# Aussagenlogik

# 2

In der Aussagenlogik werden, wie der Name schon sagt, Aussagen über logische Operatoren verknüpft. Der Satz *„die Straße ist nass"* ist eine Aussage, genauso wie *„es regnet"*. Diese beiden Aussagen lassen sich nun verknüpfen zu der neuen Aussage

$$\textit{wenn es regnet ist die Straße nass}.$$

Etwas formaler geschrieben ergibt sich

$$\textit{es regnet} \Rightarrow \textit{die Straße ist nass}.$$

Diese Schreibweise hat unter anderem den Vorteil, dass die elementaren Aussagen in unveränderter Form wieder auftreten. Um im Folgenden ganz exakt mit der Aussagenlogik arbeiten zu können, starten wir mit einer Definition der Menge aller aussagenlogischen Formeln.

## 2.1 Syntax

**Definition 2.1**
Sei $Op = \{\neg, \wedge, \vee, \Rightarrow, \Leftrightarrow, (, )\}$ die Menge der logischen Operatoren und $\Sigma$ eine Menge von Symbolen. Die Mengen $Op$, $\Sigma$ und $\{w, f\}$ seien paarweise disjunkt. $\Sigma$ heißt **Signatur** und seine Elemente sind die **Aussagevariablen**. Die Menge der Aussagenlogischen Formeln wird nun rekursiv definiert:

- $w$ und $f$ sind (atomare) Formeln.
- Alle Aussagevariablen, das heißt alle Elemente von $\Sigma$ sind (atomare) Formeln.
- Sind $A$ und $B$ Formeln, so sind auch $\neg A$, $(A)$, $A \wedge B$, $A \vee B$, $A \Rightarrow B$, $A \Leftrightarrow B$ Formeln.

W. Ertel, *Grundkurs Künstliche Intelligenz*, DOI 10.1007/978-3-8348-2157-7_2,

**Definition 2.2**
Man liest die Operatoren und Symbole folgendermaßen:

$w$: „wahr“
$f$: „falsch“
$\neg A$: „nicht $A$“ (Negation)
$A \wedge B$: „$A$ und $B$“ (Konjunktion)
$A \vee B$: „$A$ oder $B$“ (Disjunktion)
$A \Rightarrow B$: „wenn $A$ dann $B$“ (Implikation[1])
$A \Leftrightarrow B$: „$A$ genau dann, wenn $B$“ (Äquivalenz)

Diese elegante rekursive Definition der Menge aller Formeln erlaubt uns nun die Erzeugung von unendlich vielen Formeln. Mit $\Sigma = \{A, B, C\}$ sind zum Beispiel

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

Formeln. Auch $(((A)) \vee B)$ ist eine syntaktisch korrekte Formel.

Die so definierten Formeln sind bisher rein syntaktische Gebilde ohne Bedeutung. Es fehlt noch die Semantik.

## 2.2 Semantik

In der Aussagenlogik gibt es die zwei Wahrheitswerte $w$ für „wahr“ und $f$ für „falsch“. Starten wir mit einem Beispiel und fragen uns, ob die Formel $A \wedge B$ wahr ist. Die Antwort heißt: Es kommt darauf an, ob die Variablen $A$ und $B$ wahr sind. Steht zum Beispiel $A$ für „*Es regnet heute.*“ und $B$ für „*Es ist kalt heute.*“ und dies trifft beides zu, so ist $A \wedge B$ wahr. Steht jedoch $B$ für „*Es ist heiß heute.*“ (und dies trifft nicht zu) so ist $A \wedge B$ falsch.

Wir müssen offenbar den Aussagevariablen Wahrheitswerte zuordnen, die Zuständen der Welt entsprechen. Daher definieren wir

**Definition 2.3**
Eine Abbildung $I : \Sigma \to \{w, f\}$, die jeder Aussagevariablen einen Wahrheitswert zuordnet, heißt **Belegung** oder **Interpretation** oder auch **Welt**.

Da jede Aussagevariable zwei Wahrheitswerte annehmen kann, besitzt eine aussagenlogische Formel mit $n$ verschiedenen Variablen $2^n$ verschiedene Belegungen, beziehungs-

[1] Auch **materiale Implikation** genannt.

**Tab. 2.1** Definition der logischen Operatoren per Wahrheitstabelle

| $A$ | $B$ | $(A)$ | $\neg A$ | $A \wedge B$ | $A \vee B$ | $A \Rightarrow B$ | $A \Leftrightarrow B$ |
|---|---|---|---|---|---|---|---|
| $w$ | $w$ | $w$ | $f$ | $w$ | $w$ | $w$ | $w$ |
| $w$ | $f$ | $w$ | $f$ | $f$ | $w$ | $f$ | $f$ |
| $f$ | $w$ | $f$ | $w$ | $f$ | $w$ | $w$ | $f$ |
| $f$ | $f$ | $f$ | $w$ | $f$ | $f$ | $w$ | $w$ |

weise Welten. Für die erlaubten Basisoperatoren definieren wir nun deren Wahrheitswerte, indem wir in der **Wahrheitstabelle** (siehe Tab. 2.1) für alle möglichen Belegungen einen Wahrheitswert angeben.

Die leere Formel ist unter allen Belegungen wahr. Um die Wahrheitswerte für komplexe Formeln zu bestimmen, müssen wir noch die Rangfolge der logischen Operatoren angeben. Falls Ausdrücke geklammert sind, ist immer zuerst der Term in der Klammer auszuwerten. Bei ungeklammerten Formeln sind die Prioritäten wie folgt geordnet, beginnend mit der stärksten Bindung: $\neg$, $\wedge$, $\vee$, $\Rightarrow$, $\Leftrightarrow$.

Um die Äquivalenz von Formeln klar von der syntaktischen Äquivalenz zu unterscheiden, definieren wir:

**Definition 2.4**
Zwei Formeln $F$ und $G$ heißen semantisch äquivalent, wenn sie für alle Belegungen die gleichen Wahrheitswerte annehmen. Man schreibt $F \equiv G$.

Die semantische Äquivalenz wird vor allem dazu dienen, in der Metasprache, das heißt der natürlichen Sprache, über die Objektsprache, nämlich die Logik, zu reden. Die Aussage „$A \equiv B$“ etwa besagt, dass die beiden Formeln $A$ und $B$ semantisch äquivalent sind. Die Aussage „$A \Leftrightarrow B$“ hingegen ist ein syntaktisches Objekt der formalen Sprache der Aussagenlogik.

Je nachdem, in wievielen Welten eine Formel wahr ist, teilt man die Formeln ein in folgende Klassen.

**Definition 2.5**
Eine Formel heißt
- **erfüllbar**, falls sie bei mindestens einer Belegung wahr ist.
- **allgemeingültig** oder **wahr**, falls sie bei allen Belegungen wahr ist. Wahre Formeln heißen auch **Tautologien**.
- **unerfüllbar**, falls sie bei keiner Belegung wahr ist.

Jede erfüllende Belegung einer Formel heißt **Modell** der Formel.

Offenbar ist die Negation jeder allgemeingültigen Formel unerfüllbar. Die Negation einer erfüllbaren, aber nicht allgemeingültigen Formel $F$ ist erfüllbar.

Wir können nun auch für komplexere Formeln Wahrheitstafeln erstellen, um deren Wahrheitswerte zu ermitteln. Dies praktizieren wir sofort anhand einiger für die Praxis wichtiger Äquivalenzen von Formeln.

**Satz 2.1**
Die Operationen $\wedge$, $\vee$ sind kommutativ und assoziativ und folgende Äquivalenzen sind allgemeingültig:

$$
\begin{array}{rcll}
\neg A \vee B & \Leftrightarrow & A \Rightarrow B & \text{(Implikation)} \\
A \Rightarrow B & \Leftrightarrow & \neg B \Rightarrow \neg A & \text{(Kontraposition)} \\
(A \Rightarrow B) \wedge (B \Rightarrow A) & \Leftrightarrow & (A \Leftrightarrow B) & \text{(Äquivalenz)} \\
\neg(A \wedge B) & \Leftrightarrow & \neg A \vee \neg B & \text{(De Morgan'sche Regeln)} \\
\neg(A \vee B) & \Leftrightarrow & \neg A \wedge \neg B & \\
A \vee (B \wedge C) & \Leftrightarrow & (A \vee B) \wedge (A \vee C) & \text{(Distributivgesetze)} \\
A \wedge (B \vee C) & \Leftrightarrow & (A \wedge B) \vee (A \wedge C) & \\
A \vee \neg A & \Leftrightarrow & w & \text{(Tautologie)} \\
A \wedge \neg A & \Leftrightarrow & f & \text{(Widerspruch)} \\
A \vee f & \Leftrightarrow & A & \\
A \vee w & \Leftrightarrow & w & \\
A \wedge f & \Leftrightarrow & f & \\
A \wedge w & \Leftrightarrow & A &
\end{array}
$$

**Beweis** Um die erste Äquivalenz zu zeigen, berechnen wir per Wahrheitstabelle $\neg A \vee B$ und $A \Rightarrow B$ und erkennen, dass die Wahrheitswerte beider Formeln in allen Welten (Belegungen) gleich sind. Die Formeln sind also äquivalent, und in der letzten Spalte stehen daher nur „$w$"-s.

| $A$ | $B$ | $\neg A$ | $\neg A \vee B$ | $A \Rightarrow B$ | $(\neg A \vee B) \Leftrightarrow (A \Rightarrow B)$ |
|---|---|---|---|---|---|
| $w$ | $w$ | $f$ | $w$ | $w$ | $w$ |
| $w$ | $f$ | $f$ | $f$ | $f$ | $w$ |
| $f$ | $w$ | $w$ | $w$ | $w$ | $w$ |
| $f$ | $f$ | $w$ | $w$ | $w$ | $w$ |

Die Beweise für die anderen Äquivalenzen laufen analog und werden dem Leser zur Übung empfohlen (Aufgabe 2.2). □

## 2.3 Beweisverfahren

In der KI sind wir daran interessiert, aus bestehendem Wissen neues Wissen herzuleiten, beziehungsweise Anfragen zu beantworten. In der Aussagenlogik geht es darum zu zeigen, dass aus einer **Wissensbasis** *WB*, das heißt einer (eventuell umfangreichen) aussagenlogischen Formel, eine Formel $Q$[2] folgt. Zuerst definieren wir den Begriff der Folgerung.

**Definition 2.6**
Eine Formel $Q$ **folgt** aus einer Formel *WB*, wenn jedes Modell von *WB* auch ein Modell von $Q$ ist. Man schreibt dann $WB \models Q$.

In anderen Worten heißt das, dass in jeder Welt (Belegung von Variablen), in der *WB* wahr ist, auch $Q$ wahr ist. Oder noch knapper: immer dann, wenn *WB* wahr ist, ist auch $Q$ wahr. Da für den Folgerungsbegriff Belegungen von Variablen herangezogen werden, handelt es sich hier um einen semantischen Begriff.

Jede Formel, die nicht allgemeingültig ist, wählt gewissermaßen aus der Menge aller Belegungen eine Teilmenge als ihre Modelle aus. Tautologien wie zum Beispiel $A \vee \neg A$ schränken die Zahl der erfüllenden Belegungen nicht ein, denn ihre Aussage ist leer. Die leere Formel ist daher in allen Belegungen wahr. Für jede Tautologie $T$ gilt also $\varnothing \models T$. Intuitiv heißt das, dass Tautologien immer gelten, ohne Einschränkung der Belegungen durch eine Formel. Man schreibt dann auch kurz $\models T$. Nun zeigen wir einen wichtigen Zusammenhang zwischen dem Folgerungsbegriff und der Implikation.

**Satz 2.2 (Deduktionstheorem)**

$$A \models B \text{ gilt genau dann wenn } \models A \Rightarrow B.$$

**Beweis** Betrachten wir die Wahrheitstabelle der Implikation:

| $A$ | $B$ | $A \Rightarrow B$ |
|---|---|---|
| $w$ | $w$ | $w$ |
| $w$ | $f$ | $f$ |
| $f$ | $w$ | $w$ |
| $f$ | $f$ | $w$ |

[2] $Q$ steht hier für Query (Anfrage).

Eine beliebige Implikation $A \Rightarrow B$ ist offenbar immer wahr, außer bei der Belegung $A \mapsto w, B \mapsto f$. Nehmen wir an, dass $A \vDash B$ gilt. Das heißt, dass für jede Belegung, die $A$ wahr macht, auch $B$ wahr ist. Die kritische zweite Zeile der Wahrheitstabelle kommt damit gar nicht vor. Also ist $A \Rightarrow B$ wahr, das heißt $A \Rightarrow B$ ist eine Tautologie. Damit ist eine Richtung des Satzes gezeigt.

Nehmen wir nun an, dass $A \Rightarrow B$ gilt. Damit ist die kritische zweite Zeile der Wahrheitstabelle auch ausgeschlossen. Jedes Modell von $A$ ist dann auch Modell von $B$. Also gilt $A \vDash B$. □

Will man zeigen, dass $Q$ aus $WB$ folgt, so kann man also mit Hilfe der Wahrheitstafelmethode nachweisen, dass $WB \Rightarrow Q$ eine Tautologie ist. Damit haben wir ein erstes **Beweisverfahren** für die Aussagenlogik, das sich leicht automatisieren lässt. Der Nachteil dieser Methode ist die im Worst-Case sehr lange Rechenzeit. Es muss nämlich im Worst-Case bei $n$ Aussagevariablen für alle $2^n$ Belegungen der Variablen die Formel $WB \Rightarrow Q$ ausgewertet werden. Die Rechenzeit wächst also exponentiell mit der Zahl der Variablen. Somit ist dieses Verfahren für große Variablenanzahl zumindest im Worst-Case nicht anwendbar.

Wenn eine Formel $Q$ aus $WB$ folgt, so ist nach dem Deduktionstheorem $WB \Rightarrow Q$ eine Tautologie. Also ist die Negation $\neg(WB \Rightarrow Q)$ unerfüllbar. Wegen

$$\neg(WB \Rightarrow Q) \equiv \neg(\neg WB \vee Q) \equiv WB \wedge \neg Q$$

ist damit auch $WB \wedge \neg Q$ unerfüllbar. Diese einfache, aber wichtige Folgerung aus dem Deduktionstheorem formulieren wir als Satz.

**Satz 2.3 (Widerspruchsbeweis)**
$WB \vDash Q$ gilt genau dann wenn $WB \wedge \neg Q$ unerfüllbar ist.

Um zu zeigen, dass die zu beweisende Anfrage $Q$ aus der Wissensbasis $WB$ folgt, kann man also die negierte Anfrage $\neg Q$ zur Wissensbasis hinzufügen und daraus einen Widerspruch ableiten. Wegen der Äquivalenz $A \wedge \neg A \equiv f$ aus Satz 2.1 wissen wir, dass ein Widerspruch unerfüllbar ist. Damit ist also $Q$ bewiesen. Dieses in der Mathematik häufig angewendete Verfahren wird auch von verschiedenen automatischen Beweiskalkülen, unter anderem vom Resolutionskalkül oder bei der Abarbeitung von Prolog-Programmen verwendet.

Eine Möglichkeit, das Durchprobieren aller Belegungen bei der Wahrheitstafelmethode zu vermeiden, ist die syntaktische Manipulation der Formeln $WB$ und $Q$ durch Anwendung von Inferenzregeln mit dem Ziel, diese so stark zu vereinfachen, dass man am Ende sofort erkennt, dass $WB \vDash Q$. Man bezeichnet diesen syntaktischen Prozess als **Ableitung**

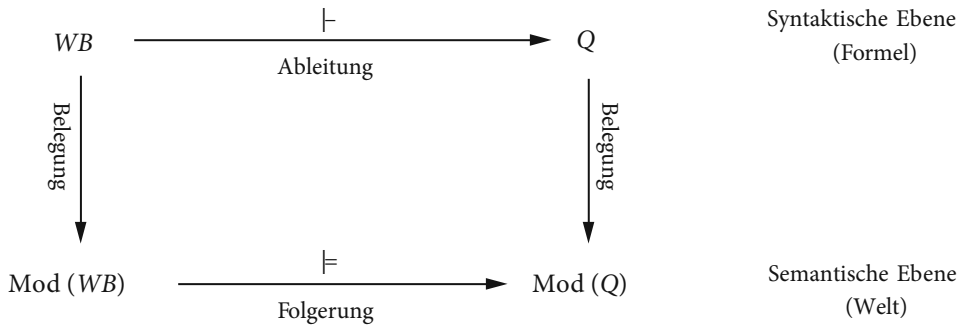


**Abb. 2.1** Syntaktische Ableitung und semantische Folgerung. Mod$(X)$ steht für die Menge der Modelle einer Formel $X$

und schreibt $WB \vdash Q$. Solch ein syntaktisches Beweisverfahren wird **Kalkül** genannt. Um sicherzustellen, dass ein Kalkül keine Fehler macht, definieren wir zwei fundamentale Eigenschaften von Kalkülen.

**Definition 2.7**
Ein Kalkül heißt **korrekt**, wenn jede abgeleitete Aussage auch semantisch folgt, das heißt, wenn für Formeln $WB$ und $Q$ gilt

$$\text{wenn} \quad WB \vdash Q \quad \text{dann} \quad WB \models Q.$$

Ein Kalkül heißt **vollständig**, wenn alle semantischen Folgerungen abgeleitet werden können, das heißt, wenn für Formeln $WB$ und $Q$ gilt

$$\text{wenn} \quad WB \models Q \quad \text{dann} \quad WB \vdash Q.$$

Die Korrektheit eines Kalküls stellt also sicher, dass alle abgeleiteten Formeln tatsächlich semantische Folgerungen aus der Wissensbasis sind. Der Kalkül produziert keine „falschen Folgerungen“. Die Vollständigkeit eines Kalküls hingegen stellt sicher, dass der Kalkül nichts übersieht. Ein vollständiger Kalkül findet immer einen Beweis, wenn die zu beweisende Formel aus der Wissensbasis folgt. Ist ein Kalkül korrekt und vollständig, so sind syntaktische Ableitung und semantische Folgerung zwei äquivalente Relationen (siehe Abb. 2.1).

Um die automatischen Beweisverfahren möglichst einfach zu halten, arbeiten diese meist auf Formeln in konjunktiver Normalform.

**Definition 2.8**
Eine Formel ist in **konjunktiver Normalform (KNF)** genau dann, wenn sie aus einer **Konjunktion**

$$K_1 \wedge K_2 \wedge \ldots \wedge K_m$$

von Klauseln besteht. Eine Klausel $K_i$ besteht aus einer **Disjunktion**

$$(L_{i1} \vee L_{i2} \vee \ldots \vee L_{in_i})$$

von Literalen. Ein **Literal** schließlich ist eine Variable (positives Literal) oder eine negierte Variable (negatives Literal).

Die Formel $(A \vee B \vee \neg C) \wedge (A \vee B) \wedge (\neg B \vee \neg C)$ ist in konjunktiver Normalform. Die konjunktive Normalform stellt keine Einschränkung der Formelmenge dar, denn es gilt

**Satz 2.4**
Jede aussagenlogische Formel lässt sich in eine äquivalente konjunktive Normalform transformieren.

**Beispiel 2.1**

Wir bringen $A \vee B \Rightarrow C \wedge D$ in konjunktive Normalform, indem wir die Äquivalenzen aus Satz 2.1 anwenden:

$$\begin{aligned}
&A \vee B \Rightarrow C \wedge D \\
&\quad \equiv \neg(A \vee B) \vee (C \wedge D) && \text{(Implikation)} \\
&\quad \equiv (\neg A \wedge \neg B) \vee (C \wedge D) && \text{(de Morgan)} \\
&\quad \equiv (\neg A \vee (C \wedge D)) \wedge (\neg B \vee (C \wedge D)) && \text{(Distributivgesetz)} \\
&\quad \equiv ((\neg A \vee C) \wedge (\neg A \vee D)) \wedge ((\neg B \vee C) \wedge (\neg B \vee D)) && \text{(Distributivgesetz)} \\
&\quad \equiv (\neg A \vee C) \wedge (\neg A \vee D) \wedge (\neg B \vee C) \wedge (\neg B \vee D) && \text{(Assoziativgesetz)}
\end{aligned}$$

Zum syntaktischen Beweisen von aussagenlogischen Formeln fehlt uns nun nur noch ein Kalkül. Wir starten mit dem **Modus Ponens**, einer einfachen intuitiven Inferenzregel, die aus der Gültigkeit von $A$ und $A \Rightarrow B$ die Ableitung von $B$ erlaubt. Formal schreibt man dies so

$$\frac{A,\ A \Rightarrow B}{B}.$$

Diese Schreibweise bedeutet, dass aus den durch Komma getrennten Formeln über der Linie die Formel(n) unter der Linie hergeleitet werden können. Modus Ponens als einzige Regel ist zwar korrekt (siehe Aufgabe 2.3), aber nicht vollständig. Fügt man weitere Regeln hinzu, so kann man einen vollständigen Kalkül erzeugen, den wir hier aber nicht betrachten wollen. Stattdessen werden wir als Alternative die **Resolutionsregel**

$$\frac{A \vee B, \quad \neg B \vee C}{A \vee C} \tag{2.1}$$

untersuchen. Die abgeleitete Klausel wird Resolvente genannt. Durch leichte Umformung erhalten wir die äquivalente Form

$$\frac{A \vee B, \quad B \Rightarrow C}{A \vee C}.$$

Setzt man hier $A$ auf $f$, so erkennt man, dass die Resolutionsregel eine Verallgemeinerung des Modus Ponens ist. Genauso ist die Resolutionsregel anwendbar, wenn $C$ fehlt oder wenn $A$ und $C$ fehlen. Im letzten Fall wird aus dem Widerspruch $B \wedge \neg B$ die leere Klausel abgeleitet (Aufgabe 2.7).

## 2.4 Resolution

Wir verallgemeinern nun die Resolutionsregel nochmal, indem wir Klauseln mit einer beliebigen Zahl von Literalen zulassen. Mit den Literalen $A_1, \ldots, A_m, B, C_1, \ldots, C_n$ lautet die **allgemeine Resolutionsregel**

$$\frac{(A_1 \vee \ldots \vee A_m \vee B), \quad (\neg B \vee C_1 \vee \ldots \vee C_n)}{(A_1 \vee \ldots \vee A_m \vee C_1 \vee \ldots \vee C_n)}. \tag{2.2}$$

Man nennt die Literale $B$ und $\neg B$ komplementär. Die Resolutionsregel löscht aus den beiden Klauseln ein Paar von komplementären Literalen und vereinigt alle restlichen Literale zu einer neuen Klausel.

Um zu beweisen, dass aus einer Wissensbasis $WB$ eine Anfrage $Q$ folgt, wird ein Widerspruchsbeweis durchgeführt. Nach Satz 2.3 ist zu zeigen, dass aus $WB \wedge \neg Q$ ein Widerspruch ableitbar ist. Bei Formeln in konjunktiver Normalform tritt ein Widerspruch in Form von zwei Klauseln $(A)$ und $(\neg A)$ auf, die zur leeren Klausel als Resolvente führen. Dass dieses Verfahren auch wirklich wie gewünscht funktioniert, sichert uns der folgende Satz.

Damit der Kalkül auch vollständig wird, benötigen wir noch eine kleine Ergänzung, wie folgendes einfache Beispiel zeigt. Sei als Wissensbasis die Formel $(A \vee A)$ gegeben. Um per Resolution zu zeigen, dass daraus $(A \wedge A)$ ableitbar ist, muss gezeigt werden, dass aus $(A \vee A) \wedge (\neg A \vee \neg A)$ die leere Klausel ableitbar ist. Mit der Resolutionsregel allein ist

das nicht möglich. Mit der **Faktorisierung**, die es erlaubt, Kopien von Literalen in Klauseln zu löschen, wird dieses Problem beseitigt. In dem Beispiel führt zweimalige Anwendung der Faktorisierung zu $(A) \wedge (\neg A)$ und ein Resolutionsschritt zur leeren Klausel.

**Satz 2.5**
Der Resolutionskalkül zum Beweis der Unerfüllbarkeit von Formeln in konjunktiver Normalform ist korrekt und vollständig. Das heißt, für jede unerfüllbare Formel kann man mit Resolution die leere Klausel herleiten und umgekehrt.

Da es Aufgabe des Resolutionskalküls ist, einen Widerspruch aus $WB \wedge \neg Q$ herzuleiten, ist es ganz wichtig, dass die Wissensbasis $WB$ **widerspruchsfrei** ist.

**Definition 2.9**
Eine Formel $WB$ heißt **widerspruchsfrei** oder **konsistent**, wenn sich aus ihr kein Widerspruch, das heißt keine Formel der Form $\phi \wedge \neg\phi$, ableiten lässt.

Andernfalls lässt sich aus $WB$ alles herleiten (siehe Aufgabe 2.8). Dies gilt nicht nur für die Resolution, sondern für viele andere Kalküle auch.

Unter den Kalkülen zum automatischen Beweisen spielt die Resolution eine herausragende Rolle. Daher wollen wir uns mit ihr etwas näher beschäftigen. Gegenüber anderen Kalkülen besitzt die Resolution nur zwei Inferenzregeln, und sie arbeitet auf Formeln in konjunktiver Normalform. Dies macht die Implementierung einfacher. Ein weiterer Vorteil gegenüber vielen Kalkülen liegt in der Reduktion der Anzahl von Möglichkeiten für die Anwendung von Inferenzregeln in jedem Schritt beim Beweisen, wodurch dann der Suchraum reduziert und die Rechenzeit verringert wird.

Wir starten mit einem einfachen Logikrätsel als Beispiel, an dem sich alle wichtigen Schritte eines Resolutionsbeweises gut zeigen lassen.

**Beispiel 2.2**

Das Logikrätsel Nr. 7 aus [Ber89] mit dem Titel *A charming English family* lautet:

> Nachdem ich sieben Jahre lang mit glänzendem Erfolg Englisch studiert habe, muss ich zugeben, dass ich, wenn ich Engländer englisch sprechen höre, vollkommen perplex bleibe. Nun habe ich neulich, von edlen Gefühlen bewegt, drei Anhalter, Vater, Mutter und Tochter, mitgenommen, die, wie ich schnell begriff, Engländer waren und folglich nur Englisch sprachen. Bei jedem ihrer Sätze zögerte ich zwischen zwei möglichen Bedeutungen. Sie sagten mir folgendes (der zweite Sinn steht in Klammern): Der Vater: „Wir fahren bis nach Spanien (wir kommen aus Newcastle)." Die Mutter: „Wir fahren nicht nach Spanien und kommen aus Newcastle (wir haben in Paris angehalten und fahren nicht nach Spanien)." Die Tochter: „Wir

kommen nicht aus Newcastle (wir haben in Paris angehalten)." What about this charming English family?

Zur Lösung derartiger Aufgaben geht man in drei Schritten vor: Formalisierung, Transformation in Normalform und Beweis. In vielen Fällen ist die Formalisierung der mit Abstand schwierigste Schritt, denn man kann dabei viele Fehler machen oder Kleinigkeiten vergessen (Daher ist das praktische Üben hier ganz wichtig. Siehe Aufgaben 2.9–2.11).

Wir verwenden hier die Variablen $S$ für „*Wir fahren bis nach Spanien*", $N$ für „*Wir kommen aus Newcastle*" und $P$ für „*wir haben in Paris angehalten*" und erhalten als Formalisierung der drei Aussagen von Vater, Mutter und Tochter

$$(S \vee N) \wedge [(\neg S \wedge N) \vee (P \wedge \neg S)] \wedge (\neg N \vee P).$$

Ausklammern von $\neg S$ in der mittleren Teilformel bringt die Formel in einem Schritt in KNF. Nummerierung der Klauseln durch tiefgestellte Indizes ergibt

$$WB \equiv (S \vee N)_1 \wedge (\neg S)_2 \wedge (P \vee N)_3 \wedge (\neg N \vee P)_4.$$

Nun starten wir den Resolutionsbeweis, vorerst noch ohne eine Anfrage $Q$. Ein Ausdruck der Form „Res(m,n): <*Klausel*>$_k$" bedeutet, dass <Klausel> durch Resolution von Klausel $m$ mit Klausel $n$ entstanden ist und die Nummer $k$ erhält.

$$\begin{aligned} &\text{Res(1,2)}: \quad (N)_5 \\ &\text{Res(3,4)}: \quad (P)_6 \\ &\text{Res(1,4)}: \quad (S \vee P)_7 \end{aligned}$$

Die Klausel $P$ hätten wir auch erhalten können durch Res(4,5) oder Res(2,7). Jeder weitere Resolutionsschritt würde zur Ableitung schon vorhandener Klauseln führen. Da sich die leere Klausel nicht ableiten lässt, ist damit gezeigt, dass die Wissensbasis widerspruchsfrei ist. Bis jetzt haben wir $N$ und $P$ abgeleitet. Um zu zeigen, dass $\neg S$ gilt, fügen wir die Klausel $(S)_8$ als negierte Anfrage zur Klauselmenge hinzu. Durch den Resolutionsschritt

$$\text{Res(2,8)}: ()_9$$

ist der Beweis geschafft. Es gilt also $\neg S \wedge N \wedge P$. Die „charming English family" kommt offenbar aus Newcastle, hat in Paris angehalten und fährt nicht nach Spanien.

**Beispiel 2.3**

Das Logikrätsel Nr. 28 aus [Ber89] mit dem Titel *Der Hochsprung* lautet:

Drei junge Mädchen üben Hochsprung für die Sportprüfung im Abitur. Die Stange wurde bei 1,20 m befestigt. „Ich wette", sagt das erste zum zweiten Mädchen, „dass mir mein Sprung

gelingen wird, wenn, und nur dann, wenn du versagst.“ Wenn das zweite junge Mädchen das gleiche zu dem dritten sagt, welches wiederum das gleiche zu dem ersten sagt, wäre es dann möglich, dass keins von den dreien seine Wette verliert?

Wir zeigen per Resolutionsbeweis, dass nicht alle drei Mädchen ihre Wette gewinnen können.

**Formalisierung:**

Der Sprung des ersten Mädchens gelingt: $A$, Wette des ersten Mädchens: $(A \Leftrightarrow \neg B)$, der Sprung des zweiten Mädchens gelingt: $B$, Wette des zweiten Mädchens: $(B \Leftrightarrow \neg C)$, der Sprung des dritten Mädchens gelingt: $C$. Wette des dritten Mädchens: $(C \Leftrightarrow \neg A)$.

Behauptung: Es können nicht alle drei die Wette gewinnen:

$$Q \equiv \neg((A \Leftrightarrow \neg B) \wedge (B \Leftrightarrow \neg C) \wedge (C \Leftrightarrow \neg A))$$

Per Resolution ist nun zu zeigen, dass $\neg Q$ unerfüllbar ist.

**Transformation in KNF:** Wette des ersten Mädchens:

$$(A \Leftrightarrow \neg B) \equiv (A \Rightarrow \neg B) \wedge (\neg B \Rightarrow A) \equiv (\neg A \vee \neg B) \wedge (A \vee B)$$

Die Wetten der beiden anderen Mädchen werden analog transformiert, und man erhält als negierte Behauptung

$$\begin{aligned}\neg Q \equiv\ & (\neg A \vee \neg B)_1 \wedge (A \vee B)_2 \wedge (\neg B \vee \neg C)_3 \wedge (B \vee C)_4 \\ & \wedge (\neg C \vee \neg A)_5 \wedge (C \vee A)_6.\end{aligned}$$

Daraus wird nun mit Resolution die leere Klausel abgeleitet:

$$\begin{aligned}&\text{Res(1,6)}: && (C \vee \neg B)_7 \\ &\text{Res(4,7)}: && (C)_8 \\ &\text{Res(2,5)}: && (B \vee \neg C)_9 \\ &\text{Res(3,9)}: && (\neg C)_{10} \\ &\text{Res(8,10)}: && ()\end{aligned}$$

Damit ist die Behauptung gezeigt.

## 2.5 Hornklauseln

Eine Klausel in konjunktiver Normalform enthält positive und negative Literale und lässt sich daher darstellen in der Form

$$(\neg A_1 \vee \ldots \vee \neg A_m \vee B_1 \vee \ldots \vee B_n)$$

mit den Variablen $A_1, \ldots, A_m$ und $B_1, \ldots, B_n$. Diese Klausel lässt sich in zwei einfachen Schritten umformen in die äquivalente Form

$$A_1 \wedge \ldots \wedge A_m \Rightarrow B_1 \vee \ldots \vee B_n.$$

Diese Implikation enthält als Prämisse (Voraussetzung) eine Konjunktion von Variablen und als Konklusion (Folgerung) eine Disjunktion von Variablen. Zum Beispiel ist „*Wenn das Wetter schön ist und Schnee liegt, gehe ich Skifahren oder ich gehe arbeiten.*“ eine Aussage von dieser Form. Der Empfänger dieser Botschaft weiß dann immerhin, dass der Absender zum Beispiel nicht schwimmen geht. Eine wesentlich klarere Aussage wäre „*Wenn das Wetter schön ist und Schnee liegt, gehe ich Skifahren.*“. Der Empfänger weiß nun definitiv Bescheid. Daher nennt man Klauseln mit höchstens einem positiven Literal auch definite Klauseln. Diese Klauseln haben den Vorteil, dass sie nur einen Schluss zulassen und daher deutlich einfacher zu interpretieren sind. Viele Zusammenhänge lassen sich mit derartigen Klauseln beschreiben. Wir definieren daher

**Definition 2.10**
Klauseln mit höchstens einem positiven Literal der Formen

$$(\neg A_1 \vee \ldots \vee \neg A_m \vee B) \quad \text{oder} \quad (\neg A_1 \vee \ldots \vee \neg A_m) \quad \text{oder} \quad B$$

beziehungsweise (äquivalent)

$$A_1 \wedge \ldots \wedge A_m \Rightarrow B \quad \text{oder} \quad A_1 \wedge \ldots \wedge A_m \Rightarrow f \quad \text{oder} \quad B.$$

heißen **Hornklauseln** (nach ihrem Erfinder). Eine Klausel mit nur einem positiven Literal heißt **Fakt**. Bei Klauseln mit negativen und einem positiven Literal wird das positive Literal **Kopf** genannt.

Zum besseren Verständnis der Darstellung von Hornklauseln möge der Leser die in der Definition verwendeten Äquivalenzen herleiten (Aufgabe 2.12).

Nicht nur im täglichen Leben, sondern auch beim formalen Schließen sind Hornklauseln einfacher zu handhaben, wie man an folgendem Beispiel erkennt. Die Wissensbasis bestehe aus folgenden Klauseln (das die Klauseln verbindende „$\wedge$“ wird hier und im Folgenden weggelassen):

$$(Wetter\_schön)_1$$
$$(Schneefall)_2$$
$$(Schneefall \Rightarrow Schnee)_3$$
$$(Wetter\_schön \wedge Schnee \Rightarrow Skifahren)_4$$

Wollen wir nun wissen, ob *Skifahren* gilt, so lässt sich dies relativ leicht folgern. Als Inferenzregel genügt hier zum Beispiel ein leicht verallgemeinerter Modus Ponens:

$$\frac{A_1 \wedge \ldots \wedge A_m,\ A_1 \wedge \ldots \wedge A_m \Rightarrow B}{B}.$$

Der Beweis für „*Skifahren*“ hat dann folgende Gestalt ($MP(i_1, \ldots, i_k)$ steht für die Anwendung des Modus Ponens auf die Klauseln $i_1$ bis $i_k$.)

$$\begin{array}{ll} \text{MP(2,3)}: & (\mathit{Schnee})_5 \\ \text{MP(1,5,4)}: & (\mathit{Skifahren})_6 \end{array}$$

Man erhält mit Modus Ponens einen vollständigen Kalkül für Formeln, die aus aussagenlogischen Hornklauseln bestehen. Bei großen Wissensbasen kann man mit Modus Ponens allerdings sehr viele unnötige Formeln ableiten, wenn man mit den falschen Klauseln startet. Besser ist es daher in vielen Fällen, einen Kalkül zu verwenden, der mit der Anfrage startet und rückwärts arbeitet, bis die Fakten erreicht sind. Solch ein Verfahren wird auch als **backward chaining** bezeichnet, im Gegensatz zum **forward chaining**, welches bei den Fakten startet und schließlich die Anfrage herleitet, so wie im obigen Beispiel mit Modus Ponens.

Für das backward chaining auf Hornklauseln wird **SLD-Resolution** verwendet. SLD steht für „*Selection rule driven linear resolution for definite clauses*“. Am obigen Beispiel, ergänzt durch die negierte Anfrage $(\mathit{Skifahren} \Rightarrow f)$

$$\begin{array}{l} (\mathit{Wetter\_schön})_1 \\ (\mathit{Schneefall})_2 \\ (\mathit{Schneefall} \Rightarrow \mathit{Schnee})_3 \\ (\mathit{Wetter\_schön} \wedge \mathit{Schnee} \Rightarrow \mathit{Skifahren})_4 \\ (\mathit{Skifahren} \Rightarrow f)_5 \end{array}$$

führt die SLD-Resolution, beginnend mit dieser Klausel folgende Resolutionsschritte aus

$$\begin{array}{ll} \text{Res(5,4)}: & (\mathit{Wetter\_schön} \wedge \mathit{Schnee} \Rightarrow f)_6 \\ \text{Res(6,1)}: & (\mathit{Schnee} \Rightarrow f)_7 \\ \text{Res(7,3)}: & (\mathit{Schneefall} \Rightarrow f)_8 \\ \text{Res(8,2)}: & () \end{array}$$

und leitet mit der leeren Klausel einen Widerspruch her. Man erkennt hier gut die „*linear resolution*“, was bedeutet, dass immer mit der gerade aktuell hergeleiteten Klausel weitergearbeitet wird. Dies führt zu einer starken Reduktion des Suchraumes. Ausserdem werden die Literale der aktuellen Klausel immer der Reihe nach in fester Reihenfolge (z.B. von links nach rechts) abgearbeitet („*Selection rule driven*“). Die Literale der aktuellen Klausel werden **Teilziele** (engl. subgoals) genannt. Die Literale der negierten Anfrage sind die **Ziele**

(engl. goals). Die Inferenzregel für einen Schritt lautet

$$\frac{A_1 \wedge \ldots \wedge A_m \Rightarrow B_1, \quad B_1 \wedge B_2 \wedge \ldots \wedge B_n \Rightarrow f}{A_1 \wedge \ldots \wedge A_m \wedge B_2 \wedge \ldots \wedge B_n \Rightarrow f}.$$

Vor Anwendung der Inferenzregel sind $B_1, B_2, \ldots, B_n$ – die aktuellen Teilziele – zu beweisen. Nach der Anwendung wird $B_1$ ersetzt durch die neuen Teilziele $A_1 \wedge \ldots \wedge A_m$. Um zu zeigen, dass $B_1$ wahr ist, muss nun gezeigt werden, dass $A_1 \wedge \ldots \wedge A_m$ wahr sind. Dieser Prozess setzt sich so lange fort, bis die Liste der Teilziele der aktuellen Klausel (der so genannte **goal stack**) leer ist. Damit ist ein Widerspruch gefunden. Gibt es zu einem Teilziel $\neg B_i$ keine Klausel mit einem komplementären Literal $B_i$ als Klauselkopf, so terminiert der Beweis und es kann kein Widerspruch gefunden werden. Die Anfrage ist also nicht beweisbar.

SLD-Resolution spielt in der Praxis eine wichtige Rolle, denn Programme in der Logikprogrammiersprache Prolog bestehen aus prädikatenlogischen Hornklauseln, und ihre Abarbeitung erfolgt mittels SLD-Resolution (siehe Aufgabe 2.13, bzw. Kap. 5).

## 2.6 Berechenbarkeit und Komplexität

Die Wahrheitstafelmethode als einfachstes semantisches Beweisverfahren für die Aussagenlogik stellt einen Algorithmus dar, der für jede Formel nach endlicher Zeit alle Modelle bestimmt. Damit sind die Mengen der unerfüllbaren, der erfüllbaren und der allgemeingültigen Formeln entscheidbar. Die Rechenzeit der Wahrheitstafelmethode für die Erfüllbarkeit wächst im Worst-Case exponentiell mit der Zahl $n$ der Variablen, denn die Wahrheitstabelle besitzt $2^n$ Zeilen. Eine Optimierung, die Methode der **Semantischen Bäume**, vermeidet die Betrachtung von Variablen, die in Klauseln nicht vorkommen, und spart daher in vielen Fällen Rechenzeit, aber im Worst-Case ist sie ebenfalls exponentiell.

Bei der Resolution wächst die Zahl der abgeleiteten Klauseln im Worst-Case exponentiell mit der Zahl der anfänglich vorhandenen Klauseln [Let03]. Für die Entscheidung zwischen den beiden Verfahren kann man daher als Faustregel angeben, dass bei vielen Klauseln mit wenigen Variablen die Wahrheitstafelmethode vorzuziehen ist und bei wenigen Klauseln mit vielen Variablen die Resolution voraussichtlich schneller zum Ziel kommen wird.

Bleibt noch die Frage, ob das Beweisen in der Aussagenlogik nicht schneller gehen kann. Gibt es bessere Algorithmen? Die Antwort lautet: Vermutlich nicht. Denn S. Cook, der Begründer der Komplexitätstheorie, hat gezeigt, dass das 3-Sat-Problem NP-vollständig ist. 3-Sat ist die Menge aller KNF-Formeln, deren Klauseln genau drei Literale haben. Damit ist klar, dass es vermutlich (modulo dem P/NP-Problem) keinen polynomiellen Algorithmus für 3-Sat und damit auch nicht allgemein geben wird. Für Hornklauseln gibt es jedoch einen Algorithmus, bei dem die Rechenzeit für das Testen der Erfüllbarkeit nur linear mit der Zahl der Literale in der Formel wächst.

## 2.7 Anwendungen und Grenzen

Aussagenlogische Beweiser gehören in der Digitaltechnik zum täglichen Handwerkszeug der Entwickler. Zum Beispiel die Verifikation digitaler Schaltungen oder die Generierung von Testmustern zum Test von Mikroporzessoren in der Fertigung gehören zu diesen Aufgaben. Hier kommen auch spezielle Beweisverfahren zum Einsatz, die auf binären Entscheidungsdiagrammen (engl. binary decision diagram, BDD) als Datenstruktur für aussagenlogische Formeln arbeiten.

In der KI kommt die Aussagenlogik bei einfachen Anwendungen zum Einsatz. Zum Beispiel kann ein einfaches Expertensystem durchaus mit Aussagenlogik arbeiten. Allerdings müssen die Variablen alle diskret sein, mit wenigen Werten, und es dürfen keine Querbeziehungen zwischen den Variablen bestehen. Komplexere logische Zusammenhänge können mit der Prädikatenlogik wesentlich eleganter ausgedrückt werden.

Eine sehr interessante und aktuelle Kombination von Aussagenlogik und Wahrscheinlichkeitsrechnung zur Modellierung von unsicherem Wissen ist die probabilistische Logik, die in Kap. 7 ausführlich behandelt wird. Auch die Fuzzy-Logik, welche unendlich viele Wahrheitswerte erlaubt, wird in diesem Kapitel besprochen.

## 2.8 Übungen

**Aufgabe 2.1** ➳ Geben Sie eine Backus-Naur-Form-Grammatik für die Syntax der Aussagenlogik an.

**Aufgabe 2.2** Zeigen Sie, dass folgende Formeln Tautologien sind:
a) $\neg(A \wedge B) \Leftrightarrow \neg A \vee \neg B$
b) $A \Rightarrow B \Leftrightarrow \neg B \Rightarrow \neg A$
c) $((A \Rightarrow B) \wedge (B \Rightarrow A)) \Leftrightarrow (A \Leftrightarrow B)$
d) $(A \vee B) \wedge (\neg B \vee C) \Rightarrow (A \vee C)$

**Aufgabe 2.3** Transformieren Sie folgende Formeln in konjunktive Normalform:
a) $A \Leftrightarrow B$
b) $A \wedge B \Leftrightarrow A \vee B$
c) $A \wedge (A \Rightarrow B) \Rightarrow B$

**Aufgabe 2.4** Überprüfen Sie die folgenden Aussagen auf Erfüllbarkeit oder Wahrheit.
a) (Lotto_spielen ∧ 6_Richtige) ⇒ Gewinn
b) (Lotto_spielen ∧ 6_Richtige ∧ (6_Richtige ⇒ Gewinn)) ⇒ Gewinn
c) ¬(¬Benzin_im_Tank ∧ (Benzin_im_Tank ∨ ¬Auto_startet) ⇒ ¬Auto_startet)

**Aufgabe 2.5** ✻✻ Programmieren Sie einen aussagenlogischen Beweiser mit der Wahrheitstafelmethode für Formeln in konjunktiver Normalform in einer Programmiersprache

Ihrer Wahl. Um einen aufwändigen Syntaxcheck der Formel zu vermeiden, können Sie die Klauseln als Listen oder Mengen von Literalen darstellen und die Formel als Liste oder Menge von Klauseln. Das Programm soll angeben, ob die Formel unerfüllbar, erfüllbar, oder wahr ist und die Zahl der unterschiedlichen Belegungen und Modelle ausgeben.

**Aufgabe 2.6**

a) Zeigen Sie, dass der Modus Ponens eine korrekte Inferenzregel ist, indem Sie zeigen, dass $A \wedge (A \Rightarrow B) \models B$.
b) Zeigen Sie, dass die Resolutionsregel (2.1) eine korrekte Inferenzregel ist.

**Aufgabe 2.7** ❄ Zeigen Sie unter Verwendung der Resolutionsregel, dass in der konjunktiven Normalform die leere Klausel der falschen Aussage entspricht.

**Aufgabe 2.8** ❄ Zeigen Sie, dass sich aus einer Wissensbasis, die einen Widerspruch enthält, mit Resolution jede beliebige Klausel „herleiten" lässt.

**Aufgabe 2.9** Formalisieren Sie folgende logischen Funktionen mit den logischen Operatoren und zeigen Sie, dass Ihre Formel korrekt ist. Stellen Sie das Ergebnis in KNF dar.

a) Die XOR-Verknüpfung (exklusives Oder) von zwei Variablen.
b) Die Aussage *mindestens zwei der drei Variablen* $A, B, C$ *sind wahr*.

**Aufgabe 2.10** ❄ Lösen Sie folgenden Kriminalfall mit Hilfe eines Resolutionsbeweises: „Hatte der Verbrecher einen Komplizen, dann ist er mit dem Wagen gekommen. Der Verbrecher hatte keinen Komplizen und hatte den Schlüssel nicht, oder er hatte einen Komplizen und den Schlüssel. Der Verbrecher hatte den Schlüssel. Ist der Verbrecher mit dem Wagen gekommen, oder nicht?"

**Aufgabe 2.11** Beweisen Sie mit Resolution, dass die Formel aus

a) Aufgabe 2.2 d eine Tautologie ist.
b) Aufgabe 2.4 c unerfüllbar ist.

**Aufgabe 2.12** Zeigen Sie folgende für das Arbeiten mit Hornklauseln wichtige Äquivalenzen

a) $(\neg A_1 \vee \ldots \vee \neg A_m \vee B) \equiv A_1 \wedge \ldots \wedge A_m \Rightarrow B$
b) $(\neg A_1 \vee \ldots \vee \neg A_m) \equiv A_1 \wedge \ldots \wedge A_m \Rightarrow f$
c) $A \equiv w \Rightarrow A$

**Aufgabe 2.13** Beweisen Sie mit SLD-Resolution, dass folgende Hornklauselmenge unerfüllbar ist.

$$
\begin{array}{lll}
(A)_1 & (D)_4 & (A \wedge D \Rightarrow G)_7 \\
(B)_2 & (E)_5 & (C \wedge F \wedge E \Rightarrow H)_8 \\
(C)_3 & (A \wedge B \wedge C \Rightarrow F)_6 & (H \Rightarrow f)_9
\end{array}
$$

**Aufgabe 2.14** ➳ In Abschn. 2.6 heißt es „Damit ist klar, dass es vermutlich (modulo dem P/NP-Problem) keinen polynomiellen Algorithmus für 3-Sat und damit auch nicht allgemein geben wird.“ Begründen Sie das „vermutlich“ in diesem Satz näher.

http://www.springer.com/978-3-8348-1677-1

Grundkurs Künstliche Intelligenz
Eine praxisorientierte Einführung
Ertel, W.
2013, XV, 353 S. 202 Abb., 2 Abb. in Farbe., Softcover
ISBN: 978-3-8348-1677-1