Dr. Winfried Teschers
Anton-Günther-Straße 26c
91083 Baiersdorf
winfried.teschers@t-online.de

Projektdokument

ASBA
Axiome, Sätze, Beweise und Ausgaben

Projekt zur maschinellen Überprüfung von mathematischen Beweisen und deren Ausgabe in lesbarer Form

Winfried Teschers

6. März 2017

Es wird ein System beschrieben, das zu eingegebenen Axiomen, Sätzen, und Beweisen letztere prüft, Auswertungen generiert und zu gegebenen Ausgabeschemata eine Ausgabe der Elemente in üblicher Formelschreibweise im LaTeX-Format erstellt.

Inhaltsverzeichnis

1. Analyse

In der Mathematik gibt es eine unüberschaubare Menge an Axiomen, Sätzen, Beweisen, Fachbegriffen[1] und Fachgebieten. Dabei soll ein *Fachgebiet* einen Teil der Mathematik mit einer zugehörigen Basis von Axiomen, Sätzen und spezifischen Fachbegriffen sein, zum Beispiel *Logik*, *Mengenlehre* und *Gruppentheorie*[2]. Zu den meisten Fachgebieten gibt es auch noch ungelöste Probleme.

Es fehlt ein System, das einen Überblick bietet und die Möglichkeit, Beweise automatisch zu überprüfen. Außerdem sollte all dies in üblicher mathematischer Schreibweise ein- und ausgegeben werden können.

Ein System mit ähnlicher Aufgabenstellung findet sich im GitHub Projekt Hilbert II (siehe [17, 18]). Einige Ideen sind von dort übernommen worden.

1.1. Fragen

Einige der Fragen, die in diesem Zusammenhang auftauchen, werden hier formuliert:

1. *Grundlagen*: Was sind die Grundlagen? Zum Beispiel welche Logik und Mengenlehre.
2. *Basis*: Welche wichtigen Axiome, Sätze, Beweise, Fachbegriffe und Fachgebiete gibt es? Welche davon sind Standard?
3. *Axiome*: Welche Axiome werden bei einem Satz oder Beweis vorausgesetzt? Allgemein anerkannte oder auch strittige, wie zum Beispiel den *Satz vom ausgeschlossenen Dritten* (*tertium non datur*) oder das *Auswahlaxiom*.
4. *Beweis*: Ist ein Beweis fehlerfrei?
5. *Konstruktion*: Gibt es einen konstruktiven Beweis?
6. *Vergleiche*: Welcher Beweis ist besser? Nach welchem Kriterium? Zum Beispiel elegant, kurz, einsichtig oder wenige Axiome. Was heißt eigentlich *elegant*?
7. *Definitionen*: Was ist mit einem Fachbegriff oder Fachgebiet jeweils genau gemeint? Zum Beispiel *Stetigkeit*, *Integral* und *Analysis*.
8. *Abhängigkeiten*: Wie heißt ein Fachbegriff oder Fachgebiet in einer anderen Sprache? Ist wirklich dasselbe gemeint? Was ist mit Fachbegriffen in verschiedenen Fachgebieten?
9. *Überblick*: Ist ein Axiom, Satz, Beweis, Fachbegriff oder Fachgebiet schon einmal – ggf. abweichend – definiert, formuliert oder bewiesen worden?
10. *Darstellung*: Wie kann man einen Satz und den zugehörigen Beweis – ggf. auch spezifisch für ein Fachgebiet – darstellen?
11. *Forschung*: Welche Probleme gibt es noch zu erforschen.

[1] *Fachbegriffe* sind Namen für Axiome, Sätze, Beweise und Fachgebiete. Symbole können als spezielle Fachbegriffe aufgefasst werden.

[2] Ein Fachgebiet kann hier sehr klein sein und im Extremfall kein einziges Element enthalten. *Umgebung* wäre in diesem Projekt eine bessere Bezeichnung, könnte aber zu Verwechslungen führen, da dies schon ein verbreiteter Fachbegriff ist.

1.2. Mission

Um zur Lösung obiger Fragen beizutragen, soll ein System entwickelt werden, das die folgenden Eigenschaften hat:

1. *Daten*: Axiome, Sätze, Beweise, Fachbegriffe und Fachgebiete können in formaler Form gespeichert werden – auch nicht oder unvollständig bewiesene Sätze. Dabei soll die übliche mathematische Schreibweise verwendet werden können.
2. *Definitionen*: Es können Fachbegriffe für Axiome, Sätze, Beweise und Fachgebiete – letztere mit eigenen Axiomen, Sätzen, Beweisen, Fachbegriffen und über- oder untergeordneten Fachgebieten – definiert werden. Die Definitionen dürfen wiederum an dieser Stelle schon bekannte Fachbegriffe und Fachgebiete verwenden.
3. *Prüfung*: Vorhandene Beweise können automatisch geprüft werden.
4. *Ausgaben*: Die Axiome, Sätze und Beweise können in üblicher Schreibweise – abhängig von Sprache und Fachgebiet – ausgegeben werden.
5. *Auswertungen*: Zusätzlich zur Ausgabe der gespeicherten Daten sind verschiedene Auswertungen möglich, unter anderem für die meisten der unter Abschnitt 1.1 auf der vorherigen Seite behandelten Fragen.

Damit das System nicht umsonst erstellt wird und möglichst breite Verwendung findet, werden noch zwei Punkte angefügt:

6. *Lizenz*: Die Software ist *Open Source*.
7. *Akzeptanz*: Das System wird von den Fachleuten akzeptiert und verwendet.

1.3. Ziele

Um die Mission zu erfüllen, soll ein System entwickelt werden, das die folgenden Anforderungen erfüllt:

1. *Daten*: Es enthält möglichst viele wichtige Axiome, Sätze, Beweise, Fachbegriffe, Fachgebiete und Ausgabeschemata[3].
2. *Form*: Die Daten liegt in formaler, geprüfter Form vor.
3. *Eingaben*: Die Eingabe von Daten erfolgt in einer formalen Syntax unter Verwendung der üblichen mathematischen Schreibweise. Folgende Daten können eingegeben werden:
   a) Axiome
   b) Sätze
   c) Beweise
   d) Fachbegriffe
   e) Fachgebiete
   f) Ausgabeschemata

   Dabei sind alle Begriffe nur innerhalb eines Fachgebietes und seiner untergeordneten Fachgebiete gültig, solange sie nicht umdefiniert werden. Das oberste Fachgebiet ist die ganze Mathematik.

[3] Um den Punkt 4 von Abschnitt 1.2 erfüllen zu können, werden noch fachgebietsspezifische Ausgabeschemata benötigt, welche die Art der Ausgaben beschreiben.

4. *Prüfung*: Vorhandene Beweise können automatisch geprüft werden.
5. *Ausgaben*: Die Ausgabe kann in einer eindeutigen, formalen Syntax gemäß vorhandener Ausgabeschemata erfolgen.
6. *Auswertungen*: Zusätzlich zur Ausgabe der Daten sind verschiedene Auswertungen möglich. Insbesondere kann zu jedem Beweis angegeben werden, wie viele Beweisschritte und welche Axiome und Sätze[4] er benötigt.
7. *Anpassbarkeit*: Fachbegriffe und die Darstellung bei der Ausgabe können mit Hilfe von – gegebenenfalls unbenannten – untergeordneten Fachgebieten angepasst werden.
8. *Individualität*: Axiome und Sätze können für jeden Beweis individuell vorausgesetzt werden. Dabei sind fachgebietsspezifische Fachbegriffe erlaubt.
9. *Internet*: Die Daten können auf mehrere Dateien verteilt sein. Ein Teil davon – oder sogar alle – können im Internet liegen.
10. *Kommunikation*: Die Kommunikation mit dem System kann mit den Fachbegriffen der einzelnen Fachgebiete erfolgen.
11. *Zugriff*: Der Zugriff auf das System kann lokal und über das Internet erfolgen.
12. *Unabhängigkeit*: Das System kann offline und online arbeiten.
13. *Rekursion*: Es kann rekursiv über alle verwendeten Dateien – auch solchen, die im Internet liegen – ausgewertet werden.
14. *Bedienbarkeit*: Das System ist einfach zu bedienen.
15. *Lizenz*: Die Software ist *Open Source*.

1.4. Zusammenhänge

Ausgehend von einer Liste der Fragen werden nun über die Zwischenstufe Mission Anforderungen an das zu realisierende System gestellt. Mit einem großen X werden die Spalten markiert, deren Punkte für die Erfüllung der Anforderungen in den Zeilen nötig sind. Idealerweise soll die Erfüllung der Anforderungen die Fragen beantworten bzw. zur Beantwortung beitragen.

Die Tabelle 1.3 auf Seite 7 ist eine Kombination aus den Tabellen 1.1 auf der nächsten Seite und 1.2 auf der nächsten Seite. Die Fragen *Akzeptanz* und *Lizenz* kommen aus Abschnitt 1.2 auf der vorherigen Seite *Mission* dazu. Mit einem kleinen x werden Spalten markiert, deren Punkte für die Erfüllung der Anforderungen in den Zeilen nicht nötig, aber von Interesse sind.

[4] Sätze, die quasi als Axiome verwendet werden.

Fragen	Mission	1 Daten	2 Definitionen	3 Prüfung	4 Ausgaben	5 Auswertungen	6 Lizenz	7 Akzeptanz
1	Grundlagen	X	X	-	X	X	-	-
2	Basis	X	X	-	X	X	-	-
3	Axiome	X	X	-	X	X	-	-
4	Beweis	X	-	X	X	-	-	-
5	Konstruktion	X	-	-	X	-	-	-
6	Vergleiche	X	-	-	-	X	-	-
7	Definitionen	X	X	-	X	-	-	-
8	Abhängigkeiten	X	-	-	X	-	-	-
9	Überblick	X	-	-	-	X	-	-
10	Darstellung	-	X	-	X	-	-	-
11	Forschung	X	-	-	-	X	-	-

Tabelle 1.1.: Fragen → Mission

Mission	Ziele	1 Daten	2 Form	3 Eingaben	4 Prüfung	5 Ausgaben	6 Auswertungen	7 Anpassbarkeit	8 Individualität	9 Internet	10 Kommunikation	11 Zugriff	12 Unabhängigkeit	13 Rekursion	14 Bedienbarkeit	15 Lizenz
1	Daten	X	X	X	-	-	-	-	-	-	-	-	-	-	-	-
2	Definitionen	X	-	X	-	-	-	-	-	-	-	-	-	-	-	-
3	Prüfung	-	-	-	X	-	-	-	-	-	-	-	-	-	-	-
4	Ausgaben	-	-	-	-	X	-	-	-	-	-	-	-	-	-	-
5	Auswertungen	-	-	-	-	-	X	-	-	-	-	-	-	-	-	-
6	Lizenz	-	-	-	-	-	-	-	-	-	-	-	-	-	-	X
7	Akzeptanz	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X

Tabelle 1.2.: Mission → Ziele (Anforderungen)

Fragen \ Ziele	1 Daten	2 Form	3 Eingaben	4 Prüfung	5 Ausgaben	6 Auswertungen	7 Anpassbarkeit	8 Individualität	9 Internet	10 Kommunikation	11 Zugriff	12 Unabhängigkeit	13 Rekursion	14 Bedienbarkeit	15 Lizenz
1 Grundlagen	X	X	X	-	X	X	x	-	-	-	-	-	-	-	-
2 Basis	X	X	X	-	X	X	x	x	-	-	-	-	-	-	-
3 Axiome	X	X	X	-	X	X	x	-	-	-	-	-	-	-	-
4 Beweis	X	X	X	X	X	-	-	x	-	-	-	-	-	-	-
5 Konstruktion	X	X	X	-	X	-	-	x	-	-	-	-	-	-	-
6 Vergleiche	X	X	X	-	-	X	-	x	-	-	-	-	-	-	-
7 Definitionen	X	X	X	-	X	-	x	-	-	-	-	-	-	-	-
8 Abhängigkeiten	X	X	X	-	X	-	x	-	-	-	-	-	-	-	-
9 Überblick	X	X	X	-	-	X	x	-	-	-	-	-	-	-	-
10 Darstellung	X	-	X	-	X	-	x	-	-	-	-	-	-	-	-
11 Forschung	X	X	X	-	-	X	x	-	-	-	-	-	-	-	-
Lizenz	-	-	-	-	-	-	-	-	-	-	-	-	-	-	X
Akzeptanz	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X

Tabelle 1.3.: Fragen → Ziele (Anforderungen)

2. Design

Diese Projekt soll Open Source sein. Daher gilt für die Dokumente die *GNU Free Documentation License (FDL)* und für die Software die *GNU Affero General Public License (APGL)*. Die *GNU General Public License (GPL)* reicht für die Software nicht, da das Programm auch mittels eines Servers betrieben werden kann und soll. Damit das Projekt gegebenenfalls durch verschiedene Entwickler gleichzeitig bearbeitet werden kann und wegen des Konfigurationsmanagements wurde es als ein GitHub Projekt erstellt (siehe [19]).

Wenn die Lizenzen nicht mitgeliefert wurden, können sie unter `http://www.gnu.org/licenses/` gefunden werden.

2.1. Anforderungen

Die Anforderungen ergeben sich zunächst aus dem Abschnitt 1.3 auf Seite 4. Die beiden Ziele 1 *Daten* und 15 *Lizenz* sind für die Entwicklung des Systems von sekundärer Bedeutung und wurden daher nicht in diesen Abschnitt übernommen. Die anderen Ziele werden noch verfeinert.

1. *Form*: Die Daten liegt in formaler, geprüfter Form vor. (siehe Ziel 2 auf Seite 4)
2. *Eingaben*: Die Eingabe von Daten erfolgt in einer formalen Syntax unter Verwendung der üblichen mathematischen Schreibweise. Folgende Daten können eingegeben werden:
   a) Axiome
   b) Sätze
   c) Beweise
   d) Fachbegriffe
   e) Fachgebiete
   f) Ausgabeschemata

   Dabei sind alle Begriffe nur innerhalb eines Fachgebietes und seiner untergeordneten Fachgebiete gültig, solange sie nicht umdefiniert werden. Das oberste Fachgebiet ist die ganze Mathematik. (siehe Ziel 3 auf Seite 4)
3. *Prüfung*: Vorhandene Beweise können automatisch geprüft werden. (siehe Ziel 4 auf Seite 5)
4. *Ausgaben*: Die Ausgabe kann in einer eindeutigen, formalen Syntax gemäß vorhandener Ausgabeschemata erfolgen. (siehe Ziel 5 auf Seite 5)
5. *Auswertungen*: Zusätzlich zur Ausgabe der Daten sind verschiedene Auswertungen möglich. Insbesondere kann zu jedem Beweis angegeben werden, wie viele Beweisschritte und welche Axiome und Sätze[1] er benötigt. (siehe Ziel 6 auf Seite 5)
6. *Anpassbarkeit*: Fachbegriffe und die Darstellung bei der Ausgabe können mit Hilfe von – gegebenenfalls unbenannten – untergeordneten Fachgebieten angepasst werden. (siehe Ziel 7 auf Seite 5)

[1] Sätze, die quasi als Axiome verwendet werden.

7. *Individualität*: Axiome und Sätze können für jeden Beweis individuell vorausgesetzt werden. Dabei sind fachgebietsspezifische Fachbegriffe erlaubt. (siehe Ziel 8 auf Seite 5)
8. *Internet*: Die Daten können auf mehrere Dateien verteilt sein. Ein Teil davon – oder sogar alle – können im Internet liegen. (siehe Ziel 9 auf Seite 5)
9. *Kommunikation*: Die Kommunikation mit dem System kann mit den Fachbegriffen der einzelnen Fachgebiete erfolgen. (siehe Ziel 10 auf Seite 5)
10. *Zugriff*: Der Zugriff auf das System kann lokal und über das Internet erfolgen. (siehe Ziel 11 auf Seite 5)
11. *Unabhängigkeit*: Das System kann offline und online arbeiten. (siehe Ziel 12 auf Seite 5)
12. *Rekursion*: Es kann rekursiv über alle verwendeten Dateien – auch solchen, die im Internet liegen – ausgewertet werden. (siehe Ziel 13 auf Seite 5)
13. *Bedienbarkeit*: Das System ist einfach zu bedienen. (siehe Ziel 14 auf Seite 5)

> > > ANFORDERUNGEN bearbeiten. < < <

2.2. Datenstruktur

> > > DATENSTRUKTUR bearbeiten. < < <

2.3. Bausteine

> > > BAUSTEINE bearbeiten. < < <

A. Anhang

A.1. Werkzeuge

Da dies ein Open Source Projekt sein soll, müssen alle Werkzeuge, die zum Ablauf der Software erforderlich sind, ebenfalls Open Source sein. Für die reine Entwicklung sollte das auch gelten.

Werkzeuge, die zum Ablauf der Software erforderlich sind

- *MiKTEX* für Dokumentation und Ausgaben in LaTeX. → https://miktex.org/ - Lizenz [10]

Werkzeuge, die für die Entwicklung verwendet werden

- *GitHub* als Online Konfigurationsmanagementsystem zur Zusammenarbeit verschiedener Entwickler. → https://github.com/ - Lizenz [6]
- GitHub benötigt *Git* als Konfigurationsmanagementsystem. → https://git-scm.com/ - Lizenz [6]
- *Visual Studio Community 2015*[1] (*VS*) als Entwicklungsumgebung für C++. → https://www.visualstudio.com/downloads/ - Lizenz [9]
- *Doxygen* als Dokumentationssystem für C++. → http://www.stack.nl/~dimitri/doxygen/ - Lizenz [6]
- Doxygen benötigt *Ghostscript* als Interpreter für Postscript und PDF. → http://ghostscript.com/ - Lizenz [4]
- Doxygen benötigt *Graphviz* mit *Dot* zur Erzeugung und Visualisierung von Graphen. → http://www.graphviz.org/Home.php - Lizenz [3]

Werkzeuge für die Entwicklung, die jeder Entwickler individuell durch andere ersetzten kann

- *TEXstudio* als Editor für LaTeX. → http://www.texstudio.org/ - Lizenz [6]
- *Notepad++* als Text-Editor. → https://notepad-plus-plus.org/ - Lizenz [5]
- *WinMerge* zum Vergleich von Dateien und Verzeichnissen. → http://winmerge.org/ - Lizenz [5]

Angedachte Werkzeuge

- In *Visual Studio Community 2015* integrierte Datenbank für Axiome, Sätze, Beweise, Fachbegriffe und Fachgebiete. - Lizenz [9]
- *RapidXml* für Ein- und Ausgabe in XML. → http://rapidxml.sourceforge.net/index.htm - Lizenz wahlweise [2] oder [12]

> > > QEDEQ Werkzeuge auflisten? < < <

[1] Visual Studio Community ist zwar nicht Open Source, darf aber zur Entwicklung von Open Source Software unentgeltlich verwendet werden.

Im Projekt *qedeq* verwendete Werkzeuge

- *Java* als Programmiersprache - Laufzeitumgebung. → https://www.java.com/de/download/win10.jsp - Lizenz [13]
- *Apache Ant* als Java Bibliothek und Kommandozeilen-Werkzeug um Java Programme zu erzeugen. → http://ant.apache.org/ - Lizenz [1]
- *Checkstyle* zur statischen Code-Analyse für Java. → http://checkstyle.sourceforge.net/ - Lizenz [7]
- *Clover*[2] als Testwerkzeug zur Analyse der Code-Abdeckung. → https://www.atlassian.com/software/clover/ - Lizenz [8]
- *Eclipse IDE for Java Developers* als Entwicklungsumgebung für Java. → http://www.eclipse.org/downloads/packages/eclipse-ide-java-developers/neon1a/ - Lizenz [14]
- *JUnit* zur Erzeugung von wiederholbaren Tests. → http://junit.org/junit4/ - Lizenz [3]
- *Xerces2* als XML-Parser in Java. → http://xerces.apache.org/xerces2-j/ - Lizenzen [1, 11, 15, 16]

A.2. Aussagenlogik

A.2.1. Konstante und Operatoren

Die Tabelle A.1 auf der nächsten Seite[3] definiert für die zweiwertige Logik die Konstanten- und Operatorsymbole über die Wahrheitswerte ihrer Anwendung. So ergeben sich, abhängig von den Wahrheitswerten der Operanden A und B, die in der Tabelle angegebenen Wahrheitswerte für die Operationen. Die mit 0, 1 und 2 benannten Spalten werden jeweils nur für die 0-, 1- und 2-stelligen Operatoren, d. h. für die Konstanten und die unären und binären Operatoren ausgefüllt. Dabei werden die Konstanten als 0-stellige Operatoren angesehen. Hat der Inhalt einer Zelle keine Relevanz, so bleibt sie leer, ist kein Wert bekannt, steht dort ein Minuszeichen. - Für die Symbole und Namen sind in der Spalte *Namen* auch Alternativen angegeben.

Um vollständig zu sein, d. h. für alle 22 möglichen Kombinationen von Wahrheitswerten für höchstens zwei Variable Operatorsymbole zu haben, enthält die Tabelle auch viele ungebräuchliche Operatoren. Am verbreitetsten sind neben den Klammern nur die logischen Operatoren $\neg, \wedge, \vee, \rightarrow$ und $\leftrightarrow$, gelegentlich auch $\leftarrow$ und $\dot{\vee}$, sowie die Konstanten $\top$ und $\bot$. Die entsprechenden Zeilen in der Tabelle sind grau hinterlegt. Zu jedem normalen Operator, getrennt nach Konstanten und unären und binären Operatoren, gibt es einen durchgestrichenen (negierten) Operator und umgekehrt. Die Wahrheitswerte W und F sind jeweils vertauscht. Das *nicht* vor geklammerten Ausdrücken darf nicht in die Klammer hineingezogen werden, da sich sonst die Bedeutung ändern würde oder unklar wäre! Die Symbole $\top, \bot, \not\top$ und $\not\bot$ werden hier nicht nur für Konstante, sondern auch als unäre und binäre Operatoren verwendet.

Wenn für eine bestimmte Kombination von Wahrheitswerten mehr als eine Operation angegeben ist, so sind diese Operationen in der zweiwertigen Aussagenlogik alle gleich. Bei der formalen Definition setzen wir aber keine Zweiwertigkeit voraus, so dass je nach Definition der Operatoren und Auswahl der Axiome die Operatoren verschieden sein können, d. h. verschiedene Ergebnisse liefern.

Identität ($=$) und Ungleichheit ($\neq$) sind im engeren Sinne keine logischen Operatoren und die Definition ($:=$) schon gar nicht, während $\leftrightarrow$ und $\nleftrightarrow$ (trotz gleicher Wahrheitswerte) über ein Axiom oder eine Definition eingeführt werden müssen. $=, \neq$ und $:=$ sind ebenfalls grau hinterlegt.

[2] Clover ist proprietäre Software, aber auf Anfrage frei für 30 Tage. Danach ist eine einmalige Lizenzgebühr fällig.
[3] Die Tabelle ist eine Erweiterung und Umsortierung der Wahrheitstafel aus Kapitel 2.2 von [26], ohne Angabe der Formeln.

Oper[1]	0	1		2				Name	Sprechweise	P[1]
A	-	W	F	W	W	F	F	-	Aussage A	
B	-	-	-	W	F	W	F	-	Aussage B	
$\top$	W							-	wahr	
$\not\perp$								-	nicht falsch	
$\perp$	F							-	falsch	
$\not\top$								-	nicht wahr	
$\top A$		W	W					-	wahr	6
$\not\perp A$								-	nicht falsch	6
(A)		W	F					Klammerung	A ist geklammert	7
$\neg A$		F	W					Negation	nicht A	6
$\perp A$		F	F					-	falsch	6
$\not\top A$								-	nicht wahr	6
$A \top B$				W	W	W	W	Tautologie	wahr	5
$A \not\perp B$								-	nicht falsch	5
$A \vee B$				W	W	W	F	Disjunktion; Adjunktion	A oder B	3
$A \leftarrow B$				W	W	F	W	Replikation; Konversion ($\subset$)	A folgt aus B	1
$A \circlearrowleft B$				W	W	F	F	Präpendenz; Identität von A ($\lfloor$)	A	2
$A \rightarrow B$				W	F	W	W	Implikation; Subjunktion; Konditional ($\supset$)	aus A folgt B	1
$A \circlearrowright B$				W	F	W	F	Postpendenz; Identität von B ($\rfloor$)	B	2
$A \leftrightarrow B$				W	F	F	W	Bijunktion; Bikonditional; Äquivalenz	A genau dann, wenn B; A ist äquivalent zu B	1
$A \not\vee B$								-	nicht (entweder A oder B)	3
$A \wedge B$				W	F	F	F	Konjunktion	A und B	4
$A \not\wedge B$				F	W	W	W	NAND[2]; Sheffer-Funktion ($\mid, \uparrow, \barwedge$)	nicht (A und B)	4
$A \dot{\vee} B$				F	W	W	F	ausschließende Disjunktion; XOR ($\veebar, \oplus$)	entweder A oder B	3
$A \not\leftrightarrow B$								Kontravalenz ($\not\equiv$)	nicht (A genau dann, wenn B)	1
$A \not\circlearrowright B$				F	W	F	W	Postnonpendenz; Negation von B ($\lceil$)	nicht B	2
$A \nrightarrow B$				F	W	F	F	Postsektion; nur A ($\dot{-}$)	nicht (aus A folgt B)	1
$A \not\circlearrowleft B$				F	F	W	W	Pränonpendenz; Negation von A ($\rceil$)	nicht A	2
$A \nleftarrow B$				F	F	W	F	Präsektion; nur B ($\dot{+}$)	nicht (A folgt aus B)	1
$A \not\vee B$				F	F	F	W	NOR[2]; Peirce-Funktion ($\downarrow, \triangledown$)	nicht (A oder B)	3
$A \perp B$				F	F	F	F	Kontradiktion	falsch	5
$A \not\top B$								-	nicht wahr	5
$A = B$				W	F	F	W	Identität	A gleich B	0
$A \neq B$				F	W	W	F	Ungleichheit	A ungleich B	0
$A \coloneqq B$								Definition	A definiert als B	0

Tabelle A.1.: Definition von aussagenlogischen Symbolen.

[1] *Oper* steht für *Operation* und *P* für *Priorität*.

[2] Diese Operationen werden auch als logische Schaltelemente in logischen Schaltungen verwendet.

A.2.2. Klammerregeln

Zur Klammerersparnis werden die üblichen Regeln verwendet, d. h. dass Operatoren mit höherer Priorität stärker binden, als solche mit niedrigerer Priorität. Bei gleicher Priorität binden Klammern von innen nach außen, unäre Operatoren von rechts nach links[4] und binäre von links nach rechts. Es gilt also mit abnehmender Priorität:

Klammern

- $(\dots)$

Unäre Operatoren

- $\neg, \top, \bot, \not\top, \not\bot$

Binäre Operatoren

- $\top, \bot, \not\top, \not\bot$
- $\wedge, \not\wedge$
- $\vee, \dot\vee, \not\vee, \not\dot\vee$
- $\reflectbox{$\rightarrow$}, \reflectbox{$\not\rightarrow$}, \not\reflectbox{$\rightarrow$}, \not\reflectbox{$\not\rightarrow$}$
- $\leftarrow, \leftrightarrow, \rightarrow, \not\leftarrow, \not\leftrightarrow, \not\rightarrow$

Nichtlogische Operatoren

- $=, \neq, :=$

A.2.3. Formalisierung

Da Computerprogramme verwendet werden, müssen die Axiome, Sätze, Beweise, etc. in streng formaler Form vorliegen. Die Formalisierung stützt sich im Wesentlichen auf [27]. (siehe auch [20, 23]).

Es werden folgende Mengen[5] definiert:

$\mathbb{N}_0$		Menge der *natürlichen Zahlen* einschließlich 0.
$\mathcal{V}$	$:= \{P_n \mid n \in \mathbb{N}_0\}$	Menge der *atomaren Formeln* (Satzbuchstaben), kurz: *Atome*.
$\mathcal{J}$	$:= \mathcal{U} \cup \mathcal{B} \cup \mathcal{G}$	Menge der *Junktoren* und Gliederungszeichen, mit:
$\mathcal{U}$	$:= \{\neg\}$	Menge der *unären Operatoren*.
$\mathcal{B}$	$:= \{\wedge, \vee, \rightarrow, \leftrightarrow\}$	Menge der *binären Operatoren*.
$\mathcal{G}$	$:= \{(,)\}$	Menge der *Gliederungszeichen*.
$\mathcal{A}$	$:= \mathcal{V} \cup \mathcal{J}$	*Alphabet* der logischen Sprache.

Die Symbole werden noch um die weiteren in der Tabelle A.1 auf der vorherigen Seite vorhandenen, bisher nicht verwendeten Symbole ergänzt.

$\mathcal{J}_\mathrm{e}$	$:= \mathcal{K} \cup \mathcal{U}_\mathrm{e} \cup \mathcal{B}_\mathrm{e} \cup \mathcal{G}$	*Erweiterte* Menge der Junktoren und Gliederungszeichen, mit:
$\mathcal{K}$	$:= \{\top, \bot, \not\top, \not\bot\}$	Menge der *Konstanten*.
$\mathcal{U}_\mathrm{e}$	$:= \mathcal{U} \cup \{\top, \bot, \not\top, \not\bot\}$	*Erweiterte* Menge der unären Operatoren.

[4] Unäre Operatoren - außer Klammern - stehen hier stets links *vor* dem Operanden, so dass es gar keine andere Möglichkeit gibt.
[5] Hier wird die naive Mengenlehre vorausgesetzt.

$\mathcal{B}_\mathrm{e} := \mathcal{B} \cup \{\leftarrow, \supset, \subset, \dot{\vee}, \top, \bot, \not\leftarrow, \not\rightarrow, \rightarrow, \leftrightarrow, \nleftarrow, \not\supset, \not\subset, \dot{\not\vee}, \overline{\top}, \underline{\bot}\}$
Erweiterte Menge der binären Operatoren.

$\mathcal{A}_\mathrm{e} := \mathcal{V} \cup \mathcal{J}_\mathrm{e}$ *Erweitertes* Alphabet der logischen Sprache.

> > > AUSSAGENLOGIK weiter bearbeiten. < < <

A.3. Prädikatenlogik

> > > PRÄDIKATENLOGIK bearbeiten. < < <

A.4. Mengenlehre

> > > PRÄDIKATENLOGIK bearbeiten. < < <

A.5. Offene Aufgaben

1. Formale Syntax definieren
2. Datenstruktur definieren
3. Prüfung der Beweise definieren
4. Axiome für das System bestimmen
5. Eingabeprogramm erstellen (liest XML)
6. Prüfprogramm erstellen
7. Ausgabeprogramm erstellen (schreibt XML)
8. Formelausgabe erstellen (erzeugt LaTeX aus XML)
9. Axiome sammeln und eingeben
10. Sätze sammeln und eingeben
11. Beweise sammeln und eingeben
12. Fachbegriffe und Symbole sammeln und eingeben
13. Fachgebiete sammeln und eingeben
14. Ausgabeschemata sammeln und eingeben

Tabellenverzeichnis

Abbildungsverzeichnis

Literaturverzeichnis

[1] *Apache License*, Version 2.0 → http://www.apache.org/licenses/LICENSE-2.0

[2] *Boost Software License* 1.0 → http://www.boost.org/users/license.html

[3] *Eclipse Public License* Version 1.0 → http://www.eclipse.org/org/documents/epl-v10.php

[4] *GNU Affero General Public License* → http://www.gnu.org/licenses/agpl

[5] *GNU General Public License* → http://www.gnu.org/licenses/old-licenses/gpl-1.0

[6] *GNU General Public License*, Version 2 → http://www.gnu.org/licenses/old-licenses/gpl-2.0

[7] *GNU Lesser General Public License*, Version 2.1 → http://www.gnu.org/licenses/old-licenses/lgpl-2.1

[8] Lizenz für *Clover* → https://www.atlassian.com/software/clover

[9] Lizenz für *Microsoft Visual Studio Community* (Microsoft Visual Studio Express 2015) → https://www.visualstudio.com/de/license-terms/mt171551/

[10] Lizenz für *MikTeX* → https://miktex.org/kb/copying

[11] Lizenz für *SAX* → http://www.saxproject.org/copying.html

[12] *MIT License* → https://opensource.org/licenses/MIT/

[13] *Oracle Binary Code License Agreement* → http://java.com/license

[14] *OSI Certified Open Source Software* → https://opensource.org/pressreleases/certified-open-source.php

[15] *W3C Document License* → http://www.w3.org/Consortium/Legal/2015/doc-license

[16] *W3C Software Notice and License* → http://www.w3.org/Consortium/Legal/2002/copyright-software-20021231.html

[17] *Hilbert II - Introduction* → http://www.qedeq.org/

[18] *Formal Correct Mathematical Knowledge*: GitHub Repository von Projekt Hilbert II → https://github.com/m-31/qedeq/

[19] *ASBA - Axiome, Sätze, Beweise und Ausgaben*. Projekt zur maschinellen Überprüfung von mathematischen Beweisen und deren Ausgabe in lesbarer Form: GitHub Repository von Projekt ASBA → https://github.com/Dr-Winfried/ASBA

[20] Meyling, Michael: *Anfangsgründe der mathematischen Logik* - 24. Mai 2013 (in Bearbeitung) → http://www.qedeq.org/current/doc/math/qedeq_logic_v1_de.pdf

[21] Meyling, Michael: *Formale Prädikatenlogik* - 24. Mai 2013 (in Bearbeitung) → http://www.qedeq.org/current/doc/math/qedeq_formal_logic_v1_de.pdf

[22] Meyling, Michael: *Axiomatische Mengenlehre* - 24. Mai 2013 (in Bearbeitung) → http://www.qedeq.org/current/doc/math/qedeq_set_theory_v1_de.pdf

[23] Meyling, Michael: *Elements of Mathematical Logic* - May 24, 2013 (in Bearbeitung) → http://www.qedeq.org/current/doc/math/qedeq_logic_v1_en.pdf

[24] Meyling, Michael: *Formal Predicate Calculus* - May 24, 2013 (in Bearbeitung) → http://www.qedeq.org/current/doc/math/qedeq_formal_logic_v1_en.pdf

[25] Meyling, Michael: *Axiomatic Set Theory* - May 24, 2013 (in Bearbeitung) → http://www.qedeq.org/current/doc/math/qedeq_set_theory_v1_en.pdf

[26] Wikipedia: *Aussagenlogik* Kapitel 2.2 *Mögliche Junktoren* - 02.03.2017 → https://de.wikipedia.org/wiki/Junktor#M.C3.B6gliche_Junktoren

[27] Wikipedia: *Aussagenlogik* Kapitel 4 *Formaler Zugang* - 24.02.2017 → https://de.wikipedia.org/wiki/Aussagenlogik#Formaler_Zugang

[28] Wikipedia: *Prädikatenlogik erster Stufe* - 24.02.2017 → https://de.wikipedia.org/wiki/Pr%C3%A4dikatenlogik_erster_Stufe

[29] Wikipedia: *Mengenlehre* - 24.02.2017 → https://de.wikipedia.org/wiki/Mengenlehre