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

27. April 2017

Es wird ein Programmsystem 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*[2]. Zu den meisten Fachgebieten gibt es 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. In diesem Dokument werden die Grundlagen für das zu entwickelnde Das zu entwickelnde Programmsystem, das **A**xiome, **S**ätze, **B**eweise und **A**uswertungen behandeln kann. (ASBA) behandelt.

Ein Programmsystem 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? Z. B. 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 z. B. 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? Z. B. elegant, kurz, einsichtig oder wenige Axiome. Was heißt eigentlich *elegant*?
7. *Definitionen*: Was ist mit einem Fachbegriff jeweils genau gemeint? Z. B. *Stetigkeit*, *Integral* und *Analysis*.
8. *Abhängigkeiten*: Wie heißt ein Fachbegriff in einer anderen Sprache? Ist wirklich dasselbe gemeint? Was ist mit Fachbegriffen in verschiedenen Fachgebieten?
9. *Überblick*: Ist ein Axiom, Satz, Beweis oder Fachbegriff 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 mathematische Elemente und Konstruktionen, z. B. Axiome, Sätze, Beweise und Fachgebiete. Symbole können als spezielle Fachbegriffe aufgefasst werden.

[2] Ein *Fachgebiet* ist ein Teil der Mathematik mit einer zugehörigen Basis an Axiomen, Sätzen und spezifischen Fachbegriffen und Darstellungen. Z. B. *Logik*, *Mengenlehre* und *Gruppentheorie*. Ein Fachgebiet kann sehr klein sein und im Extremfall kein einziges Element enthalten. *Umgebung* wäre eine bessere Bezeichnung, ist aber schon ein verbreiteter Fachbegriff, so dass hier die Bezeichnung Fachgebiet verwendet wird.

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.: 1.1 Fragen → 1.2 Eigenschaften

1.2. Eigenschaften

Ausgehend von den Fragen in Abschnitt 1.1 auf der vorherigen Seite soll ASBA 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 ASBA nicht umsonst erstellt wird und möglichst breite Verwendung findet, werden noch zwei Punkte angefügt:

6. *Lizenz*: Die Software ist *Open Source*.
7. *Akzeptanz*: ASBA 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 ASBA 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 ASBA kann mit den Fachbegriffen der einzelnen Fachgebiete erfolgen.
11. *Zugriff*: Der Zugriff auf ASBA kann lokal und über das Internet erfolgen.
12. *Unabhängigkeit*: ASBA kann online und offline arbeiten.
13. *Rekursion*: Es kann rekursiv über alle verwendeten Dateien – auch solchen, die im Internet liegen – ausgewertet werden.
14. *Bedienbarkeit*: ASBA ist einfach zu bedienen.
15. *Lizenz*: Die Software ist *Open Source*.

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

16. *Zwischenspeicher*: Wichtige Auswertungen können an vorhandenen Dateien angehängt oder separat in eigenen Dateien gespeichert 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 ASBA alle angegebenen Eigenschaften aufweisen, was wahrscheinlich ebenfalls illusorisch ist.

[3] Es sind eigentlich Anforderungen. Da dieser Begriff auch im Kapitel 3 auf Seite 19 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.: 1.2 Eigenschaften → 1.3 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.: 1.1 Fragen → 1.3 Ziele

2. Mathematische Grundlagen

2.1. Metasprache

Wenn man über eine Sprache spricht, braucht man auch eine Sprache, in der Aussagen über die erstere getroffen werden können. Wenn die zuerst genannte Sprache die der Mathematik ist, nimmt man üblicherweise die natürliche Sprache als Metasprache. Leider ist diese oft ungenau, nicht immer eindeutig und abhängig vom Zusammenhang, in dem sie gesprochen wird[1]. Um diese Probleme in den Griff zu bekommen, wird die Metasprache teilweise formalisiert. Durch diese Formalisierung erinnert sie dann teilweise schon an mathematische Formeln. Die Sprachebenen müssen aber sorgfältig unterschieden werden.

2.1.1. Metasprachliche Ausdrücke

Ein *metasprachlicher Ausdruck* ist eine in normaler Sprache verfasste Aussage, wie z. B. (a) „Morgen scheint die Sonne.“, (b) „Ich bin 1,83 m groß.“, (c) „Ich habe ein rotes Auto und das kann 200 km/h schnell fahren.“, etc. In einem erweiterten Sinne gehören auch Relationen einschließlich ihrer Operanden dazu[2], wie z. B. „$A = A$“, „$A \equiv B$“, „$A < B$“, etc.

Während die Beispiele (a) und (b) einfache, nicht mehr zerlegbare metasprachliche Ausdrücke sind, ist Beispiel (c) zusammengesetzt. Für alle drei Aussagen lässt sich feststellen, ob sie richtig sind oder nicht. Das kann man für den zweiten Teil von (c) allein aber nicht, wenn man nicht weiss worauf sich „das“ bezieht. Natürlich muss auch der Zusammenhang, in dem ein metasprachlicher Ausdruck formuliert wird, bekannt sein, denn z. B. ist die Bedeutung von „Ich“ nur dann bekannt, wenn man weiß von wem die Aussage ist. Auf eine exakte Definition von „metasprachlicher Ausdruck“ wird verzichtet, weil das intuitive Verständnis hier ausreicht. In erster Näherung können aber alle sprachlichen Ausdrücke, die im Prinzip überprüft werden können, als metasprachliche Ausdrücke betrachtet werden.

Zusammengesetzte metasprachliche Ausdrücke wie (c) können zum Teil formalisiert werden. Dies wird mit den folgenden Definitionen erreicht:

Sind A und B metasprachliche Ausdrücke, dann wird definiert:

- „$A \Rightarrow B$“ steht für „*Wenn* A [gilt], *dann* [gilt] [auch] B“.
- „$A \Leftarrow B$“ steht für „*Wenn* B [gilt], *dann* [gilt] [auch] A“.
- „$A \Leftrightarrow B$“ steht für „A [gilt] *genau dann wenn* B [gilt]“.
- „A && B“ steht für „[Es gilt] A *und* B“.
- „A || B“ steht für „[Es gilt] A *oder* B“.

[1] Noch problematischer ist es, dass man unauflösbare Widersprüche formulieren kann, z. B. „Der Barbier ist der Mann im Ort, der genau die Männer im Ort rasiert, die sich nicht selbst rasieren.“. Und der Barbier? Wenn er sich selbst rasiert, dann rasiert er sich nicht selbst, und wenn er sich nicht selbst rasiert, dann rasiert er sich selbst. Was denn nun? Das Problem ist verwandt mit dem Problem der „Menge aller Mengen, die sich nicht selbst enthalten“.

[2] Wird statt des Symbols der Name der zugehörigen Relation verwendet, ist dies unmittelbar einleuchtend. So wird z. B. aus der Formel „$A < B$“ die metasprachliche Aussage „A ist kleiner als B“.

Offensichtlich sind das alles ebenfalls metasprachliche Ausdrücke, jetzt aber zumindest teilweise formalisiert. (c) lässt sich dann ausdrücken als „‚Ich habe ein rotes Auto' && ‚das kann 200 km/h schnell fahren.'".

Um Verwechslungen mit den logischen Symbolen zu vermeiden, werden für „und" und „oder" die Symbole ‚&&' und ‚||' verwendet. Ein Symbol für „nicht" wird hier nicht gebraucht.

Metasprachliche Ausdrücke können auch geklammert werden, um die Reihenfolge der Auswertung eindeutig zu machen. ‚⇒', ‚⇐', ‚⇔', ‚&&' und ‚||' heißen *metasprachliche Operatoren*. Ihre Prioritäten werden im Unterabschnitt 2.3.2 auf Seite 12 zusammen mit anderen Operatoren definiert.

Sollen zwei metasprachliche Ausdrücke miteinander verglichen werden, muss klar sein auf welche Art; ob z. B. als Zeichenfolgen – mit oder ohne Wertung der Zwischenräume –, als Wahrheitswerte oder auf sonstige Art. Wenn die Art des Vergleichs implizit oder explizit klar ist und sich die beiden Ausdrücke damit vergleichen lassen, heißen sie *vergleichbar*.

2.1.2. Mit Gleichheit verwandte Symbole

In diesem Abschnitt wird vorausgesetzt:

- ‚=', ‚≠', ‚≡' und ‚:=' werden im selben Zusammenhang verwendet.
- ‚A', ‚B', ‚P' und ‚Q' sind vergleichbar, d. h. Ausdrücke derselben Art.

Dann werden folgende Symbole definiert:

- $=$ (*Gleichheit*): „$A = B$" heißt, dass ‚A' und ‚B' sich in den interessierenden Eigenschaften nicht unterscheiden. Welche das sind, ergibt sich normalerweise aus dem Zusammenhang[3] oder muss explizit angegeben werden. Z. B. sind zwei Operatoren gleich, wenn sie stets denselben Wahrheitswert liefern.

  Inwieweit die Begriffe *Gleichheit* und *Identität* korrelieren, wird hier nicht erörtert. siehe [29]

- $\neq$ (*Ungleichheit*): „$A \neq B$" heißt, dass ‚A' und ‚B' sich in mindestens einer der interessierenden Eigenschaften unterscheiden.

- $\equiv$ (*Äquivalenz*): „$A \equiv B$" heißt, dass ‚A' und ‚B' sich in den interessierenden Eigenschaften nicht unterscheiden. Welche das sind, ergibt sich wie bei ‚=' aus dem Zusammenhang oder wird explizit angegeben. Werden ‚=' und ‚≡' im selben Zusammenhang verwendet, muss mit „$A = B$" stets auch „$A \equiv B$" gelten, d. h. alle interessierenden Eigenschaften von ‚≡' müssen auch interessierenden Eigenschaften von ‚$A = B$' sein

- $:=$ (*Definition*): „$A := B$" bedeutet, dass ‚A' durch ‚B' definiert wird. Gewissermaßen ist ‚A' nur eine andere Schreibweise für ‚B'. ‚A' und ‚B' können sich gegenseitig ersetzten.

  Nach dieser Definition sind ‚P' und ‚Q' schon dann gleich, wenn nach der Ersetzung aller Vorkommen von ‚A' in ‚P' und ‚A' in ‚Q' jeweils durch ‚B' die resultierenden Ausdrücke ‚$\overline{P}$' und ‚$\overline{Q}$' gleich sind.

Unter den Voraussetzungen:

- $A := B$
- ‚$\overline{P}$' ergibt sich aus ‚P' durch Ersetzung aller Vorkommen (Teilausdrücke) ‚A' durch ‚B'.
- ‚$\overline{Q}$' ergibt sich aus ‚Q' durch Ersetzung aller Vorkommen (Teilausdrücke) ‚A' durch ‚B'.

gilt:

[3] Statt von einem *Zusammenhang* könnte man auch von einer *Umgebung* sprechen. Diese Bezeichnung ist aber auch ein verbreiteter Fachbegriff. Die Exaktheit der Begriffe in diesem Dokument soll für Erstellung von ASBA ausreichen; was darüber hinausgeht, ist nicht Inhalt dieses Dokuments.

- Die interessierenden Eigenschaften für ‚≡' sind auch solche für ‚='.
- Für ‚=' können – müssen aber nicht – mehr Eigenschaften von Interesse sein als für ‚≡'.
- $(A = B) \mid\mid (A \neq B)$
- $(A = B) \Rightarrow (A \equiv B)$
- $(\overline{P} = \overline{Q}) \Leftrightarrow (P = Q)$

2.2. Formale Elemente

Ein *formales Element* kann z. B. eine Menge, Zeichenfolge, Zahl, Formel, etc. sein. Zwei formale Elemente A und B sind *vergleichbar*, wenn beide von derselben Art sind, d. h. wenn z. B. jeweils beide Mengen, Zeichenfolgen, Zahlen oder Formeln – die vergleichbare Ergebnisse liefern – sind.

Intuitiv scheint klar zu sein, was damit gemeint ist. Wenn aber entschieden werden muss, ob z. B. (a) „1+1" gleich „2" oder (b) „1+1" gleich „1 + 1" ist, muss man erst entscheiden, von welcher Art die beiden zu vergleichenden Ausdrücke sind, d. h. *wie* verglichen wird. Wenn sie als jeweiliges Ergebnis der beiden Formeln verglichen werden, dann ist (a) richtig. Wenn sie als Formeln, d. h. als Zeichenfolgen, verglichen werden ist (a) falsch. Wenn die Ausdrücke in (b) als Zeichenfolgen verglichen werden, ist (b) dann richtig, wenn der Zwischenraum zwischen den einzelnen Zeichen nicht zählt. Wenn er aber zählt, ist (b) falsch.

Im Zusammenhang mit binären Relationen werden noch einige Verabredungen getroffen. Dazu seien ‚$\sim$', ‚$\simeq$', ‚$\lhd$', ‚$\rhd$', ‚$\trianglelefteq$' und ‚$\trianglerighteq$' Beispielsymbole für Relationen und ‚=' und ‚$\neq$' die in Abschnitt 2.1.2 auf der vorherigen Seite definierten Symbole für Gleichheit und Ungleichheit. Wenn dann nichts anderes gesagt wird, gelte stets:

$$((A \sim B) \mid\mid (A = B)) \quad \Leftrightarrow \quad (A \simeq B) \tag{2.1}$$
$$(A \lhd B) \quad \Leftrightarrow \quad (B \rhd A) \tag{2.2}$$
$$(A \trianglelefteq B) \quad \Leftrightarrow \quad (B \trianglerighteq A) \tag{2.3}$$

Mit der Definition einer Relation der einen Seite ist damit automatisch auch die der anderen Seite erfolgt, mit der Ausnahme, dass man „$A \sim B$" so nicht mit Hilfe von „$A \simeq B$" definieren kann. Dies könnte man zwar mit Hilfe des Ansatzes

$$(A \sim B) \quad \Leftrightarrow \quad (A \simeq B) \;\&\&\; (A \neq B) \tag{2.4}$$

versuchen, aber die so definierte Relation ‚$\sim$' kann, muss aber nicht mit der in 2.1 übereinstimmen. Allerdings lässt sich 2.1 aus 2.4 ableiten und wenn „$(A = B) \Rightarrow (A \simeq B)$" gilt, auch 2.4 aus 2.1. – Auf einen Beweis verzichten wir hier.

Es sei noch angemerkt, dass wegen 2.4 die Definition von ‚$\Leftarrow$' in Abschnitt 2.1.1 auf Seite 9 überflüssig ist und wegen der Klammerregeln (sieheUnterabschnitt 2.3.2 auf der nächsten Seite) auch alle Klammern in diesem Abschnitt 2.2. Die Prioritäten der Operatoren ‚$\lhd$', ‚$\rhd$', ‚$\trianglelefteq$' und ‚$\trianglerighteq$' unterscheiden sich normalerweise nicht; ebensowenig die der Operatoren ‚$\sim$' und ‚$\simeq$', die aber durchaus verschieden von den Prioritäten von ‚=' und ‚$\neq$' sein können.

Als Beispielsymbol für binäre Operatoren wird ‚$*$' verwendet. Damit zusammenhängende Verabredungen werden hier nicht getroffen.

2.3. Aussagenlogik

2.3.1. Konstante und Operatoren

Die Tabelle 2.1 auf der nächsten Seite [4] 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' [5], 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.

2.3.2. Klammerregeln

Zur Klammerersparnis werden die üblichen Regeln verwendet, d. h. dass Operatoren mit höherer Priorität stärker binden, als solche mit niedrigerer Priorität.

Für die Operatoren derselben Priorität gilt Rechtsklammerung[6]. Im Folgenden wird nur noch ein Teil der logischen Operatoren aus der Tabelle 2.1 auf der nächsten Seite und der metasprachlichen Operatoren von Unterabschnitt 2.1.1 auf Seite 9 berücksichtigt. Diese werden nun mit abnehmender Priorität aufgelistet:

Klammern; auch andere	$(\dots)$
Unäre logische Operatoren	$\neg$
Binäre logische Operatoren	$\wedge \quad \cdot \quad \uparrow$ $\vee \quad + \quad \downarrow$ $\leftarrow \quad \rightarrow$ $\leftrightarrow$
Mit Gleichheit verwandte Symbole; ihre Prioritäten untereinander sind nicht eindeutig und bleiben daher undefiniert.	$= \quad \neq \quad \equiv \quad :=$
Metasprachliche Operatoren	&& || $\Leftarrow \quad \Rightarrow$ $\Leftrightarrow$

[4] Die Tabelle basiert auf den Wahrheitstafeln in [27] Kapitel 2.2 und [1] Kapitel 1.1 Seite 3.
[5] Im Gegensatz zu Paragraph 2.3.3.1 auf Seite 14 können A und B hier beliebige Aussagen – auch Formeln – sein.
[6] Unäre Operatoren stehen hier stets links *vor* dem Operanden, so dass es nur Rechtsklammerung geben kann. Zur Rechtsklammerung bei binären Operationen ein Zitat aus [1] Kapitel 1.1 Seite 5: „Diese hat gegenüber Linksklammerung Vorteile bei der Niederschrift von Tautologien in →, [...]"

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
$\wedge$ & $\cdot$	-	-	-	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
$+ \veebar \varolessthan \oplus$	-	-	-	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	-
$\nrightarrow \not\Rightarrow \nsupset$	-	-	-	F	W	F	F	Postsektion		-
$\rceil$	-	-	-	F	F	W	W	Pränonpendenz	Negation von A	-
$\nleftarrow \not\Leftarrow \not\subset$	-	-	-	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*. Die Symbole ‚$\subset$', ‚$\supset$', ‚$\nsubset$' und ‚$\nsupset$' haben hier nicht die Bedeutung der entsprechenden Mengensymbole und dürfen nicht damit verwechselt werden; entsprechendes 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. Nur wenn es sowohl links-, als auch rechtsseitige unäre Operatoren gibt, muss man für diese Prioritäten definieren.

Tabelle 2.1.: Definition von aussagenlogischen Symbolen.

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

2.3.3. Formalisierung

Da sie die Grundlage – quasi das Fundament – des mathematischen Inhalts von ASBA sind, müssen die Axiome, Sätze, Beweise, etc. der Aussagenlogik in streng formaler Form vorliegen. Die Formalisierung stützt sich auf [28]; siehe auch [21, 24]. Da Computerprogramme mit der *Polnischen Notation*[7] besser umgehen können und Klammern dort überflüssig sind, werden viele Formeln auch in die Polnische Notation überführt.

2.3.3.1. Bausteine der aussagenlogischen Sprache

> > > Hier weitermachen. < < <

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

$$\mathbb{N}_0 := \quad \text{Menge der } \textit{natürlichen Zahlen} \text{ einschließlich } 0. \tag{2.5}$$
$$\mathcal{C} := \{\bot, \top\} \quad \text{Menge der } \textit{Konstanten}. \tag{2.6}$$
$$\mathcal{U} := \{\neg\} \quad \text{Menge der } \textit{unären Operatoren}. \tag{2.7}$$
$$\mathcal{B} := \{\wedge, \vee, \rightarrow, \leftrightarrow\} \quad \text{Menge der } \textit{binären Operatoren}. \tag{2.8}$$
$$\mathcal{B}_\mathrm{e} := \mathcal{B} \cup \{\cdot, +, \uparrow, \downarrow, \leftarrow\} \quad \textit{Erweiterte} \text{ Menge der binären Operatoren}. \tag{2.9}$$

Damit sind alle in der Tabelle 2.1 auf der vorherigen Seite verwendeten wesentlichen Konstanten und Operatoren[8] erfasst und es können die folgende Mengen definiert werden:

$$\mathcal{V} := \{P_n | n \in \mathbb{N}_0\} \quad \text{Menge der } \textit{atomaren Formeln} \tag{2.10}$$
$$\mathcal{J} := \mathcal{U} \cup \mathcal{B} \quad \text{Menge der } \textit{Junktoren}, \text{ bzw. Operatoren}. \tag{2.11}$$
$$\mathcal{S} := \mathcal{U} \cup \mathcal{B}_\mathrm{e} \cup \mathcal{C} = \{\neg, \wedge, \vee, \rightarrow, \leftrightarrow, \leftarrow, \uparrow, \downarrow, \cdot, +, \top, \bot\} \quad \text{Menge der } \textit{Symbole}. \tag{2.12}$$
$$\mathcal{A} := \mathcal{V} \cup \mathcal{J} \quad \textit{Alphabet der logischen Sprache}. \tag{2.13}$$
$$\mathcal{A}_\mathrm{e} := \mathcal{V} \cup \mathcal{S} \quad \textit{Erweitertes} \text{ Alphabet der logischen Sprache}. \tag{2.14}$$

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

Offensichtlich wird für alle endlichen Mengen von Formeln (siehe 2.3.3.2 auf der nächsten Seite) – und nur solche Mengen betrachten wir – 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.

[7] 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.

[8] Jeweils nur die ersten der grau hinterlegten Zeilen sowie ‚·'.

2.3.3.2. Aussagenlogische Formeln

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*:

$$\mathcal{V} \subset \mathcal{F} \tag{2.15}$$
$$A \in \mathcal{F} \quad \text{dann auch} \quad (\circ A) \in \mathcal{F} \quad \text{für} \quad \circ \in \mathcal{U} \tag{2.16}$$
$$A, B \in \mathcal{F} \quad \text{dann auch} \quad (A \circ B) \in \mathcal{F} \quad \text{für} \quad \circ \in \mathcal{B} \tag{2.17}$$

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:

$$\mathcal{V}, \mathcal{C} \subset \mathcal{F}_\mathrm{e} \tag{2.18}$$
$$A \in \mathcal{F}_\mathrm{e} \quad \text{dann auch} \quad (\circ A) \in \mathcal{F}_\mathrm{e} \quad \text{für} \quad \circ \in \mathcal{U} \tag{2.19}$$
$$A, B \in \mathcal{F}_\mathrm{e} \quad \text{dann auch} \quad (A \circ B) \in \mathcal{F}_\mathrm{e} \quad \text{für} \quad \circ \in \mathcal{B}_\mathrm{e} \tag{2.20}$$

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*:

$$\mathcal{V} \subset \mathcal{F}^\mathrm{P} \tag{2.21}$$
$$A \in \mathcal{F}^\mathrm{P} \quad \text{dann auch} \quad (\circ A) \in \mathcal{F}^\mathrm{P} \quad \text{für} \quad \circ \in \mathcal{U} \tag{2.22}$$
$$A, B \in \mathcal{F}^\mathrm{P} \quad \text{dann auch} \quad \circ AB \in \mathcal{F}^\mathrm{P} \quad \text{für} \quad \circ \in \mathcal{B} \tag{2.23}$$

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:

$$\mathcal{V}, \mathcal{C} \subset \mathcal{F}_\mathrm{e}^\mathrm{P} \tag{2.24}$$
$$A \in \mathcal{F}_\mathrm{e}^\mathrm{P} \quad \text{dann auch} \quad (\circ A) \in \mathcal{F}_\mathrm{e}^\mathrm{P} \quad \text{für} \quad \circ \in \mathcal{U} \tag{2.25}$$
$$A, B \in \mathcal{F}_\mathrm{e}^\mathrm{P} \quad \text{dann auch} \quad \circ AB \in \mathcal{F}_\mathrm{e}^\mathrm{P} \quad \text{für} \quad \circ \in \mathcal{B}_\mathrm{e} \tag{2.26}$$

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

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 2.3.3.1 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 frei von Klammern. Die Namen der Operationen finden sich in der Tabelle 2.1 auf Seite 13. Für aussagenlogische Formeln, d. h. von Elementen aus $\mathcal{X} \in \{\mathcal{F}, \mathcal{F}_\mathrm{e}, \mathcal{F}^\mathrm{P}, \mathcal{F}_\mathrm{e}^\mathrm{P}\}$, werden hier normalerweise die kleinen griechischen Buchstaben $\alpha, \beta, \gamma, \ldots$ verwendet. Das können auch atomare Formeln sein (siehe(2.10)).

2.3.4. Logische Axiome

Es werden noch weitere Mengen von Operatoren eingeführt, die jeweils ausreichen, alle anderen Operatoren und die beiden Konstanten zu definieren:

$$\mathcal{O}_{\text{bool}} := \{\neg, \wedge, \vee\} \tag{2.27}$$
$$\mathcal{O}_{\text{and}} := \{\neg, \wedge\} \tag{2.28}$$
$$\mathcal{O}_{\text{or}} := \{\neg, \vee\} \tag{2.29}$$
$$\mathcal{O}_{\text{imp}} := \{\neg, \rightarrow\} \tag{2.30}$$
$$\mathcal{O}_{\text{rep}} := \{\neg, \leftarrow\} \tag{2.31}$$
$$\mathcal{O}_{\text{nand}} := \{\uparrow\} \tag{2.32}$$
$$\mathcal{O}_{\text{nor}} := \{\downarrow\} \tag{2.33}$$

Ausgehend von ‚$\neg$' und ‚$\wedge$' werden die weiteren Operatoren und Konstanten aus $\mathcal{S}$ definiert:

$$(A \rightarrow B) := (\neg(A \wedge (\neg B))) \qquad \rightarrow AB := \neg \wedge A \neg B \tag{2.34}$$
$$(A \leftarrow B) := (B \rightarrow A) \qquad \leftarrow AB := \rightarrow BA \tag{2.35}$$
$$(A \leftrightarrow B) := ((A \rightarrow B) \wedge (A \leftarrow B)) \qquad \leftrightarrow AB := \wedge \rightarrow AB \leftarrow AB \tag{2.36}$$
$$(A \vee B) := (\neg((\neg A) \wedge (\neg B))) \qquad \vee AB := \neg \wedge \neg A \neg B \tag{2.37}$$
$$(A \uparrow B) := (\neg(A \wedge B)) \qquad \uparrow AB := \neg \wedge AB \tag{2.38}$$
$$(A \downarrow B) := (\neg(A \vee B)) \qquad \downarrow AB := \neg \vee AB \tag{2.39}$$
$$(A \cdot B) := (A \wedge B) \qquad \cdot AB := \wedge AB \tag{2.40}$$
$$(A + B) := ((A \vee B) \wedge (\neg(A \wedge B))) \qquad + AB := \wedge \vee AB \neg \wedge AB \tag{2.41}$$
$$\bot := (P_0 \wedge (\neg P_0)) \qquad \bot := \wedge P_0 \neg P_0 \tag{2.42}$$
$$\top := (P_0 \vee (\neg P_0)) \qquad \top := \vee P_0 \neg P_0 \tag{2.43}$$

Aus 2.35 folgt durch Vertauschung der Variablen unmittelbar

$$(A \rightarrow B) \equiv (B \leftarrow A) \qquad \rightarrow AB \equiv \leftarrow BA \tag{2.44}$$

Analog zu 2.34 und 2.37 sollte noch

$$(A \leftarrow B) \equiv (\neg((\neg A) \wedge B)) \qquad \leftarrow AB \equiv \neg \wedge \neg AB \tag{2.45}$$
$$(A \wedge B) \equiv (\neg((\neg A) \vee (\neg B))) \qquad \wedge AB \equiv \neg \vee \neg A \neg B \tag{2.46}$$

gelten, was aber erst noch zu beweisen ist.

Es werden nun die logischen Axiome – ohne Quantoren – mit Hilfe der Operatoren aus $\mathcal{O}_{\text{and}}$ definiert und zusätzlich alle Elemente von $\mathcal{S}$ (siehe 2.12 auf Seite 14), d. h. alle logischen Operatoren und Konstanten aus $\mathcal{S}$.

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

> > > Junktoren definieren und Übereinstimmung ableiten. < < <

2.3.4.1. Logisches Axiomensystem

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

Axiome:

$$(\alpha \to \beta \to \gamma) \to (\alpha \to \beta) \to (\alpha \to \gamma) \qquad \to\to \alpha \to \beta\gamma \to\to \alpha\beta \to \alpha\gamma \tag{2.47}$$
$$\alpha \to \beta \to \alpha \wedge \beta \qquad \to \alpha \to \beta \wedge \alpha\beta \tag{2.48}$$
$$\alpha \wedge \beta \to \alpha; \quad \alpha \wedge \beta \to \beta \qquad \to \wedge\alpha\beta\alpha; \quad \to \wedge\alpha\beta\beta \tag{2.49}$$
$$(\alpha \to \neg\beta) \to (\beta \to \neg\alpha) \qquad \to\to \alpha\neg\beta \to \beta\neg\alpha \tag{2.50}$$

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

$$(\alpha \vee \beta) \coloneqq \neg(\neg\alpha \to \beta) \qquad \vee\alpha\beta \coloneqq \neg \to \neg\alpha\beta \tag{2.51}$$
$$(\alpha \leftrightarrow \beta) \coloneqq ((\alpha \to \beta) \wedge (\beta \to \alpha)) \qquad (\alpha \leftrightarrow \beta) \coloneqq ((\alpha \to \beta) \wedge (\beta \to \alpha)) \tag{2.52}$$
$$(\alpha \cdot \beta) \coloneqq (\alpha \wedge \beta) \qquad (\alpha \cdot \beta) \coloneqq (\alpha \wedge \beta) \tag{2.53}$$
$$(\alpha + \beta) \coloneqq ((\alpha \vee \beta) \wedge \neg(\alpha \wedge \beta)) \qquad (\alpha + \beta) \coloneqq ((\alpha \vee \beta) \wedge \neg(\alpha \wedge \beta)) \tag{2.54}$$
$$(\alpha \uparrow \beta) \coloneqq \neg(\alpha \wedge \beta) \qquad (\alpha \uparrow \beta) \coloneqq \neg(\alpha \wedge \beta) \tag{2.55}$$
$$(\alpha \downarrow \beta) \coloneqq \neg(\alpha \vee \beta) \qquad (\alpha \downarrow \beta) \coloneqq \neg(\alpha \vee \beta) \tag{2.56}$$
$$(\alpha \leftarrow \beta) \coloneqq (\beta \to \alpha) \qquad (\alpha \leftarrow \beta) \coloneqq (\beta \to \alpha) \tag{2.57}$$
$$\bot \coloneqq (p_0 \wedge \neg p_0) \qquad \bot \coloneqq (p_0 \wedge \neg p_0) \tag{2.58}$$
$$\top \coloneqq \neg\bot \qquad \top \coloneqq \neg\bot \tag{2.59}$$

Zu zeigen

$$(\alpha \to \beta) \equiv \neg(\alpha \wedge \neg\beta) \qquad \to \alpha\beta \equiv \neg \wedge \alpha\neg\beta \tag{2.60}$$

2.3.5. Definition von Junktoren durch andere

2.3.5.1. nicht, und, oder

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

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

(2.61)

Zu zeigen:

$$(\alpha \vee \beta) \coloneqq \neg(\neg\alpha \wedge \neg\beta) \tag{2.62}$$

2.3.5.2. nicht, und

Gegebene Operatoren: $\neg, \wedge$

Definierter Operator: $\vee$

$$(\alpha \vee \beta) \coloneqq \neg(\neg\alpha \wedge \neg\beta) \tag{2.63}$$

Zu zeigen:

$$(\alpha \vee \beta) \equiv \neg(\neg\alpha \wedge \neg\beta) \tag{2.64}$$

Zur Definition der Operatoren $\to, \leftrightarrow, \cdot, +, \uparrow, \downarrow, \leftarrow, \bot, \top$ siehe Paragraph 2.3.5.1

2.3.5.3. nicht, oder

Gegebene Operatoren: $\neg, \vee$

Definierter Operator: $\wedge$

$$(\alpha \wedge \beta) \coloneqq \neg(\neg\alpha \vee \neg\beta)) \tag{2.65}$$

2.3.5.4. nicht, impliziert

Gegebene Operatoren: $\neg, \rightarrow$

Definierte Operatoren: $\wedge, \vee$

$$(\alpha \wedge \beta) \coloneqq \ldots \tag{2.66}$$
$$(\alpha \vee \beta) \coloneqq \ldots \tag{2.67}$$

2.3.5.5. NAND

Gegebener Operator: $\uparrow$

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

$$\neg\alpha \coloneqq \ldots \tag{2.68}$$
$$(\alpha \wedge \beta) \coloneqq \ldots \tag{2.69}$$
$$(\alpha \vee \beta) \coloneqq \ldots \tag{2.70}$$

2.3.5.6. NOR

Gegebener Operator: $\downarrow$

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

$$\neg\alpha \coloneqq \ldots \tag{2.71}$$
$$(\alpha \wedge \beta) \coloneqq \ldots \tag{2.72}$$
$$(\alpha \vee \beta) \coloneqq \ldots \tag{2.73}$$

2.3.6. Aussagenlogische Axiome

> > > Aussagenlogik weiter bearbeiten. < < <

2.4. Prädikatenlogik

> > > Prädikatenlogik bearbeiten. < < <

2.5. Mengenlehre

> > > Mengenlehre bearbeiten. < < <

3. 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.

3.1. 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 von ASBA von sekundärer Bedeutung und werden daher in diesen Abschnitt nicht übernommen. Die anderen Ziele werden noch verfeinert.

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)
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)

[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 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 ASBA kann mit den Fachbegriffen der einzelnen Fachgebiete erfolgen. (siehe Ziel 10 auf Seite 6)
10. *Zugriff*: Der Zugriff auf ASBA kann lokal und über das Internet erfolgen. (siehe Ziel 11 auf Seite 6)
11. *Unabhängigkeit*: ASBA 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*: ASBA ist einfach zu bedienen. (siehe Ziel 14 auf Seite 6)

3.2. Axiome

> > > Axiome auswählen und definieren. < < <

3.3. Beweise

> > > Schlussregeln auswählen und Beweise definieren. < < <

3.4. Datenstruktur

> > > Datenstruktur abstrakt und in XML definieren. < < <

3.5. Bausteine

> > > Bausteine? definieren. < < <

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, 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 und 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]

A.2. Offene Aufgaben

1. TODOs bearbeiten
2. Eingabeprogramm erstellen (liest XML)
3. Prüfprogramm erstellen
4. Ausgabeprogramm erstellen (schreibt XML)
5. Formelausgabe erstellen (erzeugt LaTeX aus XML)
6. Axiome sammeln und eingeben
7. Sätze sammeln und eingeben
8. Beweise sammeln und eingeben
9. Fachbegriffe und Symbole sammeln und eingeben
10. Fachgebiete sammeln und eingeben
11. Ausgabeschemata sammeln und eingeben

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

B. Ideen

B.1. Test der Referenzen

> > > Falsche Referenzen < < <
cha:Inhaltsverzeichnis: auf Seite 2 Fehler: 1 Cha zu früh (Dokumentanfang)
dic:Tabellenverzeichnis: B.1 auf der nächsten Seite Fehler: 1 Cha zu früh (Ideen)
dic:Abbildungsverzeichnis: B.1 auf Seite 26 Fehler: 1 Cha zu früh (Tabellenverzeichnis)

Tabellenverzeichnis

Abbildungsverzeichnis

Literaturverzeichnis

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

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

[3] *Boost Software License* 1.0 – 17.08.2003 → http://www.boost.org/users/license.html (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* – 19.11.2007 → http://www.gnu.org/licenses/agpl (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 – 06.1991 → http://www.gnu.org/licenses/old-licenses/gpl-2.0 (09.03.2017)

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

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

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

[11] Lizenz für *MikTeX* – 14.01.2014 → https://miktex.org/kb/copying (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* – 02.04.2013 → http://java.com/license (09.03.2017)

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

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

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

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

[1] Der Pfeil (→) verweist stets auf einen Link zu einer Seite im Internet. Das geklammerte Datum hinter dem Link gibt den Zeitpunkt an, zu dem die Seite im Rahmen der Erstellung dieses Dokuments zum letzten Mal angeschaut wurde. Das nicht geklammerte Datum gibt, je nachdem welches Datum in der Seite angegeben ist, den Stand der Seite bzw. den Zeitpunkt der letzten Änderung an. – Dies gilt für alle hier aufgelisteten Seiten im Internet.

[19] *Formal Correct Mathematical Knowledge*: GitHub Repository vom Projekt Hilbert II – 04.08.2016 → https://github.com/m-31/qedeq/ (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 vom Projekt ASBA – in Bearbeitung → https://github.com/Dr-Winfried/ASBA

[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* – 20.01.2016 → https://de.wikipedia.org/wiki/Junktor#M.C3.B6gliche_Junktoren (09.03.2017)

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

[29] Wikipedia: *Identität (Logik)* Kapitel 2.3 *Identität in der Informatik* – 24.02.2017 → https://de.wikipedia.org/wiki/Identit%C3%A4t_(Logik)#Identit.C3.A4t_in_der_Informatik – 03.03.2017 (17.04.2017)

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

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

Index

Symbolverzeichnis

Glossar

ASBA Das zu entwickelnde Programmsystem, das **A**xiome, **S**ätze, **B**eweise und **A**uswertungen behandeln kann.. 4–6, 10, 14, 19, 20

Fachbegriff Ein Name für einen mathematischen Begriff. 4–6, 19, 20, 22

Fachgebiet Ein Teil der Mathematik mit einer zugehörigen Basis von Axiomen, Sätzen und spezifischen Fachbegriffen und Darstellungen. 4–6, 19, 20, 22

formales Element Ein mathematisches Element in formaler Schreibweise. Bis auf wenige Aussagen kommen darin keine metasprachliche Ausdrücke mehr vor. 11, 32

interessierende Eigenschaften sind Eigenschaften, die bzgl. ‚=' bzw. ‚≡' von Interesse sind. Z. B. ist die interessierende Eigenschaft der Operanden der Gleichung „$y = f(x)$" normalerweise der Wert (von „y" und „$f(x)$"), und nicht deren Darstellung (Zeichenkette). 10

metasprachlicher Ausdruck Eine in normaler Sprache verfasste Aussage, die auch zusammengesetzt sein kann. 9, 10, 32

metasprachlicher Operator Operatoren, deren Operanden metasprachliche Ausdrücke sind. 10, 12

vergleichbar Zwei metasprachliche Ausdrücke bzw. formale Elemente heißen – auf eine bestimmte Art – vergleichbar, wenn sie auf diese Art verglichen werden können Die Art muss implizit bekannt oder explizit angegeben sein. Meistens genügt es zu wissen, was für metasprachliche Ausdrücke bzw. formale Elemente es sind. Beide müssen dann von derselben Art sein, z. B. Zeichenketten oder vergleichbare Ergebnisse von Formeln.. 10, 11