KAPITEL 5

Logik

"Wenn man Charaktere oder Zeichen finden könnte, die geeignet wären, alle unsere Gedanken ebenso rein und streng auszudrücken, wie die Arithmetik die Zahlen oder die analytische Geometrie die Linien ausdrückt, könnte man offenbar bei allen Gegenständen, soweit sie dem vernünftigen Denken unterworfen sind, das tun, was man in der Arithmetik und der Geometrie tut. Denn alle Forschungen, die vom vernünftigen Denken abhängen, würden durch die Umwandlung dieser Charaktere und eine Art Kalkül zustande kommen, was die Erfindung schöner Dinge ganz leicht machen würde. Denn es würde nicht nötig sein, sich den Kopf ebenso zu zerbrechen, wie man heute gezwungen ist zu tun, und man würde trotzdem sicher sein, alles was hier zu tun sein würde, tun zu können ex datis. Zudem würde man jeden von dem überzeugen, was man gefunden oder erschlossen hätte, da es leicht sein würde, den Kalkül zu prüfen, sei es, indem man ihn nachvollzieht, sei es indem man einige Proben versucht, ähnlich solchen, wie es die Neunerprobe in der Arithmetik ist. Und wenn jemand an dem, was ich vorgebracht haben würde, zweifelte, würde ich zu ihm sagen: "Rechnen wir, mein Herr!", und Feder und Tinte nehmend, würden wir uns bald aus der Verlegenheit ziehen".

Gottfried Wilhelm Leibniz (1646 - 1716)

5.1. Aussagenlogik

In der Aussagenlogik beschränkt man sich auf Aussagen, die entweder wahr oder falsch sind. Dies ist eine wirkliche Beschränkung, wie schon das Beispiel 1.1.3:

Alle Kreter lügen.

gezeigt hat. Derartige Aussagen können dann verknüpft werden um *Formeln* (der Aussagenlogik) zu erstellen. Dies wird in der folgenden Definition präzesiert:

DEFINITION 5.1.1. Formeln

- Jede Aussage ist eine Formel.
- Sind A und B Formeln, dann sind auch folgende Verknüpfungen Formeln:

$$\begin{gathered}(A \wedge B)\\(A \vee B)\\\neg A\\(A \Rightarrow B)\\(A \Leftrightarrow B)\end{gathered}$$

- Alle Formeln können durch wiederholte Anwendung obiger Methode erstellt werden.

Da Aussagen nur wahr oder falsch sind, wird dies auch für Formeln festgelegt, dies geschieht mit der folgenden *Wahrheitstafel*:

A	B	$(A \wedge B)$	$(A \vee B)$	$\neg A$	$(A \Rightarrow B)$	$(A \Leftrightarrow B)$
		und	oder	nicht	folgt	gdw
t	t	t	t	f	t	t
t	f	f	t	f	f	f
f	t	f	t	t	t	f
f	f	f	f	t	t	t

Hat man eine Formel und setzt dann für die einzelnen beteiligten Aussagen wahr und falsch Werte ein, so nennt man dies eine *Interpretation* der Formel. Verschiedene Formeln können für alle möglichen Interpretationen die gleichen Wahrheitswerte haben:

A	B	$(A \Leftrightarrow B)$	$((A \Rightarrow B) \wedge (B \Rightarrow A))$	$(A \Rightarrow B)$	$\neg A \vee B$
t	t	t	t	t	t
t	f	f	f	f	f
f	t	f	f	t	t
f	f	t	t	t	t

In diesem Fall nennt man die beiden Formeln *äquivalent*. Man kann aus obiger Tabelle auch ablesen, dass eigentlich die drei Symbole $\neg, \vee, \wedge$ ausreichend sind.

Eine Formel die für alle möglichen Interpretationen den Wert wahr annimmt, heisst *Tautologie*. Ist mindestens eine Wertebelegung möglich, die den Wert wahr liefert, so heisst die Formel *erfüllbar*. Eine Formel heisst *kontingent* wenn sie erfüllbar ist aber keine Tautologie. Wird der Wert wahr nie angenommen so ist die Formel *unerfüllbar*, d.h. sie ist dann ein *Widerspruch*. In dieser Sprechweise sind dann zwei Formeln A, B äquivalent wenn die zusammengesetzte Formel $A \iff B$ eine Tautolgie ist. Für äquivalente Formeln verwenden wir das Symbol $\equiv$.

A	B	$(A \Rightarrow B)$	$\neg A \vee B$
t	t	t	t
t	f	f	f
f	t	t	t
f	f	t	t

Obige Tabelle hat z.B. gezeigt dass: $A \Rightarrow B \equiv \neg A \vee B$. Die Äquivalenz definiert eine Äquivalenzrelation auf der Menge aller Formeln aus n Aussagen. Es gibt unendlich viele Formeln aber nur 2^{2^n} verschiedene Äquivalenzklassen. Zwei spezielle Formeln werden noch eingeführt, diese ist T (true, Tautologie) und F (false, falsch) die für alle Interpretationen wahr bzw. falsch sind.

LEMMA 5.1.2. *Eigenschaften der logischen Äquivalenz*

(1) $((X \wedge Y) \vee Z) \equiv ((X \vee Z) \wedge (Y \vee Z))$
(2) $((X \vee Y) \wedge Z) \equiv ((X \wedge Z) \vee (Y \wedge Z))$
(3) $(X \vee \neg X) \equiv T$
(4) $(X \wedge \neg X) \equiv F$
(5) $\neg\neg X \equiv X$

5.1.1. logisches Schließen. Eine Formel A ist die *logische Konsequenz* der Formeln $B_1, \ldots, B_n$ falls jede Interpretation, die bei den B_i wahr liefert auch bei A den Wahrheitswert t liefert. Dies wird geschrieben als:

$$\{B_1, \ldots, B_n\} \models A.$$

Dies wird auch in folgender Notation geschrieben:

$$\begin{array}{c} B_1 \\ B_2 \\ \vdots \\ B_n \\ \hline A \end{array}$$

Es ist möglich auf der Menge der Formeln eine Ordnung $\leq$ zu definieren:

$$A \leq B \text{ falls } B \models A.$$

Dies liefert ein Poset, denn es gilt

(1) $X \leq X$
(2) aus $X \leq Y$ und $Y \leq Z$ folgt $X \leq Z$
(3) aus $X \leq Y$ und $Y \leq X$ folgt $X \equiv Y$

Dies poset ist ein Verband, denn das supremum ist $X \vee Y$ und das infimum ist $X \wedge Y$. Dies ist der *Lindenbaum verband*. (Adolf Lindenbaum 1930).

Wichtige Methoden beim logischen Schließen:

Regel	Name
$\begin{array}{c} A \\ A \Rightarrow B \\ \hline B \end{array}$	*modus ponens*
$\begin{array}{c} \neg B \\ A \Rightarrow B \\ \hline \neg A \end{array}$	*modus tollens*
$\begin{array}{c} A \Rightarrow B \\ B \Rightarrow C \\ \hline A \Rightarrow C \end{array}$	*syllogism*
$\begin{array}{c} A \vee B \\ \neg B \\ \hline A \end{array}$	*disjunctive syllogism*
$\begin{array}{c} \neg A \Rightarrow F \\ \hline A \end{array}$	*contradictio*
$A \Rightarrow \neg(\neg A)$	
$\begin{array}{c} A \Rightarrow \neg B \\ A \Rightarrow B \\ \hline \neg A \end{array}$	*reductio ad absurdum*

5.1.2. automatisches Beweisen. Meist werden mathematische Beweise in der Weise geführt, dass aus einer Menge M von Voraussetzungen eine Aussage A folgt. Wir haben bereits gesehen, dass $M \Rightarrow A$ zu $\neg M \vee A$ äquivalent ist. Dies wiederum ist wahr wenn $M \wedge \neg A$ falsch ist. Also reicht es zu zeigen, dass nie alle Formeln aus M und $\neg A$ zugleich erfüllt werden können, um $M \Rightarrow A$ nachzuweisen. Dazu werden die Formeln aus M und $\neg A$ logisch äquivalent umgeformt, bis sich ein Widerspruch ergibt, also eine unerfüllbare Formel entsteht.

Um einen Widerspruchsbeweis algorithmisch zu führen, wird aus der Menge der Voraussetzungen M und $\neg A$ eine Menge von Formeln gebildet und in eine Normalform transformiert. Nun ist zu testen, ob diese Formel erfüllbar ist. Generell wären bei n Aussagen 2^n mögliche Bewertungen einzusetzen, bis entschieden ist, ob stets der Wahrheitswert f angenommen wird. Durch äquivalente Umformungen nach Einsetzen einiger Werte lässt sich dabei der Aufwand oft reduzieren.

EXAMPLE 5.1.3. Widerspruchsbeweis
Wir wollen

$$((\neg B \vee A) \wedge (C \vee B) \wedge \neg C) \Rightarrow A$$

beweisen. Dazu ist zu zeigen, dass

$$((\neg B \vee A) \wedge (C \vee B) \wedge \neg C) \wedge \neg A = (\neg B \vee A) \wedge (C \vee B) \wedge \neg C \wedge \neg A$$

unerfüllbar ist. Betrachte die Wahrheitstabelle der Teilformeln $(\neg B \vee A) \wedge \neg A$ und $\neg B$

A	B	$\neg B$	$(\neg B \vee A) \wedge \neg A$
t	t	f	f
t	f	t	f
f	t	f	f
f	f	t	t

und man sieht dass, es genügt die Unerfüllbarkeit von $\neg B$ zu betrachten. Bezüglich der Halbordnung des Lindemannverbands bedeutet dies: $\neg B \geq (\neg B \vee A) \wedge \neg A$. Der Formelteil mit A braucht nicht weiter untersucht werden. D.h. die Formel vereinfacht sich zu:

$$\neg B \wedge (C \vee B) \wedge \neg C.$$

Analog wird wegen

$$(C \vee B) \wedge \neg C \leq B,$$

die Formel zu

$$\neg B \wedge B \equiv F$$

vereinfacht. Damit ist der Beweis erbracht, denn aus der Unerfüllbarkeit des Ergebnis ist auch die Ausgangsformel nicht erfüllbar.

5.2. Prädikatenlogik

In der Prädikatenlogik wird die Aussagenlogik um Variablen und *Quantoren* erweitert. Der Quantor $\forall$ heisst "für alle" Quantor. Er bezieht sich auf eine Variable, die in einer Formel auftritt. Neu hinzu kommen auch *Prädikate*, dies sind Eigenschaften, die Konstanten oder Variablen haben können. Als letzte Erweiterung kommen noch Funktionen hinzu.

"Jeder Mensch muss schlafen" wird formalisiert zu

$$\forall x(MENSCH(x) \Rightarrow MUSS - SCHLAFEN(x)).$$

Der Quantor $\exists$ heisst "Existenz" Quantor. Auch er bezieht sich auf eine Variable (im Beispiel x), die in einer Formel auftritt. Prädikate werden im Folgenden stets gross geschrieben, Konstanten klein.

"Fritz hat eine Mutter" wird formalisiert zu

$$\exists x(MUTTER(Fritz, x)).$$

Fritz ist eine Konstante. Bei diesen Quantoren ist die Reihenfolge wichtig für die Bedeutung der Aussage.

"Jeder Mensch hat eine Mutter" wird zu

$$\forall x \exists y(MUTTER(x, y)),$$

während

$$\exists y \forall x(MUTTER(x, y))$$

behauptet " Da gibt es jemanden, der die Mutter aller Menschen ist".

Die Negation einer solchen Aussage erfolgt dadurch, dass $\forall$ durch $\exists$ und umgekehrt ersetzt wird und die Negation auf die quantisierte Formel angewandt wird.

"Fritz hat keine Mutter" wird also zu

$$\forall x \neg(MUTTER(Fritz, x)),$$

und "Nicht jeder Mensch hat eine Mutter" wird zu

$$\exists x \forall y \neg (MUTTER(x, y)).$$

Auch Fragen werden in derartige Ausdrücke umgewandelt, die Frage “Wo ist Klaus?” entspricht dem Nachweis, dass

$$\exists x (AT(Klaus, x))$$

ein erfüllbare Formel ist. Für die Aussage: ”Klaus Vater hat ein Fahrrad”

$$\exists x (BIKE(vater(Klaus), x))$$

wird eine Funktion benötigt. Die Vorgehensweise für die Definition von Formeln der Prädikatenlogik ist ähnlich dem Vorgehen bei der Aussagenlogik.

DEFINITION 5.2.1. Basisbausteine

Eine *atomare Formel* ist ein einzelnes Prädikat. Als Parameter dürfen sog. *Terme* auftauchen, dies sind Variablen, Konstanten und Funktionen. Eine Variable, die in einer atomaren Formel auftaucht heisst *freie* Variable.

Aus atomaren Formeln werden dann die Formeln der *Prädikatenlogik (erster Stufe)* (PL1) gebildet:

DEFINITION 5.2.2. PL1 Formeln

(1) atomare Formeln sind PL1 Formeln
(2) F, G PL1 Formeln, dann auch $F \wedge G$
(3) F, G PL1 Formeln, dann auch $F \vee G$
(4) F, G PL1 Formeln, dann auch $F \Rightarrow G$
(5) F, G PL1 Formeln, dann auch $F \Leftrightarrow G$
(6) F PL1 Formel, dann auch $\neg F$
(7) F PL1 Formel, dann auch (F)

dann noch zwei Methoden mit Quantoren:

(1) F PL1 Formel and x eine Variable, dann ist auch $(\exists x)F$ eine PL1 Formel
(2) F PL1 Formel und x eine Variable, dann ist auch $(\forall x)F$ eine PL1 Formel

Das sind wie schon bei der Aussagenlogik alle Möglichkeiten eine PL1 Formel zu bilden.

War x eine freie Variable in F dann ist sie nach diesen beiden Quantoren Konstruktionen eine *gebundene* Variable. Eine PL1 Formel ohne offene Variablen heisst geschlossene PL1 Formel. Um die Semantik einer PL1 Formelmenge (=Interpretation) zu formulieren muss man etwas mehr Aufwand treiben als im Falle der ’einfachen’ Aussagenlogik.

DEFINITION 5.2.3. Interpretation von PL1 Formeln

Eine Interpretation einer PL1 Formelmenge ist ein Paar (I, D) mit folgenden Eigenschaften:

- D ist eine nichtleere Menge (’domain’)
- I (interpretation) ist eine Abbildung mit folgenden Eigenschaften:
  - jede Konstante oder Variable wird von I auf ein Element aus D abgebildet.
  - jede $n-$stellige Funktion f wird auf eine Funktion $I(f) : D^n \to D$ abgebildet.
  - jedes $n-$stellige Prädikat P wird auf eine Funktion $I(P) : D^n \to \{t, f\}$(wahr,falsch) abgebildet.

Damit kann man dann die Semantik definieren, eine atomare Formel $P(t_1, \ldots, t_n)$ ist dann für eine gegebene Interpretation (I, D) wahr falls: $I(P)(I(t_1), \ldots, I(t_n)) = t$.

Danach definiert man analog zur Vorgehensweise bei der Aussagenlogik die Interpretation bei zusammengesetzten Formeln:

- $\neg F$ ist wahr gdw F nicht wahr ist für (I, D)
- $F \wedge G$ ist wahr gdw F und G sind wahr für (I, D)
- $F \vee G$ ist wahr gdw F oder G wahr ist für (I, D)
- $F \Rightarrow G$ ist wahr gdw $\neg F \vee G$ wahr ist für $I(D)$
- $F \Leftrightarrow G$ ist wahr gdw $F \Rightarrow G$ und $G \Rightarrow F$ sind wahr für $I(D)$

Für die Quantoren:

- $(\exists x)F$ ist wahr gdw F wahr ist für eine Interpretation (I', D), die sich von (I, D) nur im Bild der Variable x unterscheidet.
- $(\forall x)F$ ist wahr gdw F wahr ist für alle Interpretationen (I', D), die sich von (I, D) nur im Bild der Variable x unter I unterscheidet.

Eine Interpretation ist ein *Modell* der Formel F wenn F wahr ist unter dieser Interpretation.

5.3. Automatischer Widerspruchsbeweis

In diesem Abschnitt soll die Vorgehensweise aus dem Beispiel 5.1.3 für PL1 Formeln verallgemeinert werden. Das obige Beispiel kann wie folgt visualisiert werden:

$$\begin{array}{cccc}
(\neg B \vee A)\wedge & (\neg A)\wedge & (C \vee B)\wedge & (\neg C) \\
\searrow & \downarrow & \searrow & \downarrow \\
 & (\neg B) & \wedge & (B) \\
 & & \searrow & \downarrow \\
 & & & F
\end{array}$$

Die allgemeine Grundlage war die Gültigkeit von $(A \vee B) \wedge (C \vee \neg B) \Rightarrow (A \vee C)$ was aus einer Wahrheitstafel ersichtlich ist. Um dies auch für PL1 Formeln zu machen benötigt man einige Vorbereitungen:

5.3.1. Normalform einer PL1 Formel. Der folgende Algorithmus liefert die konjunktive Normalform einer PL1 Formel:

(1) Ersetze $A \iff B$ durch $A \Rightarrow B \wedge B \Rightarrow A$
(2) Ersetze $A \Rightarrow B$ durch $\neg A \vee B$
(3) Zeihe Negation nach innen durch:
   - $\neg\neg A = A$,
   - $\neg(A \vee B) = \neg A \wedge \neg B$, $\neg(A \wedge B) = \neg A \vee \neg B$,
   - $\neg(\forall x)P(x) = (\exists x)\neg P(x)$,$\neg(\exists x)P(x) = (\forall x)\neg P(x)$
(4) Mache Variablennamen eindeutig,z.B. $(\forall x)P(x)\wedge(\exists x)Q(x)$ wird $(\forall x)P(x)\wedge(\exists y)Q(y)$
(5) Alle Quantoren nach links ziehen.
(6) Ersetze den Existenz Quantor durch *Skolem Funktionen*. Dabei wird jede Variable hinter einem $\exists$ Quantor ersetzt durch eine Funktion mit soviel Variablen, wie $\forall$ Variablen im Prädikat vorkommen. e.g. $(\forall x)(\exists y) \ldots. P(x, y)$ wird $(\forall x) \ldots. P(x, f(x))$. Eine Funktion ohne Parameter wird eine Konstante.
(7) Schreibe $\forall$ Quantoren nicht mehr, jede freie Variable ist mit $\forall$ quantifiziert.
(8) Verwende $(A \wedge B) \vee C = (A \vee C) \wedge (B \vee C)$ um eine Konjunktion von Disjunktionen zu bekommen.

Eine solche Disjunktion heisst *Klausel*, die gesamte Normalform heisst auch Klauselform.

5.3.2. Algorithmus Widerspruchsbeweis. Die Idee ist die Vorgehensweise aus dem Beispiel der Aussagenlogik (5.1.3) auf eine PL1 Formel in Klauselform anzuwenden.

Folgende Schritte:

(1) Verneine die PL1 Formel
(2) Wandle in Klauselform um
(3) Versuche folgende Ersetzungen bis man zu einem Widerspruch kommt

- Nimm ein Paar von Klauseln r und t die ein gemeinsames Atom s verneint und nicht verneint enthalten. o. E. $r = x_1 \vee ... \vee x_a \vee s$ und $t = y_1 \vee ... \vee y_b \vee \neg s$
- Versuche durch Variablenersetzungen zur Form s und $\neg s$ zu kommen
- Füge die neue Klausel hinzu, die man aus r und t bekommt wenn man erst die gefundene Ersetzung vornimmt und dann das Atom $\neg s$ und s aus den Klausel entfernt. Die neue Klausel ist dann $x_1 \vee ... \vee x_a \vee y_1 \vee ... \vee y_b$.

EXAMPLE 5.3.1. automatischer Widerspruchsbeweis PL1

Man hat folgende Fakten:

(1) Marcus war ein Mann
(2) Marcus war aus Pompeii
(3) Pompeiier sind Römer
(4) Caesar war ein Staatschef
(5) Alle Römer sind loyal zu Caesar oder sie hassen ihn
(6) Jeder ist loyal zu irgendjemanden
(7) Wer versucht einen Staatschef umzubringen ist nicht loyal zu ihm
(8) Marcus versucht Caesar umzubringen

Dies werden die folgenden PL1 Formeln

(1) $MAN(Markus)$
(2) $POMPEI(Markus)$
(3) $(\forall x)POMPEI(x) \Rightarrow ROMAN(x)$
(4) $LEADER(Caesar)$
(5) $(\forall x)ROMAN(x) \Rightarrow LOYAL(x, Caesar) \vee HATE(x, Caesar)$
(6) $(\forall x)(\exists y)LOYAL(x, y)$
(7) $(\forall x)(\forall y)MAN(x) \wedge LEADER(y) \wedge TRYTOKILL(x, y) \Rightarrow \neg LOYAL(x, y)$
(8) $TRYTOKILL(Marcus, Caesar)$

Es soll bewiesen werden, dass Marcus Caesar hasst. Dazu wird diese Aussage verneint und den 8 PL1 Formeln hinzugefügt:

(9) $\neg HATE(Marcus, Caesar)$

Erster Schritt ist das Erreichen der Klauselform

(1) $MAN(Markus)$
(2) $POMPEI(Markus)$
(3) $\neg POMPEI(x1) \vee ROMAN(x1)$
(4) $LEADER(Caesar)$
(5) $\neg ROMAN(x2) \vee LOYAL(x2, Caesar) \vee HATE(x2, Caesar)$
(6) $LOYAL(x3, f1(x3))$
(7) $\neg MAN(x4) \vee \neg LEADER(y1) \vee \neg TRYTOKILL(x4, y1) \vee \neg LOYAL(x4, y1)$
(8) $TRYTOKILL(Marcus, Caesar)$
(9) $\neg HATE(Marcus, Caesar)$

Der automatische Widerspruchsbeweis kann dann wie folgt ablaufen:

$$
\begin{array}{rc}
\neg HATE(Marcus, Caesar) & \neg ROMAN(x2) \vee LOYAL(x2, Caesar) \\
 & \vee HATE(x2, Caesar) \\
\searrow & \downarrow x2 = Marcus \\
\neg POMPEI(x1) \vee ROMAN(x1) & \neg ROMAN(Marcus) \vee \\
 & LOYAL(Marcus, Caesar) \\
x1 = Marcus \searrow & \downarrow \\
POMPEI(Markus) & \neg POMPEI(Marcus) \vee \\
 & LOYAL(Marcus, Caesar) \\
\searrow & \downarrow \\
\neg MAN(x4) \vee \neg LEADER(y1) \vee & \\
\neg TRYTOKILL(x4, y1) \vee & LOYAL(Marcus, Caesar) \\
\neg LOYAL(x4, y1) & \\
y1 = Caesar \searrow & x4 = Marcus \downarrow \\
 & \neg MAN(Marcus) \vee \neg LEADER(Caesar) \vee \\
LEADER(Caesar) & \neg TRYTOKILL(Marcus, Caesar) \\
\searrow & \downarrow \\
MAN(Marcus) & \neg MAN(Marcus) \vee \\
 & \neg TRYTOKILL(Marcus, Caesar) \\
\searrow & \downarrow \\
TRYTOKILL(Marcus, Caesar) & \neg TRYTOKILL(Marcus, Caesar) \\
\searrow & \downarrow \\
 & \text{FALSE}
\end{array}
$$

Ein schwieriges Problem im Allgemeinen ist das automatische Finden der Ersetzungen (Unifikation)