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

Projektdokument

ASBA
Axiome, Sätze, Beweise und Auswertungen

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

Winfried Teschers

7. April 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 [18, 19]). 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.

Frage \ Eigenschaft	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 → Eigenschaften

1.2. Eigenschaften

Ausgehend von den Fragen in Abschnitt 1.1 auf der vorherigen Seite 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 Mathematikern akzeptiert und verwendet.

Tabelle 1.1 zeigt, wie sich die Eigenschaften zu den Fragen in Abschnitt 1.1 auf der vorherigen Seite verhalten. Mit einem X werden die Spalten einer Zeile markiert, deren zugehörige Eigenschaften zur Beantwortung der entsprechenden Frage beitragen sollen. Idealerweise sollte die Erfüllung aller angegebenen Eigenschaften alle gestellten Fragen beantworten, was allerdings illusorisch ist.

1.3. Ziele

Um die Eigenschaften von Abschnitt 1.2 auf der vorherigen Seite zu erreichen, werden für das System die folgenden Ziele[3] gesetzt:

1. *Daten*: Es enthält möglichst viele wichtige Axiome, Sätze, Beweise, Fachbegriffe, Fachgebiete und Ausgabeschemata[4].
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.
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[5] er verwendet.
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 online und offline 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*.

Der Punkt 16 wurde noch eingefügt, damit das System effizient arbeiten kann und um die Akzeptanz zu erhöhen:

16. *Zwischenspeicher*: Wichtige Auswertungen können angehängt an vorhandene oder separat in eigenen Dateien zwischengespeichert werden.

Tabelle 1.2 auf der nächsten Seite zeigt wieder, wie sich die Ziele zu den Eigenschaften in Abschnitt 1.2 auf der vorherigen Seite verhalten. Mit einem X werden wieder die Spalten einer Zeile markiert, deren zugehörige Ziele zur Sicherstellung der entsprechenden Eigenschaft beitragen sollen. Idealerweise sollte durch Erreichen aller aufgestellten Ziele das System alle angegebenen Eigenschaften aufweisen, was wahrscheinlich ebenfalls illusorisch ist.

[3] Es sind eigentlich Anforderungen. Da dieser Begriff aber auch im Kapitel 2 auf Seite 9 verwendet wird, werden die Anforderungen hier *Ziele* genannt.
[4] Um den Punkt 4 von Abschnitt 1.2 auf der vorherigen Seite erfüllen zu können, werden noch fachgebietsspezifische Ausgabeschemata benötigt, welche die Art der Ausgaben beschreiben.
[5] Sätze, die quasi als Axiome verwendet werden.

Eigenschaft \ Ziel	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	16 Zwischenspeicher
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	X

Tabelle 1.2.: Eigenschaften → Ziele

1.4. Zusammenfassung

Die Tabelle 1.3 auf der nächsten Seite ist eine Kombination aus den Tabellen 1.1 auf Seite 5 und 1.2 und zeigt, wie sich die Ziele in Abschnitt 1.3 auf der vorherigen Seite zu den Fragen in Abschnitt 1.1 auf Seite 4 verhalten. Auch hier werden mit einem X die Spalten einer Zeile markiert, deren zugehörige Ziele für die Beantwortung der entsprechenden Frage nötig sind. Mit einem kleinen x werden sie markiert, wenn sie zur Beantwortung der Fragen nicht nötig, aber von Interesse sind. Idealerweise sollte das Erreichen aller aufgestellten Ziele wieder alle gestellten Fragen beantworten, was natürlich auch illusorisch ist.

Frage \ Ziel	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	-	-	-	-	-	-	-	-
Die nächsten beiden Punkte sind Eigenschaften aus Abschnitt 1.2 auf Seite 5:															
6 Lizenz	-	-	-	-	-	-	-	-	-	-	-	-	-	-	X
7 Akzeptanz	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X

Tabelle 1.3.: Fragen → Ziele

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 [20]).

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 den Zielen in Abschnitt 1.3 auf Seite 6. Die beiden Ziele 1 *Daten* und 15 *Lizenz* sind für die Entwicklung des Systems von sekundärer Bedeutung und werden daher in diesen Abschnitt nicht übernommen. Die anderen Ziele werden noch verfeinert.

> > > ZIELE in Anforderungen umwandeln. < < <

1. *Form*: Die Daten liegt in formaler, geprüfter Form vor. (siehe Ziel 2 auf Seite 6)
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 6)
3. *Prüfung*: Vorhandene Beweise können automatisch geprüft werden. (siehe Ziel 4 auf Seite 6)
4. *Ausgaben*: Die Ausgabe kann in einer eindeutigen, formalen Syntax gemäß vorhandener Ausgabeschemata erfolgen. (siehe Ziel 5 auf Seite 6)
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 verwendet. (siehe Ziel 6 auf Seite 6)

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

6. *Anpassbarkeit*: Fachbegriffe und die Darstellung bei der Ausgabe können mit Hilfe von – gegebenenfalls unbenannten – untergeordneten Fachgebieten angepasst werden. (siehe Ziel 7 auf Seite 6)
7. *Individualität*: Axiome und Sätze können für jeden Beweis individuell vorausgesetzt werden. Dabei sind fachgebietsspezifische Fachbegriffe erlaubt. (siehe Ziel 8 auf Seite 6)
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 6)
9. *Kommunikation*: Die Kommunikation mit dem System kann mit den Fachbegriffen der einzelnen Fachgebiete erfolgen. (siehe Ziel 10 auf Seite 6)
10. *Zugriff*: Der Zugriff auf das System kann lokal und über das Internet erfolgen. (siehe Ziel 11 auf Seite 6)
11. *Unabhängigkeit*: Das System kann offline und online arbeiten. (siehe Ziel 12 auf Seite 6)
12. *Rekursion*: Es kann rekursiv über alle verwendeten Dateien – auch solchen, die im Internet liegen – ausgewertet werden. (siehe Ziel 13 auf Seite 6)
13. *Bedienbarkeit*: Das System ist einfach zu bedienen. (siehe Ziel 14 auf Seite 6)

> > > ANFORDERUNGEN bearbeiten. < < <

2.2. Datenstruktur

> > > DATENSTRUKTUR bearbeiten. < < <

2.3. Bausteine

> > > BAUSTEINE bearbeiten. < < <

A. Projektumgebung

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, muss es aber nicht.

Werkzeuge, die zum Ablauf der Software erforderlich sind

- *MiKTEX* für Dokumentation und Ausgaben in LaTeX. → https://miktex.org/ – Lizenz siehe [11]

Werkzeuge, die für die Entwicklung verwendet werden

- *GitHub* als Online Konfigurationsmanagementsystem zur Zusammenarbeit verschiedener Entwickler. → https://github.com/ – Lizenz siehe [7]
- GitHub benötigt *Git* als Konfigurationsmanagementsystem. → https://git-scm.com/ – Lizenz siehe [7]
- *Visual Studio Community 2017*[1] (*VS*) als Entwicklungsumgebung für C++. → https://www.visualstudio.com/downloads/ – Lizenz siehe [10]
- *Doxygen* als Dokumentationssystem für C++. → http://www.stack.nl/~dimitri/doxygen/ – Lizenz siehe [7]
- Doxygen benötigt *Ghostscript* als Interpreter für Postscript und PDF. → http://ghostscript.com/ – Lizenz siehe [5]
- Doxygen benötigt *Graphviz* mit *Dot* zur Erzeugung und Visualisierung von Graphen. → http://www.graphviz.org/Home.php – Lizenz siehe [4]

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

- *TEXstudio* als Editor für LaTeX. → http://www.texstudio.org/ – Lizenz siehe [7]
- *Strawberry Perl* als Interpreter für Perl. → http://strawberryperl.com/ – Lizenz: Various OSI-compatible Open Source licenses, or given to the public domain
- *Notepad++* als Text-Editor. → https://notepad-plus-plus.org/ – Lizenz siehe [6]
- *WinMerge* zum Vergleich von Dateien und Verzeichnissen. → http://winmerge.org/ – Lizenz siehe [6]

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

Angedachte Werkzeuge

- In *Visual Studio Community 2015* integrierte Datenbank für Axiome, Sätze, Beweise, Fachbegriffe und Fachgebiete. – Lizenz siehe [10]
- *RapidXml* für Ein- und Ausgabe in XML. → `http://rapidxml.sourceforge.net/index.htm` – Lizenz siehe wahlweise [3] oder [13]

Im Projekt *qedeq* verwendete Werkzeuge > > > QEDEQ Werkzeuge auflisten? < < <

- *Java* als Programmiersprache – Laufzeitumgebung. → `https://www.java.com/de/download/win10.jsp` – Lizenz siehe [14]
- *Apache Ant* als Java Bibliothek und Kommandozeilen-Werkzeug um Java Programme zu erzeugen. → `http://ant.apache.org/` – Lizenz siehe [2]
- *Checkstyle* zur statischen Code-Analyse für Java. → `http://checkstyle.sourceforge.net/` – Lizenz siehe [8]
- *Clover*[2] als Testwerkzeug zur Analyse der Code-Abdeckung. → `https://www.atlassian.com/software/clover/` – Lizenz siehe [9]
- *Eclipse IDE for Java Developers* als Entwicklungsumgebung für Java. → `http://www.eclipse.org/downloads/packages/eclipse-ide-java-developers/neon1a/` – Lizenz siehe [15]
- *JUnit* zur Erzeugung von wiederholbaren Tests. → `http://junit.org/junit4/` – Lizenz siehe [4]
- *Xerces2* als XML-Parser in Java. → `http://xerces.apache.org/xerces2-j/` – Lizenzen siehe [2, 12, 16, 17]

[2] Clover ist proprietäre Software, aber auf Anfrage frei für 30 Tage. Danach ist eine einmalige Lizenzgebühr fällig.

B. Mathematische Grundlagen

B.1. Aussagenlogik

B.1.1. Konstante und Operatoren

Die Tabelle B.1 auf der nächsten Seite [1] definiert für die zweiwertige Logik Konstanten- und Operatorsymbole über die Wahrheitswerte ihrer Anwendung. So ergeben sich, abhängig von den Wahrheitswerten der Operanden A und B [2], 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, die unären und die binären Operatoren ausgefüllt. Dabei werden die Konstanten als 0-stellige Operatoren angesehen. Hat der Inhalt einer Zelle keine Relevanz, steht dort ein Minuszeichen, ist kein Wert bekannt, so bleibt sie leer.

Für einige Junktoren, Namen und Sprechweisen sind auch Alternativen angegeben. Die durchgestrichenen (d. h. negierten) Symbole sind ungebräuchlich und nur aus formalen Gründen aufgeführt. Wenn für eine bestimmte Kombination von Wahrheitswerten mehr als eine Zeile angegeben ist, so sind die zugehörigen Operationen in der zweiwertigen Aussagenlogik alle gleich. Bei der formalen Definition setzen wir aber keine Zweiwertigkeit voraus, so dass je nach Definition die Operationen verschiedene Ergebnisse liefern können.

Um vollständig zu sein, d. h. alle 22 möglichen Kombinationen von Wahrheitswerten für höchstens zwei Variable zu berücksichtigen, enthält die Tabelle auch viele ungebräuchliche Junktoren und Operationen. Die Zeilen mit den Klammern und den gebräuchlichsten Junktoren sind in der Tabelle grau hinterlegt. Hellgrau hinterlegt sind Zeilen mit weniger gebräuchlichen Junktoren. Die restlichen Operationen sind uninteressant und brauchen daher keine Priorität.

B.1.2. Klammerregeln

Zur Klammerersparnis werden die üblichen Regeln verwendet, d. h. dass Operatoren mit höherer Priorität stärker binden, als solche mit niedrigerer Priorität, so dass redundante Klammern weggelassen werden können. Bei gleicher Priorität binden Klammern von innen nach außen. Für die Operatoren gilt Rechtsklammerung[3]. Es gilt also mit abnehmender Priorität:

Klammern

- $(\dots)$

Unäre logische Operatoren

- $\neg$

Binäre logische Operatoren

[1] Die Tabelle basiert auf den Wahrheitstafeln in [27] Kapitel 2.2 und [1] Kapitel 1.1 Seite 3.

[2] Im Gegensatz zu Paragraph B.1.3.1 auf Seite 15 können A und B hier beliebige Aussagen – auch Formeln – sein, nicht nur Atome.

[3] Unäre Operatoren – außer Klammern – stehen hier stets links *vor* dem Operanden, so dass es nur Rechtsklammerung geben kann. Zur Rechtsklammerung bei binären Operationen hier ein Zitat aus [1] Kapitel 1.1 Seite 5: *Diese hat gegenüber Linksklammerung Vorteile bei der Niederschrift von Tautologien in* $\rightarrow$*, [...]*

A	-	W	F	W	W	F	F	-	Aussage A	-
B	-	-	-	W	F	W	F	-	Aussage B	-
Junktor[1]	**0**	**1**		**2**				**Name**	**Sprechweise**[2]	**Prio**
$\top$	W	-	-	-	-	-	-	Verum	Wahr	-
$\bot$	F	-	-	-	-	-	-	Falsum	Falsch	-
	-	W	W	-	-	-	-			-
$(\dots)$	-	W	F	-	-	-	-	Klammerung[3]	A ist geklammert	6[4]
$\neg$	-	F	W	-	-	-	-	Negation	Nicht A	5[5]
	-	F	F	-	-	-	-			-
	-	-	-	W	W	W	W	Tautologie		-
$\vee$	-	-	-	W	W	W	F	Disjunktion; Adjunktion; Alternative	A oder B	3
$\leftarrow \Leftarrow \subset$	-	-	-	W	W	F	W	Replikation; Konversion	A folgt aus B	2
$\rfloor$	-	-	-	W	W	F	F	Präpendenz	Identität von A	-
$\rightarrow \Rightarrow \supset$	-	-	-	W	F	W	W	Implikation; Subjunktion; Konditional	Wenn A so B; Aus A folgt B; A nur dann wenn B	2
$\lfloor$	-	-	-	W	F	W	F	Postpendenz	Identität von B	-
$\leftrightarrow \Leftrightarrow$	-	-	-	W	F	F	W	Äquivalenz; Bijunktion; Bikonditional	A genau dann wenn B; A dann und nur dann wenn B	1
$\not\leftrightarrow$	-	-	-	"	"	"	"			-
$\wedge$ & $\cdot$	-	-	-	W	F	F	F	Konjunktion	A und B; Sowohl A als auch B	4
$\uparrow \barwedge \mid$	-	-	-	F	W	W	W	NAND; Unverträglichkeit; Sheffer-Funktion	Nicht zugleich A und B	4
$+ \dot{\vee} \veebar \oplus$	-	-	-	F	W	W	F	XOR; Antivalenz; ausschließende Disjunktion	Entweder A oder B	3
$\nleftrightarrow \not\Leftrightarrow \not\equiv$	-	-	-	"	"	"	"	Kontravalenz		-
$\lceil$	-	-	-	F	W	F	W	Postnonpendenz	Negation von B	-
$\nrightarrow \not\Rightarrow \nexists$	-	-	-	F	W	F	F	Postsektion		-
$\rceil$	-	-	-	F	F	W	W	Pränonpendenz	Negation von A	-
$\not\leftarrow \not\Leftarrow \nexists$	-	-	-	F	F	W	F	Präsektion		-
$\downarrow \barvee$	-	-	-	F	F	F	W	NOR; Nihilation; Peirce-Funktion	Weder A noch B	3
	-	-	-	F	F	F	F	Kontradiktion		-

[1] *Operatorsymbole* – '$\subset$', '$\supset$', '$\nexists$' und '$\nexists$', dürfen nicht mit den entsprechenden Mengensymbolen verwechselt werden; gleiches gilt für '$+$' und '$\cdot$' mit Addition und Multiplikation.

[2] Ist eine Zelle in dieser Spalte leer, so ist die zugehörige Zeile nur vorhanden, um alle binären Operationen aufzuführen.

[3] Klammerung ist genau genommen kein Operator und wird nicht nur bei logischen, sondern auch bei anderen Ausdrücken verwendet.

[4] Die Priorität der Klammern ist größer als die aller Operatoren.

[5] Die Priorität der unären Operatoren muss größer sein als die aller mehrwertigen, also auch der binären Operatoren. Wenn alle unären Operatoren auf derselben Seite des Operanden stehen, brauchen sie eigentlich keine Priorität, da die Auswertung nur von innen (dem Operanden) nach außen erfolgen kann.

Tabelle B.1.: Definition von aussagenlogischen Symbolen.

- $\wedge \quad \cdot \quad \uparrow$
- $\vee \quad + \quad \downarrow$
- $\leftarrow \quad \rightarrow$
- $\leftrightarrow$

Nichtlogische Zeichen

- $= \quad := \quad \equiv$

Die Prioritäten der logischen Operatoren wurden aus [1] Kapitel 1.1 Seite 5 entnommen und ergänzt.

B.1.3. Formalisierung

Da Computerprogramme verwendet werden, müssen die Axiome, Sätze, Beweise, etc. in streng formaler Form vorliegen. Die Formalisierung stützt sich auf [28]; siehe auch [21, 24].

Computerprogramme können mit der *Polnischen Notation*[4] besser umgehen und Klammern sind dort überflüssig. Daher werden viele Formeln auch in die Polnische Notation überführt.

B.1.3.1. Bausteine der aussagenlogischen Sprache

Für die Definition von neuen Elementen wird wie üblich ':=' verwendet. Damit werden zur Erfassung der logischen Symbole die folgenden Mengen[5] definiert:

$\mathbb{N}_0$	$:=$		Menge der *natürlichen Zahlen* einschließlich 0.	(B.1)
$\mathcal{C}$	$:=$	$\{\top, \bot\}$	Menge der *Konstanten*.	(B.2)
$\mathcal{U}$	$:=$	$\{\neg\}$	Menge der *unären Operatoren*.	(B.3)
$\mathcal{B}$	$:=$	$\{\wedge, \vee, \rightarrow, \leftrightarrow\}$	Menge der *binären Operatoren*	(B.4)
$\mathcal{B}_\mathrm{e}$	$:=$	$\mathcal{B} \cup \{\cdot, +, \uparrow, \downarrow, \leftarrow\}$	*Erweiterte* Menge der binären Operatoren.	(B.5)

Damit sind alle in der Tabelle B.1 auf der vorherigen Seite verwendeten wesentlichen Konstanten und Operatoren[6] erfasst und es können die folgende Mengen definiert werden:

$\mathcal{V}$	$:=$	$\{P_n \mid n \in \mathbb{N}_0\}$	Menge der *atomaren Formeln*	(B.6)
$\mathcal{J}$	$:=$	$\mathcal{U} \cup \mathcal{B}$	Menge der *Junktoren*, bzw. Operatoren.	(B.7)
$\mathcal{S}$	$:=$	$\mathcal{U} \cup \mathcal{B}_\mathrm{e} \cup \mathcal{C}$	Menge der *Symbole*.	(B.8)
$\mathcal{A}$	$:=$	$\mathcal{V} \cup \mathcal{J}$	.	(B.9)
$\mathcal{A}_\mathrm{e}$	$:=$	$\mathcal{V} \cup \mathcal{S}$	*Erweitertes* Alphabet der logischen Sprache.	(B.10)

Für Elemente aus $\mathcal{V}$ werden auch die großen lateinischen Buchstaben $A, B, C, \ldots$ verwendet.

Für alle endlichen Formeln und endlichen Mengen davon – und nur solche betrachten wir – wird jeweils nur eine endliche Teilmenge aus $\mathbb{N}_0$ gebraucht. Somit gibt es keine Schwierigkeiten mit unendlichen Mengen. Die atomaren Formeln werden auch *Satzbuchstaben* oder kurz *Atome*. genannt.

[4] Bei der *Polnischen Notation* wird eine zweistellige Operation $(A \circ B)$ dargestellt als $\circ AB$. Eine Zwischenstufe ist $\circ(A, B)$, bei der noch die redundanten Gliederungszeichen Komma und Klammern – auch andere als die runden – hinzukommen, so dass die Operationen optisch besser getrennt und dadurch für Menschen besser lesbar werden. Durch einfaches Weglassen der Gliederungszeichen ergibt sich dann die Polnische Notation.

[5] Hier wird die naive Mengenlehre vorausgesetzt.

[6] Jeweils nur die ersten der grau hinterlegten Zeilen sowie '$\cdot$'. Man beachte, dass '$= \neq :=$' und Klammerung keine logischen Operatoren sind – siehe B.1.2 auf Seite 13.

B.1.3.2. Formationsregeln

Neben dem (erweiterten) Alphabet werden noch Klammern als Gliederungszeichen verwendet. Damit können nun rekursiv drei Mengen von Formeln definiert werden:

$\mathcal{F}$ sei die Menge der auf folgende Weise definierten *aussagenlogischen Formeln*:

$$\mathcal{V} \subset \mathcal{F} \tag{B.11}$$
$$A \in \mathcal{F} \quad \text{dann auch} \quad (\circ A) \in \mathcal{F} \quad \text{für } \circ \in \mathcal{U} \tag{B.12}$$
$$A, B \in \mathcal{F} \quad \text{dann auch} \quad (A \circ B) \in \mathcal{F} \quad \text{für } \circ \in \mathcal{B} \tag{B.13}$$

Nur die auf diese Weise konstruierten Formeln sind Elemente von $\mathcal{F}$.

$\mathcal{F}_\mathrm{e}$ sei die Menge der auf folgende Weise definierten *erweiterten* aussagenlogischen Formeln:

$$\mathcal{V}, \mathcal{C} \subset \mathcal{F}_\mathrm{e} \tag{B.14}$$
$$A \in \mathcal{F}_\mathrm{e} \quad \text{dann auch} \quad (\circ A) \in \mathcal{F}_\mathrm{e} \quad \text{für } \circ \in \mathcal{U} \tag{B.15}$$
$$A, B \in \mathcal{F}_\mathrm{e} \quad \text{dann auch} \quad (A \circ B) \in \mathcal{F}_\mathrm{e} \quad \text{für } \circ \in \mathcal{B}_\mathrm{e} \tag{B.16}$$

Nur die auf diese Weise konstruierten Formeln sind Elemente von $\mathcal{F}_\mathrm{e}$.

$\mathcal{F}^\mathrm{p}$ sei die Menge der auf folgende Weise definierten aussagenlogischen Formeln in *Polnischer Notation*:

$$\mathcal{V} \subset \mathcal{F}^\mathrm{p} \tag{B.17}$$
$$A \in \mathcal{F}^\mathrm{p} \quad \text{dann auch} \quad \circ A \in \mathcal{F}^\mathrm{p} \quad \text{für } \circ \in \mathcal{U} \tag{B.18}$$
$$A, B \in \mathcal{F}^\mathrm{p} \quad \text{dann auch} \quad \circ AB \in \mathcal{F}^\mathrm{p} \quad \text{für } \circ \in \mathcal{B} \tag{B.19}$$

Nur die auf diese Weise konstruierten Formeln sind Elemente von $\mathcal{F}^\mathrm{p}$.

$\mathcal{F}_\mathrm{e}^\mathrm{p}$ sei die Menge der auf folgende Weise definierten *erweiterten* aussagenlogischen Formeln in Polnischer Notation:

$$\mathcal{V}, \mathcal{C} \subset \mathcal{F}_\mathrm{e}^\mathrm{p} \tag{B.20}$$
$$A \in \mathcal{F}_\mathrm{e}^\mathrm{p} \quad \text{dann auch} \quad \circ A \in \mathcal{F}_\mathrm{e}^\mathrm{p} \quad \text{für } \circ \in \mathcal{U} \tag{B.21}$$
$$A, B \in \mathcal{F}_\mathrm{e}^\mathrm{p} \quad \text{dann auch} \quad \circ AB \in \mathcal{F}_\mathrm{e}^\mathrm{p} \quad \text{für } \circ \in \mathcal{B}_\mathrm{e} \tag{B.22}$$

Nur die auf diese Weise konstruierten Formeln sind Elemente von $\mathcal{F}_\mathrm{e}$.

Wie man leicht sieht, ist $\mathcal{J} \subset \mathcal{S}$ und $\mathcal{X} \subset \mathcal{X}_\mathrm{e}$ für $\mathcal{X} \in \{\mathcal{A}, \mathcal{B}, \mathcal{F}, \mathcal{F}^\mathrm{p}\}$. Durch Anwendung der Klammerregeln von Paragraph B.1.3.1 auf der vorherigen Seite lassen sich in der Regel noch die meisten Klammern der Formeln aus $\mathcal{F}$ und $\mathcal{F}_\mathrm{e}$ einsparen. Die Formeln aus $\mathcal{F}^\mathrm{p}$ und $\mathcal{F}_\mathrm{e}^\mathrm{p}$ sind Klammerfrei. Die Namen der Operationen finden sich in der Tabelle B.1 auf Seite 14. Für Elemente von $\mathcal{X} \in \{\mathcal{F}, \mathcal{F}_\mathrm{e}, \mathcal{F}^\mathrm{p}, \mathcal{F}_\mathrm{e}^\mathrm{p}\}$ die kleinen griechischen Buchstaben φ, ψ, ... verwendet.

B.1.3.3. Aussagenlogische Axiome

> > > AUSSAGENLOGIK weiter bearbeiten. < < <

B.2. Prädikatenlogik

> > > PRÄDIKATENLOGIK bearbeiten. < < <

B.3. Mengenlehre

> > > PRÄDIKATENLOGIK bearbeiten. < < <

B.4. Offene Aufgaben

1. TODOs bearbeiten
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

C. Ideen

C.1. Definition von Junktoren durch andere

In den Formeln stehen jeweils links die Formeln in üblicher Schreibweise mit Klammern und rechts in Polnischer Notation (ohne Klammern).

C.1.1. logisches Axiomensystem

Gegebene Operatoren: $\neg, \wedge, \rightarrow$

Axiome:

$$(\alpha \rightarrow \beta \rightarrow \gamma) \rightarrow (\alpha \rightarrow \beta) \rightarrow (\alpha \rightarrow \gamma) \qquad \rightarrow\rightarrow \alpha \rightarrow \beta\gamma \rightarrow\rightarrow \alpha\beta \rightarrow \alpha\gamma \tag{C.1}$$

$$\alpha \rightarrow \beta \rightarrow \alpha \wedge \beta \qquad \rightarrow \alpha \rightarrow \beta \wedge \alpha\beta \tag{C.2}$$

$$\alpha \wedge \beta \rightarrow \alpha; \quad \alpha \wedge \beta \rightarrow \beta \qquad \rightarrow \wedge\alpha\beta\alpha; \quad \rightarrow \wedge\alpha\beta\beta \tag{C.3}$$

$$(\alpha \rightarrow \neg\beta) \rightarrow (\beta \rightarrow \neg\alpha) \qquad \rightarrow\rightarrow \alpha\neg\beta \rightarrow \beta\neg\alpha \tag{C.4}$$

Definierte Operatoren: $\vee, \leftrightarrow, \cdot, +, \uparrow, \downarrow, \leftarrow, \bot, \top$

$$(\alpha \vee \beta) := \neg(\neg\alpha \rightarrow \beta) \qquad \vee\alpha\beta := \neg \rightarrow \neg\alpha\beta \tag{C.5}$$

$$(\alpha \leftrightarrow \beta) := ((\alpha \rightarrow \beta) \wedge (\beta \rightarrow \alpha)) \qquad (\alpha \leftrightarrow \beta) := ((\alpha \rightarrow \beta) \wedge (\beta \rightarrow \alpha)) \tag{C.6}$$

$$(\alpha \cdot \beta) := (\alpha \wedge \beta) \qquad (\alpha \cdot \beta) := (\alpha \wedge \beta) \tag{C.7}$$

$$(\alpha + \beta) := ((\alpha \vee \beta) \wedge \neg(\alpha \wedge \beta)) \qquad (\alpha + \beta) := ((\alpha \vee \beta) \wedge \neg(\alpha \wedge \beta)) \tag{C.8}$$

$$(\alpha \uparrow \beta) := \neg(\alpha \wedge \beta) \qquad (\alpha \uparrow \beta) := \neg(\alpha \wedge \beta) \tag{C.9}$$

$$(\alpha \downarrow \beta) := \neg(\alpha \vee \beta) \qquad (\alpha \downarrow \beta) := \neg(\alpha \vee \beta) \tag{C.10}$$

$$(\alpha \leftarrow \beta) := (\beta \rightarrow \alpha) \qquad (\alpha \leftarrow \beta) := (\beta \rightarrow \alpha) \tag{C.11}$$

$$\bot := (p_0 \wedge \neg p_0) \qquad \bot := (p_0 \wedge \neg p_0) \tag{C.12}$$

$$\top := \neg\bot \qquad \top := \neg\bot \tag{C.13}$$

Zu zeigen

$$(\alpha \rightarrow \beta) \equiv \neg(\alpha \wedge \neg\beta) \qquad \rightarrow \alpha\beta \equiv \neg \wedge \alpha\neg\beta \tag{C.14}$$

C.1.2. nicht, und, oder

Gegebene Operatoren: $\neg, \wedge, \vee$

Definierte Operatoren: $\rightarrow, \leftrightarrow, \cdot, +, \uparrow, \downarrow, \leftarrow, \bot, \top$

(C.15)

Zu zeigen:

$$(\alpha \vee \beta) := \neg(\neg\alpha \wedge \neg\beta) \tag{C.16}$$

C.1.3. nicht, und

Gegebene Operatoren: $\neg, \wedge$

Definierter Operator: $\vee$

$$(\alpha \vee \beta) := \neg(\neg\alpha \wedge \neg\beta) \tag{C.17}$$

Zu zeigen:

$$(\alpha \vee \beta) \equiv \neg(\neg\alpha \wedge \neg\beta) \tag{C.18}$$

Zur Definition der Operatoren $\rightarrow, \leftrightarrow, \cdot, +, \uparrow, \downarrow, \leftarrow, \bot, \top$ siehe Unterabschnitt C.1.2 auf der vorherigen Seite

C.1.4. nicht, oder

Gegebene Operatoren: $\neg, \vee$

Definierter Operator: $\wedge$

$$(\alpha \wedge \beta) := \neg(\neg\alpha \vee \neg\beta)) \tag{C.19}$$

C.1.5. nicht, impliziert

Gegebene Operatoren: $\neg, \rightarrow$

Definierte Operatoren: $\wedge, \vee$

$$(\alpha \wedge \beta) := \ldots \tag{C.20}$$
$$(\alpha \vee \beta) := \ldots \tag{C.21}$$

C.1.6. NAND

Gegebener Operator: $\uparrow$

Definierte Operatoren: $\neg, \wedge, \vee$

$$\neg\alpha := \ldots \tag{C.22}$$
$$(\alpha \wedge \beta) := \ldots \tag{C.23}$$
$$(\alpha \vee \beta) := \ldots \tag{C.24}$$

C.1.7. NOR

Gegebener Operator: $\downarrow$

Definierte Operatoren: $\neg, \wedge, \vee$

$$\neg\alpha := \ldots \tag{C.25}$$
$$(\alpha \wedge \beta) := \ldots \tag{C.26}$$
$$(\alpha \vee \beta) := \ldots \tag{C.27}$$

D. Verzeichnisse

Abbildungsverzeichnis

Tabellenverzeichnis

Symbolverzeichnis

Literaturverzeichnis

[1] Wolfgang Rautenberg, *Einführung in die Mathematische Logik*: Ein Lehrbuch, 3. Auflage, Vieweg+Teubner 2008

[2] *Apache License*, Version 2.0 → http://www.apache.org/licenses/LICENSE-2.0 – 02.01.2004 (09.03.2017)[1]

[3] *Boost Software License* 1.0 → http://www.boost.org/users/license.html – 17.08.2003 (09.03.2017)

[4] *Eclipse Public License* Version 1.0 → http://www.eclipse.org/org/documents/epl-v10.php – (09.03.2017)

[5] *GNU Affero General Public License* – → http://www.gnu.org/licenses/agpl 19.11.2007 (09.02.2017)

[6] *GNU General Public License* → http://www.gnu.org/licenses/old-licenses/gpl-1.0 – 02.1989 (09.03.2017)

[7] *GNU General Public License*, Version 2 → http://www.gnu.org/licenses/old-licenses/gpl-2.0 – 06.1991 (09.03.2017)

[8] *GNU Lesser General Public License*, Version 2.1 → http://www.gnu.org/licenses/old-licenses/lgpl-2.1 – 02.1999 (09.03.2017)

[9] Lizenz für *Clover* → https://www.atlassian.com/software/clover – 2017 (09.03.2017)

[10] Lizenz für *Microsoft Visual Studio Express 2015* → https://www.visualstudio.com/de/license-terms/mt171551/ – 2017 (09.03.2017)

[11] Lizenz für *MikTeX* → https://miktex.org/kb/copying – 14.01.2014 (09.03.2017)

[12] Lizenz für *SAX* → http://www.saxproject.org/copying.html – 05.05.2000 (09.03.2017)

[13] *MIT License* → https://opensource.org/licenses/MIT/ – (09.03.2017)

[14] *Oracle Binary Code License Agreement* → http://java.com/license – 02.04.2013 (09.03.2017)

[15] *OSI Certified Open Source Software* → https://opensource.org/pressreleases/certified-open-source.php – 16.06.1999 (09.03.2017)

[16] *W3C Document License* → http://www.w3.org/Consortium/Legal/2015/doc-license – 01.02.2015 (09.03.2017)

[17] *W3C Software Notice and License* → http://www.w3.org/Consortium/Legal/2002/copyright-software-20021231.html – 13.05.2015 (09.03.2017)

[18] *Hilbert II – Introduction* → http://www.qedeq.org/ – 20.01.2014 (09.03.2017)

[1] Der Pfeil (→) verweist stets auf einen Link ins Internet. Das Datum hinter dem Link – sofern vorhanden – gibt die letzte Änderung des Dokuments, der Lizenz oder der entsprechenden Seite an. Das kann vom Datum der Seite oder der Copyright-Angabe abweichen. Das geklammerte Datum gibt den Zeitpunkt an, als diese Seite im Rahmen der Erstellung dieses Dokuments zum letzten Mal angeschaut wurde. Dies gilt für alle hier aufgelisteten Literaturangaben.

[19] *Formal Correct Mathematical Knowledge*: GitHub Repository von Projekt Hilbert II → https://github.com/m-31/qedeq/ – 04.08.2016 (09.03.2017)

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

[21] Meyling, Michael: *Anfangsgründe der mathematischen Logik* – 24. Mai 2013 (in Bearbeitung) → http://www.qedeq.org/current/doc/math/qedeq_logic_v1_de.pdf – (09.03.2017)

[22] Meyling, Michael: *Formale Prädikatenlogik* – 24. Mai 2013 (in Bearbeitung) → http://www.qedeq.org/current/doc/math/qedeq_formal_logic_v1_de.pdf – (09.03.2017)

[23] Meyling, Michael: *Axiomatische Mengenlehre* – 24. Mai 2013 (in Bearbeitung) → http://www.qedeq.org/current/doc/math/qedeq_set_theory_v1_de.pdf – (09.03.2017)

[24] Meyling, Michael: *Elements of Mathematical Logic* – May 24, 2013 (in Bearbeitung) → http://www.qedeq.org/current/doc/math/qedeq_logic_v1_en.pdf – (09.03.2017)

[25] Meyling, Michael: *Formal Predicate Calculus* – May 24, 2013 (in Bearbeitung) → http://www.qedeq.org/current/doc/math/qedeq_formal_logic_v1_en.pdf – (09.03.2017)

[26] Meyling, Michael: *Axiomatic Set Theory* – May 24, 2013 (in Bearbeitung) → http://www.qedeq.org/current/doc/math/qedeq_set_theory_v1_en.pdf – (09.03.2017)

[27] Wikipedia: *Aussagenlogik* Kapitel 2.2 *Mögliche Junktoren* – 02.03.2017 → https://de.wikipedia.org/wiki/Junktor#M.C3.B6gliche_Junktoren – 20.01.2016 (09.03.2017)

[28] Wikipedia: *Aussagenlogik* Kapitel 4 *Formaler Zugang* – 24.02.2017 → https://de.wikipedia.org/wiki/Aussagenlogik#Formaler_Zugang – 13.02.2017 (09.03.2017)

[29] Wikipedia: *Prädikatenlogik erster Stufe* – 24.02.2017 → https://de.wikipedia.org/wiki/Pr%C3%A4dikatenlogik_erster_Stufe – 17.07.2016 (09.03.2017)

[30] Wikipedia: *Mengenlehre* – 24.02.2017 → https://de.wikipedia.org/wiki/Mengenlehre – 03.03.2017 (09.03.2017)

Index