den Erfahrungen mit dem Einsatz zuwenden, wollen wir darauf eingehen, für welche dieser Kompetenzen von Diproche eine fördernde Wirkung zu erwarten ist und ggf. welche. Obwohl in einzelnen Punkten auch eine Auswirkung auf Kompetenzen in den Bereichen (K2), (K3) und (K4) zu erhoffen wäre, beschränken wir uns dabei auf die Bereiche (K1) ("mathematisch argumentieren"), (K5) ("mit Mathematik symbolisch/formal/technisch umgehen") und (K6) ("Mathematisch kommunizieren"), auf deren Förderung das System dem Entwurf nach als System zur automatischen Prüfung sprachlicher Darstellungen von mathematischen Argumenten unmittelbar ausgelegt ist. Wir schränken uns weiter auf diejenigen

Einzelkompetenzen ein, die anhand einer schriftlichen Lösung einer Beweisaufgabe erfasst werden können.

Im Hinblick auf (K1) liegt es inbesondere in folgenden der in [78] genannten Unterpunkte nahe, eine Wirksamkeit von Diproche zu vermuten:

1. "sammeln Argumente und präzisieren Vermutungen mithilfe von Fachbegriffen unter Berücksichtigung der logischen Struktur" (K1, Unterpunkt 1.3, [78]). Das präzise Formulieren im Rahmen einer (normierten) Fachsprache ist offenbar erforderlich, um Lösungen verfassen zu können, die von Diproche akzeptiert werden. Insbesondere durch Aspekte wie die durch die Absatzstruktur codierten Geltungsbereiche von Annahmen (s.o.) ist zu erwarten, dass Studierende sich durch die regelmäßige Verwendung des Systems in erhöhtem Maß über die logische Struktur von Argumenten Gedanken machen und so für solche Aspekte stärker sensibilisiert sind. In schriftlichen Ausarbeitungen sollte sich das in folgenden Effekten zeigen:

   - Mehr sprachliche Formulierungen (statt etwa reiner Folgen von Formeln und Pfeilen).
   - Mehr Strukturmarker auf der Makroebene, etwa durch explizite Erwähnung von Teilbeweiszielen sowie die Verwendung von – ggf. geschachtelten – Beweisanfangs- und Endmarkern.
   - Mehr Formulierungen, die den Status einer Aussage – etwa als Ziel, Annahme oder Folgerung – anzeigen sowie
   - weniger Verwechselungen von Ziel, Annahme und Behauptung, insbesondere weniger "zirkuläre" Argumente, bei denen das Beweisziel im Beweis verwendet wird.

2. "nutzen mathematische Regeln oder Sätze und sachlogische Argumente für Begründungen" (K1, Unterpunkt 2.2, [78]) Auch diese Kompetenz ist durch den regelbasierten Aufbau von Diproche offenbar erforderlich, um akzeptierte Diproche-Lösungen zu schreiben. Allerdings erfordert Diproche i.A. keine explizite Nennung dieser Regeln. In Textdarstellungen sollte sich diese Kompetenz vor allem darin zeigen, dass argumentative Schritte – auch ohne die ausdrückliche Angabe einer Schluss- oder Umformungsregel – einem bekannten Schlussmuster zugeordnet werden können und insbesondere weniger logische "Sprünge" auftauchen. Hierzu wird also die Anzahl derjenigen Behauptungen im Beweis erfasst, die nach einfachen logischen Regeln aus an dieser Stelle verfügbaren Aussagen gefolgert werden können.

3. "verknüpfen Argumente zu Argumentationsketten" (K1, Unterpunkt 2.3, [78]) Diese Kompetenz sollte sich darin zeigen, dass längere argumentierende

Texte geschrieben werden anstelle non-verbaler Begründungsstrategien wie Bildern (Venn-Diagramme) oder Wahrheitstafeln und ferner darin, dass Aufgaben, die solche Lösungen erfordern, vermehrt überhaupt bearbeitet werden.

4. "nutzen verschiedene Argumentationsstrategien" (K1, Unterpunkt 2.4, [78]) Eine Reihe solcher Strategien, etwa direkter und indirekter Beweis, Kontrapositionsbeweis, Fallunterscheidungen etc. werden durch Diproche explizit unterstützt. Zudem steht zumindest im Bereich der Aussagenlogik der Tippgeber zur Verfügung, der auf Anfrage einen Hinweis auf diese Argumentationsmuster liefert. In Beweistexten sollte sich eine Steigerung in diesem Bereich dadurch zeigen, dass solche Strategien vermehrt erkennbar zum Einsatz kommen.

5. "berücksichtigen vermehrt logische Strukturen" (K1, Unterpunkt 2.5, [78]) Dies sollte sich in Beweistexten vor allem durch die Wahl von für Aussagen des jeweiligen Typs geeigneten Beweisstrategien äußern, etwa darin, dass Äquivalenzen in zwei Implikationen aufgespalten werden oder dass Beweise von Implikationen dadurch versucht werden, dass das Antezedens angenommen und dann auf das Sukzedens hingearbeitet wird.

Im Hinblick auf die Unterpunkte 2.6 ("erklären vorgegebene Argumentationen und mathematische Beweise") sowie die dritte Phase des "Beurteilens" von Beweisen ist zwar ebenfalls auf eine förderliche Wirkung von Diproche zu hoffen, insbesondere durch Aufgaben, in denen die Ausgabe des Systems zu antizipieren und zu erläutern ist. Solche Aufgaben wurden jedoch einerseits im Verlauf des Semesters nicht gestellt; ferner sind Kompetenzen im Beurteilen von Beweistexten anhand einer schriftlichen Bearbeitung einer Beweisaufgabe kaum zu erfassen. Daher beschränken wir uns darauf, die Entwicklung im Hinblick auf diese Punkte durch die Befragung der Studierenden (s.u.) zu untersuchen.[12]

Bezüglich (K5) und (K6) halten wir ferner die folgenden Unterpunkte fest:

1. "nutzen symbolische und formale Schreibweisen" (K5, Unterpunkt 1.1, [78])

2. "verwenden Variablen, Terme und Gleichungen zum Beschreiben von Sachverhalten" (K5, Unterpunkt 1.2, [78])

3. "verwenden die Fachsprache und fachspezifische Notationen auf angemessenem Niveau" (K6, Unterpunkt 2.2, [78])

[12] Siehe dazu insbesondere Frage 4, in der die Aussage "Durch Diproche habe ich etwas darüber gelernt, wie man einen Beweistext hinsichtlich seiner Korrektheit beurteilt." zu bewerten war.

Für die Darstellung mathematischer Beweise sollten folgende Effekte zu erwarten sein:

- Grammatisch und logisch korrekte Verwendung formaler Ausdrücke im Rahmen von Beweistexten.
- Weniger syntaktische Fehler im Gebrauch von formalen Ausdrücken.
- Korrekter Gebrauch von Variablen; insbesondere weniger Typenfehler durch Verwendung nicht eingeführter Variablen oder falsche Verwendung von Variablen (etwa die Verknüpfung von Aussagen durch mengentheoretische Operatoren).

8.3 Rückmeldungen von Seiten der Studierenden

Um das System aus der Perspektive der Studierenden zu evaluieren, wurden drei Fragebögen entworfen, die die Prüfkomponente, die Mathediktate und das "Game of Def" betrafen.

Dabei sollten insbesondere folgende Punkte erfasst werden:

1. Bedienbarkeit des Systems
2. Art und Umfang der Nutzung des Systems
3. Wirkung des Systems auf das Verständnis (Beweise lesen, Formeln interpretieren...)
4. Wirkung des Systems auf eigene performative Kompetenzen (Beweise führen, Formeln schreiben...)
5. Wirkung des Systems auf Motivation und Selbstbewußtsein.
6. Einbindung des Systems in den Lehrbetrieb

Als Grundlage für die Erstellung unserer Fragebögen dienten z.T. Fragen, die im Rahmen einer ähnlichen Erhebung zu Lurch (s.o.) gestellt wurden, siehe Carter und Monks [42] sowie die Fragebögen zur Vorlesungsevaluation an der Europa-Universität Flensburg; wir danken ferner Hinrich Lorenzen und Michael Schmitz für hilfreiche Rückmeldungen zu früheren Versionen der Fragebögen. Dabei waren jeweils einige der Fragen mit einer Freitexteingabe zu beantworten, während in den übrigen nach einer Einordnung auf einer Skala von 1 (stimme überhaupt nicht zu) bis 6 (stimme voll zu) gefragt war. Die Freitextantworten wurden nach inhaltlicher Ähnlichkeit zusammengefasst und deren Häufigkeit dann quantitativ erfasst, wobei eine Antwort zu mehreren inhaltlichen Aspekten gehören konnte. Wir diskutieren nun der Reihe nach die Evalution der Prüfkomponente, der Mathediktate und des Game of Def. Eingeklammerte Zahlen oder Zahlbereiche in der Diskussion der Auswertung beziehen sich im folgenden auf die Antwortoptionen; so bedeutet etwa (1-3), dass eine der Antworten 1 (völlige Ablehnung), 2 oder 3 gegeben wurde. Mit "eher/tendenziell bejahende/positive" Antworten bezeichnen wir solche, die im Bereich 4-6 liegen; entsprechend sind "eher/tendenziell verneinende/negative" Antworten solche im Bereich 1-3.

8.3.1 Studentische Rückmeldungen zur Prüfkomponente

Der Fragebogen bestand aus 28 Bewertungsfragen mit den Antwortoptionen 1 bis 6 (s.o.) sowie vier Freitextfragen. Dem Fragebogen war folgender Erläuterungstext

vorangestellt, der klarstellte, dass hier lediglich die Beweisprüfkomponente des Systems beurteilt werden sollte; so sollte vermieden werden, dass die Fragen auf Erfahrungen aus dem Umgang mit anderen Systemkomponenten wie den Mathediktaten oder dem "Game of Def" bezogen wurden:

> Die folgenden Fragen beziehen sich auf die Beweisprüfer-Komponente des Diproche-Systems, NICHT auf andere Systemkomponenten wie die Mathediktate, das "Game of Def" oder "Vereinfachen". Eine Beispielaufgabe, die in den Übungen im HS 2020/2021 zu dieser Komponente gestellt wurde, war folgende:
> "Es sei x eine ganze Zahl. Zeige: Wenn x gerade ist, dann ist 2-3*x gerade."

Der Fragebogen wurde von 127 TeilnehmerInnen der Vorlesung ausgefüllt, also knapp 56% der 228 VorlesungsteilnehmerInnen.

Die Antworten auf die Freitextfragen (29)-(31) lassen sich der Fragestellung nach nicht apriori einer der oben genannten Kategorien (1)-(6) zuordnen. Wir beginnen daher mit der Auswertung der Freitextfragen, um die Ergebnisse in die separate Behandlung der Kategorien (1)-(6) einfließen lassen zu können.

Die Freitextfragen

Die Freitextfragen zur Prüfkomponente waren folgende[13] (Nummer der Frage im Originalfragebogen in Klammern):

1. Besonders gut gefällt mir, dass... (29)
2. Nicht so gut gefällt mir, dass... (30)
3. Konkret habe ich folgende Verbesserungsvorschläge: (31)

Zur Frage 1 wurden 49 Antworten abgegeben. Davon ließen sich 43 einem oder mehreren der folgenden inhaltlichen Punkte zuordnen:

1. Diproche gibt ein unmittelbares Feedback. (UF)
2. Diproche zeigt fehlerhafte Stellen im Beweis auf. (FS)
3. Diproche ermöglicht ein interaktives Arbeiten am Beweis unter Einbindung der Korrektur. (IA)

[13]Diese Fragen stammen aus den Fragebögen zur studentischen Evaluation von Lehrveranstaltungen an der Europa-Universität Flensburg.

4. Diproche ermöglicht eine Selbstkorrektur, ohne fremde Hilfe. (SK)

5. Diproche erhöht die Sicherheit bezüglich der Korrektheit der eigenen Beweistexte (ES)

6. Diproche hat einen Lerneffekt bezüglich der mathematischen Fachsprache. (LSp)

7. Diproche hat einen Lerneffekt im Hinblick auf Strategien und Strukturen mathematischer Beweise. (LSt)

8. Diproche hat einen positiven (steigernden) Einfluss auf die Motivation, an Beweisen zu arbeiten. (SM)

Bei den 6 übrigen Antworten handelte es sich zum einen Teil um Stellungnahmen zur Weiterentwicklung des Systems im Verlauf des Semesters, zum Teil um allgemein gehaltene (positive) Rückmeldungen zur Sinnhaftigkeit des Systems. Da sie für die Ziele dieser Auswertung kaum relevant zu sein scheinen, wurden sie ausgelassen.

Die Häufigkeit des Auftretens der oben genannten Aspekte unter den ausgewerteten 43 Antworten ergibt sich aus Abbildung 8.11; die Prozentzahlen wurden dabei auf ganze Prozente gerundet; man beachte, dass eine Antwort mehrere Aspekte ansprechen konnte und diesen dann auch allen zugeordnet wurde

Table 8.14: Auswertung der Freitextantworten zu “”Besonders gut gefällt mir, dass...”

UF	FS	IA	SK	ES	LSp	LSt	SM
25 (58%)	7 (16%)	5 (12%)	7 (16%)	2 (5%)	3 (7%)	4 (9%)	2 (5%)

Zu Frage 2 wurden 57 Antworten abgegeben. Davon ließen sich 54 einem oder mehreren der folgenden inhaltlichen Punkte zuordnen:

1. Es ist schwierig, anhand der Diproche-Fehlermeldungen den Fehler im Text zu finden. (FF)

2. Die Diproche-Korrektur ist zu “penibel”, Texte können z.B. aufgrund von Tippfehlern oder falscher Interpunktion (fehlender Klammern) nicht verarbeitet werden. (KP)

3. Diproche hat einen negativen Einfluss auf die Motivation, an Beweisen zu arbeiten. (NM)

4. Die sprachliche Standardisierung erhöht die Schwierigkeit, als korrekt gemeldete Beweistexte zu verfassen. (SSt)

5. Die logische Standardisierung ("Kleinschrittigkeit") erhöht die Schwierigkeit, als korrekt gemeldete Beweistexte zu verfassen. (LSt)

6. Die Standardisierung erhöht die Schwierigkeit, als korrekt gemeldete Beweistexte zu erfassen. (St) [umfasst die beiden letzten Punkte]

7. Einen von Diproche akzeptierten Beweistext zu verfassen, ist zu zeitaufwändig. (ZZ)

8. Die von Diproche verwendete Norm bezüglich der Korrektheit von Beweistexten weicht von den in Vorlesung bzw. Übung verwendeten ab. (AN)

9. Die Erklärungen zum System im Rahmen der Lehrveranstaltung (Vorlesung/Übungen) waren unzureichend. (LU)

10. Die sprachliche und syntaktische Standardisierung von Diproche weicht von den in Vorlesungen bzw. Übungen verwendeten Normen ab. (VSSt)

11. Diproche leistet bei der Lösungssuche unzureichende Hilfestellung. (UH)

Dabei subsumiert St die Punkte SSt und LSt; eine Antwort wurde hier eingeordnet, wenn sie die Standardisierung betraf, sich aber nicht klar auf einen der Aspekte "Sprache" bzw. "Logik" bezog.

Die oben genannten Aspekte waren über die ausgewerteten 43 Antworten wie folgt verteilt (die Prozentzahlen wurden auf ganze Prozente gerundet; man beachte, dass eine Antwort mehrere Aspekte ansprechen konnte und diesen dann auch allen zugeordnet wurde):

Table 8.15: Auswertung der Freitextantworten zu "Nicht so gut gefällt mir, dass..."'

FF	KP	NM	SSt	LSt	St	ZZ	AN	LU	VSSt	UH
29 (67%)	12 (28%)	8 (19%)	9 (12%)	3 (7%)	3 (7%)	12 (28%)	1 (2%)	5 (12%)	4 (9%)	1 (2%)

Zu Frage 3 wurden 33 Antworten abgegeben. Alle ließen sich einem oder mehreren der folgenden inhaltlichen Punkte zuordnen:

1. Das Spektrum der von Diproche akzeptierten Zeichen und Formulierungen sollte erweitert werden. (SF)

2. Das Spektrum der von Diproche akzeptierten Schlussweisen sollte erweitert werden (SW)

3. Es sollte Möglichkeiten geben, sich von den Lehrveranstaltungen unabhängig über das System zu informieren.[14] (SI)

4. Fehlermeldungen sollten ausführlicher formuliert werden. (FA)

5. Fehlermeldungen sollten von Verbesserungsvorschlägen begleitet sein. (VV)

6. Es sollte eine Hinweisfunktion geben. (HF)

7. In den Veranstaltungen sollte mehr Beschäftigung mit dem System stattfinden. (VS)

8. Beweise in Lehrveranstaltungen und Diproche-Beweise sollten sich stärker ähneln.[15] (LB)

Diese Punkte wurden mit folgenden Häufigkeiten angegeben:

Table 8.16: Antworten zur Frage nach konkreten Verbesserungsvorschlägen

SF	SW	SI	FA	VV	HF	VS	LB
5 (15%)	2 (6%)	8 (24%)	9 (27%)	2 (6%)	8 (24%)	1 (3%)	2 (6%)

Bedienbarkeit des Systems

Zur Bedienbarkeit des Systems wurden folgende Fragen gestellt (die Nummer der Frage im Evaluationsbogen ist jeweils in Klammern hinter der Frage angegeben):

1. Einen Text bei Diproche einzugeben fällt mir leicht. (1)

2. Diproche-Beweise ähneln den Beweisen, die ich in der Vorlesung und den Übungsgruppen kennen lerne. (2)

[14] Hier wurden insbesondere genannt: Eine Auflistung der Fehlermeldungen mit Erläuterung, die Verfügbarkeit von Beispiel- und Musterlösungen, sowie eine Liste der akzeptierten Formulierungen.

[15] Interessanterweise gab es hier sowohl Rückmeldungen, die Diproche als im Vergleich zu den in der Vorlesung verlangten Beweisen zu restriktiv als auch solche, die es als zu permissiv beschrieben: Hier zeigt sich erneut, dass mathematische Lehrveranstaltungen ein Interferenzfeld von Korrektheitsnormen sind, in dem Studierende sich zurechtfinden müssen und in das ein System wie Diproche eine weitere Norm einführt.

3. Bei Fehlermeldungen von Diproche kann ich oft auch einen Fehler in meinem Beweistext erkennen. (10)

4. Die Diproche-Rückmeldungen helfen mir dabei, einen Beweistext zu verbessern. (17)

5. Diproche braucht manchmal zu lange, um einen Beweis zu prüfen. (18)

Hierbei beziehen sich (1) und (2) auf die von Diproche vorgenommene Normierung der Sprache und sollen erfassen, inwieweit Studierende in der Lage sind, Diproche-Texte zu verfassen und in den Diproche-Texten die üblichen Beweisdarstellungen wiederzuerkennen (und umgekehrt). Die Fragen (3), (4) und (5) beziehen sich auf die Verständlichkeit und Nützlichkeit der Rückmeldungen des Systems; insbesondere erfasst Frage (5), ob die Prüfzeiten des Systems in einer nennenswerten Anzahl von Fällen so hoch sind, dass sie als störend empfunden werden.

Die Anzahl der Antworten, die Verteilung der Antworten sowie der Durchschnittswert ergeben sich aus folgender Tabelle:

Table 8.17: Einschätzungen zur Bedienbarkeit der Prüfkomponente

Frage	Antworten	1	2	3	4	5	6	∅
1	125	13 (10%)	40 (32%)	26 (21%)	29 (23%)	12 (10%)	5 (4%)	3,0
2	125	3 (2%)	13 (10%)	31 (25%)	35 (28%)	35 (28%)	8 (6%)	3,9
3	124	25 (20%)	47 (38%)	28 (23%)	15 (12%)	8 (6%)	1 (1%)	2,5
4	124	15 (12%)	36 (29%)	33 (27%)	31 (25%)	7 (6%)	2 (2%)	2,9
5	125	36 (29%)	40 (32%)	22 (18%)	15 (12%)	10 (8%)	2 (2%)	2,4

Betrachtet man eine Antwort von 3 oder kleiner als “eher verneinend” und eine von 4 oder höher als “eher bejahend”, erkennt man, dass eine Mehrheit der Befragten (63%) angibt, die Eingabe von Diproche-Texten falle ihnen nicht leicht. Dies kann nun zum einen an Schwierigkeiten mit der Formulierung von Beweistexten liegen, zum anderen aber auch daran, dass die Normierungen durch das System zusätzliche Schwierigkeiten aufgeworfen haben. Dass ein Aspekt der Normierung (wie etwa der sprachliche, logische, strukturelle oder syntaktische) die Abfassung von Diproche-Texten erschwert hat, wurde von immerhin 26% der abgegebenen Freitextantworten erwähnt, wobei sich allerdings nur 12% explizit auf die sprachliche Normierung bezogen. Andererseits empfindet eine ebenfalls deutliche Mehrheit (62%) die Diproche-Beweise als den aus Vorlesung und Übungen bekannten ähnlich; dies deckt sich auch mit den Freitextantworten, bei denen nur 9% eine Abweichung der sprachlichen Standardisierung von Diproche von der der Vorlesung bzw. Übungen monierten (siehe den Aspekt VSSt), während 6% sich eine stärkere Annäherung beider wünschten (Aspekt LB). Die sprachliche

Normierung scheint also durchaus eine deutliche zusätzliche Hürde gegenüber dem Schreiben von Beweistexten im Rahmen des regulären Übungsbetriebes darzustellen, wenn auch keine unüberwindliche. Dies deckt sich mit der Erfahrung aus den abgegebenen Übungsaufgaben, dass eine deutliche Mehrheit der Abgaben zu den ersten Diproche-Beweisaufgaben zur elementaren Zahlentheorie formal korrekt in der Eingabesprache von Diproche abgefasst waren.

Deutlich größere Probleme zeigen sich bei den Rückmeldungen: Lediglich 19% gaben an, bei Diproche-Fehlermeldungen auch Fehler in ihrem Beweistext erkennen zu können und nur etwa ein Drittel (33%) der Studierenden konnte die Diproche-Rückmeldungen zur Verbesserung eines Beweistextes nutzen. Auch dies findet in den Freitextantworten einen deutlichen Niederschlag: Die Schwierigkeit, anhand der Rückmeldungen die Fehler im Text zu finden, wurde in 67% der abgegebenen Antworten erwähnt (Aspekt FF); 33% der Antworten auf die Frage nach Verbesserungsvorschlägen betrafen die Fehlermeldungen (Aspekte FA und VV). Dies deckt sich ferner mit den Ergebnissen der Erfassung der Abgaben, bei denen Umformungsfehler zwar vom System gemeldet, aber nicht korrigiert wurden. Dennoch wurde in den Freitextantworten das Aufzeigen fehlerhafter Stellen (Aspekt FS) in 16%, die Möglichkeit des interaktiven Arbeitens unter Verwendung der Korrektur (Aspekt IA) in 12% sowie die Ermöglichung einer Selbstkorrektur (Aspekt SK) in 16% der Antworten positiv hervorgehoben. Ferner sieht eine deutlich Mehrheit von 79% keine größeren Schwierigkeiten mit den Laufzeiten des Systems.

Umfang und Art der Nutzung

Zu Umfang und Art der Nutzung wurden folgende Fragen gestellt (die Nummer der Frage im Evaluationsbogen ist jeweils in Klammern hinter der Frage angegeben):

1. Durch den Einsatz von Diproche habe ich mich länger mit Beweisaufgaben beschäftigt, als ich es bei den gleichen Aufgabenstellungen ohne Verwendung des Systems getan hätte. (7)

2. Ich habe mich auch über die gestellten Übungsaufgaben hinaus mit Aufgaben im Rahmen von Diproche beschäftigt. (16)

3. Auch Beweise, die ich für richtig halte, gebe ich manchmal in Diproche ein, weil es mir Freude macht. (24)

4. Bitte schätzen Sie grob, wieviel Zeit Sie pro Woche im Durchschnitt mit Diproche-Aufgaben verbracht haben. (32, Freitextfrage)

Die Anzahl der Antworten, die Verteilung der Antworten sowie der Durchschnittswert ergeben sich aus folgender Tabelle:

Table 8.18: Antworten zu Fragen zu Umfang und Art der Nutzung der Prüfkomponente

Frage	Antworten	1	2	3	4	5	6	Durchschnitt
1	125	12 (10%)	9 (7%)	13 (10%)	17 (14%)	34 (27%)	40 (32%)	4,4
2	124	57 (46%)	33 (27%)	20 (16%)	8 (6%)	4 (3%)	2 (2%)	2,0
3	124	76 (61%)	24 (19%)	12 (10%)	7 (6%)	4 (3%)	1 (1%)	1,7

Die durchschnittliche wöchentliche Beschäftigungszeit wurde als Freitexteingabe abgefragt, um nicht durch eine zu grobe Einteilung möglicherweise in den Daten auftauchende Unterschiede zu verdecken. Tatsächlich erwies sich diese Befürchtung als unbegründet: Die Mehrzahl der Angaben gab ein ganzzahliges Stundenintervall oder ein Vielfaches von 30 Minuten an, der sich entsprechend einordnen lief. Die Angaben variierten von "2 Minuten" als kleinstem zu 6 Stunden als höchstem Wert. Bei der Auswertung wurden präzise Stundeneingaben dem Intervall zugeordnet, das mit dieser Angabe endete, d.h. "2 Stunden" wurde etwa bei "$1-2$ Stunden" gezählt. Zwölf der 87 Angaben bestanden lediglich aus einer einstelligen Zahl zwischen 1 und 4 ohne Angabe einer Einheit; diese wurden als Stunden interpretiert. Eine Angabe war zu unpräzise, um sie einzuordnen. Bezüglich der durchschnittlichen wöchentlichen Beschäftigungszeit wurden somit 86 verwertbare Angaben gemacht, aus denen sich folgende Verteilung ergibt (Prozentzahlen auf ganzzahlige Prozentangaben gerundet):

Table 8.19: Angaben zu durchschnittlichen Beschäftigungszeiten mit dem Diproche-System pro Woche

Stunden	$0-1$	1-2	2-3	3-4	4-5	>5
Anteil	19 (22%)	31 (36%)	26 (30%)	5 (6%)	4 (5%)	1 (1%)

Nimmt man jeweils die Mittle eines Zeitintervalls als Annäherung an den Durchschnittswert, so ergibt sich damit eine ungefähre durchschnittliche wöchentliche Bearbeitungszeit von $\frac{19\cdot 0{,}5+31\cdot 1{,}5+26\cdot 2{,}5+5\cdot 3{,}5+4\cdot 4{,}5+1\cdot 5}{86} \approx 1,9$, also knapp 2 Stunden; legt man bei der Berechnung des Durchschnitts jeweils die obere Intervallgrenze zugrunde, ergibt sich ein etwas größerer Wert von $2,4$, also etwas mehr als zweieinhalb Stunden. Für die vier Wochen, in denen Beweisaufgaben zu Diproche gestellt wurden, ergibt sich somit eine durchschnittliche Gesamtarbeitszeit zwischen 8 und 10 Stunden. Bei den 28 in dieser Zeit gestellten Diproche-Beweisaufgaben entfällt somit auf eine Beweisaufgabe eine Beschäftigungsdauer zwischen 17 und 21 Minuten.[16]

[16]Dieser Durchschnittswert ist allerdings hinsichtlich der faktischen Verteilung zur bedingt

Eine große Mehrheit von 73% der TeilnehmerInnen der Umfrage gab an, durch den Einsatz von Diproche mehr Zeit mit Beweisaufgaben verbracht zu haben als sie es sonst getan hätten (Antwort im Bereich $4-6$). Dagegen gaben nur 11% eine überwiegend positive Antwort ($4-6$) auf die Frage, ob sie über den Übungsbetrieb hinaus mit dem System gearbeitet haben; fast die Hälfte (46%) beantworteten die Frage mit "trifft überhaupt nicht zu" (1). Noch deutlicher ist das Ergebnis bei der Frage, ob für richtig erachtete Beweise mit Diproche überprüft wurden, weil die Eingabe und das Prüfenlassen Freude machen: Hier gaben lediglich 9% eine überwiegend positive Antwort (4-6), während 61% sie mit "trifft überhaupt nicht zu" beantworteten.

Insgesamt kann somit eine längere Beschäftigungsdauer mit Beweisaufgaben durch den Einsatz des Systems als bestätigt angesehen werden; selbstständige Nutzungen des Systems, die über die im Übungsbetrieb gestellten Aufgaben hinausgehen, waren jedoch eher die Ausnahme.

Wirkung des Systems auf das Verständnis von Beweisen

Zur Wirkung auf das Verständnis von Beweisen wurden folgende Fragen gestellt (die Nummer der Frage im Evaluationsbogen ist jeweils in Klammern hinter der Frage angegeben):

1. Ich verstehe den Unterschied zwischen Logik- und Typenfehlern. (15)

2. Die Arbeit mit Diproche hilft mir dabei, besser zu verstehen, wie Beweise funktionieren.[17] (20)

3. Die Arbeit mit Diproche hilft mir dabei, Beweise, die in der Vorlesung, den Übungen etc. vorkommen, besser zu verstehen. (21)

Die Anzahl der Antworten, die Verteilung der Antworten sowie der Durchschnittswert ergeben sich aus folgender Tabelle:

Eine Mehrheit von 57% der UmfrageteilnehmerInnen gab eine überwiegend bejahende Antwort auf die Frage, ob sie den Unterschied zwischen Logik- und Typenfehlern verstehen. Die Frage, ob ihnen Diproche beim Verständnis der Funktionsweise von Beweisen hilft, wurde dagegen mehrheitlich von 60% der

aussagekräftig: Wie die Erfassung der Abgaben oben zeigt, kam es häufig vor, dass Teilaufgaben ausgelassen wurden. Andererseits kann nicht ausgeschlossen werden, dass auch in diesen Fällen eine Auseinandersetzung mit der Aufgabe stattfand, die aber eben nicht zur Abgabe einer Lösung führte.

[17] Vgl. für diese und die folgende Frage Carter und Monks [49], S. 8 in der Evaluation von Lurch, wo in einer Freitextantwort angemerkt wurde, Lurch habe dabei geholfen, zu verstehen, was ein Beweis ist.

Table 8.20: Antworten zu Fragen nach der Wirkung des Systems auf das Verständnis von Beweisen

Frage	Antworten	1	2	3	4	5	6	Durchschnitt
1	125	7 (6%)	13 (10%)	34 (27%)	26 (21%)	24 (19%)	21 (17%)	3,9
2	125	13 (10%)	31 (25%)	31 (25%)	39 (31%)	10 (8%)	1 (1%)	3,0
3	124	18 (15%)	40 (32%)	33 (27%)	26 (21%)	6 (5%)	1 (1%)	2,7

Befragen eher verneint $(1-3)$; nur 40% gaben hier eine eher bejahende Antwort $(4-6)$, wobei die deutliche Mehrheit unter diesen die Option 4 wählte; lediglich ein/e UmfrageteilnehmerIn gab auf diese Frage eine voll bejahende Antwort (6). Bei der Frage, ob die Arbeit mit Diproche dabei hilft, Beweistexte aus Vorlesung und Übung zu verstehen, fiel das Ergebnis noch deutlicher aus: 74% gaben hier eine eher verneinende Antwort.

Eine fördernde Wirkung der Verwendung des Systems bezüglich vorliegender Beweistexte scheint nach Selbsteinschätzung der Studierenden damit nur bei einer Minderheit vorzuliegen.

Wirkung des Systems auf die eigene Beweiskompetenz

Zur Wirkung auf die eigenen Beweiskompetenzen – also eigene Beweise zu finden, kritisch zu betrachten, ggf. zu korrigieren und darzustellen – wurden folgende Fragen gestellt (die Nummer der Frage im Evaluationsbogen ist jeweils in Klammern hinter der Frage angegeben):

1. Diproche hilft mir, Probleme in meinen Beweisansätzen zu erkennen. (3)
2. Durch Diproche habe ich etwas darüber gelernt, wie man einen Beweistext hinsichtlich seiner Korrektheit beurteilt. (4)
3. Diproche hat mir dabei geholfen, mich daran zu gewöhnen, Variablen vor ihrer Verwendung (mit Formulierungen wie “Sei x eine ganze Zahl.” o.ä.) einzuführen. (5)
4. Diproche hat mir geholfen, falsche Schlussweisen zu erkennen und zu vermeiden. (6)
5. Die Arbeit mit Diproche hilft mir dabei, die mathematische Fachsprache zu lernen. (22)
6. Die Arbeit mit Diproche hilft mir, zu lernen, wie man Beweistexte strukturieren kann. (23)
7. Ich habe durch die Verwendung von Diproche viel gelernt. (25)

8. Um Diproche-Aufgaben zu lösen, muss man die zu erwerbenden Kompetenzen bereits besitzen; daher ist Diproche nutzlos. (26)

Die Anzahl der Antworten, die Verteilung der Antworten sowie der Durchschnittswert ergeben sich aus folgender Tabelle:

Table 8.21: Antworten zur Wirkung der Prüfkomponente auf Beweiskompetenzen

Frage	Antworten	1	2	3	4	5	6	Durchschnitt
1	125	27 (22%)	33 (26%)	30 (24%)	24 (19%)	10 (8%)	1 (1%)	2,7
2	124	14 (11%)	21 (17%)	36 (29%)	37 (30%)	14 (11%)	2 (2%)	3,2
3	123	13 (11%)	13 (11%)	18 (15%)	24 (20%)	39 (32%)	16 (13%)	3,9
4	125	17 (14%)	36 (29%)	42 (34%)	21 (17%)	8 (6%)	1 (1%)	2,8
5	125	8 (6%)	14 (11%)	27 (22%)	47 (38%)	25 (20%)	4 (3%)	3,6
6	125	9 (7%)	11 (9%)	27 (22%)	40 (32%)	30 (24%)	8 (6%)	3,8
7	125	18 (14%)	33 (26%)	35 (28%)	30 (24%)	9 (7%)	0 (0%)	2,8
8	124	9 (7%)	31 (25%)	36 (29%)	28 (23%)	15 (12%)	5 (4%)	3,2

Wie aus den Antworten hervorgeht, wurde eine fördernde Wirkung des Systems auf die sprachlichen bzw. sprachnahen Aspekte des Beweisens (Verwendung der Fachsprache (5), strukturiertes Darstellen (6), Einführung von Variablen (3)) von den Befragten überwiegend bejaht. Bezüglich logischer Kompetenzen (Probleme in Ansätzen erkennen (1), Texte bezüglich Korrektheit beurteilen (2), Erkennen falscher Schlussweisen (4)) war das nur jeweils bei einer Minderheit der Befragten der Fall. Im Fazit gab etwa ein Drittel (31%) der Befragen an, durch den Einsatz des Systems "viel gelernt" zu haben, wobei die Option "trifft voll zu" von niemand gewählt wurde; zugleich konnten sich 60% tendenziell (1-3) nicht der Ansicht anschließen, dass das System nutzlos sei, weil man es nur bedienen könne, wenn man die fraglichen Kompetenzen bereits habe. Eine kompetenzfördernde Wirkung des Systems kann damit teilweise – in Bezug auf die Aspekte (3), (5), (6) bejaht werden, während sie in Bezug auf andere Aspekte – (1), (2), (5) – nur von einer Minderheit der Studierenden bestätigt wurde.

Wirkung des Systems auf Motivation und Selbstbewußtsein

Zur Wirkung auf Motivation und Selbstbewußtsein wurden folgende Fragen gestellt (die Nummer der Frage im Evaluationsbogen ist jeweils in Klammern hinter der Frage angegeben):[18]

1. Durch den Umgang mit Diproche bin ich im Umgang mit Beweisen sicherer und selbstbewusster geworden. (8)

[18]Die folgenden Fragen sind u.a. angeregt durch die in Carter und Monks [49] auf S. 8 zitierten Freitextantworten von Studierenden aus der Evaluation von Lurch zu diesem Aspekt.

2. Diproche hat mein Vertrauen in meine eigenen Beweistexte erhöht. (9)

3. Es ermutigt mich, wenn das System eine Lösung als korrekt meldet. (11)

4. Es ermutigt mich, wenn das System eine verbesserte Lösung als besser meldet als den vorherigen Versuch. (12)

5. Fehlermeldungen durch das Systen entmutigen mich. (13)

6. Die Arbeit mit Diproche macht mir Spaß. (14)

Die Anzahl der Antworten, die Verteilung der Antworten sowie der Durchschnittswert ergeben sich aus folgender Tabelle:

Table 8.22: Einschätzung zur Wirkung der Prüfkomponente auf Motivation und Selbstbewußtsein

Frage	Antworten	1	2	3	4	5	6	Durchschnitt
1	124	26 (21%)	43 (35%)	20 (16%)	28 (23%)	5 (4%)	2 (2%)	2,6
2	125	27 (22%)	41 (33%)	25 (20%)	26 (21%)	5 (4%)	1 (1%)	2,6
3	125	3 (2%)	6 (5%)	6 (5%)	20 (16%)	42 (34%)	48 (38%)	4,9
4	125	5 (4%)	10 (8%)	12 (10%)	44 (35%)	34 (27%)	20 (16%)	4,2
5	124	16 (13%)	30 (24%)	28 (23%)	17 (14%)	27 (22%)	6 (5%)	3,2
6	125	33 (26%)	42 (34%)	22 (18%)	20 (16%)	8 (6%)	0 (0%)	2,4

Etwa ein Drittel (35%) der Befragten gab demnach eine überwiegend bejahende (4-6) Antwort auf die Frage, ob der Umgang mit dem Diproche-System sie im Umgang mit Beweisen sicherer und selbstbewusster gemacht hat; ein erhöhtes Vertrauen in die eigenen Beweistexte hatte die Nutzung von Diproche bei etwa einem Viertel (26%) zur Folge. Eine deutliche Mehrheit von 88% empfand durch das System als korrekt gemeldete Lösungen als ermutigend, 38% antworteten hierauf sogar mit “stimme voll zu” (6); ebenso ist die Meldung, dass eine Lösung sich verbessert habe, für 78% tendenziell ermutigend (4-6). Deutlich weniger, aber immerhin 41%, werden durch Fehlermeldungen des Systems tendenziell entmutigt (4-6). Ein knappes Viertel (24%) der Befragten gab an, dass die Arbeit mit Diproche ihnen tendenziell Spaß mache (4-6); etwas mehr (26%) beantworteten diese Frage jedoch mit “trifft überhaupt nicht zu”, wohingegen die Option “trifft voll zu” von keiner und keinem der Befragten angegeben wurde.

Insgesamt zeigt sich damit, dass positive Meldungen durch das System bei einem größeren Anteil der Befragten einen ermutigenden Effekt haben als durch Fehlermeldungen entmutigt werden; eine Steigerung des Vertrauens in die eigenen Beweistexte und ein selbstbewussterer Umgang mit Beweisen zeigte sich nur bei einer Minderheit von etwa einem Viertel bzw. einem Drittel. Freude an der Arbeit mit dem System hat nur eine Minderheit der Befragten, wogegen eine klare Mehrheit dies deutlich verneint (1-2).

Einbindung in den Lehrbetrieb

Zur Einbindung in den Lehrbetrieb wurden folgende Fragen gestellt (die Nummer der Frage im Evaluationsbogen ist jeweils in Klammern hinter der Frage angegeben):[19]

1. Die Arbeit mit Diproche stellt eine sinnvolle Ergänzung zum Übungsbetrieb dar. (19)

2. Ich würde mir mehr Erklärungen zu Diproche in der Vorlesung wünschen. (27)

3. Ich würde mir mehr Beschäftigung mit Diproche in den Übungsgruppen bzw. den vorbereitenden Übungen wünschen. (28)

Die Anzahl der Antworten, die Verteilung der Antworten sowie der Durchschnittswert ergeben sich aus folgender Tabelle:

Table 8.23: Einschätzungen zur Einbindung in den Lehrbetrieb

Frage	Antworten	1	2	3	4	5	6	Durchschnitt
1	125	9 (7%)	22 (18%)	40 (32%)	34 (27%)	18 (14%)	2 (2%)	3,3
2	125	8 (6%)	22 (18%)	18 (14%)	33 (26%)	27 (22%)	17 (14%)	3,8
3	125	24 (19%)	22 (18%)	21 (17%)	28 (22%)	19 (15%)	11 (9%)	3,2

Damit betrachten 43% der Befragten die Prüfkomponente von Diproche tendenziell (4-6) als eine sinnvolle Ergänzung des Übungsbetriebes, wobei die eher ablehnenden Antworten überwiegend (56%) nur schwach ablehnend waren (3); allerdings wünscht sich eine klare Mehrheit von 62% mehr Erklärungen zum System in der Vorlesung (4-6); 43% wünschen sich tendenziell auch eine vermehrte Beschäftigung mit dem System im Rahmen des Übungsbetriebes; allerdings wird eine solche auch von immerhin 19% klar abgelehnt (1), wohingegen nur 6% sich ähnlich stark gegen mehr Erklärungen in der Vorlesung aussprechen.

In Verbindung mit den aus der Erfassung der Abgaben gewonnenen Beobachtungen läßt sich damit feststellen, dass das System stärker als bisher in den Lehrveranstaltungen erläutert und an diese angebunden werden sollte; es ist zu hoffen, dass hierdurch Schwierigkeiten in der Bedienung des Systems vermindert werden und damit auch der Anteil der Studierenden steigt, die das System als sinnvolle Ergänzung des Übungsbetriebes betrachten.

[19] Vgl. für diesen Aspekt die Frage aus der Evaluation von Lurch in Carter und Monks [49], S. 8, ob die Zeit, die mit der Vermittlung von Lurch verbracht wurde, besser auf die mathematischen Inhalte selbst hätte verwendet werden sollen.

Fazit

Zusammenfassend kann gesagt werden, dass die Bedienung des Systems den Studierenden noch Schwierigkeiten bereitete, insbesondere (i) bei der Eingabe aufgrund der sprachlichen und syntaktischen Normierung und (ii) in der Interpretation der Rückmeldungen. Ersteres läßt sich durch technische Verbesserungen des Systems nur bedingt vermeiden: Insbesondere gehört die Meldung etwa syntaktisch nicht wohlgeformter Ausdrücke als fehlerhaft ausdrücklich zu den Intentionen, mit denen das System entwickelt wurde; eine Lockerung der sprachlichen Normierung wiederum würde, wie bereits diskutiert, die Bedienung letztlich eher erschweren als erleichtern. Ein Lösungsansatz hierzu wäre, die Studierenden stärker an das System heranzuführen und ihnen zusätzlich Lehrmaterial an die Hand zu geben, das die Eingabesyntax erläutert. Auch in Bezug auf die Interpretation der Rückmeldungen wäre solches Material sicherlich hilfreich; hier könnte zusätzlich daran gearbeitet werden, die z.T. recht knappen und formalen Fehlermeldungen des Systems benutzerfreundlicher zu gestalten.

Aus der Befragung ergaben sich klare Hinweise darauf, dass der Einsatz des Systems zu einer vermehrten Beschäftigung mit Beweisaufgaben führt, auch wenn eine Nutzung über den Übungsbetrieb hinaus nur verhältnismäßig selten vorkam.

Eine kompetenzfördernde Wirkung scheint sich in Bezug auf Aspekte zu bestätigen, die sich eher auf die sprachliche Präsentation eines Beweises beziehen als auf logische bzw. inhaltliche Aspekte, weniger hingegen in Bezug etwa auf das korrekte Schlussfolgern oder die eigenständige Prüfung von Argumenten.

Die Möglichkeit, von dem System eine unmittelbare Rückmeldung zu einem Beweistext zu erhalten, wird von den Studierenden positiv wahrgenommen; ferner wirken positive Rückmeldungen durch das System mehrheitlich ermutigend. Andererseits gab nur eine Minderheit an, Freude an der Arbeit mit dem System zu haben.

Diese Ergebnisse steht durchaus im Einklang mit den im konzeptionellen Teil dieser Arbeit formulierten Erwartungen: Bezüglich Beweiskorrektheit ist Mathematik ein Interferenzfeld von Normen. Dieses Interferenzfeld stellt eine der Herausforderungen für Studienanfänger dar: Vorlesungsbeweise folgen anderen Normen als die korrigierenden Hilfskräfte, diese anderen als die TutorInnen (die untereinander ebenfalls verschiedene Vorstellungen haben können; ein noch anderer Punkt ist die Vorstellung von argumentativer Korrektheit, die der oder die Studierende schon mitbringt, sowie ggf. die von KomilitonInnen etc.). Das System stellt in diesem Interferenzfeld von Normen[20] eine weitere Norm dar, die mit den übrigen koordiniert werden muss. Diese Divergenz von Normen wird bei der Arbeit mit dem System deutlich bemerkt, tritt aber hier vielleicht nur expliziter zutage

[20]Vgl. etwa die Diskussion “sozimathematischer Normen” in [202], der den Terminus Yackel und Cobb [212] zurückführt.

als an anderen Stellen. Es ist zu vermuten, dass mehr explizit auf das System bezogene Interaktion mit den Lehrkräften den Studierenden dabei helfen würde, ein besseres Verständnis für den dem System inhärenten Korrektheitsbegriff zu erwerben, was dabei helfen sollte, diese Effekte abzufedern.

Eine solche verstärkte Interaktion wird auch von einer Mehrheit der Befragten tendenziell befürwortet, die sich eine solche allerdings eher für die Vorlesung als für die Übungsgruppen wünschen. Aus didaktischer Sicht wäre freilich zu erwarten, dass die unmittelbare Hilfestellung von Lehrkräften bei der Arbeit mit dem System, wie sie in den vorbereitenden Übungen bzw. den Übungsgruppen möglich wäre, einen stärkeren Lerneffekt zeitigen würde als zusätzliche Erklärungen bzw. Demonstrationen.

8.3.2 Studentische Rückmeldungen zu den Mathediktaten

Der Evaluationsfragebogen zu den Mathediktaten wurde von 107 Studierenden ausgefüllt, also etwa 47% der 228 VorlesungsteilnehmerInnen. Zur Verdeutlichung des Bezugs der Fragen stand dem Fragebogen folgende Bemerkung voran:

> Die folgenden Fragen beziehen sich auf das Übungsprogramm "Mathediktate". Bei den "Mathediktaten" ging es darum, natürlichsprachliche Aussagen in die Sprache der Aussagenlogik bzw. der Mengenlehre zu übersetzen. Eine der "Mathediktate"-Aufgaben im HS 2020/2021 war z.B. folgende:
> "Es bezeichne L die Aussage "Im Zoo gibt es Loewen.". Druecke die Aussage "Im Zoo gibt es keine Loewen" in der Sprache der Aussagenlogik aus."

Freitextantworten

Folgende Fragen waren mit Freitexteingaben zu beantworten:[21]

1. Besonders gut gefällt mir, dass...
2. Nicht so gut gefällt mir, dass...
3. Konkret habe ich folgende Verbesserungsvorschläge:

Zu Frage 1 wurden zwölf Antworten abgegeben. Von diesen ließen sich sieben unter einen oder mehrere der folgenden Punkte subsumieren:[22]

1. Die Mathediktate machen Spaß. (MS)
2. Das System liefert unmittelbares Feedback. (UF)
3. Die Aufgaben waren illustrativ. (AI)

Ferner wurde hervorgehoben, dass die Mathediktate geeignet sind, das Verständnis mathematischer Texte zu fördern und gegenüber den übrigen Aufgaben eine Abwechslung darstellen. Es ergab sich folgende Verteilung:

Aufgrund der geringen Anzahl auswertbarer Antworten verzichten wir auf einer weitere Analyse. Hervorzuheben ist allerdings, dass der Aspekt der unmittelbaren

[21] Die folgenden Fragen stammen aus den Fragebögen zur studentischen Evaluation von Lehrveranstaltungen an der Europa-Universität Flensburg.

[22] Zu den unten nicht erfassten Antworten gehörte u.a. die ebenso knappe wie erfreuliche Rückmeldung "alles".

Table 8.24: Verteilung der Antworten zu “Besonders gut gefällt mir, dass...”

MS	UF	AI
1	4	3

Rückmeldung mehrfach hervorgehoben wurde, wie auch bei der Prüfkomponente und dem Game of Def.

Zu Frage 2 wurden 14 Antworten abgegeben. Von diesen ließen sich fünf unter einen oder mehrere der folgenden Punkte subsumieren:[23]

1. Das System gibt keine Lösungshinweise. (LH)

2. Die Fehlermeldungen waren nicht hilfreich. (FH)

Es ergab sich folgende Verteilung:

Table 8.25: Verteilung der Antworten zu “Nicht so gut gefällt mir, dass...”

LH	FH
2	3

Da die Antworten – vermutlich auch aufgrund der recht geringen Anzahl – wenig inhaltliche Kohärenz aufwiesen, verzichten wir auch hier auf einer weitere Analyse. Die Fehlermeldungen, die ursprünglich nur zwischen “syntaktisch nicht korrekt”, “weder notwendig noch hinreichend, “notwendig, aber nicht hinreichend”, “hinreichend, aber nicht notwendig” und “korrekt” unterschieden, wurden nach den ersten Einsätzenf des Systems durch einen Gegenbeispielgenerator ergänzt, der jeweils eine Belegung der auftretenden Aussagevariablen lieferte, unter dem Eingabe und zu übersetzende Aussage verschiedene Wahrheitswerte aufwiesen. Damit scheinen hinsichtlich einer spezifischen Fehlermeldung die Möglichkeiten einer mit vertretbarem Aufwand implementierbaren Automatisierung ausgeschöpft zu sein. Hinweise könnten allenfalls als von der Lehrperson zuvor eingegebene und im Lösungsprozess abzufragende Hilfetexte umgesetzt werden. Denkbar wäre auch ein dialogisch arbeitendes System, dass z.B. erst einmal nach dem “Hauptjunktor”, dann nach den dadurch verknüpften Teilaussagen fragt und so allmählich an eine Lösung heranführt, siehe etwa das oben diskutierte System von Perikos et al. [161]. Die hierdurch bewirkte Verengung des Lösungsprozesses auf eine einzige Formulierungsmöglichkeit scheint aber wenig attraktiv (man beachte, dass die

[23]Zu den nicht erfassten Antworten gehörte u.a. die erfreuliche Stellungnahme, einer/s Studierenden, der oder die erklärte, ihr/ihm fielen “keine Kritikpunkte ein”.

Interdefinierbarkeit der Junktoren dafür sorgt, dass es eine in einem starken Sinn "falsche" Antwort auf die Frage nach dem Hauptjunktor nicht gibt).

Zu Frage 3 wurden lediglich sieben Antworten abgegeben. Diese rangierten vom Wunsch nach Hinweisen bzw. Lösungsvorschlägen (3×), genaueren Fehlermeldungen und mehr Erläuterungen in den Lehrveranstaltungen bis "mehr Aufgaben!" (jeweils einmal). Auch hier ist eine aussagekräftige Analyse durch die geringe Zahl der Antworten und deren geringe inhaltliche Kohärenz kaum sinnvoll.

Bedienbarkeit und Nutzung

Zur Evaluation von Bedienbarkeit und Nutzung der Mathediktate wurden folgende Fragen gestellt, wobei die Positionen im Originalfrageboen jeweils im Anschluss an die Frage in Klammern angegeben sind:

1. Rückmeldungen im Mathediktate-Modul helfen mir dabei, meine Lösung zu verbessern. (4)

2. Bei Fehlermeldungen im Mathediktate-Modul kann ich meistens erkennen, worin der Fehler in meiner Lösung besteht. (7)

3. Es kommt vor, dass richtige Lösungen als falsch gemeldet werden. (9)

4. Es kommt vor, dass falsche Lösungen als richtig gemeldet werden. (10)

5. Durch die Arbeit mit dem "Mathediktate"-Modul habe ich mich länger mit der Verwendung logischer Ausdrücke beschäftigt, als ich es bei den gleichen Aufgabenstellungen ohne Verwendung des Systems getan hätte. (13)

Die Anzahl der Antworten, die Verteilung der Antworten sowie der Durchschnittswert ergeben sich aus folgender Tabelle:

Table 8.26: Studentische Einschätzung zu Bedienbarkeit und Nutzung der "Mathediktate"

Frage	Antworten	1	2	3	4	5	6	Durchschnitt
1	106	5 (5%)	11 (10%)	17 (16%)	35 (33%)	27 (25%)	11 (10%)	4
2	106	12 (11%)	17 (16%)	31 (29%)	32 (30%)	12 (11%)	2 (2%)	3,2
3	102	29 (28%)	23 (23%)	19 (19%)	23 (23%)	8 (8%)	0 (0%)	2,6
4	103	54 (52%)	29 (28%)	10 (10%)	8 (8%)	1 (1%)	1 (1%)	1,8
5	105	5 (5%)	14 (13%)	14 (13%)	40 (38%)	19 (18%)	13 (12%)	3,9

Eine deutliche Mehrheit (68%) gab damit eine eher bejahende (4-6) Antwort auf die Frage, ob die Rückmeldungen des Mathediktate-Moduls bei einer Verbesserung

ihrer Lösung helfen, obwohl nur 43% angaben, bei Fehlermeldungen auch Fehler in ihrer Lösung erkennen zu können; immerhin fast ein Drittel (31%) waren der Ansicht, es komme vor, dass richtige Lösungen als falsch gemeldet werden, wohingegen nur 10% die umgekehrte Erfahrung berichteten. Ebenfalls 68% antworteten tendenziell bejahend auf die Frage, ob sie durch die Verwendung des Systems mehr Zeit mit den Aufgaben verbracht haben als ohne.

Die Mathediktate scheinen damit das Ziel zu erreichen, eine intensivere Beschäftigung mit der Syntax und Semantik logischer Ausdrücke anzuregen, indem sie fehlerhafte Lösungen für die Studierenden sofort erkennbar machen und so zur Suche nach Verbesserungen einer Lösung Anlass geben. Die Antworten auf die Fragen (2) und (3) legen indes nahe, dass die Interpretation der Rückmeldungen auch in diesem Bereich Schwierigkeiten bereitet.

Wirkung auf Kompetenzen

Zur Wirkung der Mathediktate auf Kompetenzen im Umgang mit logischer bzw. mengentheoretischer Notation wurden folgende Fragen gestellt, wobei die Positionen der Fragen im Fragebogen jeweils hinter den Fragen in Klammern stehen:

1. Ich habe durch den Gebrauch der “Mathediktate” viel gelernt. (11)
2. Die Arbeit an den Mathediktaten hat mir geholfen, die korrekte Bildung logischer Formeln zu lernen. (2)
3. Die Arbeit an den Mathediktaten hat mir geholfen, zu lernen, natürlichsprachliche Aussagen in aussagenlogischer bzw. mengentheoretischer Notation auszudrücken. (3)
4. Frühere Mathediktate-Aufgaben helfen mir oft dabei, Ansätze für spätere zu finden. (15)
5. Um Mathediktate-Aufgaben zu lösen, muss man die zu erwerbenden Kompetenzen bereits besitzen; daher sind die Mathediktate nutzlos. (16)

Zur Wirkung der Mathediktate auf das Verständnis logischer Ausdrücke wurde überdies folgende Frage gestellt:

6 Die Arbeit mit den “Mathediktaten” hat mir geholfen, logische Ausdrücke – etwa in der Vorlesung oder den Übungen – besser zu verstehen. (12)

Die Anzahl der Antworten, die Verteilung der Antworten sowie der Durchschnittswert ergeben sich aus folgender Tabelle:

Table 8.27: Studentische Einschätzung zur kompetenzfördernden Wirkung der "Mathediktate"

Frage	Antworten	1	2	3	4	5	6	Durchschnitt
1	104	3 (3%)	12 (12%)	25 (24%)	36 (35%)	24 (23%)	4 (4%)	3,8
2	105	4 (4%)	7 (7%)	8 (8%)	24 (23%)	44 (42%)	19 (18%)	4,5
3	106	3 (3%)	4 (4%)	12 (11%)	28 (26%)	40 (38%)	19 (18%)	4,5
4	105	4 (4%)	13 (12%)	22 (21%)	34 (32%)	24 (23%)	8 (8%)	3,8
5	105	18 (17%)	38 (36%)	25 (24%)	14 (13%)	9 (9%)	1 (1%)	2,6
6	105	6 (6%)	7 (7%)	21 (20%)	36 (34%)	27 (26%)	8 (8%)	3,9

Eine förderliche Wirkung der Mathediktate auf die Beherrschung logischer Syntax und Semantik wird damit von jeweils über 80% der Befragten tendenziell bejaht. Hilfreiche Zusammenhänge zwischen den Übungsaufgaben sahen tendenziell 63% der Befragten. Im Gesamtfazit gaben 62% der Befragten tendenziell an, durch die Mathediktate viel gelernt zu haben; der Ansicht, das Modul sei nutzlos, weil man es nur bedienen könne, wenn man die fraglichen Kompetenzen bereits besitze, schloss sich nur knapp ein Viertel (23%) der Befragten an, während 17% sie deutlich verneinten (1). Die Frage nach einer positiven Wirkung der Mathediktate auf das Verständnis logischer Ausdrücke wurde von 68% der Befragten tendenziell bejaht (4-6).

Diese Ergebnisse bestätigen die Erwartung, dass die Mathediktate sich als in der erwarteten Weise kompetenzfördernd auswirken.

Wirkung auf Motivation

Zur Wirkung der Mathediktate auf die Motivation wurden folgende Fragen gestellt, wobei die Positionen der Fragen im Fragebogen jeweils hinter den Fragen in Klammern stehen:

1. Die Arbeit mit den "Mathediktaten" macht mir Spaß. (1)
2. Es ermutigt mich, wenn das System eine Lösung als korrekt meldet. (5)
3. Fehlermeldungen durch das Systen entmutigen mich. (6)
4. Durch den Umgang mit den "Mathediktaten" bin ich im Umgang mit aussagenlogischen bzw. mengentheoretischen Formeln sicherer und selbstbewusster geworden. (8)

Die Anzahl der Antworten, die Verteilung der Antworten sowie der Durchschnittswert ergeben sich aus folgender Tabelle:

Damit erklärten 80% der Befragten tendenziell, dass die Arbeit mit dem System ihnen Spass mache, 25% sogar mit der Antwort "trifft voll zu". 89%

Table 8.28: Studentische Einschätzung zur Wirkung der “Mathediktate” auf die Motivation

Frage	Antworten	1	2	3	4	5	6	Durchschnitt
1	105	2 (2%)	7 (7%)	12 (11%)	24 (23%)	34 (32%)	26 (25%)	4,5
2	106	1 (1%)	2 (2%)	8 (8%)	12 (11%)	33 (31%)	50 (47%)	5,1
3	106	5 (5%)	31 (29%)	26 (25%)	28 (26%)	13 (12%)	3 (3%)	3,2
4	106	5 (5%)	7 (7%)	22 (21%)	36 (34%)	25 (24%)	11 (10%)	4

empfinden Erfolgsmeldungen von Seiten des Systems tendenziell als ermutigend, 47% antworteten hier mit “trifft voll zu”; dagegen gaben nur 41% an, durch Fehlermeldungen tendenziell entmutigt zu werden, 63% davon äußerten eine leichte Zustimmung (4). Eine Steigerung in Sicherheit und Selbstbewusstsein im Umgang mit logischen Formeln wurde von 68% der Befragten tendenziell bejaht.

Im Fazit kann damit gesagt werden, dass die “Mathediktate” sich positiv auf die Motivation auswirken, mit aussagenlogischen bzw. mengentheoretischen Ausdrücken zu arbeiten und die Sicherheit im Umgang mit solchen Ausdrücken fördern.

Einbindung in den Lehrbetrieb

Zur Einbindung der Mathediktate in den Lehrbetrieb wurden folgende Fragen gestellt, wobei die Positionen der Fragen im Fragebogen jeweils hinter den Fragen in Klammern stehen:

1. Die “Mathediktate” stellen eine sinnvolle Ergänzung des Übungsbetriebes dar. (14)

2. Die Erläuterungen zu den “Mathediktaten” auf den Übungsblättern waren ausreichend, um mit dem Programm umgehen zu können. (17)

Die Anzahl der Antworten, die Verteilung der Antworten sowie der Durchschnittswert ergeben sich aus Tabelle 8.29.

Table 8.29: Studentische Einschätzung zur Einbindung der “Mathediktate” in den Lehrbetrieb

Frage	Antworten	1	2	3	4	5	6	Durchschnitt
1	105	1 (1%)	8 (8%)	13 (12%)	30 (29%)	34 (32%)	19 (18%)	4,4
2	105	2 (2%)	15 (14%)	16 (15%)	25 (24%)	29 (28%)	18 (17%)	4,1

Damit betrachten 79% der Befragten die Mathediktate tendenziell als sinnvolle Ergänzung des Übungsbetriebes, während 69% die Erläuterungen zum System tendenziell als ausreichend betrachten.

Die Einbindung der Mathediktate in den Übungsbetrieb scheint damit insgesamt gut funktioniert zu haben. Einige zuätzliche Hilfestellungen, etwa schriftliche Erläuterungen der Eingabesyntax, könnten dazu beitragen, denen entgegen zu kommen, die zusätzliche Erläuterungen bei der Bedienung hilfreich gefunden hätten.

Fazit

Die Mathediktate sind, laut mehrheitlicher Rückmeldung der Studierenden, leicht zu bedienen, wirken sich förderlich auf das Verständnis logischer Ausdrücke sowie die Kompetenz aus, solche Ausdrücke syntaktisch korrekt zu bilden und sinnvoll zu verwenden und steigern die Motivation, sich mit logischen Ausdrücken zu befassen. Verbesserungsbedarf scheint es noch bei der Interpretation von Rückmeldungen zu geben (Frage (7)), wovon die Möglichkeit, diese zur Verbesserung von Lösungen einzusetzen, jedoch bei den meisten nicht berührt wird (Frage (4)). Die Mathediktate scheinen damit ein sinnvolles Lernformat zur Förderung des Verständnisses logischer Ausdrücke darzustellen.

8.3.3 Studentische Rückmeldungen zum "Game of Def"

Der Evaluationsfragebogen zum "Game of Def" begann mit folgendem klärenden Passus, um sicherzustellen, dass die Fragen auf das richtige Modul bezogen verstanden wurden:

> Die folgenden Fragen beziehen sich auf das Programm "Game of Def". Beim "Game of Def" ging es darum, zu einem Quadratgitter, in dem einige Felder gelb markiert waren, eine quantorenlogische Formel anzugeben, die genau auf die gelben Quadrate zutraf.

Der Fragebogen enthielt drei Fragen mit Freitextantworten, die mit denen identisch sind, die zur Evaluation der Mathediktate gestellt wurden. Auch ansonsten wurden weitgehend die gleichen Fragen gestellt, um einen möglichst direkten Vergleich zu ermöglichen.

Freitextantworten

Die folgenden Fragen waren durch eine Freitexteingabe zu beantworten:[24]

1. Besonders gut gefällt mir, dass... (21)
2. Nicht so gut gefällt mir, dass... (22)
3. Konkret habe ich folgende Verbesserungsvorschläge: (23)

Auf Frage 1 wurden zwölf Antworten abgegeben. Davon ließen sich zehn unter einen oder mehrere der folgenden Punkte subsumieren:

1. Das System liefert umgehend ein Feedback. (UF)
2. Die graphische Gestaltung war ansprechend. (GA)
3. Die graphische Gestaltung war hilfreich. (GH)
4. Der Umgang mit dem System motiviert bzw. macht Spaß. (MS)

Es ergab sich die in Tabelle 8.30 aufgeführte Verteilung.

Auf Frage 2 wurde 19 mal geantwortet. Von den Antworten ließen sich 18 unter einen oder mehrere der folgenden Punkte subsumieren:

[24] Diese Fragen stammen aus den Fragebögen zur studentischen Evaluation von Lehrveranstaltungen an der Europa-Universität Flensburg.

Table 8.30: Verteilung der Antworten zur Frage “Besonders gut gefällt mir, dass...”

UF	GA	GH	MS
1	6	1	4

1. Die Eingabe ist syntaktisch restriktiv (Klammersetzung etc.). (ER)
2. Die Rückmeldung unterscheidet nicht zwischen syntaktischen und inhaltlichen Fehlern.[25] (IS)
3. Die Beschäftigung mit dem Game of Def in den Lehrveranstaltungen war unzureichend. (EU)
4. Das Game of Def wirkt sich negativ auf die Motivation aus. (NM)
5. Es dauert lange, eine Game of Def-Aufgabe zu lösen. (ZZ)
6. Die Rückmeldung waren nicht hilfreich. (RU)

Es ergab sich folgende Verteilung:

Table 8.31: Verteilung der Antworten zur Frage “Nicht so gut gefällt mir, dass...”

ER	IS	EU	NM	ZZ	RU
2 (11%)	2 (11%)	8 (44%)	3 (17%)	1 (6%)	4 (22%)

Auf Frage 3 wurde 12 mal geantwortet. Die Antworten ließen sich durchweg unter einen oder mehrere der folgenden Punkte subsumieren:

1. Die Fehlermeldung sollte nach Fehlertyp differenzierten.[26] (FD)
2. Es sollte zusätzliche Erläuterungen zum Game of Def geben, etwa in Form von Erklärvideos oder in den Lehrveranstaltungen. (LV)
3. Es sollte Lösungsvorschläge oder eine Hilfefunktion geben (HF)

Es ergab sich folgende Verteilung:

[25] Diese Rückmeldung ist überraschend. Tatsächlich bricht das Game of Def bei einer syntaktisch nicht korrekten Eingabe mit einer entsprechenden Fehlermeldung ab, während bei einer syntaktisch korrekte, aber inhaltlich fehlerhaften Eingabe die Menge der durch die Eingabe erfassten Felder markiert wird (siehe den Abschnitt zum Game of Def weiter oben).

[26] Wie bereits oben erwähnt, ist das bereits der Fall. Vermutlich hätten hier zusätzliche Erläuterungen in den Lehrveranstaltungen geholfen, die Fehlermeldungen korrekt zu interpretieren.

Table 8.32: Verteilung der Antworten zur Frage nach Verbesserungsvorschlägen

FD	LV	HF
1 (8%)	9 (75%)	2 (17%)

Bedienbarkeit und Nutzung

Zur Bedienbarkeit zu Nutzung des "Game of Def" wurden folgende Fragen gestellt:

1. Rückmeldungen des "Game of Def" helfen mir dabei, meine Lösung zu verbessern. (4)
2. Bei Fehlermeldungen im "Game of Def" kann ich meistens erkennen, worin der Fehler in meiner Lösung besteht. (11)
3. Es kommt vor, dass richtige Lösungen als falsch gemeldet werden. (13)
4. Es kommt vor, dass falsche Lösungen als richtig gemeldet werden. (14)
5. Durch das "Game of Def" habe ich mich länger mit der Verwendung logischer Ausdrücke beschäftigt, als ich es sonst getan hätte. (16, Nutzung)

Die Anzahl der Antworten, die Verteilung der Antworten sowie der Durchschnittswert ergeben sich aus folgender Tabelle:

Table 8.33: Studentische Einschätzung zu Bedienbarkeit und Nutzung des "Game of Def"

Frage	Antworten	1	2	3	4	5	6	Durchschnitt
1	103	31 (30%)	26 (25%)	19 (18%)	15 (15%)	8 (8%)	4 (4%)	2,6
2	103	34 (33%)	31 (30%)	23 (22%)	9 (9%)	6 (6%)	0	2,2
3	95	46 (48%)	22 (23%)	15 (16%)	4 (4%)	6 (6%)	2 (2%)	2,0
4	95	59 (62%)	20 (21%)	13 (14%)	2 (2%)	1 (1%)	0 (0%)	1,6
5	103	19 (18%)	11 (11%)	15 (15%)	25 (24%)	19 (18%)	14 (14%)	3,5

Wie sich zeigt, haben die Studierenden beim "Game of Def" erhebliche Schwierigkeiten, die Rückmeldungen des Systems zu interpretieren und nutzbar zu machen: Nur etwas mehr als ein Viertel (27%) gab an, Rückmeldungen zur Verbesserung der Lösung nutzen zu können, und nur etwas mehr als ein Siebtel (15%) bestätigte, von Fehlermeldungen auf Fehler in der eigenen Lösung schließen zu können; dagegen gaben jeweils fast ein Drittel der Befragten bei beiden Fragen an, dies treffe überhaupt nicht zu (1). Die Zweifel an der Korrektheit der Rückmeldungen fallen dagegen deutlich schwächer aus als bei den Mathediktaten: Nur 12% bejahten tendenziell die Frage, ob richtige Lösungen bisweilen als falsch

gemeldet würden, eine Meldung fehlerhafter Lösungen als korrekt wurde sogar tendenziell nur von 3% der Befragten beobachtet. Das legt den Schluss nahe, dass die visuelle Rückmeldung des "Game of Def" es jedenfalls überzeugend deutlich macht, dass und wo ein Lösungsvorschlag von einer korrekten Lösung abweicht. Eine knappe Mehrheit von 56% erklärte zudem, durch die Benutzung des Systems tendenziell mehr Zeit mit der Verwendung logischer Ausdrücke verbracht zu haben; knapp ein Fünftel (18%) der Befragten gaben jedoch andererseits an, dies treffe auf sie überhaupt nicht zu.

Diese Rückmeldungen sind insofern nicht überraschend, als die meisten "Game of Def"-Aufgaben zu ihrer Lösung eine Kernidee erfordern, wie das gegebene Bild logisch aufzufassen ist, um es dann durch eine quantorenlogische Formel zu beschreiben. Bei der Suche nach solchen Ideen ist das Ausprobieren von Lösungsansätzen zwar potenziell hilfreich; andererseits geben auch präzise Fehlermeldungen nicht unbedingt einen Hinweis darauf, wie sie zu finden ist. Es besteht dann die Gefahr, dass der Lösungsversuch abgebrochen wird. Wie sich aus den Antworten auf die letzte Frage ergibt, führte der Einsatz des Systems jedoch insgesamt dazu, dass mehr Zeit mit der Verwendung quantorenlogischer Formeln verbracht wurde.

Wirkung auf Kompetenzen

Ziel des "Game of Def" ist es, Syntax und Semantik quantorenlogischer Formeln zu vermitteln, so dass diese einerseits besser interpretiert und andererseits auch eigenständig als Ausdrucksform mathematischer Aussagen verwendet werden können. Zur Evaluation der Wirkung des "Game of Def" auf diese Kompetenzen wurden folgende Fragen gestellt:

1. Die Arbeit mit dem "Game of Def" hat mir geholfen, die korrekte Bildung quantorenlogischer Formeln zu lernen. (2)
2. Die Arbeit mit dem "Game of Def" hat mir geholfen, zu lernen, anschauliche Vorstellungen in Quantorenlogik auszudrücken. (3)
3. Ich bin mit der Zeit besser im "Game of Def" geworden. (5)
4. Das "Game of Def" fällt mir schwerer als die "Mathediktate". (10)
5. Ich habe durch den Gebrauch des "Game of Def" viel gelernt. (15)
6. Frühere "Game of Def"-Aufgaben helfen mir oft dabei, Ansätze für spätere zu finden. (18)
7. Um "Game of Def"-Aufgaben lösen zu können, muss man die zu erwerbenden Kompetenzen bereits besitzen; daher ist das "Game of Def" nutzlos. (19)

8. Die Arbeit mit dem “Game of Def” hat mir geholfen, logische Ausdrücke – etwa in der Vorlesung oder in den Übungen – besser zu verstehen. (12 Verständnis)

Die Anzahl der Antworten, die Verteilung der Antworten sowie der Durchschnittswert ergeben sich aus folgender Tabelle:

Table 8.34: Studentische Einschätzung zur kompetenzfördernden Wirkung des “Game of Def”

Frage	Antworten	1	2	3	4	5	6	Durchschnitt
1	103	31 (30%)	36 (35%)	14 (14%)	18 (17%)	3 (3%)	1 (1%)	2,3
2	103	31 (30%)	33 (32%)	20 (19%)	10 (10%)	7 (7%)	2 (2%)	2,4
3	101	21 (21%)	17 (17%)	27 (27%)	18 (18%)	8 (8%)	10 (10%)	3
4	103	3 (3%)	2 (2%)	5 (5%)	12 (12%)	23 (22%)	58 (56%)	5,2
5	103	27 (26%)	23 (22%)	28 (27%)	17 (17%)	7 (7%)	1 (1%)	2,6
6	99	18 (18%)	19 (19%)	20 (20%)	20 (20%)	16 (16%)	6 (6%)	3,2
7	100	11 (11%)	24 (24%)	23 (23%)	24 (24%)	9 (9%)	9 (9%)	3,2
8	102	30 (29%)	31 (30%)	22 (22%)	11 (11%)	6 (6%)	2 (2%)	2,4

Wie sich zeigt, ist nur jeweils etwa ein Fünftel tendenziell der Ansicht, dass das “Game of Def” dabei hilft, die Syntax (21%) quantorenlogischer Formeln zu erlernen oder zu lernen, quantorenlogische Formeln als Ausdrucksmittel einzusetzen (21%) bzw. quantorenlogische Ausdrücke zu verstehen (19%); dagegen erklärte jeweils knapp ein Drittel der Befragten (30%, 30% und 29%), das treffe “überhaupt nicht zu” (1). Im Fazit war ein Viertel (25%) der Befragten tendenziell der Ansicht, durch die Verwendung des Systems viel gelernt zu haben (4-6), während 27% angaben, dies treffe “überhaupt nicht zu” (1). Der Ansicht, das System sei nutzlos, weil seine Verwendung die zu lernenden Kompetenzen voraussetze, schlossen sich tendenziell 42% der Befragten an. Für die Lösung der Aufgaben hilfreiche Zusammenhänge zwischen den Teilaufgaben konnten tendenziell 42% der Befragten erkennen; etwas mehr als ein Drittel (36%) erklärten zudem tendenziell, ihre Beherrschung des Systems habe sich mit der Zeit verbessert. Eine große Mehrheit (90%) der Befragten erklärte, das “Game of Def” falle ihnen tendenziell schwerer als die “Mathediktate” (4-6); 58% erklärten, dies treffe voll zu (6).

Eine förderliche Wirkung von der angestrebten Art wurde damit nur von rund 20% der Befragten bestätigt. Ein Lerneffekt im Durchlauf durch die Teilaufgaben bzw. über die Übungsblätter wurde deutlich häufiger, aber noch immer von einer Minderheit der Befragten bemerkt. Hierzu ist allerdings zu bemerken, dass “Game of Def”-Aufgaben lediglich auf zwei Übungsblättern in zwei aufeinanderfolgenden Wochen gestellt wurden. Bei einem längeren Einsatz wäre insbesondere dann ein stärkerer Lerneffekt zu erhoffen, wenn die Aufgaben regelmäßig besprochen und

ggf., etwa im Rahmen der vorbereitenden Übungen, in Interaktion mit Lehrkräften bearbeitet würden.

Motivation

Zur Wirkung des "Game of Def" auf Motivation und Selbstbewusstein wurden folgende Fragen gestellt:

1. Der Umgang mit dem "Game of Def" macht mir Spaß. (1)
2. Durch den Umgang mit dem "Game of Def" bin ich im Umgang mit quantorenlogischen Formeln sicherer und selbstbewusster geworden. (6)
3. Es ermutigt mich, wenn das System eine Lösung als korrekt meldet. (7)
4. Fehlermeldungen des Systems entmutigen mich. (8)
5. Die optische Darstellung macht das "Game of Def" reizvoller als Aufgaben, die nur aus Text bestehen. (9)

Die Anzahl der Antworten, die Verteilung der Antworten sowie der Durchschnittswert ergeben sich aus folgender Tabelle:

Table 8.35: Studentische Einschätzung zur Wirkung des "Game of Def" auf die Motivation

Frage	Antworten	1	2	3	4	5	6	Durchschnitt
1	102	38 (37%)	23 (23%)	14 (14%)	13 (13%)	8 (8%)	6 (6%)	2,5
2	103	28 (27%)	30 (29%)	19 (18%)	20 (19%)	4 (4%)	2 (2%)	2,5
3	103	9 (9%)	3 (3%)	6 (6%)	22 (21%)	32 (31%)	31 (30%)	4,5
4	103	17 (17%)	19 (18%)	21 (20%)	17 (17%)	15 (15%)	14 (14%)	3,3
5	103	20 (19%)	15 (15%)	12 (12%)	23 (22%)	17 (17%)	16 (16%)	3,5

Demnach erklärten 27% der Befragten tendenziell, der Umgang mit dem "Game of Def" mache ihnen Spaß (4-6), während das für deutlich mehr (37%) "überhaupt nicht" zutraf (1). Eine deutliche Mehrheit von 85% empfindet positive Rückmeldungen des Systems tendenziell als ermutigend, während nur 46% durch Fehlermeldungen tendenziell entmutigt werden; dies bestätigt die schon in den Evaluationen der Prüfkomponente und der Mathediktate beobachtete Tendenz, dass positive Rückmeldungen in einem stärkeren Maß ermutigend wirken als Fehlermeldungen entmutigen, was auch durch die Durchschnittszahlen unterstützt wird: So weicht der Durchschnitt 4,5 der Antworten auf Frage 3 um 1,2 nach oben vom Durchschnitt bei einer Gleichverteilung der Antworten (3,5) ab, der Durchschnitt bei Frage 4 dagegen um 0,2 nach unten. Eine Steigerung der

Sicherheit bzw. des Selbstbewusstseins im Umgang mit quantorenlogischen Formeln wurde tendenziell von 26% der Befragten bejaht, etwas mehr (27%) erklärten, das treffe auf sie "überhaupt nicht zu" (1). Die optische Gestaltung wurde von einer knappen Mehrheit der Befragten (55%) tendenziell als reizvoller angesehen als Aufgaben in Textform; etwa ein Fünftel (19%) fand hingegen, dies treffe "überhaupt nicht zu" (1).

Eine motivationsfördernde Wirkung des "Game of Def" kann somit nur für eine Minderheit der Studierenden bestätigt werden.

Einbindung des "Game of Def" in den Lehrbetrieb

Zur Einbindung des "Game of Def" in den Lehrbetrieb wurden folgende Fragen gestellt:

1. Das "Game of Def" stellt eine sinnvolle Ergänzung des Übungsbetriebs dar. (17)

2. Die Erläuterungen zum "Game of Def" auf den Übungsblättern waren ausreichend, um mit dem Programm umgehen zu können. (20)

Die Anzahl und Verteilung der Antworten ergibt sich aus der folgenden Aufstellung:

Table 8.36: Studentische Einschätzung zur Einbindung des "Game of Def" in den Lehrbetrieb

Frage	Antworten	1	2	3	4	5	6	Durchschnitt
1	103	25 (24%)	23 (22%)	21 (20%)	18 (17%)	13 (13%)	3 (3%)	2.8
2	103	27 (26%)	32 (31%)	19 (18%)	11 (11%)	8 (8%)	6 (6%)	2.6

Demnach betrachtet etwa ein Drittel der Befragten das "Game of Def" als sinnvolle Ergänzung des Übungsbetriebes, während nur ein Viertel der Ansicht ist, die Erläuterungen zu dieser Systemkomponente seien ausreichend gewesen, um das Programm bedienen zu können.

Der zweite Punkt verdeutlicht noch einmal, dass der Umgang mit dem System vielen Studierenden schwer gefallen ist: Wie bereits oben festgestellt wurde, bereitete es insbesondere Schwierigkeiten, die Rückmeldungen des Systems für eine Verbesserung der eigenen Lösung zu nutzen. Damit kann die angestrebte Interaktion zwischen System und BenutzerIn nur noch eingeschränkt stattfinden und die Funktion des Systems besteht im wesentlichen in einer sofortigen Korrektur, was den erhofften Lerneffekt reduziert – und damit auch den Wert des "Game of Def" als Teil des Übungsbetriebes.

Fazit

Zusammenfassend läßt sich feststellen, dass die “Game of Def”-Aufgaben deutlich kritischer beurteilt wurden als die Mathediktate: Die Bedienung des Systems fiel schwerer, die angestrebte Kompetenzförderung wurde nur von einer Minderheit der Befragten bestätigt, und auch wenn die Visualisierung mehrheitlich als Vorzug betrachtet wird, wird das System ebenfalls nur von einer Minderheit der Befragten als motivierend erlebt.

Eine mögliche Ursache hierfür mag einfach in der Schwierigkeit der Aufgaben liegen: Die “Game of Def”-Aufgaben erfordern den Umgang mit prädikatenlogischen Formeln, wozu insbesondere auch Quantoren gehören; schon in dieser Hinsicht weisen sie einen höheren Schwierigkeitsgrad auf als die Mathediktate. Hinzu kommt, dass die Umsetzung eines Bildes in einen logischen Ausdruck eine größere Abstraktionsleistung darstellt als die bloße Umformulierung eines natürlichsprachlichen Ausdrucks in einem Logikkalkül. Schließlich erfordern viele der “Game of Def”-Aufgaben eine “Grundidee” oder “Hinsicht”, mit der das Bild aufzufassen ist, ehe man die Aufgaben lösen kann. Insbesondere der letzte Aspekt hat zur Folge, dass eine allmähliche Annäherung an eine richtige Lösung unter Nutzung der automatischen Korrektur bei einigen Aufgaben schwierig ist.

Eine Möglichkeit, dem zu begegnen, wäre, die Aufgaben noch stärker als bisher inkrementell zu organisieren, so dass verschiedene Aspekte des Formalisierens separat erlernt und gefestigt werden können. Hierzu wäre zu überlegen, von der Einbindung in den traditionellen, auf wöchentlichen Abgaben basierenden Übungsbetrieb abzusehen und andere Lernformate zu konzipieren, etwa analog zu den Formalisierungskursen zur Aussagenlogik bzw. Mengenlehre (s.o.). Ebenfalls wäre es wünschenswert, durch eine stärkere Interaktion mit Lehrkräften im Umgang mit dem System stärker als bisher den Umgang mit Fehlermeldungen und deren konstruktive Nutzung vorzuführen sowie Heuristiken zur Lösungssuche zu vermitteln.

8.4 Bearbeitung von Diproche-Aufgaben und Bearbeitungserfolg bei Beweisaufgaben

Zusätzlich zu den Einschätzungen der Studierenden bezüglich der Wirkung von Diproche haben wir für eine ausgewählte Beweisaufgabe die Bearbeitungen in der Abschlussklausur zur Algebra-Vorlesung erfasst.

Um zu untersuchen, wie weit der Einsatz von Diproche Auswirkungen auf die im letzten Abschnitt herausgearbeiteten Kompetenzen hat, bietet es sich an, den Klausurerfolg am Ende einer Lehrveranstaltung zu betrachten, in der Diproche eingesetzt wurde. Um durch die Verwendung von Diproche erzielte Lerneffekte abschätzen zu können, wäre es wünschenswert gewesen, nur einige der VorlesungsteilnehmerInnen mit dem System arbeiten zu lassen und anschließend den Unterschied im Klausurerfolg zwischen dieser Gruppe und einer Kontrollgruppe von Studierenden zu erfassen, die das Semester ohne Einsatz des Systems durchlaufen haben. Aus den bereits oben besprochenen ethischen Erwägungen wurde dieser Weg abgelehnt. Da somit alle TeilnehmerInnen der Vorlesung unter Verwendung von Diproche unterrichtet worden waren, fehlte eine Kontrollgruppe für einen direkten Vergleich.

Stattdessen wurden die Bearbeitungen einer mengentheoretischen Beweisaufgabe der Abschlussklausur nach den folgenden Kriterien ausgewertet, die dann in einem zweiten Schritt mit verschiedenen Faktoren der Bearbeitung der Diproche-Übungsaufgaben in Beziehung gesetzt wurden. Auf diese Weise konnte zumindest eine Aussage darüber getroffen werden, mit welchen der Kompetenzen, auf die unserer Erwartung nach durch die Verwendung von Diproche ein förderlicher Einfluss existieren vorliegen sollte, die Bearbeitungen der Diproche-Übungsaufgaben einen signifikanten Zusammenhang aufweisen.

Die Klausuraufgabe[27] war folgende:

> Zeige: Für alle Mengen A, B, C gilt $((A \times B) \setminus (B \times C)) \subseteq ((A \setminus B) \times C)$.

Diese Aufgabe ist für die Erfassung des durch Diproche bewirkten Lernerfolges gut geeignet, da sie den im Übungsbetrieb gestellten Diproche-Beweisaufgaben aus dem Bereich der Booleschen Mengenlehre ähnelt[28] und insbesondere – nach einer kleinen Umformulierung – auch im Rahmen von Diproche gut lösbar ist:

[27] Diese Aufgabe stammt von Hinrich Lorenzen.

[28] Tatsächlich handelt es sich um eine Richtung des Beweises, dass sich das kartesischen Produkt bezüglich der mengentheoretischen Differenz distributiv verhält; zu den Diproche-Beweisaufgaben auf Blatt 11 gehörten bereits die Distributivität des kartesischen Produktes über den Vereinigungs- und Schnittoperator. Die jeweiligen Beweise verlaufen in vielen Hinsichten analog.

Table 8.37: Eine Lösung der mengentheoretischen Beweisaufgabe in Diproche

Es seien A, B, C Mengen.
Zeige: Dann gilt $((A \times C) \setminus (B \times C)) \subseteq ((A \setminus B) \times C)$.
Beweis: Es sei $(x, y) \in ((A \times C) \setminus (B \times C))$. Dann ist $(x, y) \in (A \times C)$ und $\sim (x, y) \in (B \times C)$. Also ist $x \in A$ und $y \in C$. Ferner gilt $\sim x \in B$ oder $\sim y \in C$. Also ist $\sim x \in B$. Damit ist $x \in (A \setminus B)$. Damit ist $(x, y) \in ((A \setminus B) \times C)$. qed.

Die Klausur im Herbstsemester 2020/2021 fand als Take-Home-Klausur statt, die von den Studierenden online durch Eingabe in einen Texteditor zu bearbeiten war. Da dieser Texteditor keine mathematischen Sonderzeichen zur Verfügung stellte, waren die Studierenden angehalten, formale Ausdrücke verbal zu formulieren, also etwa "$x \in y$" durch "x ist ein Element von y" zu ersetzen. Hierdurch bedingt war eine Erfassung der korrekten Verwendung formaler Ausdrücke kaum möglich. Eine Prüfung der "verbalisierten Formeln" schien nicht sinnvoll, da in diesem Fall die vor der Auswertung vorzunehmende "Übersetzung" der Verbalisierung durch einen formalen Ausdruck die Intention der bzw. des Studierenden nur noch mit großer Unsicherheit wiedergegeben hätte.

8.4.1 Operationalisierung

Zur Erfassung der in Abschnitt 8.2 herausgearbeiteten Beweiskompetenzen wurden für jeden Lösungstext zu der oben genannten Aufgabe folgende 14 Größen erhoben:[29]

1. Die Gesamtpunktzahl, die bei dieser Aufgabe erreicht wurde; diese bestand aus einer ganzen Zahl zwischen 0 und 15.

2. Das Vorliegen einer als Beweistext erkennbaren Abgabe, also eines deduktiv strukturierten Textes. Beispiele für nicht als Beweistext erkennbare Abgaben wären etwa Bilder oder bloße Aneinanderreihungen formaler Ausdrücke. Dies wurde als dichotome Variable erfasst: War die Abgabe als Beweistext erkennbar, wurde der Wert auf 1 gesetzt, sonst auf 0.

[29]Die Gesamtpunktzahl ist die, die sich durch die Korrektur der Algebra-Klausur durch die MitarbeiterInnen der Abteilung für Mathematik und ihre Didaktik der EUF ergab; an der Korrektur beteiligt waren Christian Hercher, Philipp Lampe, Hinrich Lorenzen, Michael Schmitz, Tobias Sohr, Selma Stronzik, Inga Stolz, Stefan Virchow sowie der Autor. Die übrigen Größen wurden vom Autor allein erhoben.

3. Die “zusammenhangslose” Verwendung formaler Ausdrücke ohne Einbindung in einen deduktiv strukturierten Text. Diese wurde als dichotome Variable erfasst: lag eine solche Verwendung vor, wurde dieser Wert auf 1 gesetzt, sonst auf 0.

4. Die Verwendung von Strukturmarkern wie “Beweis:” oder “qed”, Anfangsmarkierungen für Fälle bei Fallunterscheidungen etc. Wurde mindestens ein Strukturmarker verwendet, wurde dieser Wert auf 1 gesetzt, sonst auf 0.

5. Das Vorliegen logisch-operativer Fehler. Ein logisch-operativer Fehler liegt vor, wenn einer Aussage nicht der Status zugewiesen wird, den sie an dieser Stelle logisch haben müsste, etwa wenn eine Behauptung als Annahme formuliert wird oder umgekehrt. Beim Vorliegen operativer Fehler wurde dieser Wert auf 1 gesetzt, sonst auf 0.

6. Die Gesamtzahl der Folgerungsschritte, eine natürliche Zahl.

7. Die Anzahl korrekter Folgerungsschritte, eine natürliche Zahl.

8. Der Anteil der korrekten Folgerungsschritte an der Gesamtzahl der Folgerungsschritte, also der Quotient der beiden vorangegangen Größen, eine rationale Zahl.

9. Das Vorliegen einer dem Beweisziel angemessenen Beweisstruktur. Lag eine solche vor, wurde dieser Wert auf 1 gesetzt, sonst auf 0.

10. Der Versuch eines “Beweises durch Beispiel”, bei dem die fragliche Aussage lediglich an einem Einzelfall überpfüft und damit als bewiesen angesehen wurde. Wurde dieses Vorgehen versucht, wurde der Wert auf 1 gesetzt, sonst auf 0.

11. Das Vorliegen von Typenfehlern durch nicht eingeführte Variablen. Lagen solche Fehler vor, wurde dieser Wert auf 1 gesetzt, sonst auf 0.

12. Das Vorliegen von Typenfehlern durch die Verwendung von Mengen in einer nur für Aussagen sinnvollen Form, etwa dadurch, dass Mengen behauptet oder angenommen werden; lagen solche Fehler vor, wurde dieser Wert auf 1 gesetzt, sonst auf 0.

13. Das Vorliegen von Typenfehlern durch die Verwendung von Aussagen in einer nur für Mengen sinnvollen Form, etwa dadurch, dass eine mengentheoretische Differenz von zwei Aussagen gebildet wird; lagen solche Fehler vor, wurde dieser Wert auf 1 gesetzt, sonst auf 0.

14. Das Vorliegen von Typenfehlern von mindestens einer der in den letzten drei Punkten erwähnten Arten, also das Maximum der in den letzten drei Punkten erhaltenen Werte.

Der Punkt 1 ist keiner speziellen Einzelkompetenz zugeordnet, sondern erfasst die gesamte Beweiskompetenz, soweit sie sich in der Bearbeitung der Klausuraufgabe zeigt; insofern sämtliche der oben genannten Punkte in die Bewertung eingegangen sind, weist er einen Bezug zu allen anderen der angesprochenen Kompetenzen auf.

Mit Bezug auf Abschnitt 8.2 korrespondiert (3) zu den Unterpunkten "mehr sprachliche Formulierungen" sowie "mehr Formulierungen, die den Status einer Aussage (...) anzeigen", (4) zum Unterpunkt "mehr Strukturmarker auf Makroebene", (5) zu den Unterpunkten "mehr Formulierungen, die den Status einer Aussage (...) anzeigen" sowie "weniger Verwechselungen von Ziel, Annahme und Behauptung"; insgesamt erfassen die Punkte (3), (4), (5) also das Tätigkeitsfeld 1 des Kompetenzfeldes (K1). Die Punkte (7) und (8) erfassen den Punkt "Anzahl derjenigen Behauptungen im Beweis, die nach einfachen logischen Regeln aus an dieser Stelle verfügbaren Aussagen gefolgert werden können", also das Tätigkeitsfeld (2) des Kompetenzfeldes (K1). Die Punkte (2) und (6) gehören zu Tätigkeitsfeld (3) des Kompetenzfeldes (K1), (9) und (10) zu den Tätigkeitsfeldern (4) und (5). Damit dienen also (2)-(10) zur Erfassung des Zusammenhangs mit Kompetenzfeld (K1), dem mathematischen Argumentieren.

Es wäre wünschenswert gewesen, das Kompetenzfeld (K5) ("mit Mathematik symbolisch/formal/technisch umgehen") durch eine Ermittlung der Anzahl der verwendeten formalen Ausdrücke sowie des Anteils korrekt gebildeter formaler Ausdrücke zu erfassen; aus den oben erläuterten Gründen erwies sich dieses Vorhaben jedoch im konkreten Fall als undurchführbar und wurde daher fallen gelassen.

Die Punkte (10)-(14) schließlich erfassen den Punkt "korrekter Gebrauch von Variablen" und gehören damit zum Unterpunkt 2.2 des Kompetenzfeldes (K6).

Diesen 11 Größen wurden folgende Größen aus der Erfassung der Übungsaufgaben gegenüber gestellt:

1. Die Gesamtzahl bearbeiteter Diproche-Beweisaufgaben.

2. Die Gesamtzahl bearbeiteter Diproche-Beweisaufgaben zum Themengebiet Mengenlehre.

3. Die Anzahl korrekt bearbeiteter Diproche-Beweisaufgaben zum Themengebiet Mengenlehre.

4. Die Anzahl korrekt oder fast korrekt bearbeiteter Diproche-Beweisaufgaben zum Themengebiet Mengenlehre.

8.4.2 Hypothesen

Zu untersuchen ist, ob es zwischen Bearbeitungshäufigkeit und -erfolg der Diproche-Beweisaufgaben und der korrekten Bearbeitung der Klausuraufgabe einen statistischen Zusammenhang gibt. Mithilfe der nun erreichten Operationalisierung läßt diese Leitfrage sich durch eine Reihe spezifischerer Hypothesen präzisieren. Bezüglich der im letzten Abschnitt herausgearbeiteten Größen stellen wir folgende Hypothesen zum Zusammenhang der Bearbeitungen der Diproche-Aufgaben mit den Bearbeitungen der Klausuraufgabe auf:

1. Es gibt einen positiven Zusammenhang zwischen der Gesamtzahl bearbeiteter Diproche-Aufgaben und den Punkten (1), (2), (4), (6), (7), (8), (9).

2. Es gibt einen positiven Zusammenhang zwischen der Gesamtzahl bearbeiteter Diproche-Beweisaufgaben zum Themengebiet Mengenlehre und den Punkten (1), (2), (4), (6), (7), (8), (9).

3. Es gibt einen positiven Zusammenhang zwischen der Anzahl korrekt bearbeiteter Diproche-Beweisaufgaben zum Themengebiet Mengenlehre und den Punkten (1), (2), (4), (6), (7), (8), (9).

4. Es gibt einen positiven Zusammenhang zwischen der Anzahl korrekt oder fast korrekt bearbeiteter Diproche-Beweisaufgaben zum Themengebiet Mengenlehre und den Punkten (1), (2), (4), (6), (7), (8), (9).

5. Es gibt einen negativen Zusammenhang zwischen der Gesamtzahl bearbeiteter Diproche-Aufgaben und den Punkten (3), (5), (10), (11), (12), (13), (14).

6. Es gibt einen negativen Zusammenhang zwischen der Gesamtzahl bearbeiteter Diproche-Beweisaufgaben zum Themengebiet Mengenlehre und den Punkten (3), (5), (10), (11), (12), (13), (14).

7. Es gibt einen negativen Zusammenhang zwischen der Anzahl korrekt bearbeiteter Diproche-Beweisaufgaben zum Themengebiet Mengenlehre und den Punkten (3), (5), (10), (11), (12), (13), (14).

8. Es gibt einen negativen Zusammenhang zwischen der Anzahl korrekt oder fast korrekt bearbeiteter Diproche-Beweisaufgaben zum Themengebiet Mengenlehre und den Punkten (3), (5), (10), (11), (12), (13), (14).

Hierbei handelt es sich also um insgesamt 56 Hypothesen. Der Kürze halber bezeichnen wir die Hypothese, die den j-ten Punkt des i-ten Listenelements

betrifft, mit H_{ij}, so dass etwa die Existenz eines positiven Zusammenhanges zwischen der Anzahl korrekt oder fast korrekt bearbeiteter Diproche-Beweisaufgaben zur Mengenlehre und der Gesamtzahl der Folgerungsschritte mit H_{46} bezeichnet würde.

8.4.3 Durchführung

Es wurde zunächst eine Stichprobe von 101 Studierenden herangezogen, die am Übungsbetrieb teilgenommen hatten. Davon fielen 8 aus der Auswertung heraus, die an der Klausur nicht teilgenommen haben, so dass eine Gruppe von 93 Studierenden betrachtet wurde. Für diese wurden jeweils Bearbeitungshäufigkeit und -erfolg bei den Diproche-Beweisaufgaben in Beziehung gesetzt mit den oben erarbeiteten Maßzahlen zu Erfassung der Klausurbearbeitung. Zwei Umstände sind potenziell geeignet, die Aussagekraft der Daten einzuschränken: Einerseits hatten 36 der 93 untersuchten ÜbungsteilnehmerInnen, also knapp 39%, zu keiner einzigen Diproche-Beweisaufgabe eine Lösung abgegeben, während 40 TeilnehmerInnen – also über 43% – zu keiner der Diproche-Beweisaufgaben zur Mengenlehre eine Lösung abgegeben hatten. Ferner sorgt die Abgabe der Übungsaufgaben in Abgabegruppen von meist zwei Personen dafür, dass nicht festzustellen ist, von wem eine Aufgabe letztlich bearbeitet wurde; dieses Problem verschärft sich durch die Kooperation in größeren Gruppen, etwa im Rahmen der vorbereitenden Übungen. Diese Faktoren können indes allenfalls dazu beitragen, vorhandene statistische Zusammenhänge zwischen Bearbeitungs- und Klausurerfolg zu verdecken bzw. abzuschwächen; sie sind aber kaum geeignet, Zweifel an den vorgefundenen Zusammenhängen zu begründen.

Die in Bezug auf den Übungserfolg erhobenen Größen sind durchweg metrische Variablen. Als Maß für den Zusammenhang zu den in (1), (6), (7) und (8) erfassten metrischen Variablen wurde der Korrelationskoeffizient nach Pearson (vgl. z.B. Cohen et al. [38], Abschnitt 2.2) verwendet, als Maß für den Zusammenhang zu den in (3), (4), (5), (9), (10), (11), (12), (13), (14) erhobenen dichotomen Variablen der punktbiseriale Korrelationskoeffizient als Spezialfall des Pearsonschen Korrelationskoeffizienten, vgl. Cohen et al., [38], Abschnitt 2.3.3. Als Signifikanzniveau für die Annahme einer Hypothese wurde der übliche Wert $p = 0,05$ verwendet. Im Einklang mit der üblichen terminologischen Konvention (siehe z.B. [189]) bezeichnen wir Korrelationen mit einem r-Wert zwischen $0,1$ und $0,3$ bzw. zwischen $-0,1$ und $-0,3$ als schwach positiv bzw. schwach negativ, Korrelationen mit einem r-Wert zwischen $0,3$ und $0,5$ bzw. zwischen $-0,3$ und $-0,5$ als "moderat positiv" bzw. "moderat negativ" und Korrelationen mit $r > 0,5$ bzw. $r < -0,5$ als "stark positiv" bzw. "stark negativ". Ab einem Signifikanzniveau von $p < 0,05$ bezeichnen wir eine Korrelation als "signifikant", ab $p < 0,01$ als "hoch signifikant". Für $p \in (0,05, 0,1]$ ist die jeweilige Hypothese

als nicht bestätigt anzusehen; solche Korrelationen werden unten trotzdem als "schwach signifikant" erwähnt. Für $|r| > 0,1$ sprechen wir davon, dass kein Zusammenhang festgestellt werden konnte.

Der Aspekt (2) wurde nach Erfassung der Daten aus der Analyse herausgenommen, da erfreulicherweise 93 der 94 betrachteten Lösungstexte klar als Beweistexte zu erkennen waren. Die übrigen Aspekte werden nun separat behandelt.

8.4.4 Auswertung

Diproche-Übungen vs. Gesamtpunktzahl

Die durchschnittliche Gesamtpunktzahl bei der Beweisaufgabe in der gesamten Stichprobe lag mit $4,8$ geringfügig unter der in der Gruppe aller Studierenden $(5,26)$.

In der gesamten Stichprobe ergaben sich folgende Korrelationen:

- Es ergab sich kein Zusammenhang ($r = 0,096$) zwischen der Gesamtzahl bearbeiteter Diproche-Beweisaufgaben und der Gesamtpunktzahl bei der Beweisaufgabe.
- Es gab eine schwach positive ($r = 0,163$), aber nicht signifikante ($p = 0,116$) Korrelation zwischen der Gesamtzahl bearbeiteter Diproche-Beweisaufgaben zur Mengenlehre und der Gesamtpunktzahl bei der Beweisaufgabe.
- Es gab eine schwach positive ($r = 0,104$), aber nicht signifikante ($p = 0,318$) Korrelation zwischen der Anzahl korrekt bearbeiteter Diproche-Beweisaufgaben und der Gesamtpunktzahl bei der Beweisaufgabe.
- Es gab eine schwach positive ($r = 0,125$), aber nicht signifikante ($p = 0,231$) Korrelation zwischen der Anzahl korrekt oder fast korrekt bearbeiteter Diproche-Beweisaufgaben und der Gesamtpunktzahl bei der Beweisaufgabe.

Die Hypothesen H_{11}, H_{21}, H_{31} und H_{41} konnten damit nicht bestätigt werden.

Diproche-Übungen vs. Verwendung zusammenhangsloser Formeln

Zusammenhangslose Formeln, deren Funktion im Argument nicht durch verbale Formulierungen angezeigt wurde, traten lediglich bei 8 der 94 Bearbeitungen auf, also bei knapp 9% der Bearbeitungen. Entsprechend ergab sich hier kein signifikanter Zusammenhang mit der Bearbeitung bzw. dem Bearbeitungserfolg der Diproche-Übungen. Die Hypothesen H_{53}, H_{63}, H_{73} und H_{83} konnten also nicht bestätigt werden.

Die Seltenheit dieses Fehlers steht in einem gewissen Gegensatz zu den Erfahrungen vergangener Semester. Ein Grund dafür könnte der Umstand sein, dass die Lösung über ein Online-Eingabefeld über die Tastatur einzugeben war und in der Aufgabenstellung ausdrücklich darauf hingewiesen wurde, das Argument "ausschließlich in Worten" zu formulieren. Beides dürfte die Bereitschaft, lediglich eine Folge von formalen Ausdrücken zu präsentieren, vermindert haben.

Diproche-Übungen vs. Verwendung von Strukturmarkern

Von den 94 Abgaben wurde in 72, also bei einer deutlichen Mehrheit von knapp 77% der Abgaben, mindestens ein Strukturmarker wie "Beweis:" oder "qed." verwendet. Auch hier ergab sich kein signifikanter Zusammenhang zur Bearbeitung bzw. dem Bearbeitungserfolg der Diproche-Übungen. Die Hypothesen H_{14}, H_{24}, H_{34} und H_{44} konnten folglich durch die Daten nicht bestätigt werden. Allerdings zeigte sich auch, dass die Verwendung von Strukturmarkern und die Gesamtpunktzahl mit $p = 0,085 < 0,1$ unkorreliert waren; die Verwendung von Strukturmarkern scheint damit ein eher oberflächliches Kriterium zu sein, das wenig Zusammenhang mit der Kompetenz zum Finden und Darstellen mengentheoretischer Beweise aufweist.

Diproche-Übungen vs. operative Fehler

Operative Fehler traten bei bei 29 der 94 Bearbeitungen auf, was einem Anteil von knapp 31% entspricht. Diese ließen sich überwiegend einem der beiden folgenden Typen zuordnen:

- Es wurde ein Beweisziel als Annahme formuliert, also zu Beginn etwa "Es gelte ϕ" geschrieben, wobei ϕ die zu beweisende Aussage war. Nur selten wurde diese als Annahme formulierte Aussage im dann folgenden Argument auch wirklich verwendet, was einen Formulierungsfehler nahelegt. Wo sie verwendet wurde, liegt zusätzlich ein Strukturfehler (s.o.) vor.
- Es wurden Existenzbehauptungen zur Variableneinführung verwendet, so tauchten Formulierungen wie "Dann existiert $(a, b) \in (A \setminus B) \times C$." zu Beginn des Teilmengenbeweises auf (obwohl die Aufgabenstellung nicht ausschließt, dass die Menge $(A \setminus B) \times C$ leer ist), wo über "Es sei $(a, b) \in (A \setminus B) \times C$." ein Element hätte gewählt werden müssen.

In der Stichprobe ergab sich folgendes Bild:

- Es gab eine schwach negative ($r = -0,221$) Korrelation zwischen der Gesamtzahl bearbeiteter Diproche-Aufgaben und dem Vorliegen operativer Fehler, die mit $p = 0,031$ als signifikant anzusehen ist.

- Es gab eine schwach negative ($r = -0,212$) Korrelation zwischen der Anzahl bearbeiteter Diproche-Aufgaben zur Mengenlehre und dem Vorliegen operativer Fehler, die mit $p = 0,039$ als signifikant anzusehen ist.
- Es gab eine schwach negative ($r = -0,113$), aber nicht signifikante ($p = 0,274$) Korrelation zwischen der Anzahl richtig bearbeiteter Diproche-Aufgaben zur Mengenlehre und dem Vorliegen operativer Fehler.
- Es gab eine schwach negative ($r = -0,126$), aber nicht signifikante ($p = 0,225$) Korrelation zwischen der Anzahl richtig oder fast richtig bearbeiteter Diproche-Aufgaben zur Mengenlehre und dem Vorliegen operativer Fehler.

Die Hypothesen H_{55}, H_{65}, die einen negativen Zusammenhang zwischen dem Vorliegen operativer Fehler und der Anzahl insgesamt bearbeiteter Diproche-Aufgaben bzw. bearbeiteter Diproche-Aufgaben zur Mengenlehre betreffen, können damit als bestätigt gelten. Die Hypothesen H_{75} und H_{85} bezüglich eines negativen Zusammenhanges zwischen dem Vorliegen operativer Fehler und der Anzahl richtig bzw. richtig oder fast richtig bearbeiteter Diproche-Aufgaben konnten dagegen nicht bestätigt werden.

Diproche-Übungen vs. Anzahl argumentativer Schritte

Als argumentativer Schritt wurde jeder Satz einer Bearbeitung gezählt, der als Behauptung formuliert war; wurden mehrere Behauptungen durch ein "und" verbunden, wurde das entsprechend als ein Schritt gezählt. Im Durchschnitt enthielt eine Bearbeitung 5.52 Schritte, mit einem Minimum von 0 und einem Maximum von 13. Da Diproche-Übungen ein kleinschrittiges Vorgehen erfordern, wäre ein positiver Zusammenhang zwischen Diproche-Übungen und Schrittzahl zu erwarten. Die Datenauswertung ergab jedoch keinen signifikanten Zusammenhang zwischen den Diproche-Übungen und der Anzahl der Schritte. Die Hypothesen H_{16}, H_{26}, H_{36} und H_{46} sind damit durch die Daten nicht bestätigt worden.

Ein möglicher Grund hierfür könnte darin liegen, dass die Anzahl der für eine korrekte Lösung erforderlichen Schritte recht klein war: Eine von Diproche akzeptierte Lösung der Aufgabe (s.o.) bestand aus 6 Schritten, was dem Durchschnitt der auftretenden Schrittzahlen sehr nahe kommt. Deutlich höhere Schrittzahlen ergaben sich meist dadurch, dass unnötige oder nicht zielführende Argumentationsschritte vorlagen oder der Beweis von vornherein falsch strukturiert wurde (etwa als Mengengleichheitsbeweis, obwohl nur eine Teilmengenrelation zu zeigen war).

Damit liegt es nahe, statt der Schrittzahl eher die Abweichung der Schrittzahl von der für eine Diproche-Lösung erforderlichen Schrittzahl 6 zu erfassen. Hier ergab sich in der Stichprobe folgendes Bild:

- Zwischen der Anzahl insgesamt bearbeiteter Diproche-Aufgaben und der betragsmäßigen Abweichung der Schrittzahl von 6 ergab sich eine schwach negative Korrelation von $r = -0,281$, die mit $p = 0,006$ als hoch signifikant anzusehen ist.

- Zwischen der Anzahl der bearbeiteten Diproche-Aufgaben zur Mengenlehre und der betragsmäßigen Abweichung der Schrittzahl von 6 ergab sich eine schwach negative Korrelation von $r = -0,25$, die mit $p = 0,0142$ als signifikant anzusehen ist.

- Zwischen der Anzahl korrekt bearbeiteter Diproche-Aufgaben zur Mengenlehre und der betragsmäßigen Abweichung der Schrittzahl von 6 ergab sich eine moderat negative Korrelation von $r = -0,338$, die mit $p = 0,0008$ als hoch signifikant anzusehen ist.

- Zwischen der Anzahl korrekt der fast korrekt bearbeiteter Diproche-Aufgaben zur Mengenlehre und der betragsmäßigen Abweichung der Schrittzahl von 6 ergab sich eine moderat negative Korrelation von $r = -0,342$, die mit $p = 0,0007$ als hoch signifikant anzusehen ist.

Es liegt damit nahe, einen Zusammenhang zwischen Diproche-Übungen und Abweichungen von der minimalen erforderlichen Schrittzahl zu vermuten. Da diese Hypothese aber nicht zu den ursprünglich zu testenden Hypothesen gehörte, wäre hier zur Bestätigung eine erneute Datenerhebung erforderlich.

Diproche-Übungen vs. Anzahl und Anteil folgerichtiger Schritte

Wir betrachten nun die Gesamtzahl folgerichtiger Schritte sowie den Quotienten aus der Anzahl folgerichtiger Schritte und der Gesamtzahl der Schritte. Dabei gilt ein Schritt als folgerichtig, wenn er an der Stelle im Text, an der er erscheint, aus den verfügbaren Annahmen und den bisher erreichten Folgerungen mit einer elementaren Folgerungsregel ableitbar ist. Angesichts des Schwierigkeitsgrades der betrachteten Beweisaufgabe gelten hier nur solche Schritte als elementar, die aus der Anwendung der Definition eines mengentheoretische Operators oder einer einfachen aussagenlogischen Schlussregel bestehen. Syntaktisch nicht sinnvolle Folgerungen, etwa die Behauptung einer Menge oder Verknüpfungen von Aussagen durch mengentheoretische Operatoren, wurden als "nicht folgerichtig" gewertet. Bezüglich der Frage, was als elementarer Schritt anzusehen ist, ergibt sich damit die schon weiter oben diskutierte Unschärfe; allerdings gab es im Verlauf der Datenauswertung nur wenig Fälle, in denen diese Schwierigkeit auftrat: In den meisten Fällen, in denen ein nicht folgerichtiger Schritt gezählt wurde, war der fragliche Satz nicht sinnvoll oder, unabhängig von der Granularität, nicht ableitbar;

korrekte, aber nicht ausreichend begründete Folgerungen traten hauptsächlich dadurch aus, dass das Beweisziel ohne eine vorherige zielführende Argumentation behauptet wurde. Entsprechend der im vorigen Abschnitt erläuterten Zählweise von "Schritten" galt ein Satz, der mehrere Aussagen durch eine Konjunktion verknüpfte, dann als fehlerhafte Folgerung, sobald eine dieser Aussagen nicht oder nicht durch einen elementaren Schritt ableitbar war. Im Durchschnitt enthielt eine Bearbeitung 2,7 folgerichtige Schritte.

In der Stichprobe erhielten wir folgende Ergebnisse:

- Zwischen der Anzahl insgesamt bearbeiteter Diproche-Übungen und der Anzahl richtiger Schritte ergab sich eine schwach positive Korrelation ($r = 0,262$, die mit $p = 0,01$ als hoch signifikant anzusehen ist.

- Zwischen der Anzahl bearbeiteter Diproche-Übungen zur Mengenlehre und Anzahl richtiger Schritte ergab sich eine schwach positive Korrelation ($r = 0,293$), die mit $p = 0,004$ als hoch signifikant anzusehen ist.

- Zwischen der Anzahl korrekt bearbeiteter Diproche-Übungen zur Mengenlehre und der Anzahl richtiger Schritte ergab sich eine schwach positive Korrelation ($r = 0,168$), die aber nicht als signifikant anzusehen ist ($p = 0,106$).

- Zwischen der Anzahl korrekt oder fast korrekt bearbeiteter Diproche-Übungen zur Mengenlehre und der Anzahl richtiger Schritte ergab sich eine schwach positiver Korrelation ($r = 0,215$), die mit $p = 0,038$ als signifikant anzusehen ist.

- Zwischen der Anzahl insgesamt bearbeiteter Diproche-Übungen und dem Anteil richtiger Schritte ergab sich eine schwach positive Korrelation ($r = 0,239$), die mit $p = 0,02$ als signifikant anzusehen ist.

- Zwischen der Anzahl bearbeiteter Diproche-Übungen zur Mengenlehre und dem Anteil richtiger Schritte ergab sich eine schwach positive ($r = 0,263$) und signifikante Korrelation ($p = 0,0104$).

- Zwischen der Anzahl korrekt bearbeiteter Diproche-Übungen zur Mengenlehre und dem Anteil richtiger Schritte ergab sich eine schwach positive ($r = 0,192$), aber nicht signifikante Korrelation ($p = 0,064$).

- Zwischen der Anzahl korrekt oder fast korrekt bearbeiteter Diproche-Übungen zur Mengenlehre und dem Anteil richtiger Schritte ergab sich eine schwach positive ($r = 0,231$) und signifikante Korrelation ($p = 0,025$).

Die Hypothesen H_{17}, H_{27} und H_{47} werden durch die Daten also bestätigt, während H_{37} nicht bestätigt ist. Ebenso werden die Hypothesen H_{18}, H_{28} und H_{48} durch die Daten bestätigt, nicht aber H_{38}.

Insgesamt kann ein positiver Zusammenhang zwischen Bearbeitungshäufigkeit bei den Diproche-Aufgaben und richtigen Folgerungsschritten damit als bestätigt betrachtet werden. Der Zusammenhang zwischen dem Bearbeitungserfolg bei den Diproche-Aufgaben und richtigen Folgerungsschritten ist dagegen deutlich schwächer. Insbesondere ist interessant, dass die Einbeziehung "fast richtiger" Lösungen zu einem deutlich stärkeren Zusammenhang führt. Hier deutet sich an, dass die Beachtung der Normierung, die Diproche über die übliche Darstellungspraxis hinaus vornimmt (etwa durch die oben besprochene Darstellungsform für Kontrapositionsbeweise), nicht zu verbesserten Beweiskompetenzen korrespondiert, soweit diese durch eine menschliche Korrektur erfasst werden.

Diproche-Übungen vs. Vorliegen einer erkennbaren Beweisform

Wie bereits oben erwähnt, war bei 90 der 93, also fast 97% der betrachteten Lösungen eine klare Beweisform erkennbar, so dass eine Untersuchung des Zusammenhanges mit den Bearbeitungen der Diproche-Aufgaben nicht sinnvoll ist. Ein Grund für diese Abweichung von den in den Vorjahren gesammelten Erfahrungswerten dürfte in dem gegenüber früheren Klausuren deutlich veränderten Eingabemodus liegen: Statt handschriftliche Lösungen zu verfassen, mussten die Bearbeitungen über eine Tastatur eingegeben werden, was die Abgabe von Bildern, Pfeildiagramme etc. nahezu unmöglich machte.

Diproche-Übungen vs. Verwendung einer passenden Beweisstruktur

Eine angemessene Beweisstruktur wird dann als gegeben angesehen, wenn der Text ein korrekter Beweis wäre, sofern alle Folgerungsschritte richtig wären. Das ist insbesondere etwa dann nicht der Fall, wenn das Beweisziel zu Beginn des Textes angenommen wird. Im vorliegenden Fall, wo das Beweisziel eine Teilmengenbeziehung war, bot sich als Beweisstruktur an, zunächst die Variablen A, B und C als Mengenvariablen einzuführen, dann ein beliebiges Element x aus $((A \times B) \setminus (B \times C))$ zu fixieren und darauf hinzuarbeiten, dass $x \in ((A \setminus B) \times C)$. Wo diese Schritte – Variableneinführung, Wahl eines Elementes und Folgerung – vorlagen, wurde, sofern nicht etwa noch weitere Annahmen eingeführt wurden o.ä., eine richtige Beweisstruktur festgestellt; potenziell mögliche abweichende, aber ebenfalls korrekte, Strukturierungsmöglichkeiten, etwa eine Herleitung von $x \notin ((A\times B)\setminus(B\times C))$ aus $x \notin ((A\setminus B)\times C)$ wurden ebenfalls als "korrekt strukturiert" gezählt, traten aber kaum auf. Insgesamt wiesen 47 der 93 Bearbeitungen, also

etwas über der Hälfte, eine solche dem Beweisziel angemessene Strukturierung auf. Die Fehler ließen sich überwiegend einer der folgenden Kategorien zuordnen:

- Es wurde eine Beweisstruktur gewählt, die für eine Mengengleichheit angemessen wäre, nämlich eine beidseitige Mengeninklusion; dabei wurde meist nur die Richtung stichhaltig begründet, nach der in der Aufgabenstellung nicht gefragt war.

- Für die Mengenvariablen A, B, C wurden spezielle Mengen eingesetzt und die Behauptung wurde dann lediglich für dieses eine Beispiel überprüft.

- Es wurde eine Implikation bewiesen, die in die falsche Richtung ging, also zur umgekehrten Inklusion gehörte.

- Der Allquantor wurde nicht aufgelöst; stattdessen tauchten unvermittelt Variablen auf.

- Nach Auflösung des Allquantors folgte keine einer Implikation angemessene Beweisstruktur.

- Die Behauptung wurde durch Verweis auf ein "Distributivgesetz" belegt; für die mengentheoretische Differenz war ein solches aber weder aus der Vorlesung noch aus den Übungen bekannt; vielmehr stellt die Aufgabe offenbar eine Richtung des Distributivgesetzes für die mengentheoretische Differenz dar, so dass eine Bezugnahme auf das Distributivgesetz zirkulär war.

Punkt (2) wurde als besonders prominente Fehlerstrategie separat betrachtet.

Anhand der Daten konnte kein Zusammenhang zwischen Bearbeitung bzw. Bearbeitungserfolg bei den Diproche-Aufgaben und der Wahl einer angemessenen Beweisstruktur bzw. der Fehlerstrategie "Beweis durch Beispiel" festgestellt werden. Die Hypothesen H_{19}, H_{29}, H_{39}, H_{49}, H_{510}, H_{610}, H_{710} und H_{810} konnten nicht bestätigt werden.

Dies steht in einem gewissen Kontrast zum Ergebnis der Studierendenbefragung, bei der 62% der Befragten tendenziell die Frage bejaht hatten, ob die Verwendung von Diproche dabei hilft, zu lernen, Beweistexte zu strukturieren. Ein Grund hierfür könnte in der eingangs erläuterte Verzerrung der Daten infolge der Abgabe in Gruppen liegen. Möglich ist es indes auch, dass die Selbsteinschätzung der Studierenden in Bezug auf die Kompetenz zur Strukturierung von Beweistexten nicht dem Bild entsprach, dass sich aus der vorliegenden Datenauswertung ergibt.

Diproche-Übungen vs. Typenfehler

Von den untersuchten 93 Lösungstexten enthielten 57, also etwas über 61%, Typenfehler durch die Verwendung nicht eingeführter Variablen, die Verwendung von Mengen als Aussagen (etwa in Sätzen wie "Es gelte $A \times (B \setminus C)$" oder die Verwendung von Aussagen als Mengen (etwa in der Verknüpfung zweier Aussagen mit dem mengentheoretischen Differenzoperator). Dabei trat der Fehler "Menge als Aussage" am häufigsten auf: 47 der 93 Lösungstexte, also über 51%, enthielten mindestens eine Verwendung einer Menge als Aussage. Nicht eingeführte Variablen traten in 26 Bearbeitungen auf, was einem Anteil von knapp 28% entspricht. Die Verwendung von Aussagen als Mengen war demgegenüber verhältnismäßig selten und fand sich in 12 Bearbeitungen, also knapp 13%. Die Verwendung von Mengen als Aussage korrelierte moderat negativ ($r = -0,347$) mit der Gesamtpunktzahl bei der Beweisaufgabe in der Klausur; die Korrelation ist mit $p = 0,0006$ als hoch signifikant anzusehen. Da das Bewertungsschema der Klausur keine starken Punktabzüge für diesen Fehler vorsah, ist das ein Hinweis darauf, dass die Verwendung von Mengen als Aussagen auf eine Schwierigkeit im Umgang mit Mengen hinweist, die die Bearbeitung von Beweisaufgaben beeinträchtigt.

Zwischen Bearbeitungshäufigkeit bzw. Bearbeitungserfolg bei den Diproche-Aufgaben und dem Vorliegen der Fehler "Menge als Aussage", "Aussage als Menge", der Einführung der verwendeten Variablen sowie dem Vorliegen von Typenfehler im Allgemeinen konnte kein signifikanter Zusammenhang festgestellt werden; in allen Fällen lag der Korrelationskoeffizient unter 0,17 und das Signifikanzniveau über $0,1$. Ein schwach negativer Zusammenhang zeigte sich zwischen der Anzahl richtig bearbeiteter Beweisaufgaben zur Mengenlehre und der Verwendung von Mengen als Aussagen ($r = -0,114$), der aber mit $p = 0,273$ nicht signifikant ist; interessant ist hier allerdings, dass diese Korrelation deutlich stärker ist als die zwischen der Anzahl richtig oder fast richtig bearbeiteter Beweisaufgaben zur Mengenlehre und der Verwendung von Mengen als Aussagen ($r = -0,08$). In Bezug auf das Vorliegen von Typenfehler ergaben sich ebenfalls schwach negative Korrelationen, die jedoch auch nicht signifikant waren; hervorzuheben ist jedoch auch hier, dass die Korrelation zwischen dem Vorliegen von Typenfehlern und der Anzahl korrekt bearbeiteter Beweisaufgaben zur Mengenlehre unter den betrachteten die stärkste ist.

Schließlich ergab sich ein schwach negativer Zusammenhang ($r = -0,105$) zwischen der Anzahl richtig bearbeiteter Aufgaben zur Mengenlehre und der Verwendung von Aussagen als Mengen, der aber ebenfalls nicht signifikant ist ($p = 0,314$). In allen anderen Fällen waren die jeweiligen Größen mit $0 > r > -0,1$ unkorreliert.

Die Hypothesen H_{53}, H_{55}, H_{510}, H_{511}, H_{512}, H_{513}, H_{514}, H_{63}, H_{65}, H_{610}, H_{611}, H_{612}, H_{613}, H_{614}, H_{73}, H_{75}, H_{710}, H_{711}, H_{712}, H_{713}, H_{714}, H_{83}, H_{85}, H_{810}, H_{811},

Table 8.38: Statistischer Zusammenhang zwischen Bearbeitungshäufigkeit und -erfolg bei den Diproche-Beweisaufgaben und dem Vorliegen von Typenfehlern

	r	p
Insgesamt bearbeitet	$-0,147$	$0,158$
Bearbeitet Mengenlehre	-0,122	0,158
Mengenlehre richtig/fast richtig	-0,147	0,158
Mengenlehre richtig	-0,169	0,104

H_{812}, H_{813}, H_{814} ließen sich daher durch die Daten nicht bestätigen.

Dieses Ergebnis ist insofern überraschend, als die Typenprüfung von Diproche die hier erfassten Fehlertypen erkennt und abfängt; bei einer regelmäßigen Arbeit mit dem System wäre also zu erwarten gewesen, dass eine Tendenz zu solchen Fehlern bemerkt und korrigiert wird. Ferner wurde die Frage, ob die Verwendung von Diproche dabei helfe, sich an die Einführung von Variablen vor ihrer Verwendung zu gewöhnen, in der Befragung der Studierenden von 65% tendenziell bejaht. Diese Einschätzung wird durch die hier erhobenen Daten aber nicht gestützt.

8.4.5 Fazit

Signifikante Zusammenhänge konnten festgestellt werden zwischen der Bearbeitungshäufigkeit und dem Vorliegen operativer Fehler sowie der absoluten Zahl und dem Anteil korrekter Folgerungsschritte. Es kann also als bestätigt gelten, dass zwischen der Beschäftigung mit den Diproche-Übungsaufgaben und dem korrekten Schlussfolgern sowie der korrekten Verwendung der Fachsprache ein positiver Zusammenhang besteht. Ob die Verwendung von Diproche eine förderliche Wirkung auf die betreffenden Kompetenzen hatte oder ob sprachlich und logisch kompetentere Studierende stärker dazu neigten, die Diproche-Aufgaben zu bearbeiten, geht aus den Daten freilich nicht hervor (wohl aber gibt die Befragung der Studierenden in dieser Hinsicht einigen Aufschluss).

Für die anderen erfassten Größen waren die Effekte zu schwach, um als Bestätigung der jeweiligen Hypothesen gelten zu können. Insbesondere im Hinblick auf die Verwendung einer geeigneten Beweisstruktur ist dieses Ergebnis überraschend, da Diproche einerseits strukturell fehlerhafte Texte auch als fehlerhaft meldet, andererseits die Diproche-Übungsaufgaben zur Mengenlehre die Verwendung der hier angemessenen Beweisstruktur zum Beweis einer Teilmengenbeziehung immer wieder erforderten. Ferner war die hier betrachtete Aufgabe, wie schon oben erwähnt, den Teilaufgaben (3) und (4) der Diproche-Aufgabe von Blatt 11 inhaltlich wie formal sehr ähnlich, so dass bei einer

korrekten Bearbeitung dieser Aufgaben auch eine weitgehende Bewältigung der Klausuraufgabe zu erwarten gewesen wäre. Als Ursachen hierfür kommen die bereits zu Beginn dieses Abschnitts erläuterten Umstände in Betracht, dass (i) ein erheblicher Anteil der Studierenden keine Lösungen zu den Diproche-Aufgaben abgegeben haben und (ii) die paarweise Abgabe und die Zusammenarbeit in größeren Gruppen die Zuordnung von Abgaben und Klausurerfolg störten.

Kapitel 9

Ausblick: Entwicklungsmöglichkeiten

Der Diproche-Code wird bald nach Abschluss dieser Arbeit im Internet unter einer Open-Source-Lizenz frei zur Verfügung stehen. Jede/r Interessierte/r ist herzlichen eingeladen, Verbesserungen, Ergänzungen oder andere Änderungen vorzunehmen, solange die so produzierten neuen Versionen wiederum frei zur Weiterentwicklung und -verwendung zur Verfügung gestellt werden. Wir zeigen an dieser Stelle einige Möglichkeiten auf, das System weiterzuentwickeln und einzusetzen, die einerseits für uns selbst als Arbeitsprogramm, andererseits aber auch anderen EntwicklerInnen als Anregung dienen mögen.

9.1 Zur Perspektive des Einsatzes in der Schule

Bereits im vorigen Kapitel wurde disktuiert, inwieweit Diproche zur Förderung mathematischer Grundkompetenzen geeignet ist, die in den Bildungsstandards der Kultusministerkonferenz für das Fach Mathematik ([122]) aufgeführt sind. Gerade zur Förderung argumentative Kompetenzen bietet sich ein System zur automatischen Prüfung von Argumenten durchaus auch für den schulischen Einsatz an. Als Vorbild kann hier der Einsatz des GEOBEWEIS-Systems von G. Holland (s.o.) im Geometrie-Unterricht der Mittelstufe durch Hinrich Lorenzen ([138]) angesehen werden. Wie weit sich das Diproche-System für den Einsatz im Schulunterricht eignet, wie die Einbindung aussehen könnte und welche Erfolge sich damit ggf. erzielen lassen ist eine Frage, die in künftigen Projekten untersucht werden sollte.

9.2 Bewertung von Beweisschritten

Einige Systeme, wie etwa das oben beschriebene QED-Tutrix, bieten eine Evaluation von Beweisschritten im Hinblick auf die Frage an, ob diese “zielführend”

sind. In speziellen Kontexten, in denen die Zahl "sinnvoller" Beweise überschaubar bleibt, wie etwa der Aussagenlogik oder der Booleschen Mengenlehre (und möglicherweise auch einigen Beweisen zum Thema gerade/ungerade Zahlen), könnte die dort verwendete Technik unter Verwendung des HPDIC-Graphen sich hierfür als nützlich erweisen. Für Themenfelder und Aufgaben, in denen diese Möglichkeit nicht mehr besteht – etwa bei Aufgaben zur Teilbarkeit – könnten die in Zinn [217] diskutierten Techniken eingesetzt werden, Beweistexte mithilfe von automatischen Beweisplanern mit einem Vorrat an Beweisstrategien abzugleichen, auf diese Weise den Beweisplan zu "raten" und Schritte danach zu evaluieren, wie gut sie sich in den Plan integrieren lassen.

9.3 Erkennung unnötiger Beweisschritte

Sowohl für die Beurteilung fertiger Beweistexte als auch als Hilfestellung beim Konstruieren und Verschriftlichen von Beweisen dürfte es hilfreich sein, auf Textstellen hinzuweisen, an denen ein Beweisziel sich mit wenigen Schritten erreichen läßt. Hierzu könnte ein im Hintergrund arbeitender ATP wie beim Tippgeber versuchen, den bisher eingegebenen Text zu einem vollständigen Beweis des aktuellen Zieles zu ergänzen. Die so erhaltene Information könnte dann auf mindestens zwei Arten zu einer Rückmeldung an den bzw. die BenutzerIn führen: Einerseits könnte das System bei der Beurteilung eines Beweistextes auf "Umwege" hinweisen, indem es Textstellen hervorhebt, von denen aus das fragliche Beweisziel sich in deutlich weniger Schritte erreichen läßt als im Text gebraucht wurden. Andererseits könnte die Anzahl der Schritte, die der ATP benötigt, um den vorhandenen Text zu einem vollständigen Beweis zu ergänzen, als Maß für den Fortschritt im Verlauf eines Beweistextes verwendet werden; hierdurch könnten einerseits Deduktionsschritte automatisch als mehr oder weniger zielführend bewertet werden, andererseits könnte die Aufmerksamkeit der/des Benutzers/In auf Stellen gelenkt werden, an denen sie oder er "so gut wie fertig" ist. Auf diese Weise könnten sich einige der Effekte, die im Rahmen des oben diskutierten Systems "QED-Tutrix" durch die Verwendung des HPDIC-Graphen erreicht werden womöglich in deutlich weniger stark kontrollierte Kontexte übertragen. Wie viel für Menschen heuristisch nützliche Information sich allerdings tatsächlich aus der Länge des kürzesten formalen Beweises, den ein ATP findet, gewinnen läßt, ist eine Frage, die sich nur empirisch beantworten läßt. Immerhin ist zu erwarten, dass ein solcher Ansatz in Bereichen zielführend ist, in denen formale und natürlichsprachliche Beweise sprachlich und methodisch noch recht ähnlich funktionieren, wie etwa in der Aussagenlogik oder der Booleschen Mengenlehre.

9.4 Erweiterung der Ausdrucksmittel auf diagrammatische Beweise

Wie in der Eingangsbetrachtung diskutiert (vgl. Abschnitt 4.5), eignet sich das Diproche-System vor allem zur Einübung solcher Aspekte und Techniken des mathematischen Beweisens, die einen deduktiven und regelhaften Charakter tragen; Argumente, die auf dem kreativen Einsatz von Anschauung, Perspektivwechseln oder Gedankenexperimenten beruhen (wie sie z.B. für die Kombinatorik typisch sind, wo es häufig darum geht, eine zu zählende Menge auf die richtige Art zu "sehen"), sind dagegen kaum mit vertretbarem Aufwand in einem Sprachstil formulierbar, der für ein System wie Diproche zugänglich ist.[1]

Besonders eindrucksvolle Beispiele für Beweise dieses Typs sind "Bildbeweise", die im Extremfall gänzlich ohne sprachliche Anteile durchaus geeignet sind, eine rational begründete Überzeugung hinsichtlich der Richtigkeit einer Behauptung zu generieren.

Es würde zweifellos eine deutliche Steigerung der Reichweite von Diproche bedeuten, zumindest die automatische Verifikation auch auf solche Beweise zu erweitern und dadurch eine automatisch unterstützte Förderung von mathematischer Kreativität und Anschauung zu ermöglichen.

Tatsächlich ist die Möglichkeit zur automatischen Verifikation diagrammatischer Beweise bereits eingehend erwogen und ein entsprechendes System entwickelt worden, nämlich M. Jamniks "Diamond", siehe z.B. Jamnik [119], Bundy und Jamnik [23] oder Jamnik, Bundy und Green [118]. Für den didaktischen Einsatz ist dieses System in der aktuellen Form ungeeignet: Es verfügt z.B. weder über ein angenehm handhabbares Interface noch über die Möglichkeit, differenzierte Rückmeldungen zu liefern etc. Was die Arbeiten zu diesem Thema aber jedenfalls zeigen, ist die prinzipielle Möglichkeit, das Einsatzgebiet von Diproche auch auf diagrammatische Beweise auszudehnen.

9.5 Systematisierung der Fehlerdiagnose

Die Fehlschlussdiagnose durch den Anti-ATP beruht derzeit auf der persönlichen Erfahrung des Autors mit der Korrektur von Klausuren und Übungsaufgaben, ergänzt durch Hinweise von KollegInnen, vereinzelte Arbeiten zu Umformungsfehlern, sowie einigen Beiträgen der Argumentationstheorie zu formalen Fehlschlüssen. Es wäre ein sicherlich lohnendes Projekt, den Anti-ATP

[1] Siehe hierzu aber die sehr anregenden Betrachtungen aus Tanswell [191] am Beispiel des bekannten Färbungsargumentes dafür, dass ein Schachbrett nach Entfernung zweier diagonal gegenüberliegender Eckfelder nicht mehr mit 2×1-Dominosteinen überdeckbar ist.

durch die systematische Nutzung von Ergebnissen aus der Argumentationstheorie, der Psychologie und der Didaktik zu zu systematisieren (insbesondere, was die Diagnose von Fehlern bei Termumformungen angeht) und empirisch zu evaluieren: Wie viele Fehler können durch das System diagnostiziert werden? Wie häufig treten verschiedene Fehlertypen auf? Lassen Fehlertypen sich zu größeren Klassen zusammenfassen?[2] Dies sind Fragen, deren Beantwortung voraussichtlich zu einer erheblichen Weiterentwicklung der Fehlerdiagnose beitragen kann.

9.6 Automatisiertes Erkennen neuer Fehlermuster

Der Anti-ATP, sowohl der für logische Fehlschlüsse wie auch der für fehlerhafte Umformungen zuständige Teil, ist derzeit darauf beschränkt, Fehlermuster aus einer Liste vorgegebener Typen zu erkennen. Eine gewisse Erweiterung wird ferner durch die Einführung von Meta-Typen erreicht wie etwa "allgemeine Distributivität" sowie durch die Verwendung der "Analogie-Diagnose", wie im Abschnitt zum Anti-ATP erläutert. Es ist zu erwarten, dass die im vorigen Abschnitt vorgeschlagene Systematisierung zu einer verbesserten Funktion des Anti-ATP führt, sowohl hinsichtlich der Anzahl diagnostizierbarer Fehler als auch bezüglich der Verlässlichkeit der Diagnose, d.h. der Frage, ob der vom System als Möglichkeit vorgeschlagene Fehlschluss dem fraglichen Beweisschritt tatsächlich zugrunde lag. Auf der anderen Seite scheint es neben den verbreiteten Fehlschlüssen auch immer wieder individuelle Fehlermuster zu geben, die keine noch so umfängliche Korpusanalyse erschöpfend auflisten wird.

Unter anderem aus diesem Grund wäre es also wünschenswert, wenn das System über die oben erwähnten Möglichkeiten hinaus Fehlertypen selbstständig bilden und erkennen könnte. Sobald ein/e BenutzerIn das System über längere Zeit verwendet und ein nennenswerter Vorrat an Bearbeitungsversuchen vorliegt, kann eine automatische Analyse der aufgetretenen Fehlern erfolgen. Für den Fall fehlerhafter Termumformungen könnte eine solche Analyse wie folgt vorgehen:

1. Auf eine gegebene (falsche) Gleichung $T_0 = T_1$ werden zunächst die entsprechenden 'allgemeineren Gleichungen' gewonnen, indem Teilterme

[2] So haben verschiedene KollegInnen in Gesprächen über den Anti-ATP angemerkt, dass viele Fehler Spezialfälle allgemeinerer Fehlvorstellungen zu sein scheinen, etwa der Annahme, dass Operatoren sich grundsätzlich distributiv verhalten. Unter diesem Gesichtspunkt sind Fehler wie etwa $(a+b)^2 = a^2 + b^2$ und $\neg(a \wedge b) \leftrightarrow (\neg a \wedge \neg b)$ Ausdruck derselben Schwierigkeit. Ein weiteres Beispiel, wenn auch weniger allgemeines, für eine solche "Fehlerklasse" sind Fehler wie $\frac{n^2}{n^3} = \frac{2}{3}$ und $\frac{n^2-1}{n^4-1} = \frac{n^2}{n^4}$; hier besteht anscheinend eine rein formale Vorstellung vom Kürzen, derzufolge es darin besteht, "über und unter dem Bruchstrich das gleiche zu entfernen". Es ist eine interessante Herausforderung, die automatische Fehlerdiagnose auf solche allgemeineren Fehlertypen auszudehnen.

durch Variablen ersetzt werden (wobei gleiche Variablen stets für gleiche Teilterme stehen müssen, aber nicht notwendigerweise auch umgekehrt); dadurch entsteht eine Liste $L = [G_1, ..., G_n]$ von allgemeineren Gleichungen, aus denen sich $T_0 = T_1$ durch Einsetzen ergibt. Diese Liste L ist durch die Relation 'allgemeiner als' partiell geordnet.[3]

2. Durch eine Analyse der vorliegenden Texte eines/r Benutzers/In wird für jeden solchen 'allgemeineren' Typ von Gleichung festgestellt, wie häufig Fehler dieses Typs auftreten.

3. Schließlich werden mit einer geeigneten Heuristik solche Gleichungen ausgewählt, die einerseits möglichst speziell sind und es andererseits erlauben, einen möglichst großen Anteil an Fehlern zu erklären.

So ließen sich aus der Gleichung $(a + 1)^2 = a^2 + 1$ zunächst (u.a.) folgende allgemeineren Gleichungen bilden:

1. $(a + b)^2 = (a^2 + b^2)$
2. $(X + b)^2 = (X^2 + b^2)$
3. $(X + Y)^2 = (X^2 + Y^2)$
4. $X^2 = Y + Z$
5. $X^2 = Y$
6. $X = Y$

Von diesen sind (1) und (2) sicherlich zu speziell – die Erkennung eines Fehlers sollte nicht von der Benennung der Variablen abhängen – während (4)-(6) zu allgemein sind und so die "Struktur" des Fehlers verdecken. Hier sollte sicherlich (3) als "korrekte" Diagnose ausgewählt werden.

Ebenfalls lohnend dürfte die Frage sein, wie weit etwa die in VanLehn [198] für den Bereich der elementaren Arithmetik vorgestellten Techniken zur automatischen Generierung von Fehlermustern sich auf den hier vorliegenden Kontext anwenden lassen.

[3]Man beachte, dass jede Gleichung, die durch so eine "Abstraktion" aus einer fehlerhaften Gleichung entsteht, selbst fehlerhaft sein muss – schließlich läßt die ursprüngliche fehlerhafte Gleichung sich aus ihnen durch Substitution gewinnen.

9.7 Interaktive Aspekte der Fehlerdiagnose

Die Fehlerdiagnose durch den Anti-ATP dient dazu, bis zu einem gewissen Grad die Hilfestellung zu automatisieren, die ein/e menschliche/r KorrektorIn dadurch leistet, dass ein zugrundeliegendes Fehlschlussmuster identifiziert und als solches benannt und erläutert wird. Diese Erläuterung kann etwa durch ein Beispiel erfolgen, in dem das fragliche Muster zu einer offensichtlich falschen Folgerung führt.[4] Durch solche Beispiele wird einerseits die Fehlerhaftigkeit des Schlussmusters deutlicher, andererseits kann ein gut gewähltes Beispiel auch als Gedächtnisstütze dienen. In der derzeitigen Form beschränken sich die Rückmeldungen des Anti-ATP darauf, Fehlschlüsse zu benennen. Dies ließe sich etwa dadurch ausbauen, dass bei erkannten Fehlschlüssen eine weitergehende Erläuterung angeboten wird, die einerseits aus anschaulichen Beispielen, andererseits aber auch aus einigen Fragen zur Überprüfung des Verständnisses bestehen könnte.

9.8 Weitere "Spielwiesen"

Offensichtlich beschränken sich die Einsatzmöglichkeiten von Diproche nicht auf die bisher realisierten Themenfelder Aussagenlogik, Boolesche Mengenlehre, reelle Zahlen, Gruppentheorie, axiomatische Geometrie und elementare Zahlentheorie. Andererseits ist auch nicht jedes Themenfeld gleich gut für einen Einsatz von Diproche geeignet. Einige Kriterien für geeignete Themenfelder sind:

- Eine einfache und stabile Ontologie: Es sollte eine feststehende und überschaubare Menge von relevanten Objekttypen geben, die über eine überschaubare Anzahl an Relationen zusammenhängen. Schon aus diesem Grund ist etwa elementare Zahlentheorie geeigneter als analytische Zahlentheorie, in der außer über natürliche Zahlen noch über Mengen natürlicher Zahlen, reelle Zahlen, komplexe Zahlen, komplexe Funktionen etc. zu reden wäre.

- Eine überschaubare Menge an relevanten Eigenschaften und Begriffen, die weiterhin möglichst wenig "geschachtelt" sein sollten. Viele mathematische Theorien bauen komplexe hierarchische Begriffsnetze auf, in denen wiederholt bereits definierte Begriffe zur Definition neuer Begriffe herangezogen werden. Das Durchdringen eines Beweises in einem solchen

[4] Ein bekanntes Beispiel zur Erläuterung der Unzulässigkeit von Umkehrschlüssen ist etwa, dass man aus "Wenn es regnet, wird die Straße nass" nicht schließen kann, dass es immer, wenn die Straße nass ist, regnet – schließlich kann die Straße auch auf andere Weise nass werden oder es vom letzten Regen noch sein.

Bereich erfordert es dann, solche Schachtelungen bis zu einem gewissen Punkt wieder "aufzufalten". Dagegen ist Diproche in seiner derzeitigen Form darauf angewiesen, auf der Grundlage eines begrenzten und festen Regelsatzes zu arbeiten.

- Eine begrenzte Anzahl an Axiomen und Methoden. Unter anderem aus diesem Grund etwa die axiomatische Mengenlehre als Einsatzgebiet für Diproche eher ungeeignet, auch wenn gewisse Fragmente – wie etwa die Theorie der Ordinalzahlen – sich als zugänglicher erweisen könnten.

- Formalisierungsnähe: Diproche kann nur Beweise verarbeiten, die die fragliche Behauptung durch eine Folge kleiner logischer Schritte demonstrieren. In vielen Bereichen der Mathematik, die für StudienanfängerInnen durchaus geeignet sind, wie etwa der Elementargeometrie, der abzählenden Kombinatorik oder der Graphentheorie, wird dagegen anschaulich argumentiert, wobei formale Definitionen eher "pro forma" eingeführt und zugunsten eines klaren anschaulichen Verständnisses rasch in den Hintergrund rücken. (Kaum jemand dürfte etwa beim Beweis des Kriteriums für die Existenz von Euler-Kreisen an die Definition eines Graphen als geordnetes Paar aus einer Menge und einer Relation denken.) Oft wird auf der Basis mehr oder weniger aussagekräftiger Skizzen argumentiert, ohne die das Argument unverständlich ist. Es ist offensichtlich, dass solche Bereiche für eine Behandlung mit Diproche ungeeignet sind.

Bereiche, auf die diese Kriterien zutreffen und die zugleich für StudienanfängerInnen geeignet sind, sind etwa:

- Theorien algebraischer Strukturen wie die Theorien der Vektorräume, der angeordneten Gruppen, Ringe, Körper oder der Boolesche Algebren.

- Ordnungstheorien, wie etwa die Theorien der linearen Ordnungen, der partiellen Ordnungen, der dichten linearen Ordnungen (mit und ohne Endpunkte), der Wohlordnungen etc. Auch hier findet sich das Zusammenwirken von formaler und anschaulicher Ebene, das wir oben als Vorzug der axiomatischen Geometrie erwähnt haben.

- Die Theorie der komplexen Zahlen. Hierzu wäre es nötig, die Spielwiese "reelle Zahlen" entsprechend anzupassen.

- Trigonometrische Funktionen, Gleichungen und Ungleichungen.

9.9 Andere Sprachen

Da Diproche sich auf ein sehr einfaches und kontrolliertes Fragment der natürlichen mathematischen deutschen Fachsprache beschränkt, ist eine Übertragung in andere Sprachen ohne größere Schwierigkeiten möglich. Hierzu muß lediglich der Parser im Textannotationsmodul, der u.a. Typ (Annahme, Behauptung, Annotation, Deklaration) und Inhalt eines Satzes identifiziert, entsprechend ersetzt und die Rückmeldungen an den/die BenutzerIn übersetzt werden. Beides sollte zumindest für Sprachen, die das lateinische Alphabet verwenden und grammatisch gut verstanden sind, ohne größeren Aufwand möglich sein (insbesondere gilt das für das Englische und die romanischen Sprachen).

Eine Übertragung von Diproche in andere Sprachen hätte zum einen den offensichtlichen Effekt, das System für einen breiteren Kreis an NutzerInnen zugänglich zu machen. Als Folge davon wäre auf weitere produktive Rückmeldungen ebenso zu hoffen wie auf Entwicklungen, die durch die Eingliederung in anders aufgebaute Curricula nahegelegt werden (etwa weitere Themenfelder). Zugleich würde der potenzielle Kreis der EntwicklerInnen erweitert. Ein weiteres Einsatzgebiet wäre der Einsatz im deutschen Sprachraum, wo es helfen könnte, erste Fähigkeiten in der Formulierung mathematischer Beweise in einer Fremdsprache, insbesondere im Englischen, zu erwerben.

Die in dieser Hinsicht besonders wichtige Übertragung ins Englische ist für die meisten Diproche-Komponenten bereits vorgenommen worden: Insbesondere steht der Beweisprüfer für Texte zur Verfügung, die in einem kontrollierten Fragment des Englischen verfasst sind und gibt auch englische Rückmeldungen.

9.10 Verfeinerung und Ergänzung der Eingabesprache

Die derzeitige Sprache von Diproche fängt einen guten Teil der Formulierungen ein, in denen formal korrekte Lösungen von Übungsaufgaben im Anfängerbereich ausgedrückt werden. Dennoch bleibt hier natürlich Raum für Verbesserungen. Sicherlich gibt es einige Formulierungen für Behauptungen, Folgerungen, Annahmen etc., die im System derzeit nicht vorgesehen sind. Eine umfangreiche und systematische Korpusstudie an Anfängerübungen bzw. den zugehörigen Musterlösungen wäre ein Weg, derartige Lücken aufzuspüren und zu schließen.

Jenseits der Ergänzung einzelner Wörter und Formulierungen sind verschiedene grundlegendere Erweiterungen der Ausdrucksmöglichkeiten denkbar. So ist es beispielsweise der Lesbarkeit oft zuträglich, Termumformungen im Fließtext wiederzugeben und die verwendeten Schritte dabei zu beschreiben, etwa in Wendungen wie den folgenden:

- Damit folgt $e^x = 2$. Anwenden des natürlichen Logarithmus auf beiden

Seiten der Gleichung liefert $x = \ln(2)$.

- Nun haben wir $\sqrt{x} = 4$. Quadrieren führt also auf $x = 16$.
- Also gilt $\frac{x+x}{x} = \sqrt{y}^2$. Vereinfachen auf beiden Seiten ergibt $2 = y$.

Die Formulierungen, mit denen Umformungen und ihre Ergebnisse gewöhnlich wiedergegeben werden, bilden einen hinreichend begrenzten Bereich, um ihn mit den in Diproche verwendeten linguistischen Methoden zu erfassen. Hier liegt eines von zahlreichen kleinen Teilprojekten, die zur Verbesserung von Diproche angesetzt werden könnten.

9.11 Interaktive Skripte

Ein interessantes Einsatzgebiet für Diproche könnte die Integration in interaktive Lernmedien wie Lehrbücher oder Vorlesungsskripte sein. Hierzu könnten zentrale Lehr- und Hilfssätze sowie Definitionen und ggf. Schluss- und Umformungsregeln in dem jeweiligen Lehrmaterial mit Bezeichnern versehen werden, die es ermöglichen, sich bei der Verwendung des Systems durch Formulierungen wie "Mit Lemma 5 folgt..." oder "Nach Definition einer Primzahl gilt..." darauf zu beziehen. Besonders attraktive Möglichkeiten ergeben sich hierbei aus der Verwendung von Hypertext; so könnten etwa Bearbeitungsfenster für Übungsaufgaben mit Korrektur- und Tippfunktion direkt in einen Lehrtext integriert werden. Solche Texte sind bereits für Lurch geschrieben worden (siehe Magnus [144]); ebenso existiert ein solches Skript mit interaktiven Elementen für die oben diskutierte Lean-basierte Einführung in das mathematische Beweisen, siehe Avigad et al. [137].

9.12 Verfeinerungen des Tippgebers

Die sicherlich hinsichtlich ihrer Implementierung anspruchsvollste der drei Arten von Lösungshinweisen, die dem/r AnwenderIn von Diproche zur Verfügung stehen, sind die "advanced hints" (s.o.), bei denen ein partieller Beweis automatisch ergänzt und dann ein passender Zwischenschritt vorgeschlagen wird. Die Güte dieser Art von Hilfestellung hängt offenbar einerseits von der Qualität des verwendeten automatischen Theorembeweisers ab, andererseits aber auch davon, wie geeignet der jeweilige Zwischenschritt als Hinweis tatsächlich ist.

Der im Hintergrund von Diproche arbeitende ATP ist speziell zu dem Zweck entworfen worden, solche Schritte nachzuvollziehen, die im Sinne einer Übungsaufgabe auf einem bestimmten Niveau als "elementar" gelten können. Zum

Auffinden von Beweisen ist er nur bedingt geeignet. Ein deutlicher Fortschritt bezüglich der Möglichkeit, Beweise auch größerer Länge automatisch zu ergänzen sollte sich durch die Anbindung professioneller ATPs wie LEAN (siehe etwa [133]) erreichen lassen.

Die auf diese Weise erzeugten Beweise sind allerdings im Allgemeinen weit von einer für einen Menschen gut lesbaren und strategisch durchschaubaren Darstellung entfernt. Ein beliebiger Zwischenschritt aus einem von einem solchen System erzeugten Beweis wird im Zweifel eine längliche Zeichenkette sein, die weder leichter zu beweisen ist als das ursprüngliche Beweisziel, noch zu diesem in einem erkennbaren Zusammenhang steht. Sinnvollere Hinweise sollten sich jedoch durch die Verwendung von automatischen Beweisassistenten oder Beweisplanern erzeugen lassen, die explizit strategische Muster verwenden, wie sie auch Menschen in Beweisen einsetzen.[5]

9.13 Integration von Beweisframes

Als Unterstützung bei der Lösungssuche wäre es potenziell nützlich, Textstrukturen für einige grundlegende Beweisstrategien wie den Widerspruchsbeweis, den Kontrapositionsbeweis, den Induktionsbeweis, Fallunterscheidungen oder eine Biimplikation als "Templates" oder "Frames"[6] zur Verfügung zu stellen, die auf Wunsch im Eingabefenster angezeigt werden und nun "nur" noch geeignet ausgefüllt werden müssen. Auf diese Weise wäre es für Studierende, die in der Wahl einer geeigneten Beweisstruktur und deren sprachlicher Darstellung noch unsicher sind, möglich, verschiedene strukturelle Ansätze bei der Lösung einer Aufgabe zu erproben. Ein solcher Ansatz wurde, allerdings nicht im Rahmen eines digitalen Systems, unter dem Namen "Proof Frameworks" in Selden et al. ([188]) vorgestellt.

Bei Auswahl von "Induktion" und aktuellem Beweisziel $\forall n \in \mathbb{N}\phi(n)$ würde sich etwa folgendes Schema ergeben:

[5] Siehe z.B. http://dream.inf.ed.ac.uk/projects/proof-plans-faq.html

[6] Zur Bedeutung von Frames in mathematischen Beweisen siehe etwa Fisseni, Sarikaya, Schmitt und Schröder [72] sowie Carl, Cramer, Fisseni, Sarikaya und Schröder [37].

Wir zeigen $\forall n \in \mathbb{N}\phi(n)$.

Induktionsanfang: Wir zeigen $\phi(1)$.
Beweis.
qed.

Induktionsschritt: Wir zeigen $\forall n \in \mathbb{N}(\phi(n) \to \phi(n+1))$.
Beweis.
qed.

Nun folgt mit Induktion $\forall n \in \mathbb{N}\phi(n)$.
qed.

Ebenfalls denkbar wäre eine Funktion, die eine für das Beweisziel passende Beweisstruktur automatisch ermittelt – eine solche Routine ist bereits als Teil des "Tippgebers" implementiert – und dann das dazu passende Schema anzeigt; so könnte etwa bei dem Beweisziel

$\forall n \in \mathbb{N}(\phi(n) \to \psi(n))$

der Beweis durch Fixierung einer beliebigen natürlichen Zahl n, gefolgt von einem direkten Implikationsbeweis, als passende Beweisstrategie identifiziert werden, womit dem/der BenutzerIn dann etwa das folgende, bereits partiell ausgefüllte, Strukturschema angeboten werden könnte:

Wir zeigen $\forall n \in \mathbb{N}(\phi(n) \to \psi(n))$.
Beweis. Es sei $n \in \mathbb{N}$. Angenommen, es gilt $\phi(n)$.
...
Also gilt $\psi(n)$.
qed.

Es ist zu hoffen, dass eine solche Unterstützung bei Wahl und Verwendung einer passenden Beweisstruktur dazu anregt, mit verschiedenen argumentativen Strukturen zu experimentieren ihre Möglichkeiten und Grenzen auf diese Weise besser kennen zu lernen sowie die Anzahl abgebrochener Lösungsversuche aufgrund fehlender Strategien zu verringern. Wie jede Form der Hilfestellung birgt natürlich auch diese die Gefahr, den durch das System erreichbaren Übungseffekt letztlich zu verringern: Indem etwa durch die Verwendung von strukturellen Beweistemplates die Anforderung entfällt, die zur Darstellung einer Beweisstruktur nötigen verbalen Marker selbst niederzuschreiben, findet eine geringere Gewöhnung an

die Verwendung solcher Marker statt.

9.14 Einsatz als Untersuchungsmittel für didaktische Fragestellungen

Durch die Erstellung von Sitzungsprotokollen, etwa die Speicherung der jeweils geprüften Texte, bietet Diproche die Möglichkeit, die Arbeit Studierender an Beweisaufgaben zu beobachten, ohne dass etwa von seiten der Studierenden eine Verbalisierung ihrer Gedankengänge erforderlich wäre, wie sie z.B. den Untersuchungen von Schoenfeld [178] zugrunde liegt. Durch den begleitenden Einsatz in der Lehre lassen sich so über längere Zeiträume Entwicklungen der Beweiskompetenzen beobachten. Dies für didaktische Untersuchungen zum Beweisenlernen fruchtbar zu machen ist ein für die Zukunft vorgesehenes Einsatzgebiet für Diproche.

Kapitel 10

Anhang A: Die Formelsyntax von Diproche

In diesem Teil gehen wir noch einmal detailliert auf die in Diproche verwendeten Typen formaler Ausdrücke und ihre Syntax ein.

10.1 Terme

Abhängig vom jeweiligen Teilgebiet gehört ein gewisser Vorrat an Funktions- und Konstantenzeichen zum festen Grundbestand an Ausdrücken in einem mathematischen Diskurs. Im Diproche-System schlägt sich das im Termkonzept nieder, das mit der jeweiligen "Spielwiese" variiert.

10.1.1 Aussagenlogische Terme

Zur Bildung eines aussagenlogischen Terms steht zunächst ein Vorrat von aussagenlogischen Variablen zur Verfügung; als Aussagevariablen dienen große und kleine lateinische Buchstaben (wobei zwischen Klein- und Großschreibung unterschieden wird), optional gefolgt von einer natürlichen Zahl als Index. Beispiele für Aussagevariablen sind also a, B, $x15$ oder $C3$.

Ferner können zwei aussagenlogische Ausdrücke durch einen der Junktoren $\wedge$, $\vee$, $\rightarrow$ und $\leftrightarrow$ zu einem neuen aussagenlogischen Ausdruck verbunden werden. Nach strikter Auslegung der üblichen Syntaxregeln müsste dabei auf vollständige Klammerung geachtet werden, d.h. mit ϕ und ψ wären auch $(\phi \wedge \psi)$, $(\phi \vee \psi)$ etc. aussagenlogische Ausdrücke, nicht aber etwa $\phi \vee \psi$. Diese Regel ist in Diproche dahingehend gelockert, dass äußere Klammern fortgelassen werden können. Ferner ist mit ϕ auch $(\neg\phi)$ ein aussagenlogischer Ausdruck; auch hier kann die Klammerung entfallen, wobei dann die Negation auf den ersten vollständigen aussagenlogischen Ausdruck bezogen wird, der auf sie folgt – so würde etwa $\sim \phi \vee \psi$ interpretiert als $((\sim \phi) \vee \phi)$. Auf die

Einführung anderer üblicher Konventionen zur Erleichterung der Notation – etwa das aufgrund der Assoziativität unproblematische Fortlassen von Klammern bei längeren Konjunktionen und Disjunktionen oder die Prioritätsregeln, die eine eindeutige Lesart von Ausdrücken wie $(a \wedge b \rightarrow c)$ erlauben – wurde bewusst verzichtet, um BenutzerInnen zu einer strikten Beachtung syntaktischer Regeln anzuhalten. Das erscheint insbesondere deshalb gerechtfertigt, als die Aussagenlogik beim intendierten Einsatz von Diproche in der Lehre zu einem frühen Zeitpunkt eingesetzt werden soll, in der zunächst einmal ein Verständnis für die im strengsten Sinn korrekte Ausdrucksform anzustreben ist. Erst wenn dieses erreicht ist, also die Notwendigkeit und der korrekte Gebrauch der Klammerung eingesehen ist und vereinfachte Schreibweisen als Abkürzungen von im strikten Sinn korrekt gebildeten Ausdrücken gesehen werden können, sind solche weitergehenden abkürzenden Konventionen legitim und sinnvoll. Da vor Erreichen eines solchen Verständnisses aber von der Aussagenlogik bereits zu anderen Themengebieten übergegangen werden sollte, haben wir uns im Rahmen der Aussagenlogik-Spielwiese zum Verzicht auf solche Konventionen entschieden.

Aussagenlogische Ausdrücke sind also z.B. $(a \rightarrow (\neg(b \vee (c \wedge (\neg a)))))$ oder $(a \rightarrow (b \rightarrow c(\rightarrow d)))$.

10.1.2 Prädikatenlogische Ausdrücke

Zur Formulierung inhaltlicher mathematischer Aussagen ist die Aussagenlogik ungeeignet. Die Ausdruckskraft wird deutlich erweitert durch die Einführung prädikatenlogischer Ausdrücke (genauer: durch die Einführung von Ausdrücken der erststufigen Prädikatenlogik; höherstufige Logiken sind zwar für viele Bereiche der Mathematik sehr nützlich (siehe etwa Simpson [182]), werden aber für die im Rahmen von Diproche behandelten Gebiete nicht benötigt), die zusätzlich zu den aussagenlogischen Junktoren noch Variablen, Quantoren, Funktions-, Relations- und Konstantenzeichen verwenden können. Die Diproche-Syntax der Prädikatenlogik ist dabei die übliche (siehe z.B. Ebbinghaus et al. [63]), wobei der Vorrat an Konstanten-, Funktions- und Relationszeichen von der jeweiligen Spielwiese abhängt:

- Als Variablen haben wir kleine und große lateinische Buchstaben, optional gefolgt von einer natürlichen Zahl. Jede Variable ist ein prädikatenlogischer Term.
- Jedes Konstantenzeichen ist ein prädikatenlogischer Term.
- Ist f ein k-stelliges Funktionszeichen und sind $a_1, ..., a_k$ prädikatenlogische Terme, so ist auch $f(a_1, ..., a_k)$ ein prädikatenlogischer Term.

- Abhängig von der jeweiligen Spielwiese können gewisse Funktionen – wie etwa $+$, $\cdot$, $\cup$ oder $\cap$ – auch infix notiert werden.

- Ist p ein k-stelliges Prädikatensymbol und sind $a_1, ..., a_k$ Variablen bzw. Konstantenzeichen, so ist $p(a_1, ..., a_k)$ ein prädikatenlogischer Ausdruck.

- Ist ϕ ein prädikatenlogischer Ausdruck, so auch $(\neg\phi)$.

- Sind ϕ und ψ prädikatenlogische Ausdrücke, so auch $(\phi \wedge \psi)$, $(\phi \vee \psi)$, $(\phi \rightarrow \psi)$ und $(\phi \leftrightarrow \psi)$. Wie bei der Aussagenlogik werden zwar das Fortlassen äußerer Klammern, nicht aber weitergehende abkürzende Konventionen wie das Fortlassen von Klammern in längeren Disjunktionen und Konjunktionen oder Prioritätsregeln zwischen den Junktoren unterstützt.

- Ist ϕ ein prädikatenlogischer Ausdruck und ist x eine Variable, so sind auch $\forall x : \phi$ und $\exists x : \psi$ prädikatenlogische Ausdrücke.

10.1.3 Boolesche Mengenterme

Zur Bildung eines Booleschen Mengentermes stehen als atomare Bestandteile Mengenvariablen zur Verfügung, zur Zeit sind das alle großen und kleinen lateinischen Buchstaben, optional gefolgt von einer natürlichen Zahl. Ferner können zwei Boolesche Mengenterme durch die Operatoren $\cup$ (Vereinigung), $\cap$ (Schnitt) sowie / (mengentheoretische Differenz) zu einem neuen Booleschen Mengenterm verbunden werden, wobei wiederum auf die Klammerung zu achten ist (auf abkürzende Konventionen wurde aus dem gleichen Grund wie in der Aussagenlogik verzichtet). Ferner ist mit T auch $(-T)$ ein Boolescher Mengenterm, der das Komplement von T bezeichnet.

Beispiele für Boolesche Mengenterme sind also etwa $(a \cap (b \cup c))$ und $-((\bar{a} \cup (-b)) \cap (-(b \cap c)))$.

10.1.4 Arithmetische Terme

Arithmetische Terme sind Teil der Spielwiesen "reelle Zahlen" und "elementare Zahlentheorie" und, in einer vereinfachten Form, der Spielwiese "Gruppentheorie".

Ein arithmetischer Term verwendet kleine lateinische Buchstaben als Variablen. Ferner sind als atomare Bestandteile in der Spielwiese "elementare Zahlentheorie" beliebige natürliche Zahlen zugelassen. Sind t_0 und t_1 arithmetische Terme, so auch $(t_0 + t_1)$, $(t_0 - t_1)$, $(t_0 \cdot t_1)$, $\frac{t_0}{t_1}$ und $t_0^{t_1}$. Als vereinfachende Konvention sind ferner Summen und Produkte beliebiger Länge ohne Klammerung zulässig; sind also etwa $t_0, ..., t_{10}$ arithmetische Ausdrücke, so auch $(t_0 - t_1 + t_2 - t_3 + t_4)$ und

$(t_0 \cdot t_1 \cdot t_2 \cdot t_3 \cdot t_4)$. Das verbreitete Fortlassen des Multiplikationszeichen wird dagegen derzeit (aus technischen, nicht aus didaktischen Gründen) nicht unterstützt.

Zulässig sind dagegen indizierte Summen und Produkte. Ist I eine Variable und sind t_0, t_1, t_2 arithmetische Terme, so sind auch $\Sigma(i = t_0)(t_1)t_2$ als Darstellung von $\Sigma_{i=t_0}^{t_1} t_2$ und $\Pi(i = t_0)(t_1)t_2$ als Darstellung für $\Pi_{i=t_0}^{t_1} t_2$ arithmetische Terme.

Beispiele für arithmetische Terme sind also z.B. $(\Sigma(i = 1)(n)2)^2$ oder $\Pi_{i=1}^{n}((i^2 + i + 1)/(i + 1))$.

Gruppen werden additiv geschrieben, das neutrale Element wird mit 0 bezeichnet. Zulässige Terme in der Spielwiese "Gruppentheorie" verwenden daher nur die Operatoren + und −, die Grundterme sind die Variablen. Wegen der Assoziativität der Gruppenoperation ist dabei keine Klammerung erforderlich. Abkürzende Schreibweisen wie etwa $3 \cdot a$ für $a + a + a$ (genauer: für $(a + (a + a))$ werden dagegen derzeit nicht unterstützt, ebensowenig wie indizierte Summen und Produkte. Ein Beispiel für einen zulässigen Term in der Spielwiese "Gruppentheorie" ist also $(a + b - c + 0 + a)$.

10.2 Umformungsketten

Termumformungen spielen als Bestandteile mathematischer Beweistexte in den meisten von Diproche bisher abgedeckten Themenfeldern eine wichtige Rolle, so in der Aussagenlogik, der Booleschen Mengenlehre und der elementaren Zahlentheorie oder der Gruppentheorie (weniger allerdings in der weiter oben behandelten Variante der axiomatischen Geometrie). Solche Umformungen können im Prinzip als eine Folge einzelner Gleichungen bzw. Äquivalenzen dargestellt werden, aus der zum Schluss die gewünschte Gleichung bzw. Äquivalenz mit der Transitivität der Gleichheit bzw. der logischen Äquivalenz gefolgert werden kann. Eine derartige Darstellungsweise ist jedoch ebenso lästig wie unnatürlich. In mathematischen Texten werden derartige Folgen daher als Ketten von (Un-)Gleichungen bzw. Äquivalenz- oder Folgerungsbeziehungen dargestellt, so dass statt "Es gilt $a = b$. Es gilt $b \geq c$. Es gilt $c = d$." etwa einfach "Es gilt $a = b \geq c = d$." geschrieben wird, woraus dann $a \geq d$ folgt. Streng genommen handelt es sich bei Ausdrücken wie "$a = b \geq c = d$" oder "$A \leftrightarrow B \leftrightarrow C$" um syntaktisch fehlerhafte Verwendungen der binären Operatoren $=$, $\geq$, $\leftrightarrow$ etc.; doch sind solche "Ketten" in der Praxis derart verbreitet und stellen gegenüber der "schrittweisen" Darstellung auch einen derartigen Vorteil dar, dass es im Sinn des Ziels einer der üblichen Darstellungspraxis bei Übungsaufgaben nahen Eingabesprache ist, sie im Rahmen von Diproche zu erlauben.

Derzeit erlaubt Diproche folgende Kettenausdrücke:

10.2.1 Folgerungsketten in der Aussagenlogik

In längeren aussagenlogischen Beweisen, und nachdem eine Reihe aussagenlogischer Regeln wie die Assoziativität von $\wedge$ und $\vee$, die Regeln für das "Hereinziehen" von Negationen in zusammengesetzte Ausdrücke etc. eingeführt wurden, kommen öfter Passagen vor, die eher den Charakter von Symbolmanipulationen als von Argumentationen haben und für die die von Diproche vorgesehene Darstellungsform für argumentative Beweise daher umständlich und künstlich wäre. So würde beispielsweise eine Passage wie die folgende

> Es gilt $\neg(a \wedge (b \vee c))$. Also gilt $\neg a \vee \neg(b \vee c)$. Damit folgt $\neg a \vee (\neg b \wedge \neg c)$. Also gilt auch $(\neg a \vee \neg b) \wedge (\neg a \vee \neg c)$.

üblicherweise eher dargestellt als:

> Es gilt $\sim (a\&(b \vee c)) \Leftrightarrow (\neg a \vee \neg(b \vee c)) \Leftrightarrow (\neg a \vee (\neg b\&\neg c)) \Leftrightarrow ((\neg a \vee \neg b)\&(\neg a \vee \neg c))$.

wodurch zugleich auch noch in knapper Form mitgeteilt werden kann, dass es sich jeweils nicht nur um Implikationen, sondern um Äquivalenzen handelt.

In Diproche sind daher innerhalb der Spielwiese "Aussagenlogik" aussagenlogische Implikationsketten zulässig; hierbei handelt es sich um Symbolfolgen, die abwechselnd aus aussagenlogischen Ausdrücken und einem der Symbole $\Rightarrow$ und $\Leftrightarrow$ bestehen, wobei das erste und das letzte Element aussagenlogische Ausdrücke sein müssen und die doppelt gestrichenen Pfeile verwendet werden, um einen Folgerungsschritt innerhalb des Beweises von der Implikation als Teil einer aussagenlogischen Formel zu unterscheiden. Dabei ist darauf zu achten, dass innerhalb von aussagenlogischen Äquivalenzketten äußere Klammern um die verwendeten Ausdrücke nur bei Negationen fortgelassen werden können.

Die oben angegebene Kette ist ein Beispiel für eine aussagenlogische Implikationskette, die von Diproche akzeptiert wird.

Durch eine aussagenlogische Implikationskette gilt die Äquivalenz des ersten und letzten Ausdrucks als bewiesen, wenn sämtliche Schritte Äquivalenzen, ausgedrückt durch $\Leftrightarrow$, sind; treten lediglich $\Leftrightarrow$ und $\Rightarrow$ auf, so gilt die Implikation vom ersten zum letzten Ausdruck als bewiesen; treten lediglich $\Leftrightarrow$ und $\Leftarrow$ auf, so gilt die Implikation vom letzten zum ersten Ausdruck als bewiesen. Treten sowohl $\Rightarrow$ als auch $\Leftarrow$ auf, so können die Schritte der Kette zwar einzeln verifiziert werden, es wird dadurch aber keine Information für weitere Folgerungsschritte zur Verfügung gestellt.

10.2.2 Ungleichungsketten in der Booleschen Mengenlehre

Wie in der Aussagenlogik gibt es auch in der Booleschen Mengenlehre typische Beweispassagen, die gewöhnlich – und sinnvollerweise – eher als Termumformungen denn als Argumentationsketten dargestellt werden, wie etwa die folgendes:

$$\overline{(A \cup (B \cap C))} = \bar{A} \cap \overline{B \cap C} = \bar{A} \cap (\bar{B} \cup \bar{C}) = (\bar{A} \cap \bar{B}) \cup (\bar{A} \cap \bar{C})$$

Eine Boolesche Kette im Diproche-Sinn ist eine Symbolfolge, die abwechselnd aus Booleschen Ausdrücken (wie oben definiert) und einem der Symbole $\subset$, $\supset$ und $=$ bestehen, wobei das erste und letzte Element Boolesche Ausdrücke sein müssen. Wiederum ist das oben gegebene Beispiel eines, das von Diproche im Rahmen der Spielwiese "Boolesche Mengenlehre" akzeptiert wird.

Eine Boolesche Kette wird von Diproche aufgefaßt als ein Teilbeweis, der einen Zusammenhang zwischen dem ersten und dem letzten Element der Kette zeigt. Entsprechend ermittelt Diproche nach Prüfung der einzelnen Schritte einer solchen Kette den stärksten Zusammenhang zwischen Anfangs- und Endglied, der sich aus der Kette ergibt (bei einer Folge aus $\subset$ und $=$, in der mindestens einmal $\subset$ auftritt, wäre das etwa $\subset$) und nimmt diese als Ausgangspunkt für weitere Argumentationsschritte. Es ist dagegen nicht möglich, sich im weiteren Verlauf des Beweistextes auf Zwischenschritte der Kette zu beziehen.

10.2.3 Ungleichungsketten in der Arithmetik

Termumformungen bei arithmetischen Termen sind – nach den schriftlichen Rechenverfahren, die aber gewöhnlich nicht als solche aufgefaßt bzw. herausgestellt werden – die ersten Formen symbolischer Argumentation, die in der schulmathematischen Ausbildung begegnet. Entsprechend ist zu erwarten, dass Studienanfänger mit dieser Form der Argumentation vertraut sind, weshalb Diproche hier gegenüber den üblichen Darstellungsformen recht entgegenkommend ist. Typische (Un)gleichungsketten wie etwa die folgende:

$$\Sigma_{i=1}^{n+1} i = \Sigma_{i=1}^{n} i + (n+1) = \frac{n(n+1)}{2} + (n+1) = \frac{n(n+1) + 2(n+1)}{2}$$

$$= \frac{(n+1)(n+2)}{2} = \frac{n^2 + 3n + 2}{2} > n^2 + n + 1$$

werden von Diproche als syntaktisch korrekt akzeptiert. Formal ist eine arithmetische Ungleichungskette eine Symbolfolge, die abwechselnd aus arithmetischen Ausdrücken und einem der Symbole $\geq$, $>$, $\leq$, $<$ und $=$ besteht, wobei das erste und letzte Glied arithmetische Ausdrücke sein müssen.

Ähnlich wie bei der Booleschen Mengenlehre gilt durch eine Kette die logisch stärkste Beziehung zwischen Anfangs- und Endterm als bewiesen, die sich aus der Kette ergibt; treten nur Gleichheiten auf, ist das die Gleichheit, tritt $=$ zusammen mit $\leq$ auf, ist das $\leq$, bei $=$, $<$ und $\leq$ ist das $<$ usw. Wenn sich aus der Kette keine Information bezüglich des Anfangs- und Endgliedes ergibt – etwa weil sowohl $<$ als auch $>$ auftreten – so wird durch die Kette auch keine Information für die weitere Verarbeitung zur Verfügung gestellt.

10.2.4 Implikationsketten in der Booleschen Mengenlehre

Neben Symbolmanipulationen Boolescher Ausdrücke, die die repräsentierte Menge unverändert lassen, gibt es Umformungen, die den Wahrheitswert von Gleichungen bzw. Teilmengenrelationen erhalten. Solche Folgerungsketten werden in der Darstellungspraxis mathematischer Übungsaufgaben häufig als Symbolketten geschrieben, wie etwa im folgenden Beispiel:

$$(a \subseteq b) \Rightarrow (a \cup b) = b \Leftrightarrow b = (a \cup b) \Leftrightarrow \overline{b} = \overline{a \cup b} \Leftrightarrow \overline{b} = \overline{a} \cap \overline{b}$$

Formal ist eine Boolesche Implikationskette eine Symbolfolge, die abwechselnd aus Booleschen Formeln und einem der Symbole $\Rightarrow$, $\Leftarrow$ und $\Leftrightarrow$ besteht. Boolesche Formeln sind dabei Boolesche Kombinationen von Formeln der folgenden Art:

- Formeln der Form $x \in a$ und $x \notin a$, wobei a ein Boolescher Ausdruck ist und x eine Variable.
- Formeln der Form $a \subseteq b$, $a \subset b$, $a \supseteq b$, $a \supset b$, $a = b$, wobei a und b Boolesche Ausdrücke sind.

Boolesche Implikationsketten werden von Diproche als syntaktisch korrekte Ausdrücke akzeptiert und geprüft. Als Ergebnis einer Booleschen Implikationskette wird für die weitere Verarbeitung die stärkste logische Beziehung zur Verfügung gestellt, die sich aus der Kette bezüglich Anfangs- und Endglied ergibt; treten ausschließlich Äquivalenzen auf, ist das die Äquivalenz, bei $\Rightarrow$ und $\Leftrightarrow$ ist es die Implikation von der ersten zur letzten Aussage und bei $\Leftarrow$ und $\Leftrightarrow$ die Implikation von der letzten zur ersten. Treten sowohl $\Rightarrow$ als auch $\Leftarrow$ auf, so wird die Kette zwar geprüft und als verifizierbar oder nicht verifizierbar gemeldet, trägt aber zum weiteren Beweistext keine Information bei. Ergibt sich auf diese Weise eine Implikation $A \rightarrow B$ oder eine Äquivalenz $A \leftrightarrow B$ zwischen zwei Aussagen A und B, von denen bei einer Implikation A bzw. bei einer Äquivalenz eine der Aussagen A bzw. B bereits gezeigt ist, so wird automatisch ebenfalls B bzw. die jeweils andere der beiden Aussagen den verfügbaren Aussagen hinzugefügt. Es ist also, wenn etwa A sowie $A \Rightarrow B \Leftrightarrow C \Rightarrow D$ gezeigt sind, nicht erforderlich, D noch einmal explizit als Folgerung aufzuführen.

10.2.5 Äquivalenzketten in der Arithmetik

Eine weitere Art von Umformungskette im Kontext von arithmetischen Termen sind Äquivalenzumformungen von Gleichungen und Ungleichungen. Diese kommen entweder durch äquivalente Termumformungen auf beiden Seiten zustande oder dadurch, dass auf beide Seiten dieselbe Operation angewendet wird; typische Operationen wären etwa, zu beiden Seiten dieselbe Konstante zu addieren oder beide Seiten mit der gleichen von 0 verschiedenen Zahl zu multiplizieren. Das wird oft durch eine zeilenweise geschriebene Folge von Gleichungen bzw. Ungleichungen geschrieben, die durch Äquivalenzpfeile verbunden sind, wobei zur Rechtfertigung der Äquivalenz eine "Regieanweisung" gegeben wird, die die jeweils auf beiden Seiten vorzunehmende Umformung anzeigt. Ein Beispiel für eine solche Umformungskette ist die folgende:

$$
\begin{aligned}
& x^2 + px = q \mid -q \\
\leftrightarrow\ & x^2 + px - q = 0 \\
\leftrightarrow\ & x^2 + px + (\tfrac{p}{2})^2 - (\tfrac{p}{2})^2 - q = 0 \\
\leftrightarrow\ & x^2 + 2\tfrac{p}{2}x + (\tfrac{p}{2})^2 - ((\tfrac{p}{2})^2 + q) = 0 \mid + ((\tfrac{p}{2})^2 + q) \\
\leftrightarrow\ & (x + \tfrac{p}{2})^2 = (\tfrac{p}{2})^2 + q
\end{aligned}
$$

Formal haben wir also eine Symbolfolge, deren Elemente abwechselnd Gleichungen bzw. Ungleichungen zwischen arithmetischen Termen und Äquivalenzzeichen sind, wobei die Äquivalenzzeichen zusätzlich durch eine Umformungsvorschrift kommentiert sein können (aber nicht müssen). Ketten dieser Art werden von Diproche akzeptiert. Die Ermittlung des dadurch für den weiteren Beweistext erreichten Ergebnisses funktioniert wie für Äquivalenzketten in der Booleschen Mengenlehre.

Kapitel 11

Anhang B: “Game of Def”-Aufgaben auf den Übungsblättern 7 und 8

11.1 Blatt 7

(a) Aufgabe 1

(b) Aufgabe 2

(a) Aufgabe 3

(b) Aufgabe 4

(a) Aufgabe 5

(b) Aufgabe 6

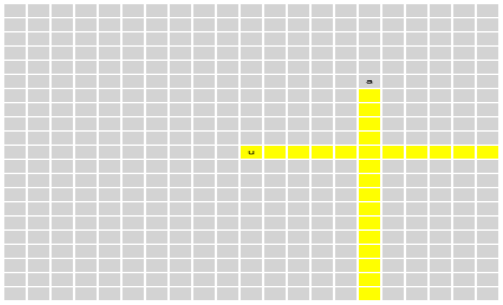
(a) Aufgabe 7

(b) Aufgabe 8

11.2 Blatt 8

(a) Aufgabe 9

(b) Aufgabe 10

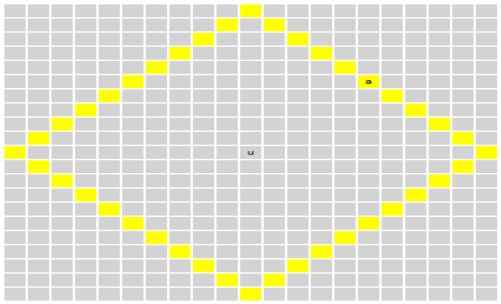


(a) Aufgabe 1

(b) Aufgabe 2

(a) Aufgabe 3

(b) Aufgabe 4



(a) Aufgabe 5

(b) Aufgabe 6

(a) Aufgabe 7

(b) Aufgabe 8

(a) Aufgabe 9

Kapitel 12

Anhang C: Fragebögen zur Evaluation der Diproche-Komponenten

12.1 Fragebogen zur Prüfkomponente

1. Einen Beweis in Diproche einzugeben, fällt mir leicht.

2. Diproche-Beweise ähneln den Beweisen, die ich in der Vorlesung und den Übungsgruppen kennen lerne.

3. Diproche hilft mir, Probleme in meinen Beweisansätzen zu erkennen.

4. Durch Diproche habe ich etwas darüber gelernt, wie man einen Beweistext hinsichtlich seiner Korrektheit beurteilt.

5. Diproche hat mir dabei geholfen, mich daran zu gewöhnen, Variablen vor ihrer Verwendung (mit Formulierungen wie "Sei x eine ganze Zahl." o.ä.) einzuführen.

6. Diproche hat mir geholfen, falsche Schlussweisen zu erkennen und zu vermeiden.

7. Durch den Einsatz von Diproche habe ich mich länger mit Beweisaufgaben beschäftigt, als ich es bei den gleichen Aufgabenstellungen ohne Verwendung des Systems getan hätte.

8. Durch den Umgang mit Diproche bin ich im Umgang mit Beweisen sicherer und selbstbewusster geworden.

9. Diproche hat mein Vertrauen in meine Beweistexte erhöht.

10. Bei Fehlermeldungen von Diproche kann ich oft auch einen Fehler in meinem Beweistext erkennen.

11. Es ermutigt mich, wenn das System eine Lösung als korrekt meldet.

12. Es ermutigt mich, wenn das System eine verbesserte Lösung als besser meldet als den vorherigen Versuch.

13. Fehlermeldungen durch das Systen entmutigen mich.

14. Die Arbeit mit Diproche macht mir Spaß.

15. Ich verstehe den Unterschied zwischen Logik- und Typenfehlern.

16. Ich habe mich auch über die gestellten Übungsaufgaben hinaus mit Aufgaben im Rahmen von Diproche beschäftigt.

17. Die Diproche-Rückmeldungen helfen mir dabei, einen Beweistext zu verbessern.

18. Diproche braucht manchmal zu lange, um einen Beweis zu prüfen.

19. Die Arbeit mit Diproche stellt eine sinnvolle Ergänzung zum Übungsbetrieb dar.

20. Die Arbeit mit Diproche hilft mir dabei, besser zu verstehen, wie Beweise funktionieren.

21. Die Arbeit mit Diproche hilft mir dabei, Beweise, die in der Vorlesung, den Übungen etc. vorkommen, besser zu verstehen.

22. Die Arbeit mit Diproche hilft mir dabei, die mathematische Fachsprache zu lernen.

23. Die Arbeit mit Diproche hilft mir, zu lernen, wie man Beweistexte strukturieren kann.

24. Auch Beweise, die ich für richtig halte, gebe ich manchmal in Diproche ein, weil es mir Freude macht.

25. Ich habe durch die Verwendung von Diproche viel gelernt.

26. Um Diproche-Aufgaben zu lösen, muss man die zu erwerbenden Kompetenzen bereits besitzen; daher ist Diproche nutzlos.

27. Ich würde mir mehr Erklärungen zu Diproche in der Vorlesung wünschen.

28. Ich würde mir mehr Beschäftigung mit Diproche in den Übungsgruppen bzw. den vorbereitenden Übungen wünschen.

Freie Fragen:

29. Besonders gut gefällt mir, dass:

30. Nicht so gut gefällt mir, dass:

31. Konkret habe ich folgende Verbesserungsvorschläge:

32. Bitte schätzen Sie grob, wieviel Zeit Sie pro Woche im Durchschnitt mit Diproche-Aufgaben verbracht haben.

12.2 Fragebogen zu den “Mathediktaten”

1. Die Arbeit an Mathediktaten macht mir Spaß.
2. Die Arbeit an den Mathediktaten hat mir geholfen, die korrekte Bildung logischer Formeln zu lernen.
3. Die Arbeit an den Mathediktaten hat mir geholfen, zu lernen, natürlichsprachliche Aussagen in aussagenlogischer bzw. mengentheoretischer Notation auszudrücken.
4. Rückmeldungen im Mathediktate-Modul helfen mir dabei, meine Lösung zu verbessern.
5. Es ermutigt mich, wenn das System eine Lösung als korrekt meldet.
6. Fehlermeldungen durch das Systen entmutigen mich.
7. Bei Fehlermeldungen im Mathediktate-Modul kann ich meistens erkennen, worin der Fehler in meiner Lösung besteht.
8. Durch den Umgang mit den ‘Mathediktaten’ bin ich im Umgang mit aussagenlogischen bzw. mengentheoretischen Formeln sicherer und selbstbewusster geworden.
9. Es kommt vor, dass richtige Lösungen als falsch gemeldet werden.
10. Es kommt vor, dass falsche Lösungen als richtig gemeldet werden.
11. Ich habe durch den Gebrauch des Mathediktate-Moduls viel gelernt.
12. Die Arbeit mit den ‘Mathediktaten’ hat mir geholfen, logische Ausdrücke – etwa in der Vorlesung oder den Übungen – besser zu verstehen.
13. Durch das Mathediktate-Modul habe ich mich länger mit der Verwendung logischer Ausdrücke beschäftigt, als ich es bei den gleichen Aufgabenstellungen ohne Verwendung des Systems getan hätte.
14. Die “Mathediktate” stellen eine sinnvolle Ergänzung des Übungsbetriebs dar.
15. Frühere Mathediktat-Aufgaben helfen mir oft dabei, Ansätze für spätere zu finden.
16. Um Mathediktate-Aufgaben zu lösen, muss man die zu erwerbenden Kompetenzen bereits besitzen; daher sind die Mathediktate nutzlos.

17. Die Erläuterungen zu den “Mathediktaten” auf den Übungsblättern waren ausreichend, um mit dem Programm umgehen zu können.

Freie Fragen:

18. Besonders gut gefällt mir:

19. Nicht so gut gefällt mir:

20. Konkret habe ich folgende Verbesserungsvorschläge:

12.3 Fragebogen zum "Game of Def"

1. Der Umgang mit dem "Game of Def" macht mir Spaß.
2. Die Arbeit mit dem "Game of Def" hat mir geholfen, die korrekte Bildung logischer Formeln zu lernen.
3. Die Arbeit mit dem "Game of Def" hat mir geholfen, zu lernen, anschauliche Vorstellungen in Quantorenlogik auszudrücken.
4. Rückmeldungen im "Game of Def"-Modul helfen mir dabei, meine Lösung zu verbessern.
5. Ich bin mit der Zeit besser im "Game of Def" geworden.
6. Durch den Umgang mit dem "Game of Def" bin ich im Umgang mit quantorenlogischen Formeln sicherer und selbstbewusster geworden.
7. Es ermutigt mich, wenn das System eine Lösung als korrekt meldet.
8. Fehlermeldungen durch das Systen entmutigen mich.
9. Die optische Darstellung macht das "Game of Def"-Modul reizvoller als Aufgaben, die nur aus Text bestehen.
10. Das "Game of Def" fällt mir schwerer als die "Mathediktate".
11. Bei Fehlermeldungen im Game of Def kann ich meistens erkennen, worin der Fehler in meiner Lösung besteht.
12. Die Arbeit mit dem "Game of Def" hat mir geholfen, logische Ausdrücke – etwa in der Vorlesung oder den Übungen – besser zu verstehen.
13. Es kommt vor, dass richtige Lösungen als falsch gemeldet werden.
14. Es kommt vor, dass falsche Lösungen als richtig gemeldet werden.
15. Ich habe durch den Gebrauch des "Game of Def" viel gelernt.
16. Durch das "Game of Def" habe ich mich länger mit der Verwendung logischer Ausdrücke beschäftigt, als ich es sonst getan hätte.
17. Das "Game of Def" stellt eine sinnvoller Ergänzung des Übungsbetriebs dar.
18. Frühere "Game of Def"-Aufgaben helfen mir oft dabei, Ansätze für spätere zu finden.

19. Um “Game of Def”-Aufgaben zu lösen, muss man die zu erwerbenden Kompetenzen bereits besitzen; daher ist das “Game of Def” nutzlos.

20. Die Erläuterungen zum “Game of Def” auf den Übungsblättern waren ausreichend, um mit dem Programm umgehen zu können.

Freie Fragen:

21. Besonders gut gefällt mir:

22. Nicht so gut gefällt mir:

23. Konkret habe ich folgende Verbesserungsvorschläge:

Bibliographie

[1] S. Arora, B. Barak. Computational Complexity: A Modern Approach. Cambridge University Press. (2009)

[2] P. Andrews, M. Bishop, C. Brown, S. Issar, F. Pfennig, H. Xi. ETPS: A System to Help Students Write Formal Proofs. Journal of Automated Reasoning, vol. 32 (2004)

[3] S. Autexier, D. Dietrich, M. Schiller. Towards an Intelligent Tutor for Mathematical Proofs. I: P. Quaresma, R. Back (eds.): THedu'11, EPTCS 79 (2012)

[4] J. Alama. Formal Proof and Refutations. Dissertation, Stanford (2009)

[5] J. Avigad, J. Harrison. Formally Verified Mathematics. Communications of the ACM, Vol. 57 (4) (2014)

[6] Automath Archive. `https://www.win.tue.nl/automath/` (zugegriffen 14.10.2021)

[7] Aristoteles. Erste Analytik. Organon Band 3. Meier Philosophische Bibliothek. (1997)

[8] D. Arndt. Semantik und Korrektheit von Prolog-Programmen im Naproche-Projekt. Diplomarbeit, Bonn (2009)

[9] M. Aschbacher. The Status of the Classification of the Finite Simple Groups. Notices of the AMS, vol. 51(7) (2004)

[10] K. Ambos-Spies, P. Fejer. Degrees of Unsolvability. Handbook of the History of Logic, vol. 9 (2014)

[11] J. Avigad. Learning Logic and Proof with an Interactive Theorem Prover. In: G. Hanna, D. Reid, M. de Villiers (eds.). Proof

Technology in Mathematics Research and Teaching. Springer International Publishing (2019)

[12] J. Avigad, K. Dnnelly, D. Gray, P. Raff. A Formally Verified Proof of the Prime Number Theorem. ACM Transactions on Computational Logic doi.org/10.1184/R1/6490715.v1 (2007)

[13] K. Appel, W. Haken. Every Planar Map is Four Colorable. American Mathematical Society (1989)

[14] J. Azzouni. How and Why Mathematics is Unique as a Social Practice. In: B. van Kerkhove, J. van Bendegem (eds): Perspectives On Mathematical Practices. Logic, Epistemology, and the Unity of Science, vol 5. Springer, Dordrecht (2007)

[15] J. Azzouni. Tracking Reason. Oxford University Press (2008)

[16] Y. Bertot, P. Casteran. Interactive Theorem Proving and Program Development. Coq'Art: The Calculus of Inductive Constructions. doi.org/10.1007/978-3-662-07964-5 Springer (2004)

[17] The Elfe System – Verifying mathematical proofs for undergraduate students. In: Proceedings of the 10th International Conference on Computer Supported Education (CSEDU 2018), (2018)

[18] A. Blass, N. Dershowitz, Y. Gurevich. When Are Two Algorithms the Same? The Bulletin of Symbolic Logic, vol. 15(2) (2009)

[19] Elfe Homepage. https://elfe-prover.org/ Zugegriffen: 07.10.2020

[20] A. Beutelspacher, R. Danckwarts, G. Nickel, S. Spies, G. Wickel. Mathematik Neu Denken. Impulse für die Gymnasiallehrerbildung an Universitäten. Vieweg+Teubner (2011)

[21] S. Beller, H. Hoppe. Deductive error reconstruction and classification in a logic programming framework. In: P. Brna, S. Ohlsson, H. Pain (eds.). Proceedings of the World Conference on Artificial Intelligence in Education. Charlottesville: Association for the Advancement of Computing in Education. (1993)

[22] G. Brandl. Analyse von Rechenfehlern im Grundschulbereich. Ein Beitrag zur Behebung von Rechenschwäche/Arithmasthenie. Ars Una München (1992)

[23] A. Bundy, M. Jamnik. A Common Type of Rigorous Proof that resists Hilbert's program. In: G. Hanna, D. Reid, M. de Villiers (eds.): Proof Technology in Mathematics Research and Teaching. Mathematics Education in the Digital Era, vol 14. Springer, Cham. https://doi.org/10.1007/978-3-030-28483-1_3 (2019)

[24] S. Böhne, C. Kreitz. Learning how to Prove: From the Coq Proof Assistant to Textbook Style. In: P. Quaresma and W. Neuper (Eds.): 6th International Workshop on Theorem proving components for Educational software (ThEdu'17) EPTCS 267, 2018, pp. 1–18, doi:10.4204/EPTCS.267.1

[25] N. Budesca, A. Moreno. Mathematical Logic Tutor – Propositional Calculus. Available online: https://www.researchgate.net/publication/228898409_Mathematical_Logic_Tutor-Propositional_Calculus (2000)

[26] G. Boy. The Handbook of Human-Machine Interaction: A Human-Centered Design Approach. CRC Press (2011)

[27] P. Braselmann, P. Koepke. A formal proof of Gödel's completeness theorem. Formalized Mathematics, vol. 13 (2005)

[28] R. Brown. Philosophy of Mathematics: A Contemporary Introduction to the World of Proofs and Pictures. Routledge (2008)

[29] R. Burton. Diagnosing bugs in a simple procedural skill. In: D. Sleeman, J. Brown. (eds). Intelligent Tutoring Systems. Academie Press (1982)

[30] W. Byers. How Mathematicians Think. Using Ambiguity, Contradiction, and Paradox to Create Mathematics. Princeton University Press (2010)

[31] M. Carl. Formal and natural proof – a phenomenological approach. In: S. Centrone, D. Kant, D. Sarikaya (eds.): Reflections on the Foundations of Mathematics. Synthese Library (Studies in Epistemology, Logic, Methodology, and Philosophy of Science), vol 407. Springer, Cham. https://doi.org/10.1007/978-3-030-15655-8_14 (2019)

[32] M. Carl. Number Theory and Axiomatic Geometry in the Diproche System. In: Proceedings 9th International Workshop

on Theorem Proving Components for Educational Software, Electronic Proceedings in Computer Science (2020)

[33] M. Carl. Automatic Evaluation of Formalization Exercises in Mathematis. Preprint (2020) arXiv:2006.01800v2

[34] M. Carl, R. Krapf. Das Diproche-System – ein automatisierter Tutor für den Einstieg ins Beweisen. In: Proceedings of the Herbsttagung des Arbeitskreises Mathematikunterricht und digitale Werkzeuge 2019. (2020)

[35] G. Cantor. Über unendliche lineare Punktmannigfaltigkeiten. In: E. Zermelo (hrsg.). Georg Cantor Gesammelte Abhandlungen mathematischen und philosophischen Inhalts. Springer Berlin (1932)

[36] W. Carnielli, M. Coniglio. Paraconsistent Set Theory. In: Paraconsistent Logic: Consistency, Contradiction and Negation. Logic, Epistemology, and the Unity of Science, vol 40. Springer, Cham. https://doi.org/10.1007 (2016)

[37] M. Carl, M. Cramer, D. Sarikaya, B. Schröder. How to Frame Understanding in Mathematics: A Case Study Using Extremal Proofs. Axiomathes:1-2 (forthcoming)

[38] J. Cohen, P. Cohen, S. West, L, Aiken. Applied Multiple Regression/Correlation Analysis for the Behavioral Sciences. Third Edition. Lawrence Erlbaum Associates, Publishers Mahwah, New Jersey, London (2003)

[39] R. Cherubin, M. Mannucci. A very short history of ultrafinitism. In J. Kennedy & R. Kossak (Eds.), Set Theory, Arithmetic, and Foundations of Mathematics: Theorems, Philosophies (Lecture Notes in Logic, pp. 180-199). Cambridge: Cambridge University Press. doi:10.1017/CBO9780511910616.010 (2011)

[40] M. Carl, P. Koepke. Interpreting Naproche – An algorithmic approach to the derivation-indicator view. In: Proceedings of the International Symposium on Mathematical Practice and Cognition (2010)

[41] N. Carter, K. Monks. Using the Proof-Checking Word Processor Lurch to Teach Proof-Writing. In: R. Schwell, J. Franko, A. Steurer (eds.). Beyond Lecture: Resources and Pedagogical

Techniques for Enhancing the Teaching of Proof-Writing Across the Curriculum. Mathematical Association of America (2015)

[42] N. Carter, K. Monks. Lurch: A Word Processor that Can Grade Student's Proofs. In: C. Lange et al. (eds.) Workshops and Work in Progress at CICM, CEUR Workshop Proceedings (2013)

[43] M. Carl, M. Cramer, D. Kühlwein. Chapter 1 of Landau in Naproche. Available online: https://korpora-exp.zim.uni-duisburg-essen.de/naproche/inc/downloads.php (2009)

[44] M. Cramer, P. Koepke, B. Schröder. Parsing and Disambiguation of Symbolic Mathematics in the Naproche System. In: J. Davenport, W. Farmer, J. Urban, F. Rabe (eds.): Intelligent Computer Mathematics. CICM 2011. Lecture Notes in Computer Science, vol 6824. Springer, Berlin, Heidelberg. https://doi.org/10.1007/978-3-642-22673-1_13 (2011)

[45] M. Cramer, P. Koepke, D. Kühlwein, B. Schröder. The Naproche System. Calculemus 2009 Proceedings. Emerging Trends (2009)

[46] M. Cramer, P. Koepke, D. Kühlwein, B. Schröder. Premise Selection in the Naproche System. In: J. Giesl, R. Hähnle (eds.): Automated Reasoning. IJCAR 2010. Lecture Notes in Computer Science, vol 6173. Springer, Berlin, Heidelberg. https://doi.org/10.1007/978-3-642-14203-1_37 (2010)

[47] M. Cramer, B. Fisseni, P. Koepke, D. Kühlwein, B. Schröder, J. Veldman. The Naproche Project – Controlled Natural Language Proof Checking of Mathematical Texts. In: N. Fuchs (ed.): Controlled Natural Language. CNL 2009. Lecture Notes in Computer Science, vol 5972. Springer, Berlin, Heidelberg. https://doi.org/10.1007/978-3-642-14418-9_11 (2010)

[48] R. Clower. Axiomatics in Economics. Southern Eonomic Journal, vol. 62(2) (1995)

[49] W. Clocksin, C. Mellish. Programming in Prolog. Springer (2003)

[50] M. Colyvan. Who's Afraid of Inconsistent Mathematics? ProtoSociology vol 25 (2008)

[51] G. Chartrand, A. Polimeni, P. Zhang. Mathematical Proofs – A Transition to Advanced Mathematics. Third Edition. Pearson (2013)

[52] J. O'Connor, E. Robertson. The four colour theorem. https://mathshistory.st-andrews.ac.uk/HistTopics/The_four_colour_theorem/ zugegriffen: 09.10.2020

[53] M. Cramer. Proof-checking mathematical texts in controlled natural language. Dissertation, Bonn (2013)

[54] M. Cramer. Implicit dynamic function introduction and its connections to the foundations of mathematics. In: Proceedings of the International interdisciplinary conference on Philosophy, Mathematics, Linguistics: Aspects of interaction (PhML 2012) (2012)

[55] J. Copeland. The Church-Turing Thesis. Standford Encyclopedia of Philosophy https://plato.stanford.edu/entries/church-turing/ (2017)

[56] J. Dawson. Why prove it again? Birkhäuser Basel (2015)

[57] G. Debreu. Theory of Value: An Axiomatic Analysis of Economic Equilibrium. Yale University Press (1977)

[58] K. Devlin. The Joy of Sets. Fundamentals of Contemporary Set Theory. Springer–Verlag New York (1993)

[59] A. De Lon, P. Koepke, A. Lorenzen. Interpreting Mathematical Texts in Naproche-SAD. In: Benzmüller C., Miller B. (eds) Intelligent Computer Mathematics. CICM 2020. Lecture Notes in Computer Science, vol 12236. Springer, Cham. (2020)

[60] D. Kühlwein. A calculus for Proof Representation Structures. Diplomarbeit, Bonn (2009)

[61] E. David, D. Zazkis. Characterizing introduction to proof courses: A survey of U.S. R1 and R2 course syllabi. International Journal of MAthematical Education in Science and Technology, vol. 51(3) (2020)

[62] M. Dore. ELFE – An interactive theorem prover for undergraduate students. Bachelorarbeit, RWTH Aachen (2017)

[63] H. Ebbinghaus, J. Flum, W. Thomas. Einführung in die mathematische Logik. Springer Spektrum (2018)

[64] Edukera Website https://www.edukera.com/ (Zugriff erfolgreich am 03.04.2021)

[65] R. Elwes. An enormous theorem: the classification of finite simple groups. + plus magazine https://plus.maths.org/content/enormous-theorem-classification-finite-simple-groups (2006)

[66] E-Proofs Homepage. http://e-proof.weebly.com/ (zugegriffen 14.10.2021)

[67] E-Proofs Homepage. "Limitations". http://e-proof.weebly.com/limitations.html (zugegriffen 14.10.2021)

[68] K. Erk, L. Priese. Theoretische Informatik: Eine umfassende Einführung. Springer-Verlag Berlin Heidelberg (2008)

[69] M. Tessier-Baillargeon, N. Leduc, L. Font,P. Richard, M. Gagnon. QED-Tutrix: Creating and expanding a problem database towards personalized problem itineraries for proof learning. 10th Congress of European Research in Mathematics Education (CERME 2017) (2017)

[70] B. Fisseni. Die Entwicklung einer Annotationssprache für natürlichsprachliche mathematische Beweise. Magisterarbeit, Bonn (2003).

[71] K. Flasch. Das philosophische Denken im Mittelalter. Von Augustin zu Machiavelli. Reclam (2013)

[72] B. Fisseni, D. Sarikaya, M. Schmitt, B. Schröder. How to Frame a Mathematician. Modelling the Cognitive Background of Proofs. In: Centrone S., Kant D., Sarikaya D. (eds) Reflections on the Foundations of Mathematics. Synthese Library (Studies in Epistemology, Logic, Methodology, and Philosophy of Science), vol 407. Springer, Cham. https://doi.org/10.1007/978-3-030-15655-8_19 (2019)

[73] L. Font, P. Richard, M. Gagnon. Improving QED-Tutrix by Automating the Generation of Proofs. In: P. Quaresma, W. Neuper (eds.): Proceedings 6th International Workshop on Theorem proving components for Educational software, ThEdu 2017, Electronic Proceedings in Computer Science vol. 267 (2017)

[74] M. Tessier-Baillargeon. GeoGebraTUTOR: Developpment d'un systeme tutoriel autonome pour l'accompagnement d'eleves en situation de resolution de problemes de demosntration en geometrie plane et genese d'un espace de travail geometrique idoine. Dissertation, Montreal (2015).

[75] N. Leduc. QED-Tutrix: systeme tutoriel intelligent pour l'accompagenement d'eleves en situation de resolution de problemes de demonstration en geometrie plane. Dissertation, Montreal (2016)

[76] P. Feyerabend. Against Method. Outline of an Anarchistic Theory of Knowledge. Third Edition. Verso Books (1993)

[77] S. Frerix, P. Koepke. Making Set Theory Great Again: The Naproche-SAD Project. Extended Abstract zum Vortrag auf der 4th Conference on Artificial Intelligence and Theorem Proving (AITP) (2019)

[78] Fachanforderungen Mathematik. Ministerium für Schule und Berufsbildung des Landes Schleswig-Holstein. Kiel `https://fachportal.lernnetz.de/mathematik.html` (2014) (zugegriffen 18.08.2021)

[79] E. Focke. Computerunterstützte Prävention und Frühförderung bei Rechenschwäche. Dissertation, Münster (2004)

[80] G. Fritsch, R. Fritsch. The Four-Color Theorem: History, Topological Foundations, and Idea of Proof. Springer Science & Business Media (1998)

[81] A. Furdek. Fehler-Beschwörer. Books on Demand GmbH Norderstedt (2001)

[82] H. Gadamer. Wahrheit und Methode. J. C. B. Mohr (Paul Siebeck) Tübingen (1960)

[83] M. Ganesalingam. The Language of Mathematics. A Linguistic and Philosophical Investigation. Lecture Notes in Computer Science 7805. Springer Berlin Heidelberg (2013)

[84] M. Gaidoschik. Wie Kinder rechnen lernen – oder auch nicht. Peter Lang GmbH, Internationaler Verlag der Wissenschaften (2010)

[85] M. Gardner. Geometrie mit Taxis, die Köpfe der Hydra und andere mathematische Spielereien. (über. A. Ehlers) Birkhäuser Verlag Basel Boston Berlin (1997)

[86] G. Gentzen. Untersuchungen über das logische Schließen. I. Mathematische Zeitschrift. 39 (2): 176–210 (1934)

[87] G. Gentzen. Untersuchungen über das logische Schließen. II. Mathematische Zeitschrift. 39 (3): 405–431. (1935)

[88] M. Gerwig. Beweisen verstehen im Mathematikunterricht. Aximatik, Pythagoras und Primzahlen als Exempel der Lehrkunstdidaktik. Springer Spektrum (2015)

[89] M. Giaquinto. Mathematical Proofs: The Beautiful and the Explanatory. Journal of Humanistic Mathematics vol. 6(1) (2016)

[90] B. Girnat. Geometrische Hintergrundtheorien des Beweisens im Schulalltag: Auszüge aus einer qualitativen Studie über Lehreransichten. In: M. Ludwig, R. Oldenburg, J. Roth (eds.). Argumentieren, Beweisen und Standards im Geometrieunterricht. AK Geometrie 2007/2008. Franzbecker Hildesheim Berlin (2008)

[91] K. Gödel. Über formal unentscheidbare Sätze der Principia Mathematica und verwandter Systeme. Monatshefte für Mathematik und Physik vol. 38 (1931)

[92] K. Gödel. Postscriptum zu "On Undecidable Propositions of Formal Mathematical Systems". In: M. Davis (ed.). The Undecidable. Raven Press Hewlett, New York (1965)

[93] D. Gorenstein (ed). The Classification of Finite Simple Groups. Springer US (1983)

[94] A. Hacker. The Math Myth. And Other STEM Illusions. The New Press (2016)

[95] T. Hales. The Jordan curve theorem, formally and informally. The American Mathematical Monthly, vol. 114 (10), S. 882–894. (2007)

[96] T. Hales. Introduction to the Flyspeck Project. In: Mathematics, Algorithms, Proofs. Dagstuhl Seminar Proceedings 05201. (2005)

[97] G. Hanna. Proof, Explanation and Exploration: An Overview. Educational Studies in Mathematics, vol. 11(1/2) (2000)

[98] G. Hanna. The Ongoing Value of Proof. Journal für Mathematik-Didaktik, vol. 18 (1997)

[99] History of the Church-Turing thesis. Wikipedia-Eintrag, https://en.wikipedia.org/wiki/History_of_the_Church\%E2\%80\%93Turing_thesis#Sieg_and_axiomatic_definitions zugegriffen: 09.10.2020

[100] B. Heintz. Die Herrschaft der Regel: zur Grundlagengeschichte des Computers. Dissertation (1993)

[101] H. Hemme. Heureka! Unterhaltsame Mathematik in 95 Rätseln mit ausführlichen Lösungen. Vandenhoeck & Ruprecht (2004)

[102] G. Hardy, E. Wright. An Introduction to the Theory of Numbers. Oxford University Press (2008) (Erstausgabe 1938)

[103] G. Holland. GEOLOG-WIN: Konstruieren, Berechnen, Beweisen, Problemlösen mit dem Computer im Geometrie-Unterricht der Sekundarstufe. Dümmler (1996)

[104] G. Holland. Aufgabenorientierte tutorielle Systeme for den Mathematikunterricht. In: U. Glowalla, E. Schoop (eds.): Hypertext und Multimedia. Informatik aktuell. Springer Berlin, Heidelberg. https://doi.org/10.1007/978-3-642-77758-5_13 (1992)

[105] G. Holland. Innovative Lernsoftware für den Geometrieunterricht in der Sekundarstufe I. In: H. Hoppe, W. Luther (eds.): Informatik und Lernen in der Informationsgesellschaft. Informatik aktuell. Springer, Berlin, Heidelberg. https://doi.org/10.1007/978-3-642-60894-0_12 (1997)

[106] G. Holland. Konstruieren mit dem Computer. In: K. Graf (eds.): Computer in der Schule 3. Vieweg+Teubner Verlag. https://doi.org/10.1007/978-3-322-92129-1_2 (1990)

[107] J. Horgan. The Death of Proof. Scientific American (1993)

[108] D. Hilbert. Grundlagen der Geometrie. B. G. Teubner Leipzig (1903)

[109] D. Hilbert. Mathematische Probleme. Nachrichten der Königlichen Gesellschaft der Wissenschaften zu Göttingen, mathematisch-physikalische Klasse. vol. 3 (1900)

[110] M. Heidegger. Gelassenheit. Verlag Karl Alber (2015)

[111] M. Heidegger. Grundbegriffe der Metaphysik. Vittorio Klostermann (2004)

[112] C. Hoare. An Axiomatic Basis for Computer Programming. Communications of the ACM, vol. 12(10) (1969)

[113] Homotopy Type Theory. Univalent Foundations of Mathematics. Institute for Advanced Study https://homotopytypetheory.org/book/ (2013)

[114] J. Hintikka, G. Sandu. Game-Theoretical Semantics. In: J. van Benthem, A. ter Meulen (eds.): Handbook of Logic and Language. North Holland (1996)

[115] M. Inglis, L, Alcock. Expert and Novice Approaches to Reading Mathematical Proofs. Journal for Research in Mathematics Education, vol. 43, (2012)

[116] M. Inglis, A. Aberdein. Beauty Is Not Simplicity: An Analysis of Mathematician's Proof Appraisals. Philosophia Mathematica vol. 23(1) (2014)

[117] M. Inglis, J. Mejia-Ramos, K. Weber, L. Alcock. On Mathematicians' Different Standards When Evaluating Elementary Proofs. Topics in Cognitive Science, vol. 5(2), (2013)

[118] A. Bundy, M. Jamnik, I. Green. On Automating Diagrammatic Proofs of Arithmetic Arguments. Journal of Logic, Language and Information, vol. 8 https://doi.org/10.1023/A:1008323427489 (1999)

[119] M. Jamnik. Mathematical Reasoning with Diagrams. From Intuition to Automation. Chicago University Press. https://doi.org/10.1080/17498430600803532 (2001)

[120] I. Kant. Kritik der reinen Vernunft. (1781)

[121] K. Lorenz, P. Lorenzen. Dialogische Logik. Wissenschaftliche Buchgesellschaft (1978)

[122] Bildungsstandards im Fach Mathematik für die Allgemeine Hochschulreife. Beschluss der Kultusministerkonferenz vom 18.10.2012. https://www.kmk.org/fileadmin/Dateien/veroeffentlichungen_beschluesse/2012/2012_10_18-Bildungsstandards-Mathe-Abi.pdf Zugegriffen: 20.05.2021

[123] R. Jamison. Learning the Language of Mathematics. Language and Learning Across the Disciplines vol. 4(1) (2000)

[124] S. Johnson, S. Steinerberger. Intuitions about mathematical beauty: A case study in the aesthetic experience of ideas. Cognition vol. 189 (2019)

[125] E. Klarreich. Titans of Mathematics Clash Over Epic Proof of ABC Conjecture. Quanta Magazine https://www.quantamagazine.org/titans-of-mathematics-clash-over-epic-proof-of-abc-conjecture-20180920/ (2018)

[126] P. Koepke. Mathematics and the New Technologies Part II: Computer-Assisted Formal Mathematics and Mathematical Practice. In: P. Schroeder-Heister, G. Heinzmann, W. Hodges, P. Bour (Hrsg.). Logic, Methodology and Philosophy of Science. Proceedings of the Fourteenth International Congress (2014)

[127] T. Koetsier. Lakatos' Philosophy of Mathematics, A Historical Approach, Amsterdam, North-Holland, (1991)

[128] H. Kamp, U. Reyle. From Discourse to Logic. Kluwer Academic Publishers Dordrecht/Boston/London (1993)

[129] I. Lakatos. Proofs and Refutations. Cambridge University Press (1976)

[130] E. Landau. Grundlagen der Analysis. Akademische Verlagsgesellschaft (1930)

[131] T. Lesar, L. Briceland, D. Stein. Factors Related to Errors in Medication Prescribing. JAMA vol. 277(4) (1997)

[132] Lean Homepage. https://leanprover.github.io/ Zugegriffen: 15.04.2021

[133] L. de Moura, S. Kong, J. Avigad, F. van Doorn, J. von Raumer. The Lean Theorem Prover. 25th International Conference on Automated Deduction (CADE-25). (2015)

[134] C. Lehner. Prolog und Linguistik. Oldenbourg Wissenschaftsverlag (1990)

[135] S. Levy. An Axiomatic Approach to Human Behaviour. Universal Journal of Psychology vol. 5(2) (2017)

[136] G. Lakoff, R. Nunez. Where Mathematics Comes from: How the Embodied Mind Brings Mathematics into Being. Basic Books (2001)

[137] J. Avigad, R. Lewis, F. van Doorn. Logic and Proof. An introductory course for undergraduates. https://leanprover.github.io/logic_and_proof/ (2017)

[138] H. Lorenzen. Geolog, Geobeweis und Geokon – Erfahrungen und Konzepte zum Unterricht. In: Berich üüber die 17. Arbeitstagung des Arbeitskreises "Mathematikunterricht und Informatik" in der Gesellschaft für Didaktik der Mathematik e.V., Wolfenbüttel. DIVerlag franzbecker (1999)

[139] H. Lorenzen. Zur Didaktik des begrifflichen Denkens in der Geometrieausbildung. Universität Kiel. Habilitationsschrift (2002)

[140] B. Löwe. Mathematics and the New Technologies Part I: Philosophical relevance of a changing culture of mathematics. In: P. Schroeder-Heister, G. Heinzmann, W. Hodges, P. Bour (Hrsg.). Logic, Methodology and Philosophy of Science. Proceedings of the Fourteenth International Congress (2014)

[141] B. Löwe, T. Müller. Mathematical Knowledge is context dependent. Grazer Philosophische Studien 76 (2008)

[142] Lurch Homepage. http://lurchmath.org/ Zugegriffen: 06.10.2020

[143] P. Maddy. Believing the Axioms 1. Journal of Symbolic Logic, vol. 53(2) doi:10.1017/S0022481200028425 (1988)

[144] P. Magnus, N. Carter. forallx in Lurch. An Introduction to Formal logic in Lurch. http://web.bentley.edu/empl/c/ncarter/faxil (2017)

[145] D. Marker. Model Theory. An Introduction. Springer New York (2002)

[146] R. Moore, M. Byrne, S. Hanusch, T. Fukawa-Connelly. When we Grade Students' Proofs, Do They Understand our Feedback? In: T. Fukawa-Connelly, N. Infante, M. Wawro, S. Brown (eds.): Proceedings of the 19th Annual Conference on Research in Undergraduate Mathematics Education (2016)

[147] N. Matsuda, K. VanLehn. Advanced Geometry Tutor: An Intelligent Tutoring System for Proof-Writing with Constructions. In: Proceedings of the Japan National Conference on Information and Systems in Education. (2005)

[148] Th. Mormann. Argumentieren, begründen, verallgemeinern: zum Beweisen im Mathematikunterricht. Scriptor (1981)

[149] C. Mortensen. Inconsistent Mathematics. Springer Netherlands (1995)

[150] C. Mortensen. Inconsistent Mathematics: Some Philosophical Implications. In: A. Irvine (ed.): Handbook of the Philosophy of Science, Philosophy of Mathematics. North-Holland (2009)

[151] R. Moore. Mathematics Professors' Evaluation of Student's Proofs: A Complex Teaching Practice. Int J. Res. Undergrad. Math. Ed. (2016)

[152] J. Mumma. Proofs, pictures and Euclid. Synthese, vol. 175(2). (2010)

[153] S. Naumann, H. Langer. Parsing. B. G. Teubner Stuttgart (1994)

[154] E. Niehaus, M. Platz, M. Krieger, K. Winter. Elektronische Beweise in der Lehre. Beiträge zum Mathematikunterricht. (2016)

[155] M. Stein. Beweisen. verlag franzbecker (1984)

[156] J. Siekmann, C. Benzmüller, V. Brezhnev, L. Cheikhrouhou, A. Fiedler, An. Franke, H. Horacek, M. Kohlhase, A. Meier, E. Melis, M, Moschner, I. Normann, M. Pollet, V. Sorge, C. Ullrich, C. Wirth, J. Zimmer. Proof Development with ΩMEGA. In: A. Voronkov (ed.): Automated Deduction—CADE-18. CADE 2002. Lecture Notes in Computer Science, vol 2392. Springer, Berlin, Heidelberg. https://doi.org/10.1007/3-540-45620-1_12 (2002)

[157] F. Padberg, S. Wartha. Didaktik der Bruchrechnung. Springer Spektrum (2017)

[158] A. Paskevich. Methodes de formalisation des connaissances et de rasonnements mathematiques: aspects appliques et theoretiques. Dissertation, Paris (2007)

[159] A. Paskevich. The syntax and semantics of the ForThel language. (2007)

[160] I. Perikos, F. Grivokostopoulou, I. Hatzilygeroudis. Help Generation in a System for Learning Natural Language to First Order Logic Conversion. In: Proceedings of the IASTED International Conference Computers and Advanced Technology in Education (CATE 2011). (2011)

[161] I. Perikos, F. Grivokostopoulou, I. Hatzilygeroudis. Assistance and Feedback Mechanism in an Intelligent Tutoring System for Teaching Conversion of Natural Language into Logic. International Journal of Artificial Intelligence in Education, vol. 27 (2017)

[162] M. Platz, M, Krieger, E. Niehaus, K. Winter. Electronic proofs in mathematics education – A South African Teacher Professional Development (TPD) course informing the conceptualisation of an e-proof system authoring support workshop. 017 IST-Africa Week Conference (IST-Africa) (2017)

[163] G. Polya. Vom Lösen mathematischer Aufgaben. Springer Basel (1979)

[164] G. Polya. Schule des Denkens. Vom Lösen mathematischer Probleme. A. Francke (1995)

[165] M. Polson, J. Richardson. Foundations of Intelligent Tutoring Systems. Lawrence Erlbaum Associates Publisher Hillsdale, New Jersey, Hove, London. (1988)

[166] ProofScape Website. `https://royalroadmath.org/newpfsc.html` (zugegriffen 14.10.2021)

[167] S. Payne, H. Squibb. Algebra Mal-Rules and Cognitive Accounts of Error. Cognitive Science 14 (1990)

[168] R. Nelsen. Proofs without Words. Exercises in Visual Thinking. The Mathematical Association of America (1997)

[169] R. Nelsen. Proofs without Words II. More Exercises in Visual Thinking. The Mathematical Association of America (2000)

[170] QED-Website. https://www.math.ucla.edu/~tao/QED/QED.html (zugegriffen 14.10.2021)

[171] "The QED Manifesto" in "Automated Deduction – CADE 12", Springer-Verlag, Lecture Notes in Artificial Intelligence, Vol. 814, pp. 238-251, 1994. Online verfügbar: https://www.cs.ru.nl/~freek/qed/qed.html

[172] Y. Rav. Why do we prove theorems? Philosophia Mathematica, vol. 7 (1999)

[173] Y. Rav, "A Critique of a Formalist-Mechanist Version of the Justification of Arguments in Mathematicians' Proof Practices," in Philosophia Mathematica, vol. 15, doi: 10.1093/philmat/nkm023 (2007)

[174] G.-C. Rota. The Phenomenology of Mathematical Beauty. Synthese vol. 111(2) (1997)

[175] G. Link (ed.). One Hundred Years of Russell's Paradox. Mathematics, Logic, Philosophy. De Gruyter New York. (2004)

[176] B. Russell, A. Whitehead. Principia Mathematica. Cambridge University Press (1962)

[177] D. Sleeman, J. Brown. (eds). Intelligent Tutoring Systems. Academie Press (1982)

[178] A. Schoenfeld. Mathematical Problem Solving. Academic Press (2016)

[179] A. Selden, J. Selden. Validations of proofs considered as texts: Can undergraduates tell whether an argument proves a theorem? Journal for Research in Mathematics Education, vol. 34. (2003)

[180] Inconsistent Mathematics. Stanford Encyclopedia of Philosophy. https://iep.utm.edu/math-inc/#SH1a zugegriffen: 18.10.2020

[181] G. Shimura. Yutaka Taniyama and his Time. Very Personal Recollections. Bulletin of the London Mathematical Society, vol. 21 (1989)

[182] S. Simpson. Subsystems of Second-Order Arithmetic. Cambridge University Press (2010)

[183] D. Sleeman. Basic Algebra Revisited: A Study with 14 Year Olds. International Journal of Man-Machine Studies, vol. 22(2) (1985)

[184] D. Sleeman. Assessing competence in basic algebra. In: D. Sleeman, J. Brown. (eds). Intelligent Tutoring Systems. Academie Press (1982)

[185] E. Scholz. G.W. Leibniz als Mathematiker. In: F. Knipping, S. Mangoldt, G. Walther (eds.). Europa und die Wissenschaft, Trier. Wissenschaftlicher Verlag (2007)

[186] D. Spalt. Vom Mythos der Mathematischen Vernunft. Eine Archäologie zum Grundlagenstreit der Analysis oder Dokumentation einer vergeblichen Suche nach der Einheit der mathematischen Vernunft. (Dissertation) Wissenschaftliche Buchgesellschaft (1981)

[187] B. de Spinoza. Ethik in geometrischer Ordnung dargestellt. Meiner (2015)

[188] A. Selden, J. Selden, A. Benkhalti. Proof Frameworks – A Way to Get Started. PRIMUS, vol. 28(1) (2018)

[189] StatistikGuru. Pearson Produkt-Moment Korrelation: Ergebnisse interpretieren. `https://statistikguru.de/spss/produkt-moment-korrelation/ergebnisse-interpretieren.html` Zugegriffen: 08.04.2021

[190] M. Heidegger. Sein und Zeit. Max Niemeyer Verlag Tübingen (2006)

[191] F. Tanswell. Proof and Prejudice: Why Formalising Doesn't Make You a Formalist. Master's thesis, Amsterdam (2012)

[192] R. Thiele, L. Wos. Hilbert's Twenty-Fourth Problem. Journal of Automated Reasoning, vol. 29 (2002)

[193] H. Thimbleby. Improving Safety in Medical Devices and Systems. 2013 IEEE International Conference on Healthcare Informatics.

[194] O. Toeplitz. Die Entwicklung der Infinitesimalrechnung Eine Einleitung in die Infinitesimalrechnung Nach der Genetischen Methode. Erster Band. Hrsg. von W. Blaschke, R. Grammel, E. Hopf, F. Schmidt, B. van der Waerden, G. Köthe. Springer-Verlag Berlin Heidelberg (1949)

[195] A. Turing. On Computable Numbers, with an Application to the Entscheidungsproblem. Proceedings of the London Mathematical Society, vol. s2-42(1) (1937)

[196] T. Tymoczko. The Four-Colour Problem and its Philosophical Significance. The Journal of Philosophy, vol. 76(2) (1979)

[197] Beurteilung einiger Programme. Interaktives Skript an der Universität Freiburg. http://home.mathematik.uni-freiburg.de/didaktik/material_download/Geometrie_Aufsatz/node10.html#SECTION00023000000000000000) (zugegriffen 16.01.2020)

[198] K. VanLehn. Mind Bugs: The Origins of Procedural Misconceptions (Learning, Development, and Conceptual Change). The MIT Press (2008)

[199] K. Verchinine, A. Lyaletski, A. Paskevich. System for Automated Deduction (SAD): a tool for proof verification. In: Pfenning F. (eds) Automated Deduction – CADE-21. CADE 2007. Lecture Notes in Computer Science, vol 4603. Springer, Berlin, Heidelberg (2007)

[200] M. Wagenschein. Ursprüngliches Verstehen und exaktes Denken. Ernst Klett Verlag Stuttgart (1970)

[201] K. Weber. How Mathematicians Determine If an Argument is a Valid Proof. Journal for Reserach in Mathematics Education, vol 38(4) (2008)

[202] K. Weber. Research Sampler 8: Students' difficulties with proof. https://www.maa.org/programs/faculty-and-departments/curriculum-department-guidelines-recommendations/teaching-and-learning/ (zugegriffen 14.10.2021) research-sampler-8-students-difficulties-with-proof Zugriff: 29.09.2020

[203] M. Wehrmann. Qualitative Diagnostik von Rechenschwierigkeiten im Grundlagenbereich Arithmetik. Verlag Dr. Köster (2011)

[204] K. Winter. Entwicklung von Item-Distraktoren mit diagnostischem Potential zur individuellen Defizit- und Fehleranalyse. – Didaktische Überlegungen, empirische Untersuchungen und konzeptionelle Entwicklung für ein internetbasiertes Mathematik-Self-Assessment. Dissertation, Münster (2010)

[205] H. Winter. Zur Problematik des Beweisbedürfnisses. Journal für Mathematik-Didaktik, vol. 4 https://doi.org/10.1007/BF03339229 (1983)

[206] Alexander I. Wittenberg. Bildung und Mathematik. Klett (1963)

[207] E. Wittmann. Das Wesen der Mathematik liegt in ihrer Freiheit: Fachliche Argumente für individuelle Lern- und Lösungswege. 18. Symposium mathe 2000: Individuelle Förderung im Mathematikunterricht der Grundschule (2008)

[208] J. Woodger. The Axiomatic Method in Biology. Cambridge University Press (1937)

[209] D. Velleman. How to Prove it: A Structured Approach. Cambridge University Press (2006)

[210] M. Wolska. Student's language in computer-assisted tutoring of mathematical proofs. Dissertation, Saarbrücken (2013)

[211] D. Walton, C. Reed, F. Macagno. Argumentation Schemes. Cambridge University Press, Cambridge, New York (2008)

[212] E. Yackel, P. Cobb. Sociomathematical Norms, Argumentation, and Autonomy in Mathematics. Journal for Research in Mathematics Education. Vol. 27(4) (1996)

[213] B. Yushau, M. Bokhari. Language and Mathematics: A Mediational Approach to Bilingual Arabs. International Journal for Mathematics Teaching and Learning. (2005)

[214] E. Zalta. Principia Metaphysica. http://mally.stanford.edu/principia.pdf (2021)

[215] O. Zaslavsky, K. Shir. Students' Conceptions of Mathematical Definition. Journal for Research in Mathematics Education. (2005)

[216] S. Zittermann. Entwicklung des Naproche-Proof-State-Datentyps. Masterarbeit, Köln (2011)

[217] K. Zinn. Understanding Informal Mathematical Discourse. Dissertation, Universität Tübingen (2004)