
# Einführung in die Logik

Jiří Adámek
mit Anpassungen von Jürgen Koslowski

Sommersemester 2013 + 2014

Version vom 22. Juni 2015

Institut für Theoretische Informatik

# Inhaltsverzeichnis

# 1. Einleitung: Logische Systeme

Formale Logik kann man als den Versuch auffassen, das Denken zu mechanisieren. Als „wahr“ gilt nicht, was aufgrund einer argumentativen Begründung plausibel gemacht, sondern das, was mittels einer systematischen Abfolge mechanischer Arbeitsschritte nachgewiesen werden kann.

Ein logisches System überprüft dabei nicht den Wahrheitsgehalt einzelner Aussagen, sondern es ermöglicht festzustellen, was zusätzlich wahr ist, wenn eine Menge von Aussagen als wahr vorausgesetzt wird. Diese Erweiterung bereits vorhandenen Wissens ist das Forschungsgebiet der Logik.

Es gibt diverse Arten formaler logischer Systeme, die sich zum Teil unterschiedliche Fragestellungen widmen.

In diesem Skript werden nur einige dieser Systeme eingehender betrachtet: klassische Aussagenlogik, Gleichungslogik und Prädikatenlogik. Die große Auswahl existierender logischer Systeme entspricht ähnlich wie die Vielfalt an Programmiersprachen den unterschiedlichen Typen von Anwendungen, die Logik in der Informatik und auch in anderen Gebieten hat.

Die hier vorgestellten Logiken arbeiten mit streng formulierten Sätzen – Formeln –, die aus einem klar definierten Vorrat an Zeichen – Symbolen – zusammengesetzt werden. Allerdings gilt nicht alles was formulierbar ist, als wahr. Deshalb gehört zu jeder Logik immer auch ein Teil, der prüft, welche der korrekt formulierten Sätze auch tatsächlich im System gültig, das heißt „wahr“, sind.

Jedes der behandelten logischen Systeme besitzt die folgenden Grundbausteine:

(a) **Sprache:** Diese besteht aus allen Symbolen, die die spezifische Logik nutzt, um formal korrekte Formeln aufzubauen.

Dazu zählt etwa die Mengen

- ▷ aller *atomaren Aussagen* (oder *Variablen*) „$A_0$“, „$A_1$“, „$A_2$“, …;
- ▷ aller logischen *Junktoren* zum Verknüpfen einer bestimmten Anzahl von Aussagen, so etwa „$\top$“ (**wahr**, 0-stellig), „$\bot$“ (**falsch**, 0-stellig), „$\wedge$“ (**und**, 2-stellig), „$\vee$“ (**oder**, 2-stellig) und „$\neg$“ (**nicht**, 1-stellig) und ggf. weitere, sowie
- ▷ der *Hilfssymbole* wie etwa der Klammern „ ( “ und „ ) “.Kommas „ , “ .

(b) **Syntax:** Dies sind die Regeln, die bestimmen, wie man formal korrekte Formeln aus den Symbolen der Sprache zusammensetzt.

Meist werden diese Regeln wie folgt in zwei Schritten angegeben.

- ▷ Zuerst legt man fest, was die *atomaren*, nicht weiter zerlegbaren Formeln sind. Normalerweise kommen hier die atomaren Aussagen und evtlexFormel!atomare. die 0-stelligen Junktoren $\top$ und $\bot$ vor.

- ▷ Anschließend wird bestimmt, wie aus diesen mit Hilfe der höherstelligen Junktoren *zusammengesetzte* Formeln korrekt aufgebaut werden können.

Dieses zweistufige Verfahren wird als *strukturelle Rekursion* bezeichnet. Mit der zugehörigen *strukturellen Induktion* lassen sich dann Eigenschaften korrekt gebildeter Formeln beweisen.

In der unten eingeführten Aussagenlogik ist etwa die Formel „ $A_0 \vee A_1$ “ formal korrekt, während „ $A_0 \vee A_1 \wedge$ “ nicht formal korrekt ist.

(c) **Semantik:** Diese weist den Formeln ihre „Bedeutung“ zu, was verschiedene Formen annehmen kann. Auf jeden Fall handelt es sich um eine *Abbildung* von Formeln auf geeignete Wahrheitswerte, meist 0 oder 1, s.u.. Ihre Definition bedient sich, wie auch andere Abbildungen auf der Menge aller Formeln, meist der oben erwähnten strukturellen Rekursion, indem zuerst atomaren Formeln eine Semantik zugewiesen wird, bevor man die Semantik zusammengesetzter Formeln aus der Semantik ihrer Bestandteile aufbaut.

Achtung: damit das funktionieren kann, muß die Menge der semantischen Werte eine innere Struktur aufweisen, die es erlaubt, Werte entsprechend den verwendeten logischen Junktoren zu kombinieren!

In der *Aussagenlogik* geht es darum den *Wahrheitsgehalt* von korrekt gebildeten Formeln zu bestimmen, sie können entweder wahr oder falsch sein. Diese Wahrheitswerte werden oft binär mit 1 für „wahr“ und 0 für „falsch“ dargestellt; die Semantik-Abbildung hat als Zielbereich also die Menge $2 = \{0, 1\}$.

Alternativ befasst sich die *intuitionistische Logik* mit der *Beweisbarkeit* von Aussagen, was einen fundamentalen Unterschied darstellt.

(d) **Logische Kalküle:** Diese beinhalten Regeln zur syntaktischen Transformation korrekter Formeln jenseits ihres Aufbaus mittels struktureller Induktion aber *unter Beibehaltung ihrer Semantik*.

Um die Semantik einer Formel $F$ zu bestimmen kann es neben der strukturellen Induktion also andere, im Idealfall effizientere Methoden geben, etwa indem man $F$ als Ergebnis einer zulässigen Transformation einer Formel $E$ von bekannter Semantik darstellt.

Das Hauptinteresse der Logik gilt natürlich der Semantik und den logischen Kalkülen, Syntax als Selbstzweck ist eher uninteressant, aber als Grundlage für Semantik und Kalküle unverzichtbar.

# Teil I

# Aussagenlogik

# 2. Aussagenlogik

Einige Sätze der Umgangssprache haben einen Wahrheitswert „wahr“ oder „falsch“. Z.B. „Prag liegt in Europa“ ist wahr. „Jede Zahl ist größer als 3“ ist falsch. Solche Sätze heißen *Aussagen*. Manche von ihnen sind aus einfacheren Aussagen zusammengesetzt, etwa „Prag liegt genau dann in Europa, wenn jede Zahl größer als 3 ist“. Ziel der Aussagenlogik ist es, orientiert zunächst an der Alltagslogik und später an mathematischen Schlußweisen, eine Semantik und ein Kalkül für derartige Aussagen zu finden, wobei die Semantik atomarer Aussagen entweder von außen vorgegeben ist oder frei gewählt werden kann. Z.B. ist

$$\text{wenn } x = 2 \text{ dann gilt } y = 3$$

eine zusammengesetzte Aussage, die die Form einer sogenannten *Implikation* hat. Auch ohne zu wissen, ob die atomaren Aussagen

$$A: \quad x = 2 \qquad \text{und} \qquad B: \quad y = 3$$

wahr oder falsch sind, wird es möglich sein

$$A \Rightarrow B$$

zumindest in solchen Situationen als wahr zu bewerten, in denen $x \neq 2$ gilt.

## 2.i Sprache und Syntax der Aussagenlogik

**Definition 2.1.** Die **Sprache der Aussagenlogik** besteht aus

(a) einer abzählbaren Menge von **atomaren Aussagen** $A_0$, $A_1$, $A_2$, ...

(b) Operatoren vorgegebener Stelligkeit auf der Menge der atomaren Aussagen, den sogenanten **Junktoren**

- $\bot$ **Absurdität**, „bottom“, 0-stellig
- $\neg$ **Negation**, „nicht“, 1-stellig
- $\wedge$ **Konjunktion**, „und“, 2-stellig
- $\vee$ **Disjunktion**, „oder“, 2-stellig

(c) und Hilfssymbolen

- ( ) Klammern

Vereinfachend schreiben wir oft $A,\ B,\ C, \ldots$ statt $A_0,\ A_1,\ A_2, \ldots$ um atomare Aussagen zu bezeichnen.

**Bemerkung 2.2.** Die Namen und Stelligkeiten der Junktoren sind natürlich im Hinblick auf die intendierte Semantik gewählt, die im folgenden Abschnitt beschrieben wird. Werden die Junktoren nach Stelligkeit sortiert, so bezeichnet man die resultierende Folge $\Sigma$ von Mengen $\Sigma_0 = \{\bot\}$, $\Sigma_1 = \{\neg\}$, $\Sigma_2 = \{\wedge, \vee\}$ und $\Sigma_n = \varnothing$ für $n > 2$ auch als *Signatur*.

**Definition 2.3.** [Syntax der Aussagenlogik, Infix-Version]
Die Menge der **Formeln** der Aussagenlogik (auch **Aussagen** genannt) ist die kleinste Menge, so dass

(a) $\bot$ und jede atomare Aussage eine Formel ist;

(b) falls $F$ und $G$ Formeln sind, dann gilt dies auch für

$$(\neg F) \quad , \quad (F \wedge G) \quad \text{sowie} \quad (F \vee G)$$

**Bemerkung 2.4.** Aus mathematischer Sicht ist die Menge der Formeln die *freie* $\Sigma$-*Algebra* über der Menge der atomaren Aussagen. Dabei verweist der Begriff „frei“ darauf, dass abgesehen von syntaktischer Gleichheit (von Zeichenketten) keine weiteren Gleichungen gelten.

**Bemerkung 2.5.** Die Verwendung der *Infix-Schreibweise* für die binären Junktoren $\wedge$ und $\vee$ erzwingt die Verwendung von Klammern, ähnlich wie in der Arithmetik. Wenn man Klammern einsparen möchte, kann man sich entweder mit sogenannten *Bindungskonventionen* behelfen, siehe Abschnitt 2.iv, oder man muß die Infix-Notation zugunsten von Präfix- oder *polnischer* Notation, besser noch Postfix- oder *umgekehrt polnischer* Notation (RPN) aufgeben. Erstere wurde 1924 vom polnischen Logiker Jan Łukasiewicz eingeführt, wird aber heute nicht mehr häufig verwendet, letztere wurde Mitte der 1950'er Jahre von Burks, Warren und Wright [BWW54] sowie vom australischen Philosophen und Informatiker Charles Hamblin vorgeschlagen und weiterentwickelt [Ham57]. In diesem Zusammenhang dürfen auch die Miterfinder des Kellerprinzips (oder Stacks), Friedrich Ludwig Bauer und Klaus Samelson nicht unerwähnt bleiben [BS57]. RPN findet sich beispielsweise in wissenschaftlichen Taschenrechnern von Hewlett Packard, in der Programmiersprache Forth und in der Seitenbeschreibungssprache PostScript. Nur Teil (b) der obigen Definition wäre betroffen und nimmt in RPN folgende Form an:

(b)' falls $F$ und $G$ Formeln sind, dann gilt dies auch für

$$F\neg \quad , \quad FG\wedge \quad \text{sowie} \quad FG\vee$$

**Beispiel 2.6.** Für atomare Aussagen $A$ und $B$ sind etwa $((A \vee B) \wedge (\neg B)))$ und $(\neg(\neg(A \vee B)))$ Infix-Formeln, deren RPN-Form etwas gewöhnungsbedürftig aussieht: $AB \vee B\neg\wedge$ bzw. $AB \vee \neg\neg$.

Wir kommen nun zum Betriff der *Länge* einer Formel. Dabei stellt sich die Frage, ob die Länge einer Formel von der Art ihrer Präsentation abhängen soll, genauer, ob die Klammern als Hilfssymbole bei Verwendung der Infix-Notation mitgezählt werden sollen, oder nicht. Speziell auch im Hinblick auf die in Kürze einzuführenden Regeln zur Klammerersparnis erscheint das Mitzählen der Klammern problematisch, das funktioniert nur bei vollständig geklammerten Ausdrücken. Die logische Signifikanz der Länge einer Formel mit allen Klammern liegt zwar über derjenigen der Farbe, in der sie aufgeschrieben wurde, erschwert aber immer noch die Kommunikation mit Logikern aus anderen Kulturkreisen. Um Mißverständnisse zu vermeiden werden wir zwei Typen von Längen definieren:

**Definition 2.7.** Die **inheränte Länge** $|F|$ wie auch die **Präsentationslänge** $\|F\|$ einer syntaktisch korrekten Formel $F$ definieren wir mit Hilfe struktureller Induktion, also über den Aufbau von $F$ wie folgt:

- ▷ Falls $F$ mit $\bot$ übereinstimmt oder atomar ist, gelte

$$|F| := 1 =: \|F\|$$

- ▷ Falls $F$ die Form $(\neg G)$ hat und $|G|$, bzw. $\|F\|$, bekannt ist, setze

$$|(\neg G)| := |G| + 1 \qquad \text{bzw.} \qquad \|(\neg G)\| := \|G\| + 3$$

- ▷ Falls $F$ die Form $(G \wedge H)$ oder $(G \vee H)$ hat und $|G|$ sowie $|H|$, bzw. $\|G\|$ sowie $\|H\|$, bekannt sind, setze

$$|(G \wedge H)| := |G| + |H| + 1 \qquad \text{bzw.} \qquad \|(G \wedge H)\| := \|G\| + \|H\| + 3$$

  und analog für die Disjunktion.

## 2.ii Semantik der Aussagenlogik

Die Semantik der Aussagenlogik untersucht, wie aus Belegungen der atomaren Aussagen die Wahrheitswert aller korrekten Formeln mechanisch abgeleitet werden können. Interpretiert man 1 als „wahr“ und 0 als „falsch“, so geht es darum beliebige Funktionen von der Menge aller atomaren Aussagen in die Menge $2 = \{0, 1\}$ auf die Menge aller korrekten Formeln sinnvoll fortzusetzen.

Gegeben sei eine Formel $F$. Ihre atomare Aussagen mögen in einer endliche Untermenge $\mathcal{M}$ der Menge von $\{A_0, A_1, A_2 \ldots\}$ aller atomaren Aussagen liegen. Falls wir die Wahrheitswerte der Elemente von $M$ kennen, soll der Wahrheitswert der ganzen Formel $F$ bestimmt werden können. Während für einzelne Formeln die Beschränkung auf endliche Teilmengen der Variablenmenge sinnvoll erscheint, sind später, beim Kompaktheitssatz, nicht notwendig endliche Mengen von Formeln zu betrachten, in denen auch unendlich viele Variablen auftreten können. Daher:

**Definition 2.8.** Eine **Belegung** ist eine Abbildung $\alpha$ von einer (nicht notwendig endlichen) Menge $\mathcal{M}$ atomarer Aussagen in die Menge $\{0, 1\}$. Kurz:

$$\alpha : \mathcal{M} \longrightarrow \{0, 1\} \quad \text{für} \quad \mathcal{M} \subseteq \{A_0, A_1, A_2, \ldots\}$$

**Beispiel 2.9.** Falls $\mathcal{M} = \{A, B\}$ mit der Belegung $\alpha(A) = 1$ und $\alpha(B) = 0$ ist, legt die intuitive Bedeutung der Wörter „oder“ und „nicht“ nahe, dass $(A \vee B)$ mit 1 und $(\neg(A \vee B))$ mit 0 bewertet werden sollten. Dies gilt es nun zu formalisieren.

**Definition 2.10.** Für jede Menge $\mathcal{M} \subseteq \{A_0, A_1, A_2, \ldots\}$, jede Formel $F$, die syntaktisch mit Hilfe der atomaren Aussagen aus $\mathcal{M}$ gebildet werden kann, und jede Belegung $\alpha : \mathcal{M} \longrightarrow \{0, 1\}$ definieren wir den **Wert** $\widehat{\alpha}(F)$ der Formel $F$ **unter der Belegung** $\alpha$ mittels struktureller Induktion, d.h., Induktion über den Aufbau der Formel $F$:

$|F| = 1$ Entweder stimmt $F$ mit der Absurdität überein, dann setzen wir

$$\widehat{\alpha}(\bot) := 0$$

Oder $F$ ist eine atomare Formel aus $\mathcal{M}$ und wir setzten

$$\widehat{\alpha}(F) := \alpha(F)$$

$|F| > 1$ Hat $F$ eine Länge $> 1$, so hat $F$ die Form $(\neg G)$ oder $(G \wedge H)$ oder $(G \vee H)$, wobei die Teilformeln $G$ und $H$ echt kürzer sind. Für sie ist $\widehat{\alpha}$ durch die Induktion also bereits definiert. Entsprechend definieren wir $\widehat{\alpha}(F)$ durch

$$\widehat{\alpha}(\neg G) := \begin{cases} 1 & \text{falls } \widehat{\alpha}(G) = 0 \\ 0 & \text{sonst} \end{cases}$$

$$\widehat{\alpha}(G \wedge H) := \begin{cases} 1 & \text{falls } \widehat{\alpha}(G) = 1 \text{ und } \widehat{\alpha}(H) = 1 \\ 0 & \text{sonst} \end{cases}$$

$$\widehat{\alpha}(G \vee H) := \begin{cases} 1 & \text{falls } \widehat{\alpha}(G) = 1 \text{ oder } \widehat{\alpha}(H) = 1 \\ 0 & \text{sonst} \end{cases}$$

**Bemerkung 2.11.** Warum wird $\widehat{\alpha}$ gerade so definiert? Und wie verhält es sich mit der Anmerkung auf Seite 5, dass die Menge der semantischen Werte, hier $2 = \{0, 1\}$, eine innere Struktur benötigt, die das Nachspielen der logischen Junktoren erlaubt?

- ▷ Die Negation soll offenbar die Wahrheitswerte in ihr Komplement umwandeln, was ausgedrückt werden kann durch

$$\widehat{\alpha}(\neg G) = 1 - \widehat{\alpha}(G)$$

- ▷ Damit eine Konjunktion wahr ist, darf keine der Teilformeln einen geringeren Wahrheitswert als 1 annehmen, anders ausgedrückt

$$\widehat{\alpha}(G \wedge H) = \inf\{\widehat{\alpha}(G), \widehat{\alpha}(H)\}$$

Da die Infimum-Operation auf $2 = \{0, 1\}$, speziell in Infix-Schreibweise, auch gern mit $\wedge$ bezeichnet wird, können wir schreiben

$$\widehat{\alpha}(G \wedge H) = \widehat{\alpha}(G) \wedge \widehat{\alpha}(H)$$

was eine Interpretation der Konjunktion als verallgemeinerter Infimum-Operation nahelegt. (Wir bevorzugen das Infimum gegenüber dem Minimum, weil ersteres nicht auf linear geordnete Mengen beschränkt ist.)

▷ Bei der Interpretation von „oder“ gibt es unterschiedliche Ansätze. In der Mathematik und in der theoretischen Informatik meint „oder“ immer die nicht-exklusive Form, die mit $\vee$ bezeichnet wird. Nach dieser Präzisierung ist klar, dass eine Disjunktion schon wahr ist, wenn mindestens eine der Teilformeln wahr ist, anders ausgedrückt

$$\widehat{\alpha}(G \vee H) = \sup\{\widehat{\alpha}(G), \widehat{\alpha}(H)\} = \widehat{\alpha}(G) \vee \widehat{\alpha}(H)$$

Diese Formeln legen es nahe, die Junktoren $\wedge$ und $\vee$ in einem noch zu präzisierenden Sinn als „dual“ zu verstehen, vergleiche die Abschnitte ???.

Das „exklusive Oder“, häufig mit $\oplus$ bezeichnet, läßt sich auch modellieren, selbst wenn wir es nicht als Teil der Aussagenlogik eingeführt haben, vergleiche Abschnitt 2.iii.

**Beispiel 2.12.** Die Semantik der Formel

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

ist bei jeder Belegung $\alpha$ der Atome $A, B, C$ definiert. Die Formel ist wahr, falls $A$ wahr ist oder $(B \wedge (\neg C))$ wahr ist. Letzteres erfordert sowohl die Wahrheit von $B$ als auch die von $\neg C$.

**Definition 2.13.** Eine Belegung $\alpha : \mathcal{M} \longrightarrow \{0, 1\}$ heißt **passend** für eine Formel $F$ falls all deren atomaren Aussagen in $\mathcal{M}$ liegen.

Jeder Formel werden genau durch die passenden Belegungen Werte 0 oder 1 zugeordnet, andere Belegungen sind dafür uninteressant.

## 2.iii Abgeleitete Symbole

Andere Autoren zählen die zwei-stelligen Junktoren $\Rightarrow$ (Implikation) und $\Leftrightarrow$ (Äquivalenz) sowie den null-stelligen Junktor $\top$ (Tautologie) zu den Grundbausteinen der Aussagenlogik. Wir hingegen werden sie, ebenso wie das exklusive Oder $\oplus$, als abgeleitete Symbole verwenden, definiert als Abkürzungen längerer Formeln, die nur $\neg$, $\wedge$ oder $\vee$ verwenden. Das ergibt sich aus der Analyse der *intendierten klassischen Semantik* für diese Junktoren. Dazu erweitern wir zunächst die Syntax um diese zwei-stelligen Junktoren.

▷ Die Tautologie $\top$ als Komplement der Absurdität $\bot$ ist unproblematisch.

▷ Die Implikation $(G \Rightarrow H)$ soll als Abstraktion von „wenn $G$, dann $H$“ verstanden werden. Das bedeutet rein formal, dass der Wahrheitswert von $G$ kleiner oder gleich dem Wahrheitswert von $H$ sein soll. Achtung: inhaltlich braucht kein Zusammenhang zwischen den Aussagen $G$ und $H$ zu bestehen, was die Akzeptanz der obigen Interpretation erschweren mag. Wem dies nicht gefällt, muß eine bessere Semantik suchen, Kandidaten dafür gibt es.

- ▷ Die Äquivalenz $(G \Leftrightarrow H)$ soll „$G$ genau dann wenn $H$“ ausdrücken und kann als Konjunktion zweier entgegengesetzter Implikationen $(G \Rightarrow H)$ und $(H \Rightarrow G)$ aufgefasst werden. Mit anderen Worten, die Wahrheitswerte von $G$ und $H$ stimmen überein.

- ▷ Schließlich soll $(G \oplus H)$ genau dann wahr sein, wenn genau eine der Teilformeln wahr ist, mit anderen Worten, wenn beide Teilformeln verschiedene Wahrheitswerte haben.

Diese Vorgaben lassen sich wie folgt umsetzen:

**Definition 2.14.** Für jede, bzw. jede für $G$ und $H$ passende Belegung setzen wir

$$\begin{aligned}
\widehat{\alpha}(\top) &:= 1 \\
\widehat{\alpha}(G \Rightarrow H) &:= \begin{cases} 1 & \text{falls } \widehat{\alpha}(G) \leq \widehat{\alpha}(H) \\ 0 & \text{sonst} \end{cases} \\
\widehat{\alpha}(G \Leftrightarrow H) &:= \begin{cases} 1 & \text{falls } \widehat{\alpha}(G) = \widehat{\alpha}(H) \\ 0 & \text{sonst} \end{cases} \\
\widehat{\alpha}(G \oplus H) &:= \begin{cases} 1 & \text{falls } \widehat{\alpha}(G) \neq \widehat{\alpha}(H) \\ 0 & \text{sonst} \end{cases}
\end{aligned}$$

Wenn es gelingt, die spezifizierten Relationen zwischen den Wahrheitswerten in eine oder mehrere Gleichungen umzusetzen, die die Konstanten 0 oder 1 verwenden, können wir die neu eingeführten Junktoren durch die ursprünglichen ausdrücken.

**Proposition 2.15.** *In der Aussagenlogik mit erweiterter Syntax sind folgenden Formeln immer semantisch gleichwertig, d.h., haben bzgl. jeder passenden Belegung denselben Wahrheitswert:*

(a) $\top$ *und* $\neg\bot$*;*

(b) $(G \Rightarrow H)$ *und* $((\neg G) \vee H)$*;*

(c) $(G \Leftrightarrow H)$ *und* $((G \Rightarrow H) \wedge (H \Rightarrow G))$ *und* $((G \wedge H) \vee ((\neg G) \wedge (\neg H)))$*;*

(d) $(G \oplus H)$ *und* $(\neg(G \Leftrightarrow H))$ *und* $((G \wedge (\neg H)) \vee ((\neg G) \wedge H))$.

*Folglich können wir die Junktoren $\top$, $\Rightarrow$, $\Leftrightarrow$ und $\oplus$ ohne Einbuße an Ausdrucksfähigkeit wieder aus der Syntax der Aussagenlogik entfernen.*

*Beweis.* Im Folgenden ist $\alpha$ eine beliebige passende Belegung.

(a) Klar wegen $1 = 1 - 0$.

(b)
$$\begin{aligned}
\widehat{\alpha}(G) \leq \widehat{\alpha}(H) \quad &\text{gdw} \quad \widehat{\alpha}(G) = 0 \quad \text{oder} \quad \widehat{\alpha}(H) = 1 \\
&\text{gdw} \quad \widehat{\alpha}(\neg G) = 1 \quad \text{oder} \quad \widehat{\alpha}(H) = 1
\end{aligned}$$

(c) Die erste Variante ergibt sich aus

$$\widehat{\alpha}(G) = \widehat{\alpha}(H) \quad \text{gdw} \quad \widehat{\alpha}(G) \leq \widehat{\alpha}(H) \quad \text{und} \quad \widehat{\alpha}(H) \leq \widehat{\alpha}(G)$$

Alternativ erhalten wir

$$\widehat{\alpha}(G) = \widehat{\alpha}(H) \quad \text{gdw} \quad \widehat{\alpha}(G) = 1 = \widehat{\alpha}(H) \quad \text{oder} \quad \widehat{\alpha}(G) = 0 = \widehat{\alpha}(H)$$

(d) Die erste Variante ist unmittelbar klar. Alternativ erhalten wir

$$\begin{aligned} \widehat{\alpha}(G) \neq \widehat{\alpha}(H) \quad &\text{gdw} \quad \widehat{\alpha}(G) = 1 = \widehat{\alpha}(\neg H) \quad \text{oder} \quad \widehat{\alpha}(G) = 0 = \widehat{\alpha}(\neg H) \\ &\text{gdw} \quad \widehat{\alpha}(G) = 1 = \widehat{\alpha}(\neg H) \quad \text{oder} \quad \widehat{\alpha}(\neg G) = 1 = \widehat{\alpha}(H) \end{aligned}$$ □

In Abschnitt 2.vii werden wir vermöge semantischer Gleichwertigkeit eine nützlichere Relation als die syntaktischen Gleichheit zwischen Formeln einzuführen.

## 2.iv Bindungskonventionen

Wie gerade gesehen, können selbst Formeln recht simpler Semantik schnell eine komplizierte und unübersichtliche Syntax annehmen. Um unnötige Klammern zu vermeiden, d.h., zur Vereinfachung der Syntax, vereinbaren wir die folgende

**Notationelle Konvention 2.16.**

(a) Die äußeren Klammern um zusammengesetzte Aussagen können weggelassen werden

(b) $\neg$ bindet stärker als alle zwei-stelligen Junktoren ($\wedge$, $\vee$, $\Rightarrow$, $\Leftrightarrow$, $\oplus$).

(c) $\wedge$ und $\vee$, die „alten“ zwei-stelligen Junktoren, binden gleich stark, aber stärker als die abgeleiteten Junktoren $\Rightarrow$, $\Leftrightarrow$ und $\oplus$.

**Beispiel 2.17.** $\neg A \Rightarrow B \vee \neg C$ wird dadurch eine korrekte Formel. Es ist die Abkürzung für die „offizielle“ Langfassung $\big((\neg A) \Rightarrow (B \vee (\neg C))\big)$. Auch die durch $G \oplus H$ abgekürzte Formel vereinfacht sich zu $\neg((H \vee \neg G) \wedge (G \vee \neg H))$.

Dennoch können nicht alle Klammern entfernt werden. Ausdrücke wie $A \vee B \wedge C$ oder $A \wedge B \wedge C$ oder auch $A \Rightarrow B \Rightarrow C$ lassen sich nicht aufgrund der obigen Bindungskonventionen aus einer Langfassung herleiten. Mit Hilfe späterer Ergebnisse kann man $A \wedge B \wedge C$ interpretieren, während $A \Rightarrow B \Rightarrow C$ eine zusätzliche Konvention erfordert. Aber $A \vee B \wedge C$ wird sich als uninterpretierbar herausstellen.

## 2.v Wahrheitstafeln

Die Semantik jeder Formel $F$, die atomare Aussagen aus $\mathcal{M} := \{A_0, A_1, \ldots, A_{n-1}\}$ benutzt, kann man als **Wahrheitstafel** darstellen: jede Zeile entspricht einer der $2^n$

Belegungen $\alpha : \mathcal{M} \longrightarrow \{0,1\}$. Diese sind gemäß der Binärdarstellung der Zahlen von 0 bis $2^n - 1$ angeordnet:

| $A_0$ | $A_1$ | $\ldots$ | $A_{n-1}$ | $F$ |
|---|---|---|---|---|
| 0 | 0 | $\ldots$ | 0 | $\widehat{\alpha}(F)$ für $\alpha(A_i) = 0$ für alle $i < n$ |
| 0 | 0 | $\ldots$ | 1 | $\widehat{\alpha}(F)$ für $\alpha(A_n) = 1,$ sonstige $\alpha(A_i) = 0$ |
| $\vdots$ | $\vdots$ | $\ddots$ | $\vdots$ | $\vdots$ |
| 1 | 1 | $\ldots$ | 1 | $\widehat{\alpha}(F)$ für $\alpha(A_i) = 1$ für alle $i < n$ |

Letztendlich lassen sich alle Wahrheitstafeln auf diejenigen für Negation, Konjunktion und Disjunktion zurückführen, die wir zur Referenz auflisten:

$\neg$ :

| $A$ | $\neg A$ |
|---|---|
| 0 | 1 |
| 1 | 0 |

$\wedge$ :

| $A$ | $B$ | $A \wedge B$ |
|---|---|---|
| 0 | 0 | 0 |
| 0 | 1 | 0 |
| 1 | 0 | 0 |
| 1 | 1 | 1 |

$\vee$ :

| $A$ | $B$ | $A \vee B$ |
|---|---|---|
| 0 | 0 | 0 |
| 0 | 1 | 1 |
| 1 | 0 | 1 |
| 1 | 1 | 1 |

Der Übersichtlichkeit halber und zu Vergleichszwecken ist es auch zulässig, Hilfsspalten mit Teilergebnissen einzufügen. Diese können auch semantischen Spezifikationen umsetzen. Für die Implikation und die Äquivalenz erhalten wir nun

$\Rightarrow$:

| $A$ | $B$ | $\neg A$ | $\neg A \vee B$ | $\alpha(A) \leq \alpha(B)$ |
|---|---|---|---|---|
| 0 | 0 | 1 | 1 | 1 |
| 0 | 1 | 1 | 1 | 1 |
| 1 | 0 | 0 | 0 | 0 |
| 1 | 1 | 0 | 1 | 1 |

$\Leftrightarrow$:

| $A$ | $B$ | $A \Rightarrow B$ | $B \Rightarrow A$ | $(A \Rightarrow B) \wedge (B \Rightarrow A)$ | $\alpha(A) = \alpha(B)$ |
|---|---|---|---|---|---|
| 0 | 0 | 1 | 1 | 1 | 1 |
| 0 | 1 | 1 | 0 | 0 | 0 |
| 1 | 0 | 0 | 1 | 0 | 0 |
| 1 | 1 | 1 | 1 | 1 | 1 |

während das exklusive Oder folgende Tabelle hat:

$\oplus$ :

| $A$ | $B$ | $\neg(A \Leftrightarrow B)$ | $\alpha(A) \neq \alpha(B)$ |
|---|---|---|---|
| 0 | 0 | 0 | 0 |
| 0 | 1 | 1 | 1 |
| 1 | 0 | 1 | 1 |
| 1 | 1 | 0 | 0 |

**Beispiel 2.18.** Wahrheitstafeln ermöglichen eine Analyse der jeweils dargestellten For-

mel, etwa von $F := C \Rightarrow A \vee (B \wedge A)$:

| $A$ | $B$ | $C$ | $F$ |
|---|---|---|---|
| 0 | 0 | 0 | 1 |
| 0 | 0 | 1 | 0 |
| 0 | 1 | 0 | 1 |
| 0 | 1 | 1 | 0 |
| 1 | 0 | 0 | 1 |
| 1 | 0 | 1 | 1 |
| 1 | 1 | 0 | 1 |
| 1 | 1 | 1 | 1 |

Gemäß der Tafel ist die Formel fast immer wahr, außer in zwei Fällen:

- $\alpha(A) = 0, \alpha(B) = 0$ und $\alpha(C) = 1$
- $\alpha(A) = 0, \alpha(B) = 1$ und $\alpha(C) = 1$

Daraus ist zu erkennen, dass der Wert von $F$ *unabhängig* von dem Wert von $B$, also von $\alpha(B)$, ist. Wir können also die $B$-Spalte aus der Tabelle entfernen:

| $A$ | $C$ | $F$ |
|---|---|---|
| 0 | 0 | 1 |
| 0 | 1 | 0 |
| 1 | 0 | 1 |
| 1 | 1 | 1 |

Damit ist $\widehat{\alpha}(F) = 1$, falls $\alpha(A) = 1$ oder $\alpha(C) = 0$ gilt. Damit können wir kürzere Formeln mit dieselbe Semantik angeben

$$G := A \vee \neg C \qquad \text{oder noch kürzer} \qquad H := C \Rightarrow A$$

## 2.vi Erfüllbarkeit und Tautologien

Grundsätzlich kann man Formeln danach klassifizieren, ob überhaupt passende Belegungen existieren, die sie wahr bzw. falsch machen.

**Definition 2.19.** Eine Formel $F$ heißt

(a) **erfüllbar**, falls sie unter wenigstens einer Belegung wahr ist;

(b) **Tautologie**, falls sie für jede passende Belegung wahr ist (in Anlehnung an den 0-stelligen Junktor $\top$).

Hinsichtlich der Wahrheitstabelle bedeutet dies, dass die rechte Spalte mindestens eine Eins enthält, bzw. nur aus Einsen besteht. Offenbar enthält die Wahrheitstabelle der Negation jeder Tautologie nur Nullen in der rechten Spalte, folglich sind Negationen von Tautologien nicht erfüllbar. Das läßt sich auch positiv formulieren:

**Satz 2.20.** *Eine Formel $F$ ist genau dann erfüllbar, wenn $\neg F$ keine Tautologie ist.*

*Beweis.* Wenn $F$ erfüllbar ist, gibt es eine passende Belegung $\alpha$ mit $\widehat{\alpha}(F) = 1$ und folglich $\widehat{\alpha}(\neg F) = 1 - \widehat{\alpha}(F) = 0$. Daher kann $\neg F$ keine Tautologie sein. Ist umgekehrt $\neg F$ keine Tautologie, so gibt es eine passende Belegung $\alpha$ mit $\widehat{\alpha}(\neg F) = 1 - \widehat{\alpha}(F) = 0$. Daraus folgt $\widehat{\alpha}(F) = 1$, was $F$ erfüllbar macht. □

**Beispiel 2.21.** Die Formel $A \Rightarrow B$ ist erfüllbar, da sie etwa dann wahr wird, wenn man $B$ mit 1 belegt. Das gilt sogar im Fall $B = \neg A$, was zunächst irritierend sein mag, aber der sehr simplen Semantik der Aussagenlogik geschuldet ist, die nur die Wahrheitswerte der Teilformeln betrachtet.

**Beispiel 2.22.** Die Formel $F \vee \neg F$ ist eine Tautologie, denn $F$ und $\neg F$ nehmen aufgrund der Semantik von $\neg$ immer verschiedene Werte an. Damit tritt immer genau einmal der Wert 1 auf, so dass aufgrund der Semantik von $\vee$ das Infimum den Wert 1 annimmt.

Auch $(\neg F \Rightarrow \neg G) \Rightarrow (G \Rightarrow F)$ ist eine Tautologie, wie man anhand der Wahrheitstabelle unschwer feststellt. Sie bildet die Grundlage für das Beweisverfahren durch *Contraposition*. Wenn einem der Berweis von $F \Rightarrow G$ nicht gelingt, kann man sich stattdessen an $\neg G \Rightarrow \neg F$ versuchen. Im Erfolgsfall folgt daraus die ursprüngliche Behauptung.

**Beispiel 2.23.** Eine evtl. überraschende Tautologie ist

$$(F \Rightarrow G) \vee (G \Rightarrow F)$$

In der Tat, bei jeder passenden Belegung $\alpha$ gilt entweder $\widehat{\alpha}(F) = 0$, woraus die Wahrheit von $F \Rightarrow G$ folgt, oder $\widehat{\alpha}(F) = 1$, woraus die Wahrheit von $G \Rightarrow F$ folgt. Und für die Wahrheit einer Disjunktion genügt die Wahrheit einer Teilformel.

Die obige Formel besagt nur, dass der Wertebereich $2 = \{0, 1\}$ der Semantik der Aussagenlogik total geordnet ist, also neben $0 \leq 0$ und $1 \leq 1$ auch genau eine der Bedingungen $0 \leq 1$ oder $1 \leq 0$ erfüllt sein muß (natürlich die erste). Damit ergibt sich für die beiden Wahrheitswerte $\widehat{\alpha}(F)$ und $\widehat{\alpha}(G)$ trivialerweise (hier ist das Wort tatsächlich mal erlaubt)

$$\widehat{\alpha}(F) \leq \widehat{\alpha}(G) \qquad \text{oder} \qquad \widehat{\alpha}(G) \leq \widehat{\alpha}(F)$$

Es wäre irreführend, die obige Aussage dahingehend zu interpretieren, dass für zwei beliebige Aussagen $F$ und $G$ immer mindestens eine aus der anderen *hergeleitet werden kann*. Das setzt eine andere Semantik voraus, die auf Beweisbarkeit statt auf Wahrheit basiert. Dort kann die obige Formel auch formuliert werden, ist aber keine Tautologie mehr. Das unterstreicht, dass der Begriff „Tautologie“ nur relativ zu einer gegebenen Semantik Sinn ergeben kann.

Im „realen Leben“, etwa in der Bahnindustrie, werden zur Steuerung von Weichenanlagen Formeln mit bis zu 300.000 atomaren Aussagen formuliert und deren Erfüllbarkeit getestet. Dieses Problem wird in der Theoretischen Informatik intensiv erforscht und heißt dort **SAT** (von englisch „satisfiable“). Im übernächsten Kapitel entwickeln wir einen Algorithmus dafür.

## 2.vii Äquivalenz von Formeln

Nachdem wir schon in Proposition 2.15 das Konzept der semantischen Gleichwertigkeit verwendet hatten, um die Einführung von abgeleiteten Symbolen zur Abkürzung komplexer Formeln als unbedenklich nachzuweisen, wollen wir es nun auch auf Formeln in der Originalsyntax anwenden. Zunächst brauchen wir aber einen einprägsameren bzw. kürzeren Namen:

**Definition 2.24.** Zwei Formeln $F$ und $G$ der Aussagenlogik heißen **äquivalent**, wenn sie semantisch gleichwertig sind, d.h. für jede Belegung $\alpha : \mathcal{M} \longrightarrow \{0,1\}$ die zu $F$ und $G$ passend ist, gilt

$$\widehat{\alpha}(F) = \widehat{\alpha}(G)$$

Wir führen dafür die Notation $F \equiv G$ ein.

Bitte beachten: $\equiv$ ist *kein* Junktor und $F \equiv G$ ist *keine* Formel der Aussagenlogik, sondern beide gehören zur *Metasprache*, mit der wir über Aussagenlogik sprechen.

**Beispiel 2.25.** $C \Rightarrow A \vee (B \wedge C)$ ist äquivalent zu $(A \vee \neg C)$. Dies haben wir in Beispiel 2.18 durch Analyse der Wahrheitstafel festgestellt.

Der Begriff „äquivalent“ läßt sich in zweifacher Hinsicht rechtfertigen:

**Satz 2.26.** *Zwei aussagenlogische Formeln $F$ und $G$ sind genau dann äquivalent, wenn $F \Leftrightarrow G$ eine Tautologie ist.*

*Beweis.* Dies ist nur eine Umformulierung der semantischen Interpretation von $F \Leftrightarrow G$: die Wahrheitswerte von $F$ und $G$ stimmen für jede passende Belegung überein. □

Dieses Ergebnis rechtfertigt es, die Relation $\equiv$ auf Formeln als „Externalisierung“ des Junktors $\Leftrightarrow$ aufzufassen; diesen könnte man präziser mit „interner Äquivalenz“ bezeichnen.

**Satz 2.27.** *Die Relation $\equiv$ ist eine Äquivalenzrelation auf der Menge aller aussagenlogischen Formeln, d.h., sie ist reflexiv, transitiv und symmetrisch.*

*Beweis.* Alle drei Eigenschaften lassen sich auf die entsprechenden Eigenschafen der Gleichheit „=“ im semantischen Wertebereich $2 = \{0,1\}$ zurückführen, die von allen passenden Belegungen „reflektiert“ werden. □

Es gilt sogar noch mehr: $\equiv$ ist zudem mit den Junktoren „verträglich“, was $\equiv$ zu einer sogenannten *Kongruenzrelation* macht:

**Satz 2.28.** *Falls $F \equiv F'$, dann gilt*

$$\neg F \equiv \neg F' \quad \textit{und} \quad F \vee G \equiv F' \vee G \quad \textit{sowie} \quad F \wedge G \equiv F' \wedge G \quad \textit{für alle Formeln } G$$

*Beweis.* Ist $\alpha$ eine für $F$ und $F'$ passende Belegung, dann gilt nach Voraussetzung $\widehat{\alpha}(F) = \widehat{\alpha}(F')$, und folglich

$$\widehat{\alpha}(\neg F) = 1 - \widehat{\alpha}(F) = 1 - \widehat{\alpha}(F') = \widehat{\alpha}(\neg F')$$

Falls $\alpha$ zudem für $G$ passend ist, erhalten wir

$$\widehat{\alpha}(F) \wedge \widehat{\alpha}(G) = \widehat{\alpha}(F') \wedge \widehat{\alpha}(G)$$

und analog für die Disjunktion. □

**Folgerung 2.29.** *Aus $F \equiv F'$ und $G \equiv G'$ folgt*

$$F \wedge G \equiv F' \wedge G' \quad , \quad F \vee G \equiv F' \vee G' \quad \textit{sowie} \quad F \Rightarrow G \equiv F' \Rightarrow G' \qquad \square$$

Wir hatten festgestellt, dass die Menge der korrekten aussagenlogischen Formeln „frei“ erzeugt worden war, d.h., Gleichheit wirklich syntaktische Gleichheit von Zeichenketten bedeutet. Das ist nicht besonders interessant. Die eben eingeführte Äquivalenz $\equiv$ ist eine „bessere Gleichheit“ auf der Menge der Formeln, da sie semantische Aspekte mit einbezieht. Wir sind letztlich nur an Formeln „modulo Äquivalenz“ interessiert.

# 3. Eigenschaften von Formeln

Neben den Eigenschaften der Gleichheit auf dem Semantischen Wertebereich $2 = \{0, 1\}$ für die klassische Aussagenlogik lassen sich auch Eigenschaften der dortigen Operationen in die Menge der Formeln modulo $\equiv$ reflektieren, so dass diese Quotienten- oder Faktormenge schließlich ebenfalls die Struktur einer Boole'schen Algebra annimmt.

## 3.i Eigenschaften der Negation

**Satz 3.1.** *Modulo $\equiv$ ist die Negation **selbstinvers**, d.h., für jede Formel $F$ der klassischen Aussagenlogik gilt $\neg\neg F \equiv F$.*

*Beweis.* Jede passende Belegung $\alpha$ erfüllt

$$\widehat{\alpha}(\neg\neg F) = 1 - \widehat{\alpha}(\neg F) = 1 - (1 - \widehat{\alpha}(F)) = \widehat{\alpha}(F) \qquad \square$$

Dieses Ergebnis ist die Basis für den *Beweis durch Widerspruch*: statt $F$ zu beweisen kann man $\neg F$ widerlegen, was einem Beweis von $\neg\neg F$ gleichkommt. Nach Satz 2.26 ist nun $\neg\neg F \Leftrightarrow F$ eine Tautologie, deren Langfassung die Form $(\neg\neg F \Rightarrow F) \wedge (F \Rightarrow \neg\neg F)$ ist. Insbesondere müssen dann beide Teilformeln $G = (\neg\neg F \Rightarrow F)$ und $H = (F \Rightarrow \neg\neg F)$ auch Tautologien der Aussagenlogik sein. Während $G$ in so gut wie jedem logischen System gilt, ist das für $H$ keineswegs der Fall. In solchen Systemen reicht die Widerlegung von $\neg F$ nicht als Nachweis von $F$.

**Satz 3.2.** *Modulo $\equiv$ gelten in der klassischen Aussagenlogik die **De Morganschen Regeln**, d.h., alle Formeln $F$, $G$ der Aussagenlogik erfüllen*

(a) $\neg(F \wedge G) \equiv \neg F \vee \neg G$

(b) $\neg(F \vee G) \equiv \neg F \wedge \neg G$

*Beweis.* Nach Definition der Semantik gilt für jede passende Belegung $\alpha$

(a) $$1 - \inf\{\widehat{\alpha}(F), \widehat{\alpha}(G)\} = \sup\{1 - \widehat{\alpha}(F), 1 - \widehat{\alpha}(G)\}$$

(b) $$1 - \sup\{\widehat{\alpha}(F), \widehat{\alpha}(G)\} = \inf\{1 - \widehat{\alpha}(F), 1 - \widehat{\alpha}(G)\} \qquad \square$$

$\square$

## 3.ii Eigenschaften von Kon- und Disjunktion

**Satz 3.3.** *Die Konjunktion $\wedge$, modulo $\equiv$,*

- ▷ *ist* ***kommutativ:*** $F \wedge G \equiv G \wedge F$;
- ▷ *ist* ***assoziativ:*** $(F \wedge G) \wedge H \equiv F \wedge (G \wedge H)$;
- ▷ *ist* ***idempotent:*** $F \wedge F \equiv F$
- ▷ *hat $\top$ als* ***neutrales Element:*** $F \wedge \top \equiv F$.
- ▷ *hat $\bot$ als* ***absorbierendes Element:*** $F \wedge \bot \equiv \bot$.

*Beweis.* Das folgt aus den entsprechenden Eigenschaften der Infimums-Operation $\wedge$ auf dem semantischen Wertebereich $2 = \{0, 1\}$. Diese werden von jeder passenden Belegung $\alpha$ reflektiert. □

Völlig analog erhält man

**Satz 3.4.** *Die Disjunktion $\vee$ ist modulo $\equiv$ kommutativ, assoziativ, idempotent und hat $\bot$ als neutrales sowie $\top$ als absorbierendes Element.* □

**Notationelle Konvention 3.5.**

1. Die Assoziativität erlaubt es Formeln wie $A \wedge B \wedge C$ oder $A \wedge B \wedge C \wedge D$ usw. ohne gruppierende Klammern zu schreiben, analog für die Disjunktion.
2. Abkürzend schreiben wir auch $\bigvee_{i<n} F_i$ statt $F_0 \vee \ldots \vee F_{n-1}$ und $\bigwedge_{i<n} F_i$ statt $F_0 \wedge \ldots \wedge F_{n-1}$. Genauer: wir definieren das Symbol $\bigvee_{i<n} F_i$ induktiv:

$$\bigvee_{i<0} F_i = \bot \qquad \text{und} \qquad \bigvee_{i<n+1} F_i = \Big(\bigvee_{i<n} F_i\Big) \vee F_n$$

   Analoges gilt für $\bigwedge_{i<n} F_i$, speziell $\bigwedge_{i<0} F_i = \top$.

**Bemerkung 3.6.** Die De Morganschen Regeln kann man wie folgt verallgemeinern:

$$\neg \bigwedge_{i=1}^{n} F_i \equiv \bigvee_{i=1}^{n} \neg F_i \qquad \text{sowie} \qquad \neg \bigvee_{i=1}^{n} F_i \equiv \bigwedge_{i=1}^{n} \neg F_i$$

Der Beweis erfolgt analog zu Satz 3.2.

**Satz 3.7.** *Modulo $\equiv$ erfüllen Konjunktion und Disjunktion die* ***Distributivgesetze****, d.h., für alle Formeln $F$ und $G_i$, $i < n$ gilt*

$$F \wedge \bigvee_{i<n} G_i \equiv \bigvee_{i<n} (F \wedge G_i) \qquad \textit{sowie} \qquad F \vee \bigwedge_{i<n} G_i \equiv \bigwedge_{i<n} (F \vee G_i)$$

*Beweis.* Für $n = 0$ stimmt die Aussage mit der Absorbtionseigenschaft von $\bot$ bzw. $\top$ bzgl. der Konjunktion bzw. Disjunktion überein, vergleiche Satz 3.3.

Für $n = 1$ besteht syntaktische Gleichheit, woraus wegen der Reflexivität von $\equiv$ die Behauptung folgt.

Aufgrund der Assoziativität von $\equiv$ genügt nun der Beweis für den Fall $n = 2$. Aus Symmetriegründen beschränken wir uns auf das erste Distributivgesetz. Zu zeigen ist

$$F \wedge (G_0 \vee G_1) \equiv (F \wedge G_0) \vee (F \wedge G_1)$$

Ist $\alpha$ eine passende Belegung, so gilt es in $2 = \{0, 1\}$

$$\widehat{\alpha}F \wedge (\widehat{\alpha}(G_0) \vee \widehat{\alpha}(G_1)) = (\widehat{\alpha}(F) \wedge \widehat{\alpha}(G_0)) \vee (\widehat{\alpha}(F) \wedge \widehat{\alpha}(G_1))$$

nachzuweisen. Die linke Seite hat genau denn den Wert 1, wenn beide Argumente der Konjunktion den Wert 1 haben, d.h., wenn $F$ und mindestens eine der Formel $G_i$, $i < 2$, wahr ist.

Die rechte Seite hat genau dann den Wert 1, wenn mindestens ein Argument der Disjunktion den Wert 1 hat, d.h., wenn $F$ und mindestens eine der Formeln $G_i$, $i < 2$, wahr sind. Damit ist die Behauptung beweisen. □

**Bemerkung 3.8.** Äquivalente Formeln haben dieselbe Wahrheitstabelle. Aufgrund der Kommutativität und der Idempotenz der Konjunktion wie der Disjunktion haben wir damit genau 4 potentiell verschiedene Formeln, die nur ein Atom $A$ verwenden:

$$A,\ \neg A,\ A \vee \neg A \quad \text{und} \quad A \wedge \neg A$$

Ob diese Formeln wirklich verschieden sind, entnehmen wir den Wahrheitstabellen

| $A$ | $A$ |
|---|---|
| 0 | 0 |
| 1 | 1 |

| $A$ | $\neg A$ |
|---|---|
| 0 | 1 |
| 1 | 0 |

| $A$ | $A \vee \neg A$ |
|---|---|
| 0 | 1 |
| 1 | 1 |

| $A$ | $A \wedge \neg A$ |
|---|---|
| 0 | 0 |
| 1 | 0 |

In der Tat werden alle Möglichkeiten ausgeschöpft.

Analog haben die Wahrheitstabellen für genau 2 verschiedene Atome 4 Zeilen mit $2^4 = 16$ möglichen rechten Spalten. Also können modulo $\equiv$ höchstens 16 nicht äquivalente Formeln auftreten.

Allgemein können wir mit genau $n$ verschiedenen Atomen höchstens $2^{2^n}$ paarweise nicht-äquivalente Formeln aufbauen.

## 3.iii Adäquate Junktoren

Die De Morganschen Regeln und die Tatsache, dass $\neg$ selbstinvers ist, zeigen, dass die von uns verwendeten „Basis-Junktoren“ $\bot$, $\neg$, $\wedge$ und $\vee$ noch Redundanz aufweisen. So ist $A_0 \wedge \neg A_0$ äquivalent zu $\bot$. Weiterhin kann z.B. $\vee$ eliminiert werden: aus $\neg(F \vee G) \equiv \neg F \wedge \neg G$ folgt durch Negation beider Seiten

$$F \vee G \equiv \neg(\neg F \wedge \neg G)$$

**Definition 3.9.** Eine Menge $\mathcal{J}$ von Junktoren vorgegebener Semantik heißt **adäquat**, falls jede $\{\bot, \neg, \wedge, \vee\}$-Formel zu einer $\mathcal{J}$-Formel äquivalent ist, d.h., zu einer Formel, in der nur Junktoren aus $\mathcal{J}$ vorkommen.

**Beispiel 3.10.**

(a) Wie wir gesehen haben, formen $\wedge$ und $\neg$ eine adäquate Menge. Analog sind auch $\vee$ und $\neg$ adäquat.

(b) Die Junktoren $\Rightarrow$ und $\neg$ sind adäquat. In der Tat: Disjunktion können wir wie folgt ersetzen:
$$F \vee G \equiv \neg\neg F \vee G = \neg F \Rightarrow G$$
und dann benutzen wir die Tatsache, dass $\vee$ und $\neg$ adäquat sind.

(c) Die Junktoren $\Rightarrow$ und $\bot$ sind adäquat. Da $F \Rightarrow \bot$ eine Abkürzung für $\bot \vee \neg F$ ist, impliziert die Neutralität von $\bot$ bzgl. $\vee$ sofort $F \Rightarrow \bot \equiv \neg F \vee \bot \equiv \neg F$. Nun folgt die Behauptung aus (b).

**Beispiel 3.11.** Die Junktoren $\wedge$ und $\bot$ sind nicht adäquat. In der Tat, für eine atomare Aussage $A$ ist die Formel $\neg A$ zu keiner Formel $F$ äquivalent, die nur $\wedge$ und $\bot$ als Junktoren verwendet. Falls nämlich die Formel $F$ das Symbol $\bot$ nicht enthält, hat sie die Form $F = \bigwedge_{i<n} B_i$, mit $B_i$ atomar. Dann nehmen wir die Belegung $\alpha$, die jeder atomaren Aussage $B_i$ den Wert 1 zuordnet: es gilt $\widehat{\alpha}(\neg A) = 0$ und $\widehat{\alpha}(F) = 1$. Falls andererseits $F$ das Symbol $\bot$ enthält, gilt $\widehat{\alpha}(F) = 0$ für jede Belegung $\alpha$. Wählen wir speziell die Belegung, die konstant den Wert 0 hat, dann gilt $\widehat{\alpha}(\neg A) = 1$ und $\widehat{\alpha}(F) = 0$.

**Beispiel 3.12.** Der NAND-Junktor $\uparrow$ (auch als "Sheffer-Stroke" bekannt) ist der Junktor "nicht beide", der durch
$$A \uparrow B = \neg(A \wedge B)$$
definiert wird. Die Aussagenlogik kann durch den NAND Junktor allein aufgebaut werden; die Menge $\{\uparrow\}$ ist adäquat:
**Negation:** $\neg A \equiv \neg(A \wedge A) \equiv A \uparrow A$
**Konjunktion:** $A \wedge B \equiv \neg(A \uparrow B) \equiv (A \uparrow B) \uparrow (A \uparrow B)$
**Disjunktion:** $A \vee B \equiv \neg(\neg A \wedge \neg B) \equiv (\neg A) \uparrow (\neg B) = (A \uparrow A) \uparrow (B \uparrow B)$
**Absurdität:** $\bot \equiv A \wedge \neg A \equiv (A \uparrow (A \uparrow A)) \uparrow (A \uparrow (A \uparrow A))$

Der Preis für die geringe Anzahl an Junktoren ist natürlich eine gewisse Unübersichtlichkeit der resultierenden Formeln (speziell bei Verwendung umgekehrt polnischer Notation, vergl. HA).

# 4. Normalformen

Es ist oft wichtig mit Formeln einer gewissen „schönen“ Gestalt arbeiten zu können. Die wichtigste solcher Formen heißt konjunktive Normalform (KNF) und wir zeigen hier, wie sie berechnet werden kann. Zwei andere Normalformen werden ebenfalls eingeführt.

## 4.i Negation-Normalform (NNF)

Die De Morganschen Regeln und die Tatsache, dass $\neg$ selbstinvers ist, ermöglichen es, Formeln so umzugestalten, dass die Negation höchstens vor atomaren Aussagen auftritt.

**Beispiel 4.1.**

$$\begin{aligned}(A \vee B) \Rightarrow (B \wedge \neg(A \vee \neg C)) &\equiv \neg(A \vee B) \vee (B \wedge \neg(A \vee \neg C)) \\ &\equiv (\neg A \wedge \neg B) \vee (B \wedge \neg A \wedge \neg\neg C) \\ &\equiv (\neg A \wedge \neg B) \vee (B \wedge \neg A \wedge C)\end{aligned}$$

**Definition 4.2.** Eine Formel $F$ liegt in **Negation-Normalform** (NNF) vor, falls nur die Junktoren $\neg$, $\wedge$ und $\vee$ vorkommen, und die Negationen in $F$ höchstens direkt vor atomaren Aussagen auftritt.

Wir können für jede Formel $F$ ihre NNF berechnen indem wir die De Morganschen Regeln rekursiv benutzen: wegen

$$F = \neg(G \vee H) \equiv \neg G \wedge \neg H$$

sind nur die NNF für $\neg G$ sowie $\neg H$ berechnen. Analog für $F = \neg(G \vee H)$. Und falls $F = \neg\neg G$, gilt $F \equiv G$. Dies formalisiert der folgende rekursive Algorithmus:

**Algorithmus 4.3** (NNF)**.**
**Eingabe:** Eine Formel $F$ mit Junktoren $\vee, \wedge$ und $\neg$.
(Die Junktoren $\Rightarrow$ und $\Leftrightarrow$ müssen ggf. erst ersetzt werden.)
**Ausgabe:** Eine zu $F$ äquivalente Formel in NNF, bezeichnet als $\mathsf{NNF}(F)$

Wir verwenden strukturelle Rekursion:

0. $F$ atomar: wir setzen
$$\mathsf{NNF}(F) := F$$

1. $F = G_0 \wedge G_1$: wir setzen
$$\mathsf{NNF}(G_0 \wedge G_1) := \mathsf{NNF}(G_0) \wedge \mathsf{NNF}(G_1)$$

2. $F = G_0 \vee G_1$: wir setzen

$$\mathsf{NNF}(G_0 \vee G_1) := \mathsf{NNF}(G_0) \vee \mathsf{NNF}(G_1)$$

3. $F = \neg G$: Hier müssen wir eine zweite Rekursion starten, diesmal in $G$.

   3.0 $G$ atomar: wir setzen

   $$\mathsf{NNF}(\neg G) := \neg G$$

   3.1 $G = H_0 \wedge H_1$: wir setzen

   $$\mathsf{NNF}(\neg(H_0 \wedge H_1)) := \mathsf{NNF}(\neg H_0) \vee \mathsf{NNF}(\neg H_1)$$

   3.2 $G = C_0 \vee C_1$: wir setzen

   $$\mathsf{NNF}(\neg(H_0 \vee H_1)) := \mathsf{NNF}(\neg H_0) \wedge \mathsf{NNF}(\neg H_1)$$

   3.3 $G = \neg H$: wir setzen

   $$\mathsf{NNF}(\neg\neg H) := \mathsf{NNF}(H)$$

**Satz 4.4.** *Der Algorithmus NNF ist korrekt: für jede Formel $F$ wird NNF(F) in endlich vielen Schritten berechnet, ist zu $F$ äquivalent und liegt in NNF vor.*

*Beweis.* Die endlich vielen Schritte folgen aus der Tatsache, dass für eine Formel $F$ der Länge $n$ in einem rekursiven Durchlauf der Algorithmus NNF ein oder zweimal verwendet wird, jedes Mal an Formeln der Länge höchstens $n-1$. Da für die Formeln der Länge 1 nur ein Schritt notwendig ist, haben wir insgesamt höchstens $n$ Durchläufe. Die Äquivalenz $F \equiv \mathsf{NNF}(F)$ und die Tatsache, dass NNF(F) in NNF vorliegt, müssen wieder per struktureller Induktion bewiesen werden. Falls $F$ atomar ist, gilt $F = \mathsf{NNF}(F)$, dies hat NNF. Gilt die Aussage für $G_0$ und $G_1$, verwenden wir für $F = G_0 \wedge G_1$ die Definition

$$\mathsf{NNF}(F) := \mathsf{NNF}(G_0) \wedge \mathsf{NNF}(G_1)$$

und benutzen dann die Kongruenzeigenschaft aus Satz 2.28: aus $G_i \equiv \mathsf{NNF}(G_i)$ für $i < 2$ folgt

$$\mathsf{NNF}(F) \equiv G_0 \wedge G_1 = F$$

Da $\mathsf{NNF}(G_0)$ und $\mathsf{NNF}(G_1)$ beide in NNF vorliegen, folgt dies unmittelbar auch für $\mathsf{NNF}(F)$.
Die Fälle $F = G_0 \vee G_1$ und $F = \neg G$ sind analog zu bearbeiten. □

**Beispiel 4.5.** Wir wollen eine $\mathsf{NNF}$ der Formel

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

berechnen. Bevor wir den rekursiven Algorithmus starten, müssen wir $\Rightarrow$ ersetzen. Wir arbeiten mit der Formel

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

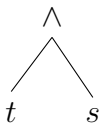
Die rekursive Berechnung wird separat auf der linken und der rechten Seite entwickelt:

$$\begin{aligned}
\mathsf{NNF}(G) &= \big(\mathsf{NNF}(\neg C) \vee \mathsf{NNF}(\neg A \vee B)\big) \wedge \big(\mathsf{NNF}(\neg(A \wedge C) \vee (\neg B \vee C)\big) \\
&= \big(\neg C \vee \mathsf{NNF}(\neg A) \vee \mathsf{NNF}(B)\big) \wedge \big(\mathsf{NNF}(\neg(A \wedge C)) \vee \mathsf{NNF}(\neg B \vee C)\big) \\
&= \big(\neg C \vee \neg A \vee B\big) \wedge \big(\mathsf{NNF}(\neg A) \vee \mathsf{NNF}(\neg C) \vee \mathsf{NNF}(\neg B) \vee \mathsf{NNF}(C)\big) \\
&= \big(\neg C \vee \neg A \vee B\big) \wedge \big(\neg A \vee \neg C \vee \neg B \vee C\big) \\
&\equiv \big(\neg C \vee \neg A \vee B\big) \wedge \top \\
&\equiv \big(\neg C \vee \neg A \vee B\big)
\end{aligned}$$

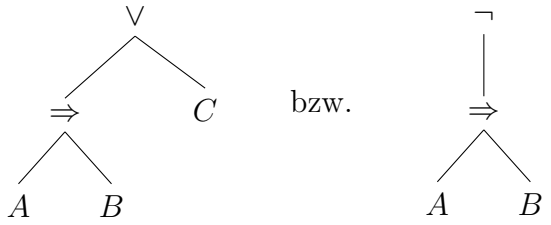
# 4.ii Baum-Notation von Formeln

Es ist oft vorteilhaft, sich die Struktur einer Formel $F$ (wie sie durch ihre Unterformeln aufgebaut ist) als einen Syntax-Baum mit Wurzel vorzustellen, dessen Blätter mit den atomaren Aussagen von $F$ und dessen innere Knoten mit geeigneten Junktoren markiert sind. Dabei stimmt die Stelligkeit des markierenden Junktors mit der Anzahl der Kinder überein. Für Formeln $F = G \vee H$ steht $\vee$ in der Wurzel (analog mit $\wedge$, $\Rightarrow$ oder $\Leftrightarrow$). Für $F = \neg G$ steht $\neg$ in der Wurzel.

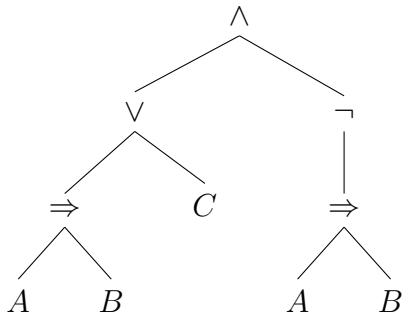
**Beispiel 4.6.** $((A \Rightarrow B) \vee C) \wedge \neg(A \Rightarrow B)$ ist eine Formel mit Hauptjunktor $\wedge$; der entsprechende Baum hat die Form




wobei $t$ der Baum für $(A \Rightarrow B) \vee C$ und $s$ der für $\neg(A \Rightarrow B)$ ist. Wir sehen schnell, dass $t$ und $s$ die Form

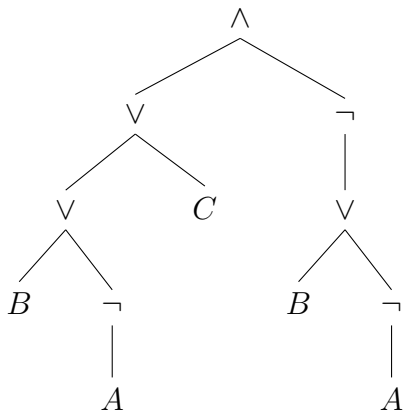


haben, deswegen wird $F$ durch den Baum

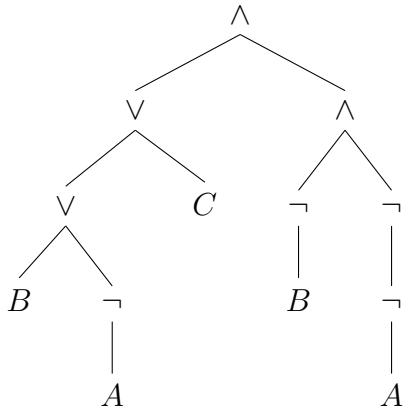



dargestellt.

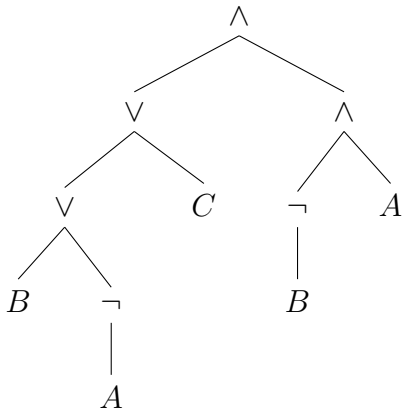
**Bemerkung 4.7.** Der Algorithmus NNFverschiebt die ¬-Knoten nach unten bis sie sich mit anderen ¬-Knoten aufheben, oder nur vor den Blättern landen. Dies illustrieren wir im obigen Beispiel: erst müssen wir ⇒ ersetzen




Jetzt wird im rechten Unterbaum ¬∨ mit Hilfe der De Morganschen Regel umgewandelt:



und schließlich wird die doppelte Negation $\neg\neg$ entfernt:




In linearer Form erhalten wir nun $(B \vee \neg A \vee C) \wedge \neg B \wedge A$.

# 4.iii Konjunktive Normalform

**Definition 4.8.** Unter einem **Literal** verstehen eine atomare Aussage oder die Negation einer solchen. Wir sprechen auch von positiven Literalen $A$ oder negativen Literalen $\neg A$ ($A$ atomar). Da die Negation $\neg$ selbstinvers ist, sind die Literale hinsichtlich Äquivalenz unter Negation abgeschlossen.

**Definition 4.9.** Eine Disjunktion von Literalen heißt **Klausel**.
**Beispiel 4.10.**

(a) Die Formel $A \vee A \vee \neg B \vee C$ ist eine Klausel. Sie ist äquivalent zur kürzeren Formel $A \vee \neg B \vee C$, die auch eine Klausel ist.

(2) Für $n$ atomare Aussagen $X_0, \ldots, X_{n-1}$ ist $\bigvee_{i<n} X_i$ eine Klausel. Ihr Wert ist 1 in genau für die passenden Belegungen $\alpha$, für die ein Index $i < n$ mit $\alpha(X_i) = 1$ existiert. Im Fall $n = 0$ ist die Klausel $\bot$ und somit falsch.

**Bemerkung 4.11.** Die Formeln, die die Form $F_0 \wedge \ldots \wedge F_{n-1}$ einer Konjunktion von Klauseln haben, sind „übersichtlicher“ als die allgemeinen Formeln: der Wert unter einer Belegung ist genau dann wahr, wenn alle Klauseln $F_i$ wahr sind. Dazu muß für jedes $i < n$ ein positives Literal $A$ mit $\alpha(A) = 1$ in $F_i$ vorkommen, oder ein negatives Literal $\neg A$ mit $\alpha(A) = 0$.

**Beispiel 4.12.** Die Formel $(B \wedge \neg A \wedge C) \vee (\neg A \wedge \neg B)$ wollen wir als Konjunktion von Klauseln darstellen. Dazu benutzten wir die Distributivgesetze:

$$\begin{aligned}
&(B \wedge \neg A \wedge C) \vee (\neg A \wedge \neg B) \\
\equiv\ &((B \wedge \neg A \wedge C) \vee \neg A) \wedge ((B \wedge \neg A \wedge C) \vee \neg B) \\
\equiv\ &(B \vee \neg A) \wedge (\neg A \vee \neg A) \wedge (C \vee \neg A) \wedge (B \vee \neg B) \wedge (\neg A \vee \neg B) \wedge (C \vee \neg B) \\
\equiv\ &(B \vee \neg A) \wedge \neg A \wedge (C \vee \neg A) \wedge (\neg A \vee \neg B) \wedge (C \vee \neg B)
\end{aligned}$$

Zum Schluss kamen die Idempotenz von $\vee$ und die Neutralität von $\top$ zur Anwendung.

**Definition 4.13.** Eine Formel in **Konjunktive Normalform (KNF)** besteht aus einer Konjunktion von Klauseln.

Die letzte Zeile des obigen Beispiels hat also bereits KNF. Wir stellen jetzt einen Algorithmus vor, der eine KNF berechnet. Sein Kern besteht in der Benutzung des Distributivgesetzes. Wir formulieren erst diesen Kern, und fügen diesen dann in unserem Algorithmus ein:

**Satz 4.14.** *Falls $F$ und $G$ in KNF mit Klauseln $F_i$ mit $i < n$ bzw. $G_j$ mit $j < m$ vorliegen, ist $F \vee G$ äquivalent zur Konjunktion der Klauseln $F_i \vee G_j$ für $i < n$ und $j < m$.*

*Beweis.* Es gilt, aufgrund des Distributivgesetzes:

$$F \vee G = (\bigwedge_{i<n} F_i) \vee G \equiv \bigwedge_{i<n} (F_i \vee G)$$

Für jedes $i$ berechnen wir analog

$$F_i \vee G \equiv F_i \vee (\bigwedge_{j<m} G_j) \equiv \bigwedge_{j<m} (F_i \vee G_j)$$

Daraus folgt:

$$F \vee G \equiv \bigwedge_{i<n} (F_i \vee G) \equiv \bigwedge_{i<n} \bigwedge_{j<m} (F_i \vee G_j)$$

Weil die Formeln $F_i$ und $G_j$ Klauseln sind ist aufgrund der Assoziativität von $\vee$ auch $F_i \vee G_j$ eine Klausel. Somit liegt die Gesamtformel in KNF vor. □

Der folgende Algorithmus verwendet wieder strukturelle Induktion.

**Algorithmus 4.15** (KNF)**.**
**Eingabe:** Eine Formel $F$ in NNF
**Ausgabe:** Eine KNF, äquivalent zu $F$, bezeichnet als KNF(F)

0. $F$ atomar: wir setzen
$$\mathsf{KNF}(F) := F$$

1. $F = G \wedge H$, wobei $\mathsf{KNF}(G)$ und $\mathsf{KNF}(H)$ bekannt sind: wir setzen
$$\mathsf{KNF}(A) := \mathsf{KNF}(G) \wedge \mathsf{KNF}(H)$$

2. $F = G \vee H$, wobei $\mathsf{KNF}(G)$ und $\mathsf{KNF}(H)$ folgende Form als Konjunktion von Klauseln haben
$$\mathsf{KNF}(G) = \bigwedge_{i<n} G_i \quad \text{bzw.} \quad \mathsf{KNF}(H) = \bigwedge_{j<m} H_j$$
Dann setzen wir gemäß Satz 4.14
$$\mathsf{KNF}(G \vee H) := \bigwedge_{i<n} \bigwedge_{j<m} (G_i \vee H_j)$$

3. $F = \neg G$. Hier muss $G$ atomar sein, da $F$ nach Voraussetzung in $\mathsf{NNF}$ vorliegt. Wir setzen
$$\mathsf{KNF}(\neg G) := \neg G.$$

**Satz 4.16.** *Der Algorithmus $\mathsf{KNF}$ ist korrekt: für jede Formel $F$ berechnen wir in endlich vielen Schritte eine zu $F$ äquivalente Formel in $\mathsf{KNF}$.*

*Beweis.* Der Beweis hat dieselbe Struktur wie der von Satz 4.4. Der einzige nichttriviale Schritt ist zu beweisen, dass $\mathsf{KNF}(F)$ für $F = G \vee H$ zu $F$ äquivalent ist. Aber dies folgt aus Satz 4.14. □

**Beispiel 4.17.** Die $\mathsf{KNF}$ der Formel
$$F = A \Leftrightarrow (B \Rightarrow \neg C \wedge A)$$
berechnen wir in 3 Schritten:

a) $\Leftrightarrow$ und $\Rightarrow$ sind zu ersetzen, vergleiche Proposition 2.15 (b) und (c):
$$\begin{aligned} & A \Leftrightarrow (B \Rightarrow \neg C \wedge A) \\ \equiv\ & A \Leftrightarrow \neg B \vee (\neg C \wedge A) \\ \equiv\ & (A \wedge (\neg B \vee (\neg C \wedge A))) \vee (\neg A \wedge \neg(\neg B \vee (\neg C \wedge A))) \end{aligned}$$

b) Der Algorithmus $\mathsf{NNF}$ liefert
$$\begin{aligned} F &\equiv (A \wedge (\neg B \vee (\neg C \wedge A))) \vee (\neg A \wedge B \wedge \neg(\neg C \wedge A)) \\ &\equiv (A \wedge (\neg B \vee (\neg C \wedge A))) \vee (\neg A \wedge B \wedge (C \vee \neg A)) \end{aligned}$$

c) Algorithmus $\mathsf{KNF}$ wird angewendet. Das zweite Argument der äußeren Disjunktion liegt bereits in $\mathsf{KNF}$ vor. In ihrem ersten Argument hat das zweite Argument der Konjunktion (rot) noch nicht die Form einer $\mathsf{KNF}$, es handelt sich aber um die Disjunktion zweier Formeln in $\mathsf{KNF}$. Behandlung gemäß Schritt 2, d.h. Anwendung von Satz 4.14 liefert hier eine Formel in $\mathsf{KNF}$ (grün):
$$F \equiv (A \wedge (\neg B \vee \neg C) \wedge (\neg B \vee A)) \vee (\neg A \wedge B \wedge (C \vee \neg A))$$
Auf diese Disjunktion zweier Formeln in $\mathsf{KNF}$ kann nun ebenfalls Satz 4.14 angewendet werden, was die Kombination von jeweils 3 Klauseln erfordert:
$$\begin{aligned} \mathsf{KNF}(F) = (A \vee \neg A) \wedge (A \vee B) \wedge (A \vee C \vee \neg A) \wedge (\neg B \vee \neg C \vee \neg A) \\ \wedge (\neg B \vee \neg C \vee B) \wedge (\neg B \vee \neg C \vee C \vee \neg A) \\ \wedge (\neg B \vee A \vee \neg A) \wedge (\neg B \vee A \vee B) \\ \wedge (\neg B \vee A \vee C \vee \neg A) \end{aligned}$$
Schließlich kann man noch die offensichtlichen Tautologien zusammenfassen, und dann Klauseln mit $\top$ entfernen:
$$\begin{aligned} \mathsf{KNF}(F) &\equiv \top \wedge (A \vee B) \wedge (\top \vee C) \wedge (\neg B \vee \neg C \vee \neg A) \\ &\quad \wedge (\neg C \vee \top) \wedge (\top \vee \neg B \vee \neg A) \\ &\quad \wedge (\top \vee \neg B) \wedge (A \vee \top) \wedge (\neg B \vee C \vee \top) \\ &\equiv (A \vee B) \wedge (\neg B \vee \neg C \vee \neg A) \end{aligned}$$

**Beispiel 4.18.** Für $G_2 = (A_0 \wedge B_0) \vee (A_1 \wedge B_1)$ ergibt sich $\mathsf{KNF}(G_2)$ aufgrund von Satz 4.14, d.h. nur unter Verwendung des Distributivgesetzes, zu:

$$\mathsf{KNF}(G_2) = (A_0 \vee A_1) \wedge (A_0 \vee B_1) \wedge (B_0 \vee A_1) \wedge (B_0 \vee B_1)$$

Analog: für die Formel $G_{n-1} = \bigvee_{i<n}(A_i \wedge B_i)$ ergibt sich $\mathsf{KNF}(G_{n-1})$ als Konjunktion aller Klauseln des Typs

$$A_0 \vee A_1 \vee \ldots A_{n-1}$$

$$B_0 \vee A_1 \vee \ldots A_{n-1}$$

$$\vdots$$

$$B_0 \vee B_1 \vee \ldots B_{n-1}$$

wobei immer $n$ Atome mit aufsteigenden Indizes $i < n$ auftreten und als Namen zunächst $B$, dann später $A$.

**Bemerkung 4.19.** Die Formel $G_{n-1}$ hat Länge $4n - 1 \in O(n)$. Aber $\mathsf{KNF}(G_{n-1})$ hat $2^n$ Klauseln, da für jeden Index $1, \ldots, n$ die Wahl zwischen $A$ oder $B$ vorkommt. Deswegen liegt die Länge von $\mathsf{KNF}(G_{n-1})$ in $O(2^n)$. Dies bedeutet, dass der Algorithmus $\mathsf{KNF}$ nicht effizient ist: für Formeln der Länge 200 ist es i.A. praktisch unmöglich eine $\mathsf{KNF}$ zu berechnen. In der Tat: die Lichtgeschwindigkeit ist $3 \cdot 10^{8m}/_{sec}$ und der Durchschnitt eines Protons ist $10^{-15}m$ dann lieferte ein Rechner, der jede Operation in der Zeit erledigt, die das Licht braucht um den Durchmesser eines Protons zurückzulegen, $3 \cdot 10^{23}$ Operationen pro Sekunde. Wegen $10 < 2^4$ gilt $3 \cdot 10^{23} < 2^{100}$. Falls also ein Algorithmus $2^{200}$ Schritte braucht, müsste auch ein so schneller Rechner wenigstens $2^{100}$ Sekunden rechnen - dies ist aber mehr, als die Zeit ($4 \cdot 10^{17} < 2^{59}$ Sekunden) seit dem Urknall.

## 4.iv Disjunktive Normalform

**Definition 4.20.** Eine Formel hat **disjunktive Normalform** (DNF), falls sie als Disjunktion von Co-Klauseln vorliegt, wobei Co-Klauseln Konjunktionen von Literalen sind.

**Beispiel 4.21.** Wir wandeln die Formel $C \wedge A \Rightarrow (B \wedge \neg C)$ in DNF um:

$$\begin{aligned} C \wedge A \Rightarrow (B \wedge \neg C) &\equiv \neg(C \wedge A) \vee (B \wedge \neg C) \\ &\equiv \neg C \vee \neg A \vee (B \wedge \neg C) \end{aligned}$$

**Algorithmus 4.22.** Um eine Formel $G$ in eine DNF zu bringen, können wir die de Moganschen Regeln auf eine KNF für $\neg G$ anwenden.

0. Wir berechnen eine $\mathsf{KNF}$ für $\neg G$
1. Für jede Klausel $K_i$ von $\mathsf{KNF}(\neg G)$ ist $\neg K_i$ äquivalent zur Konjunktion von Literalen, also einer Co-Klausel: die Literale in $K_i$ werden negiert.

2. Es gilt:

$$\begin{aligned} G &\equiv \neg\mathsf{KNF}(\neg G) \\ &\equiv \neg(K_0 \wedge K_1 \wedge \ldots \wedge K_{n-1}) \\ &\equiv \neg K_0 \vee \neg K_1 \vee \ldots \vee \neg K_{n-1} \end{aligned}$$

Ersetzen aller negierten Klauseln gemäß der Vorüberlegung in 1. liefert eine DNF.

# 5. Resolutionsmethode der Aussagenlogik

Dieses Kapitel stellt einen Algorithmus vor, der die Erfüllbarkeit einer Formel bestimmt, die sogenannte Resolutionsmethode. Die Idee basiert auf den folgenden Beobachtungen:

**Bemerkung 5.1.** Gegeben ist eine Formel $F$ in $\mathsf{KNF}$.

(a) Falls $F$ die Literale $A$ und $\neg A$ als Klauseln enthält, ist $F$ unerfüllbar, da für jede passende Belegung nicht beide Literale gleichzeitig wahr sein können.

(b) Falls $F$ die Klauseln $A \vee C$ und $B \vee \neg C$ enthält, dürfen wir die Klausel $A \vee B$ hinzufügen – ohne dass sich die Erfüllbarkeit ändert. Dies wird später in verallgemeinerter Form als „Resolutionslemma“ bewiesen.

Iterierte Anwendung von (b) unter Berücksichtigung von (a) ermöglicht eine Entscheidung darüber, ob $F$ erfüllbar ist. Den zugehörigen Algorithmus entwickeln wir in diesem Abschnitt.

**Notationelle Konvention 5.2** (Mengentheoretische Schreibweise)**.**

0. Für eine Klausel $X_0 \vee \ldots \vee X_{n-1}$ verwendet man auch die mengentheoretische Notation $\{X_0, \ldots, X_{n-1}\}$. Aufgrund der Idempotenz der Disjunktion brauchen mehrfach auftretende Literale dabei nicht wiederholt werden. Die leere Klausel $\bot$ wird dann mit $\varnothing$ bezeichnet. Das Vorkommen eines Literals $X$ in einer Klausel $K$ kann nunmehr mit Hilfe der Element-Relation $\in$ ausgedrückt werden: entweder gilt $X \in K$ oder $X \notin K$. Klauseln sind unter Vereinigung abgeschlossen.

1. Leider entspricht die Konjunktion von Klauseln *nicht* dem Durchschnitt derselben, stattdessen verbinden wir die entsprechenden Mengen weitherhin mit $\wedge$. Z.B. läßt sich $\mathsf{KNF}(G_2)$ in Beispiel 4.18 wie folgt darstellen

$$\{A_0, A_1\} \wedge \{A_0, B_1\} \wedge \{B_0, A_1\} \wedge \{B_0, B_1\}$$

2. Falls die Klausel $K$ das Literal $X$ enthält, bezeichnet $K \backslash X$ jene Klausel, die durch Entfernen von $X$ aus $K$ entsteht. Für $X \notin K$ ist $K \backslash X$ undefiniert.

**Beispiel 5.3.** Für $K = \{A, \neg B\}$ gilt $K \backslash A = \{\neg B\}$ und $K \backslash \neg B = \{A\}$. Für $L = \{A, \neg B, C\}$ haben wir $L \backslash C = \{A, \neg B\}$, aber $L \backslash \neg A$ ist nicht definiert. Schließlich gilt $\{A\} \backslash A = \varnothing$.

**Definition 5.4.** Sind $K$ und $K'$ Klauseln und ist $X$ ein Literal mit $X \in K$ und $\neg X \in K'$, so nennen wir die Klausel

$$(K \backslash X) \cup (K' \backslash \neg X)$$

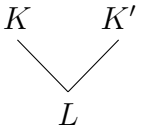
eine **Resolvente** von $K$ und $K'$.

Genauer: Sind zwei Klauseln $K := \{X_0, \ldots, X_{n-1}\}$ und $K' := \{X'_0, \ldots, X'_{m-1}\}$ gegeben mit $X_i = \neg X'_j$ für ein Paar von Indezes $i < n$ und $j < m$, dann ist die Klausel

$$\{X_0, \ldots, X_{i-1}, X_{i+1}, \ldots, X_{n-1}, X'_0, \ldots, X'_{j-1}, X'_{j+1}, \ldots, X'_{m-1}\}$$

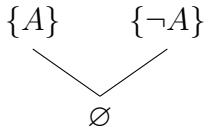
eine Resolvente von $K$ und $K'$.

**Notationelle Konvention 5.5.** Ist $L$ eine Resolution von $K$ und $K'$, so verwenden wir folgende **Baum-Notation**

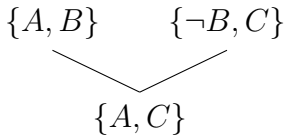



**Beispiel 5.6.**

(a) $\{A\}$ und $\{\neg A\}$ haben die Resolvente




(b) $\{A, B\}$ und $\{\neg B, C\}$ haben die Resolvente $\{A, C\}$, in Baumform:




(c) $\{A, B, \neg C\}$ und $\{A, \neg B, C\}$ haben zwei Resolventen


die aufgrund der Tautologien und der absorbierenden Eigenschaft von $\top$ beide zu $\top$ äquivalent sind. Insofern stellt sich die Frage, ob die beiden obigen Resolventen überhaupt noch für weitere Resolventenbildung verwendet werden sollten, was nach der Ersetzung durch $\top$ nicht mehr möglich ist. Letztendlich ist abzuwägen, was aufwändiger ist: das Entfernen oder das Mitschleppen von zu $\top$ äquivalenten Resolventen.

Aufpassen! Die Resolventenbildung erfolgt immer nur bezogen auf **eine** atomare Aussage und das zugehörige Paar aus einem positiven und einem negativen Literal. Daher tritt $\{A\}$ hier nicht als Resolvente auf. Wollte man eine Resolvente bzgl. $B \vee \neg C$ bilden, brauchte man als Gegenstück die Negation $\neg(B \vee \neg C) \equiv \neg B \wedge C$, was als Konjunktion nicht Teil einer Klausel sein kann.

**Satz 5.7** (Resolutionslemma)**.** *Für zwei Klauseln $K$ und $K'$ mit Literalen $X \in K$ sowie $\neg X \in K'$ und Resolvente $L := (K \backslash X) \cup (K' \backslash \neg X)$ ist $K \wedge K'$ zu $K \wedge K' \wedge L$ äquivalent, d.h., für jede passende Belegung gilt $\widehat{\alpha}(K \wedge K') = \widehat{\alpha}(K \wedge K' \wedge L)$.*

*Beweis.* Nach Konstruktion ist jede für $K \wedge K'$ passende Belegung auch passend für $L$. Wir unterscheiden zwei Fälle:

- ▷ Aus $\widehat{\alpha}(K \wedge K') = 1$ folgt nach Definition der Semantik $\widehat{\alpha}(K) = 1 = \widehat{\alpha}(K')$. Falls $\alpha(X) = 0$, dann impliziert $\widehat{\alpha}(K) = 1$ sofort $\widehat{\alpha}(K \backslash X) = 1$, denn $K$ muss ein von $X$ verschiedenes wahres Literal enthalten. Analog folgt aus $\alpha(X) = 1$ sofort $\widehat{\alpha}(K' \backslash \neg X) = 1$, denn $\neg X$ kann nicht das wahre Literal von $K'$ sein. In beiden Fällen gilt:

  $$\widehat{\alpha}(L) = \widehat{\alpha}(K \backslash X) \vee \widehat{\alpha}(K' \backslash \neg X) = 1$$

  und folglich

  $$\widehat{\alpha}(K \wedge K' \wedge L) = \widehat{\alpha}(K \wedge K') \wedge \widehat{\alpha}(L) = 1 \wedge 1 = 1$$

- ▷ Aus $\widehat{\alpha}(K \wedge K') = 0$ folgt $\widehat{\alpha}(K \wedge K' \wedge L) = 0$. □

Wir wollen nun die Erfüllbarkeit einer Formel in KNF analysieren, indem wir sie unter Resolution „sättigen“, d.h., solange neue Resolventen bestimmen, bis entweder $\varnothing$ auftritt, oder keine weiteren Resolventen mehr gebildet werden können. Hier ist es zweckmäßig, Formeln in KNF als Mengen von Klauseln aufzufassen.

**Definition 5.8.** Für eine Formel $F$ in KNF entsteht $\mathrm{Res}(F)$ durch Hinzunahme aller Resolventen zweier $F$-Klauseln, oder in herkömmlichr Notation als Konjunktion

$$\mathrm{Res}(F) := F \wedge \bigwedge \{\, L \,:\, L \not\equiv \top \text{ ist neue Resolvente zweier } F\text{-Klauseln} \,\}$$

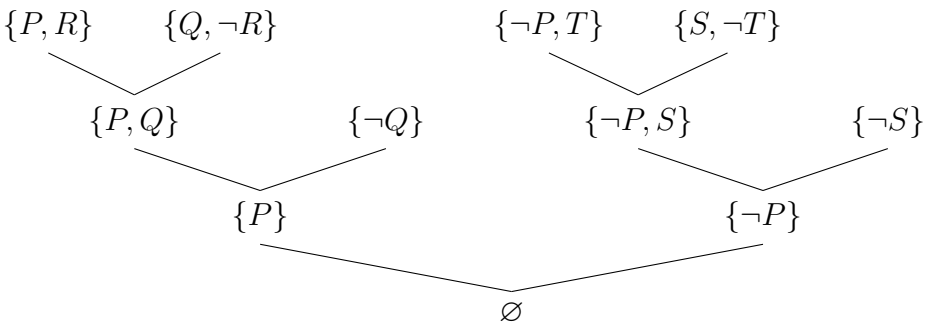
Zwecks Arbeitserleichterung, werden dabei entstehende Klauseln, die Tautologien enthalten, wie in Beispiel 5.6(c) sofort durch $\top$ ersetzt, was wegen der Neutralität bzgl. $\wedge$ eliminiert werden kann.

**Beispiel 5.9.** Für $F = (P \vee R) \wedge (Q \vee \neg R) \wedge \neg Q \wedge (\neg P \vee T) \wedge \neg S \wedge (S \vee \neg T)$ wollen wir die Res-Funktion iterieren:

$$\begin{aligned}
F &= \{P, R\}\,\{Q, \neg R\}\,\{\neg Q\}\,\{\neg P, T\}\,\{\neg S\}\,\{S, \neg T\} \\
\mathrm{Res}(F) &= F\,\{P, Q\}\,\{\neg R\}\,\{R, T\}\,\{\neg P, S\}\,\{\neg T\} \\
\mathrm{Res}^2(F) &= \mathrm{Res}(F)\,\{R, S\}\,\{P\}\,\{Q, T\}\,\{\neg P\}\,\{Q, S\}\,\{T\}\,\{R\} \\
\mathrm{Res}^3(F) &= \mathrm{Res}^2(F)\,\{Q\}\,\{S\}\,\varnothing \\
\mathrm{Res}^4(F) &= \mathrm{Res}^3(F)
\end{aligned}$$

Die Iteration muß nach endlich vielen Schritten ein stabiles Ergebnis liefern, da die Anzahl $k$ der verfügbaren Variablen die Größe der möglichen Klauseln beschränkt. Sobald $\varnothing$ als Resolvente auftritt, kann man das Verfahren abbrechen, denn aufgrund des Resolutionslemmas ist die ursprüngliche Formel ebenso wie die aktuelle Formel nicht erfüllbar.

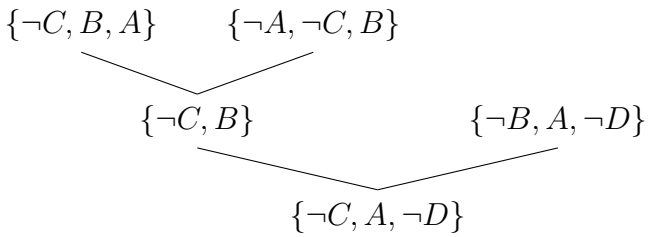
In diesem konkreten Beispiel läßt sich mindestens ein Resolventenbaum mit Wurzel $\varnothing$ erstellen:



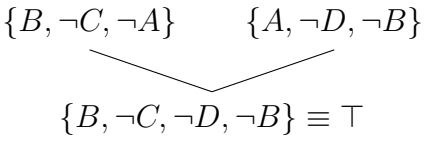

Offenbar kann $\varnothing$ nur als Resolvente eines Literals und seiner Negation auftreten. Die zugehörigen Resolventenbäume mit Blättern in der Ausgangsformel werden i.A. aber nicht disjunkt sein, mit anderen Worten, dieselbe Klausel kann mehrfach als Knoten auftreten.

Viel wichtiger ist aber die Frage, ob eine Formel erfüllbar ist, wenn $\varnothing$ nicht bei der Iteration der Resolventen auftritt.

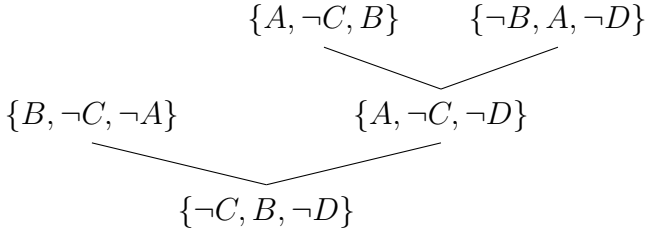
**Beispiel 5.10.** $F := (A \vee B \vee \neg C) \wedge (\neg A \vee B \vee \neg C) \wedge (A \vee \neg B \vee \neg D)$ Wir betrachten zunächst Resolventenbäume mit Blättern in $F$:




(0)




(1)




(2)

$$\begin{array}{ccc} \{\neg A, \neg C, B\} & & \{\neg B, A, \neg D\} \\ & \diagdown\!\!\diagup & \\ & \{\neg A, \neg C, A, \neg D\} \equiv \top & \end{array} \tag{3}$$

In diesem Fall sind keine weiteren Resolventen konstruierbar. Zwar können wir aus den Resolventen keine Belegung ablesen, mit der $F$ wahr wird. Trotzdem hoffen wir, dass aus der Abwesenheit von $\varnothing$ die Erfüllbarkeit von $F$ folgt.

**Satz 5.11.** *Eine Formel in KNF ist genau dann erfüllbar wenn $\varnothing$ keine Resolvente ist.*

*Beweis.* Tritt $\varnothing$ als Resolvente auf, so gilt nach dem Resolutionslemma $F \equiv F \wedge \bot \equiv \bot$. Somit ist $F$ nicht erfüllbar.

Wenn $\varnothing$ nicht als Resolvente auftritt, beweisen wir die Erfüllbarkeit von $F$ per Induktion über die Anzahl $k$ an verschiedenen Literalen, die $F$ enthält:
**Induktionsanfang** $k = 0$
Dann ist $F = \top$.
**Induktionsvoraussetzung** $k > 0$
Wir nehmen als Induktionsvoraussetzung an, dass alle Formeln mit $< k$ verschiedenen Literalen die Eigenschaft haben erfüllbar zu sein, wenn aus ihren $\varnothing$ nicht als Resolvente ableitbar ist.
**Induktionsschritt** $k > 0$
OBdA können wir annehmen, dass $F$ keine tautologischen Klauseln enthält. Wir wählen ein Literal $X$ in $F$ und sortieren die Klauseln disjunkt wie folgt in Teilformeln:

$$F = F_X \wedge F_{\neg X} \wedge F_O \qquad \text{mit}$$

$F_X$ die Konjunktion aller Klauseln, die $X$ enthalten

$F_{\neg X}$ die Konjunktion aller Klauseln, die $\neg X$ enthalten

$F_O$ die Konjunktion aller Klauseln, die weder $X$ noch $\neg X$ enthalten.

Mit $\widetilde{F}_X$ bezeichnen wir die Formel, die aus $F_X$ entsteht, wenn alle Klauseln $K$ durch $K \backslash X$ ersetzt werden.

**Behauptung:** Entweder ist $F$ erfüllbar, oder die Resolvente $\{X\}$ kann erzeugt werden.

Falls $\tilde{F}_X \wedge F_O$ erfüllbar ist, gibt es eine Belegung $\alpha$ mit $\widehat{\alpha}(\tilde{F}_X \wedge F_O) = 1$ und $\alpha(X) = 0$, weil weder $X$ noch $\neg X$ als Literal in $\tilde{F}_X \wedge F_O$ vorkommen. Dann gilt aber auch $\widehat{\alpha}(F_X \wedge F_O) = 1$ und wegen $\widehat{\alpha}(\neg X) = 1$ ergibt sich $\widehat{\alpha}(F_{\neg X}) = 1$, woraus $\widehat{\alpha}(F) = 1$ folgt.

Ist hingegen $\widetilde{F}_X \wedge F_O$ nicht erfüllbar, tritt nach Induktionsvoraussetzung $\varnothing$ als Resolvente dieser Formel auf. Dabei werden immer nur von $X$ und $\neg X$ verschiedenen Literale eliminiert. Im letzten Schritt der Konstruktion von $\varnothing$ als Resolvente von $\widetilde{F}_X \wedge F_O$ sind Resolventen $\{Y\}$ und $\{\neg Y\}$ für geeignetes $Y \neq X$ beteiligt. Da nach Voraussetzung $\varnothing$ keine Resolvente von $F$ und daher auch nicht von $F_X \wedge F_O$ ist, muß $\{Y\}$ oder $\{\neg Y\}$ Resolvente von $\widetilde{F}_X$ sein. Spielt man nun deren Konstruktion für $F_X$ nach, erhält man $\{X, Y\}$ oder $\{X, \neg Y\}$, was im nächsten Resolutionsschritt $\{X\}$ liefert.

Aus Symmetriegründen gilt nun aber auch die Behauptung, dass entweder $F$ erfüllbar ist, oder die Resolvente $\{\neg X\}$ erzeugt werden kann.

Zusammen bedeutet dies, dass entweder $F$ erfüllbar ist, oder die Resolventen $\{X\}$ und $\{\neg X\}$ erzeugt werden können. Aber letzteres impliziert die Erzeugbarkeit der Resolvente $\varnothing$. Also scheidet die zweite Alternative aus und $F$ muß erfüllbar sein. □

Damit erhalten wir folgenden Algorithmus zur Bestimmung der Erfüllbarkeit von aussagenlogischen Formeln:

**Algorithmus 5.12** (Resolutionsmethode)**.**
**Eingabe:** Eine Formel $F$ in KNF.
**Ausgabe:** Entscheidung über die Erfüllbarkeit von $F$.
Iteriere die Res-Funktion so lange, bis

- ▷ entweder die leere Resolvente $\varnothing$ entsteht, Ausgabe „$F$ ist unerfüllbar“;
- ▷ oder die Res-Funktion sich stabilisiert hat, d.h., keine weiteren Resolvenen mehr gebildet werden können, und alle Resolventen nichtleer sind, Ausgabe „$F$ ist erfüllbar.“

**Beispiel 5.13.** Familie A will im kommenden Jahr eine Waschmaschine, ein Auto und ein Moped anschaffen. Aber falls Herr A seinen üblichen Bonus nicht bekommt, können sie sich nicht alles leisten. Die Waschmaschine ist aber unverzichtbar. Zudem braucht die Familie mindestens ein Fortbewegungmittel. Wenn sie in den Urlaub fahren will, kann sie sich kein Auto leisten. Wenn sie nicht in den Urlaub fahren, muß sie aber ein Moped kaufen, um den Sohn zu besänftigen.

Zeigen Sie, dass Familie A ein Moped und kein Auto anschafft, sofern Herr A seinen üblichen Bonus nicht bekommt.

Die Übersetzung in atomare Aussagen und aussagenlogische Formeln in Form von Klauseln liefert:

- “Aber falls Herr A seinen üblichen Bonus nicht bekommt, können sie sich nicht alles leisten.”

$$\neg B \Rightarrow \neg(W \wedge A \wedge M) \equiv \neg A \vee B \vee \neg M \vee W$$

- “Die Waschmaschine ist aber unverzichtbar.”

$$W$$

- “Die Familie braucht mindestens ein Fortbewegungsmittel.”

$$A \vee M$$

- “Wenn sie in den Urlaub fahren will, kann sie sich kein Auto leisten.”

$$U \Rightarrow \neg A \equiv \neg A \vee \neg U$$

- “Wenn sie nicht in den Urlaub fahren, muß sie aber ein Moped kaufen.”

$$\neg U \Rightarrow M \equiv M \vee U$$

Zusammen mit der Voraussetzung $\neg B$ liefert die entsprechende KNF

$$F = \neg B \wedge (\neg A \vee B \vee \neg M \vee \neg W) \wedge W \wedge (A \vee M) \wedge (\neg A \vee \neg U) \wedge (M \vee U)$$

bzw. in Mengenschreibweise

$$F = \{\neg B\}\{\neg A, B, \neg M, \neg W\}\{W\}\{A, M\}\{\neg A, \neg U\}\{M, U\}$$

iterativ folgende nicht-tautologische Resolventen

$$\begin{aligned}\operatorname{Res} F &= F\{\neg A, \neg M, \neg W\}\{\neg A, B, \neg M\}\{\neg A, B, U, \neg W\}\{M, \neg U\}\{\neg A, M\}\\ \operatorname{Res}^2 F &= \operatorname{Res} F\{\neg A, \neg M\}\{\neg A, U, \neg W\}\{\neg A, B, \neg U, \neg W\}\{\neg A, B, \neg W\}\\ &\qquad \{\neg A, B, U\}\{B, M, U, \neg W\}\{M\}\{\neg A, \neg U, \neg W\}\{\neg A, \neg W\}\\ &\qquad \{\neg A, B, \neg U\}\{\neg A, B\}\{\neg A, B, M, \neg W\}\end{aligned}$$

An dieser Stelle ist klar, dass die zusätzliche Voraussetzung $A$ die Resolvente $\neg M$ und dann $\varnothing$ liefert, ein Autokauf also nicht mit den übrigen Bedingungen verträglich ist.

Was den Kauf eines Mopeds angeht, so reicht es zu zeigen, dass $\neg M$ in den iterierten Resolventen von $F$ nicht auftritt:

$$\begin{aligned}\operatorname{Res}^3 F &= \operatorname{Res}^2 F\{\neg A, U\}\{M, U, \neg W\}\{\neg A\}\{\neg A, M, \neg W\}\{B, M, U\}\{\neg A, B, M\}\\ &\qquad \{B, M, U, \neg W\}\{B, M, \neg W\}\{M, \neg U, \neg W\}\{M, \neg W\}\{B, M, \neg U\}\\ &\qquad \{B, M\}\\ \operatorname{Res}^4 F &= \operatorname{Res}^3 F\end{aligned}$$

Also ist der Kauf eines Mopeds mit den anderen Bedingungen verträglich.

Leider ist die Resolutionsmethode nicht effizient, schon die obige Berechnung ist überraschend aufwendig (siehe aber auch Beispiel 6.6). Bei Formeln der Länge $O(n)$ kann bereits die Umwandlung in KNF eine Zeitkomplexität von $O(2^n)$ aufweisen. Aber selbst bei KNF-Formeln kann die Resolutionsmethode exponentiell viele Schritte in der Anzahl der Klauseln erfordern.

**Proposition 5.14.** *Die Anwendung der Resolutionsmethode kann eine Formel mit $n$ Klauseln in eine Formel mit $2^n - 1$ Klauseln verwandeln.*

*Beweis.* Wir konstruieren induktiv Formeln $F_k$ in denen die ersten $k$ Variablen vorkommen und die $k$ Klauseln haben, so dass Anwendung der Resolutionsmethode eine Formel mit $2^k - 1$ Klauseln liefert.

$k = 0$: $F_0 = \bot$ bzw. $\varnothing$. Die Resolutionsmethode liefert $2^0 - 1 = 0$ Klauseln.

Annahme: Die Konjunktion $F_k = \bigwedge_{i<k} K_{k,i}$ der Klauseln $K_{k,i}$, $i < k$, hat die gewünschte Eigenschaft.

$k + 1$: Definiere $F_{k+1}$ als Konjunktion der Klauseln $K_{k+1,i} := K_{k,i} \cup \{A_k\}$, $i < k$, mit $K_{k+1,k} := \{\neg A_k\}$.

Resolventenbildung für $F_{k+1}$ ohne Beteiligung von $K_{k+1,k}$ liefert $2^k - 1$ Klauseln, durch Hinzufügen von $A_k$ zu allen Resolventen von $F_k$. Aus diesen lassen sich unter Verwendung von $\{\neg A_k\}$ die ursprünglichen $2^k - 1$ Resolventen von $F_k$ gewinnen. Zusammen mit $K_{k+1,k}$ gibt das $2 \cdot \left(2^k - 1\right) + 1 = 2^{k+1} - 1$ viele Klauseln. □

In Kapitel 8 zeigen wir, dass die Resoltionsmethode zumindest für eine bestimmte Klasse von Formeln, die sogenannten „Hornformeln“, effizient ist.

# 6. Semantische Folgerungen

Ganz ähnlich, wie die Äquivalenz $\equiv$ den Junktor $\Leftrightarrow$ externalisiert hat, wollen wir auch mit dem Junktor $\Rightarrow$ für die Implikation verfahren. Die im Folgenden beschriebene Methode wird außerdem die Konjunktion $\wedge$ externalisieren.

**Definition 6.1.** Betracht eine Menge $\Gamma$ von Formeln und eine einzelne Formel $F$.

(a) Eine für $\Gamma$ passende Belegung $\alpha$ (siehe Definition 2.13) **erfüllt** $\Gamma$, wenn $\widehat{\alpha}$ jedes Element $G \in \Gamma$ auf 1 abbildet. Existiert keine derartige Belegung, so heißt $\Gamma$ **nicht erfüllbar**.

(b) Die Formel $F$ **folgt** aus der Menge $\Gamma$, geschrieben $\Gamma \models F$, wenn jede für $\Gamma \cup \{F\}$ passende und $\Gamma$ erfüllende Belegung auch $F$ erfüllt.

Ausdrücke der Form $\Gamma \models F$ heißen auch **semantische Sequenzen**, und die Elemente von $\Gamma$ werden als **Prämissen** bezeichnet.

Wir stellen fest, dass die Folge-Relation $\models$ letztendlich mittels der Inklusion zwischen bestimmten Mengen von Belegungen definiert ist. Insbesondere läßt sich die Nicht-erfüllbarkeit von $\Gamma$ durch $\Gamma \models \bot$ charakterisieren. In den HA wird bewiesen:

**Satz 6.2.** *$\Gamma \models F$ gilt genau dann, wenn $\Gamma \cup \{\neg F\}$ nicht erfüllbar ist.* □

**Notation 6.3.** Für endliche Mengen $\Gamma = \{G_0, \ldots, G_{n-1}\}$ lässt man die Mengenklammern weg und schreibt

$$G_0, \ldots, G_{n-1} \models F$$

**Satz 6.4.** *Für jede endliche Menge $\Gamma$ von Formeln und jede Formel $F$ sind folgende Aussagen der Metasprache äquivalent:*

(a) $\Gamma \models F$;

(b) $\bigwedge \Gamma \Rightarrow F$ *ist eine Tautologie;*

(c) $\bigwedge \Gamma \models F$.

(d) $\bigwedge \Gamma \wedge \neg F$ *ist nicht erfüllbar.*

*Beweis.* Die Implikation $\bigwedge \Gamma \Rightarrow F$ ist eine Tautologie wenn jede für $\Gamma \cup \{F\}$ passende Belegung $\alpha$ eine der folgenden Bedingungen erfüllt: $\widehat{\alpha}(\bigwedge \Gamma) = 0$ oder $\widehat{\alpha}(F) = 1$. Aber dies sagt genau, dass jede passende Belegung $\alpha$ mit $\widehat{\alpha}(\bigwedge \Gamma) = \bigwedge\{\, \widehat{\alpha}(G) \,:\, G \in \Gamma \,\} = 1$ auch $\widehat{\alpha}(F) = 1$ erfüllt, was gleichbedeutend mit $\widehat{\alpha}\,(\bigwedge \Gamma \wedge \neg F) = 0$ ist. Man beachte, dass $\bigwedge\{\, \widehat{\alpha}(G) \,:\, G \in \Gamma \,\} = 1$ genau dann gilt, wenn jedes $G \in \Gamma$ die Bedingung $\widehat{\alpha}(G) = 1$ erfüllt. □

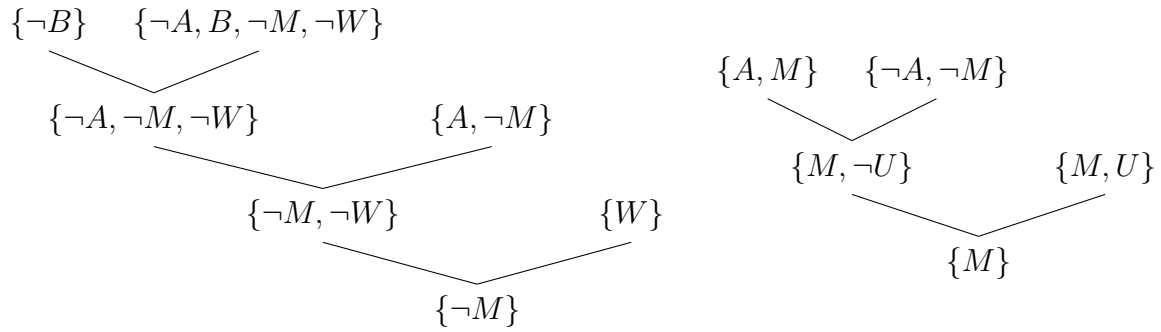
**Folgerung 6.5.** *Ist $G$ eine Formel in KNF und ist $F$ eine Resolvente von $G$, so folgt $F$ aus der Menge $\Gamma$ der Klauseln von $G$.*

*Beweis.* Nach dem Resolutionslemma 5.7 sind $G$ und $G \wedge F$ äquivalent, also ist $G \wedge \neg F = \bigwedge \Gamma \wedge \neg F$ nicht erfüllbar. □

**Beispiel 6.6.** Unter diesem Aspekt läßt sich Beispiel 5.13 etwas geschickter angehen. Wir fügen zu

$$F = \neg B \wedge (\neg A \vee B \vee \neg M \vee \neg W) \wedge W \wedge (A \vee M) \wedge (\neg A \vee \neg U) \wedge (M \vee U)$$

die negierte Schlußfolgerung $\neg(\neg A \vee M) \equiv A \vee \neg M$ als Klausel hinzu und versuchen, $\varnothing$ als Resolvente zu finden. Das gelingt recht schnell:




**Bemerkung 6.7.**

(a) Achtung: In Kapitel 5 hatten wir eine Mengen-Schreibweise für Klauseln verwendet, der implizit eine Disjunktion der Elemente zugrunde lag. Im Gegensatz dazu sind die Elemente der Mengen $\Gamma$ links des $\models$-Zeichens durch Konjunktion verbunden. Das Weglassen der Mengenklammern im endlichen Fall, siehe Notation 6.3, sollte helfen, Verwirrungen zu vermeiden.

(b) Die Folge-Relation ist unerwartet asymmetrisch. Es ist aber möglich, auch allgemeinere Sequenzen zu betrachten, bei denen auf der rechten Seite mehr als eine Formel vorkommen, etwa $A, B \models C, D$. Dies soll als $A \wedge B \models C \vee D$ verstanden werden, d.h., die Interpretation des Kommas ist „kontextsensitiv“: links von $\models$ dient es als Konjunktion, rechts von $\models$ als Disjunktion. Auf der syntaktischen Ebene, d.h., in der **Beweistheorie** (Kapitel 7) ist das beim Sequenzenkalkül für die klassische Logik von Vorteil, siehe etwa [Cal11].

Satz 6.4 gilt speziell für $\Gamma = \varnothing$. Das führt zu einer alternativen Charakterisierung von Tautologien.

**Folgerung 6.8.** *Eine Formel $F$ ist genau dann eine Tautologie, wenn $\models F$ gilt, was zu $\top \models F$, und somit zur Tautologie-Eigenschaft von $\top \Rightarrow F$ gleichwertig ist.* □

**Beispiel 6.9.**

(a) $A \wedge B \models A$.

(b) $A, B \models A \wedge B$.

(c) $A \Rightarrow B, A \models B$

**Satz 6.10.** *Die folgenden fünf Aussagen sind gleichwertig:*

(a) $A \wedge B \models F$

(b) $A, B \models F$

(c) $A \models B \Rightarrow F$

(d) $\models A \Rightarrow (B \Rightarrow F)$

(e) $\models A \wedge B \Rightarrow F$

*Beweis.* Die Gleichwertigkeit von (a) und (b), ebenso wie die von (c) und (d), bzw. von (e) und (a) folgt unmittelbar aus Satz 6.4.
Nachzuweisen bleibt die Gleichwertigkeit von (b) und (c), d.h., dass die Prämissen einzeln, und nicht nur zur Gänze nach rechts transportiert werden können. Aber aus Satz 6.4 folgt die Gleichwertigkeit von $B \models F$ mit $\models B \Rightarrow F$, eine Aussage die alle für $B$ und $F$ passenden Belegungen betrifft. Die Einschränkung auf diejenigen Belegungen, die zudem $A$ wahr machen, liefert die Behauptung. □

**Bemerkung 6.11.** Wir haben absichtlich in der Formulierung des Satzes den Begriff „gleichwertig“ anstelle des sonst üblichen „äquivalent“ verwendet, um potentielle Verwirrung zu vermeiden. Es wäre nämlich schon die dritte Variante von „Äquivalenz“ gewesen:

- ▷ Der Junktor $\Leftrightarrow$ in der Sprache der Aussagenlogik heißt „Äquivalenz“.
- ▷ Diesen Junktor hatten wir im vorigen Abschnitt externalisiert zu einer Relation $\equiv$, die auch als „Äquivalenz“ bezeichnet wurde. Dies ist Teil der sogenannten *Meta-Sprache*, mit der wir über die Aussagenlogik sprechen.
- ▷ Satz 6.10 vergleicht schließlich mehrere meta-sprachliche Aussagen hinsichtlich ihres Wahrheitsgehalts und stellt fest, dass sie alle denselben haben. Üblicherweise wird das als „Äquivalenz“ dieser meta-sprachlichen Aussagen formuliert, womit wir uns aber im Bereich der Meta-Meta-Sprache bewegen!

Diese Probleme könnten umgangen werden, wenn man die Sprach-Ebene mit einfließen läßt. Allerdings ist es auf Dauer doch zu umständlich, von „0-Äquivalenz“, „1-Äquivalenz“, „2-Äquivalenz“ usw. zu sprechen.

Im Hinblick auf die Sätze 6.4 und 6.10 stellt sich die Frage, wozu die Externalisierung $\models$ der Implikation $\Rightarrow$ überhaupt gut ist. Beachte, dass die Definition von $\Gamma \models F$ keine Größenbeschränkung für $\Gamma$ enthielt, insbesondere kann $\Gamma$ unendlich sein. Und dafür gibt es offenbar keine interne Entsprechung.

**Beispiel 6.12.**

(a) Die Menge $\Gamma$ möge aus allen Aussagen bestehen, die höchstens $\vee$ oder $\wedge$ als Junktoren enthalten. Gilt für Atome $A$ und $B$ die Sequenz

$$\Gamma \models A \Rightarrow B \quad ?$$

Jede Belegung, die $B$ wahr macht, macht auch $A \Rightarrow B$ wahr. Und falls eine Belegung alle Formeln aus $\Gamma$ wahr macht, macht sie insbesondere auch $B \in \Gamma$ wahr. Daher ist die obige Aussage korrekt.

(b) Wie steht es mit der Sequenz

$$\Gamma \models \bot$$

für die Menge $\Gamma$ aller Literale? Auch diese Sequenz gilt, weil keine Belegung existiert, die alle Formeln aus $\Gamma$ wahr macht: für jedes Atom $A$ enthält $\Gamma$ die Formeln $A$ sowie $\neg A$. Folglich ist für keine Belegung zu überprüfen, ob sie $\bot$ auf 1 abbildet.

**Bemerkung 6.13.** Für jede Formelmenge $\Gamma' \subseteq \Gamma$ ist es klar, dass aus der Gültigkeit von $\Gamma' \models F$ auch die von $\Gamma \models F$ folgt: jede Belegung, die alle Formeln aus $\Gamma$ wahr macht, macht alle Formeln aus $\Gamma'$ wahr und somit auch $F$.

Insofern werden Sequenzen der Form $\Gamma \models F$ umso schwächer, je größer die Mengen $\Gamma$ werden, weil mehr Voraussetzungen die Menge der zu betrachtenden Belegungen einschränken.

Nachdem wir in Beispiel 6.12 unendliche Mengen $\Gamma$ betrachtet haben, wollen jetzt beweisen, dass die Gültigkeit einer Sequenz $\Gamma \models F$ immer schon mit einer endlichen Untermenge von $\Gamma$ überprüft werden kann.

**Kompaktheitssatz 6.14.** *Wenn $\Gamma \models F$, existieren endlich viele Formeln $G_i \in \Gamma$, $i < n$, mit $G_0, \ldots, G_{n-1} \models F$.*

*Beweis.* Wir unterscheiden zwei Fälle:

(0) Falls alle Formeln aus $\Gamma \cup \{F\}$ nur endlich viele, etwa $k$, Atome enthalten, gibt es nach Bemerkung 3.8 $n \leq 2^{2^k}$ Formeln $G_i$ in $\Gamma$, so dass alle übrigen zu einer von diesen äquivalent sind. Dann ist aber $G_0, \ldots, G_{n-1} \models F$ gültig: jede Belegung, die $G_0, \ldots, G_{n-1}$ wahr macht, macht auch alle anderen Formeln in $\Gamma$ wahr und damit auch $F$.

(1) Ist hingegen die Menge aller Atome in den Formeln aus $\Gamma \cup \{F\}$ unendlich, etwa $\{ C_i : i \in \mathbb{N} \}$, beweisen wir die Contraposition: ist für jede endliche Teilmenge $\Gamma_0 \subseteq \Gamma$ die Sequenz $\Gamma_0 \models F$ ungültig, dann auch $\Gamma \models F$.

Für $k \in \mathbb{N}$ bezeichne $\Gamma_k$ die Menge alle Formeln aus $\Gamma$, in denen höchstens die Atome $C_i$, $i < k$, vorkommen. Die Contraposition von (0) zeigt, dass die Sequenz $\Gamma_k \models F$ nicht gültig ist. Damit existiert eine Belegung $\alpha_k$ der Atome $C_i$, $i < k$, mit

$$\widehat{\alpha}_k(F) = 0 \qquad \text{und} \qquad \widehat{\alpha}(G) = 1 \quad \text{für alle} \quad G \in \Gamma_k$$

Wir konstruieren nun eine Belegung $\alpha$ aller Variablen $C_i$, $i \in \mathbb{N}$, die alle Formeln in $\Gamma$ wahr macht, nicht aber $F$; folglich gilt $\Gamma \models F$ nicht. Wir setzen erst

$$\alpha(C_0) := \begin{cases} 1 & \text{falls } \alpha_k(C_0) = 1 \text{ für unendlich viele Indizes } k \\ 0 & \text{sonst} \end{cases}$$

Jetzt entfernen wir alle Belegungen $\alpha_k$ mit $\alpha(A_0) \neq \alpha_k(C_0)$ aus unserer Liste von Belegungen. Nach Konstruktion verbleiben unendlich viele Belegungen $\alpha_k$ und wir setzen

$$\alpha(C_1) := \begin{cases} 1 & \text{falls } \alpha_k(C_1) = 1 \text{ für unendlich viele der restlichen Indizes } k \\ 0 & \text{sonst} \end{cases}$$

usw. Die resultierende Belegung $\alpha$ hat folgende Eigenschaft: für jede Zahl $n$ existieren unendlich viele Indizes $k$, so dass die Werte $\alpha(C_0), \ldots, \alpha(C_{n-1})$ mit $\alpha_k(C_0), \ldots, \alpha_k(C_{n-1})$ übereinstimmen.

Da $F$ nur endlich viele Atome enthält, existiert ein $n$, so dass alle diese Atome zu $\{C_0, \ldots, C_{n-1}\}$ gehören. Wähle ein $k \geq n$ mit $\alpha(C_i) = \alpha_k(C_i)$ für $i < n$. Dann gilt $\widehat{\alpha}(F) = \widehat{\alpha}_k(F)$. Aus der Konstruktion der Belegung $\alpha_k$ folgt aber $\widehat{\alpha}(F) = 0$.

Analog wird jede Formel $G \in \Gamma$ von $\widehat{\alpha}$ auf 1 abgebildet, denn auch $G$ enthält nur endlich viele Atome und es gibt deswegen einen Index $l$ mit $\widehat{\alpha}(G) = \widehat{\alpha}_l(G) = 1$.

Die Belegung $\alpha$ realisiert somit den erwünschte Widerspruch zur Gültigkeit von $\Gamma \models F$. □

# 7. Beweistheorie

Die Externalisierung von Äquivalenz und Implikation waren semantischer Natur, da mittels Belegungen definiert. Wir wollen uns nun der Aufgabe zuwenden, *Beweise, wie sie in der Mathematik üblich sind,* zu formalisieren und zu analysieren. Diese operieren nicht semantisch, sondern versuchen *Schlussfolgerungen* über die Wahrheit von Formeln aus *Annahmen* über die Wahrheit bestimmter *Voraussetzungen* zu ziehen.

Aber Vorsicht: typische mathematische Beweise, wie man sie in Fachaufsätzen, Lehrbüchern oder Vorträgen findet, etwa der obige Beweis des Kompaktheitssatzes, wirken häufig informell und und sind mehr oder weniger verkürzt, gemäß den Vorkenntnissen der intendieren Leser/Hörer, die die fehlenden oder nur angedeuteten Schritte ergänzen können sollten. Dennoch folgen solche Beweise bestimmten Regeln, auf die sich die (meisten) Mathematiker verständigt haben.

Es gilt also zunächst, die typischen Schlußregeln solcher Beweise zu extrahieren, bevor man sie formalisieren kann, denn erst formale Beweise sind selber mathematischen Methoden zugänglich.

Diese Schlußregeln können ebenfalls mit Hilfe von Sequenzen formuliert werden; es handelt sich hier letztendlich um eine syntaktische Vorgehensweise, die unter den anfangs erwähnten Begriff des *logischen Kalküls* fällt. Es gibt aber auch andere Möglichkeiten.

In jedem Fall stellen sich unmittelbar zwei entscheidende Fragen:

- ▷ Ist alles wahr, was syntaktisch hergeleitet (und somit formal bewiesen) werden kann? Dann nennt man das Kalkül **korrekt**.
- ▷ Kann alles, was wahr ist, syntaktisch hergeleitet werden? Im positiven Fall nennt man das Kalkül **vollständig**.

Solange wir die Antworten auf diese Fragen nicht kennen, müssen wir zwischen syntaktischen und semantischen Sequenzen (vergl. Kapitel 6) unterscheiden. Entsprechend benötigen wir neben dem semantischen Sequenzen-Symbol $\models$ eine syntaktische Variante $\vdash$, die die **Herleitbarkeit** gemäß des Kalküls zum Ausdruck bringt. Damit lassen sich die Fragen nach der Korrektheit bzw. Vollständigkeit als Inklusionen dieser Relationen formulieren:

$$\text{Korrektheit:} \quad \models \subseteq \vdash \qquad \text{bzw.} \qquad \text{Vollständigkeit} \quad \models \supseteq \vdash$$

Erst wenn beide Fragen positiv beantwortet worden sind, also $\models \; = \; \vdash$ gilt, könnte man diese Unterscheidung wieder vergessen.

Im Rahmen der Prädikatenlogik bemühte man sich seit der zweiten Hälfte des 19. Jahrhundersts um geeignete Formalisierungen. Ursprünglich bediente man sich dabei der *axiomatischen Methode*, bei der man für das Gebiet von Interesse gewisse „offensichtliche“ oder „unwiderlegbare“ Grundannahmen, die sogenannten *Axiome*, festlegte, und dann versuchte, soviele korrekte Aussagen über dieses Gebiet aus den Axiomen herzuleiten, wie möglich [Hil99]. Ein weiteres bekanntes Beispiel dürfte der axiomatische Zugang zur Mengenlehre durch Peano sein. Hilberts Programm sah die vollständige Formalisierung der Mathematik vor. Mit der axiomatischen Methode verbinden sich neben dem Namen von Hilbert auch die von Frege und Łukasiewizc (dem Erfinder der „polnischen Notation“).

Leider zeigten es sich zu Beginn des 20. Jahrhunderts Schwierigkeiten, Hilberts Programm umzusetzen. 1930 konnte Kurt Gödel schließlich zeigen, dass die von Hilbert angestrebte Formalisierung im Rahmen jedes hinreichend ausdrucksstarken Systems prinzipiell nicht erreichbar ist. (Das betrifft zwar nicht die Aussagenlogik, aber die Prädikatenlogik.) Zudem modellierte die axiomatische Methode nicht wirklich die typische Arbeitsweise von Mathematikern. Sie war so umständlich, dass selbst ihre Protagonisten sie nicht konsequent verwendeten. Im Jahr 1934 erschienen unabhängig voneinander gleich zwei Vorschläge für ein System, das die tatsächliche Arbeitsweise von Mathematikern besser wiederspiegeln sollte: die sogenannte **natürliche Deduktion**: von Stanisław Jaśkowski [Jaś34] (mit Vorarbeiten seit mindestens 1929) und Gerhard Gentzen [Gen34] (basierend auf seiner Dissertation von 1933).

## 7.i Natürliche Deduktion (ND)

Die englische Wikipedia-Seite bietet einen einen ordentlichen Überblick über die Facetten dieses interessanten Themas und weiterführende Hinweise.

Anstatt alle Sätze mit wenigen Schlußregeln aus Axiomen herleiten zu wollen, werden bei der natürlichen Deduktion typischerweise Sätze unter bestimmten Voraussetzungen oder globalen Prämissen formuliert, die ggf. wieder entfernt und in die Schlußfolgerung integriert werden können. Dazu kommen eine Reihe von Schlußregeln, deren lokale Prämissen nicht nur Formeln, sondern in manchen Fällen auch ganze Herleitungen mit bestimmten vorgegebenen globalen Prämissen sein können, es werden aber keine Axiome benötigt.

Die natürliche Deduktion war zunächst für Beweise in der intuitionistischen Logik konzipiert. Durch Hinzufügen einer Schlußregel kann sie aber auf die klassische Logik erweitert werden. Beweise von Formeln, die klassisch gelten, intuitionistisch aber nicht, werden daher von manchen Autoren als “artifiziell” abqualifiziert. Was hauptsächlich als störend empfunden wird ist die Tatsache, dass Teilformeln, die im Beweis benötigt werden, nicht notwendig in der Schlußfolgerung auftreten müssen, sofern diese nicht intuitionistisch gültig ist. Das kann zudem die Suche nach einem Beweis erschweren, siehe etwa [D’A05].

Die Präsentation von Beweisen in natürlicher Deduktion kann im Wesentlichen auf zwei Weisen erfolgen: mittels einer Baumstruktur, mit zusätzlichen Markierungen von Teil-Herleitungen, die als lokale Prämissen dienen (was auf Gentzen zurückgeht), oder

in tabellarischer Form mit Annotationen, aus welchen früheren Zeilen die aktuelle Formel aufgrund einer Schlußregel hergeleitet wurde und welchen Gültigkeitsbereich (Skope) bestimmte Teilherleitungen haben (was auf Jaśkowski zurückgeht, dessen ursprüngliches Verfahren allerdings auch diagrammatischer Natur war). Beide Varianten der Darstellung sind im Laufe der Zeit von einer Reihe von Autoren optimiert und verfeinert worden, wobei die Baumdarstellung eher für die Erforschung des Kalküls nützlich ist, und die tabellearische Darstellung anwendungsorientierer zu sein scheint. Ab der Mitte des vergangenen Jahrhunderts war die natürliche Deduktion im angloamerikanischen Raum die Methode der Wahl, die in Lehrbüchern für Logik sehr weitgehende Verbreitung erfuhr, und damit Generationen von Logik-Nutzern (Philosophen wie Mathematikern und Informatikern) prägte. In Europa hielt sich die axiomatische Methode länger.

In der Fortsetzung [Gen35] des oben genannten Artikels stellte Gentzen auch noch eine zweite Beweismethode vor, den sogenannten **Sequenzenkalkül**. Ursprünglich als technisches Hilfmittel gedacht, hat sich dieser seit seiner modernen Präsentation durch Kleene [Kle52] zu einem ernstzunehmenden Konkurrenten der natürlichen Deduktion entwickelt, speziell im Bereich der klassischen Logik, siehe etwa [Cal11].

Folgende Aspekte finden sich in praktisch allen Spielarten der natürlichen Deduktion, auch wenn sie dieselbe nicht charakterisieren:

- ▷ Kaum Einschränkung auf bestimmte Junktoren oder Normalformeln.
- ▷ Die Regeln (inference rules) der Natürlichen Deduktion treten häufig paarweise auf, zur Einführung und zur Eliminierung von Junktoren. Erstere dienen dazu, Schlussfolgerungen aus einfacheren Voraussetzungen ziehen zu können, letztere zur Auflösung von Voraussetzungen in ihre Bestandteile. Allerdings erfolgt der "Informationsfluß" bei diesen Regeln in entgegengesetzter Richtung, was einer Automatisierung des Verfahren entgegensteht.
- ▷ Die Regeln sind „natürlich“ oder „intuitiv“ richtig, bilden also Denk- und Argumentationsmuster ab, die beim Verwendung natürlicher Sprache typisch sind. (Das ist natürlich hochgradig subjektiv.)

In der hier präsentierten Variante der Natürlichen Deduktion für die Aussagenlogik finden die Junktoren $\wedge$, $\vee$, $\neg$, $\Rightarrow$ und $\bot$ Verwendung. Nur $\Leftrightarrow$ wollen wir wie gewohnt durch $(A \Rightarrow B) \wedge (B \Rightarrow A)$ ersetzen. Die natürlichen Deduktionsregeln haben die Form

$$P_0 \vdash Q \qquad \text{oder} \qquad P_0, P_1 \vdash Q \qquad \text{oder} \qquad P_0, P_1, P_2 \vdash Q$$

wobei $P_i$ die Prämissen der Regel sind und $Q$ die Folgerung. Alternativ sind vertikale, spezifisch für eine Baum-Darstellung von Beweisen geeignete Varianten dieser Regeln verbreitet:

$$\frac{P_0}{Q} \qquad \text{oder} \qquad \frac{P_0 \quad P_1}{Q} \qquad \text{oder} \qquad \frac{P_0 \quad P_1 \quad P_2}{Q}$$

Interpretation: Aus jeder Obermenge $\Gamma$ der Prämissen läßt sich die Folgerung herleiten. In graphentheoretischer Hinsicht sollte man die „Bruchstriche“ als Knoten mit durch Prämissen gelabelten Input-Kanten und durch die Schlußfolgerung gelabelter Output-Kante interpretieren. (Das oben kurz erwähnte Sequentzen-Kalkül verwendet

zwar ebenfalls „Bruchstriche“, aber dort wird spezifiziert, wie eine oder mehrere Sequentzen im „Zähler“ in eine neue Sequenz im „Nenner“ transformiert werden dürfen. Während man die Sequenzen selber auch hier diagrammatisch interpretieren kann, ist das für die „Brüche“ im Sequenzenkalkül nicht der Fall.)

## 7.ii ND-Regeln für die Konjunktion

Gehören die Prämissen $G$ und $H$ zu $\Gamma$, dann läßt sich ihre Konjunktion aus $\Gamma$ herleiten, geschrieben $\Gamma \vdash G \wedge H$. Die zugehörige Regel formulieren wir für die Minimalversion von $\Gamma$

$$G, H \vdash G \wedge H$$

Diese Regel, nennen wir **Introduktion der Konjunktion** oder kürzer $(\wedge i)$. Sie wird typischerweise verwendet, um ein gewünschtes (Zwischen-)Ergebnis aus einfacheren Bausteinen zusammenzusetzen.

Wir benötigen jedoch auch eine Regel zur **Elimination der Konjunktion**, kurz $(\wedge e)$, etwa um eine komplizierte Voraussetzung in ihre Bestandteile zu zerlegen. Da unsere Sequenzen rechts des Relationszeichens nur eine Formel aufweisen, brauchen wir diese Regel in zwei Varianten. Zusammengefasst:

**Definition 7.1.** Die Regeln der Konjunktion lauten

$$G, H \vdash G \wedge H \ (\wedge i) \qquad \text{und} \qquad G \wedge H \vdash G \ (\wedge e) \quad \text{sowie} \quad G \wedge H \vdash H \ (\wedge e)$$

oder in Baum-Form

$$\frac{G \quad H}{G \wedge H}\ (\wedge i) \qquad \text{und} \qquad \frac{G \wedge H}{G}\ (\wedge e) \qquad \text{sowie} \qquad \frac{G \wedge H}{H}\ (\wedge e)$$

Mit diesen ersten Regeln der natürlichen Deduktion lässt sich bereits etwas beweisen.

**Beispiel 7.2.** Wir demonstrieren das Verfahren, indem wir Assoziativität der Konjunktion nachweisen. Konkret wollen wir die Gültigkeit der syntaktischen Sequenz $F \wedge (G \wedge H) \vdash (F \wedge G) \wedge H$ zunächst tabellarisch zeigen:

*Beweis.*

| | | |
|---|---|---|
| 1 | $F \wedge (G \wedge H)$ | Prämisse der Sequenz |
| 2 | $F$ | $(\wedge e)$, Zeile 1 |
| 3 | $G \wedge H$ | $(\wedge e)$, Zeile 1 |
| 4 | $G$ | $(\wedge e)$, Zeile 3 |
| 5 | $H$ | $(\wedge e)$, Zeile 3 |
| 6 | $F \wedge G$ | $(\wedge i)$, Zeilen 2 und 4 |
| 7 | $(F \wedge G) \wedge H$ | $(\wedge i)$, Zeilen 6 und 5 |

Der eigentliche Beweis steht links in nummerierten Zeilen, während die Kommentare auf der rechen Seite die angewendete Regel angeben und die Zeilen, in denen deren Voraussetzungen zu finden sind. Im Folgenden lassen wir das Wort „Zeile“ jedoch weg.

Und nun alternativ als Baum:

$$\dfrac{\dfrac{F \wedge (G \wedge H)}{F}\,(\wedge e) \qquad \dfrac{\dfrac{F \wedge (G \wedge H)}{G \wedge H}\,(\wedge e)}{G}\,(\wedge e)}{F \wedge G}\,(\wedge i) \qquad \dfrac{\dfrac{F \wedge (G \wedge H)}{G \wedge H}\,(\wedge e)}{H}\,(\wedge e)}{(F \wedge G) \wedge H}\,(\wedge i)$$

Man beachte, dass dieselbe Prämisse mehrfach verwendet werden kann, was in der Baumstruktur deutlicher zum Ausdruck kommt. Bäume werden zudem gern von unten nach oben entwickelt, da man die Anzahl der benötigten Zweige nicht vorher kennt.

**Beispiel 7.3.** Als weiteres Beispiel soll die Kommutativität von $\wedge$ bewiesen werden:

$$H \wedge G \vdash G \wedge H$$

*Beweis.*

| | | |
|---|---|---|
| 1 | $H \wedge G$ | Prämisse |
| 2 | $H$ | $(\wedge e), 1$ |
| 3 | $G$ | $(\wedge e), 1$ |
| 4 | $G \wedge H$ | $(\wedge i), 3, 2$ |

Alternativ als Baum:

$$\dfrac{\dfrac{H \wedge G}{G}\,(\wedge e) \qquad \dfrac{G \wedge H}{H}\,(\wedge e)}{G \wedge H}\,(\wedge i)$$

## 7.iii ND-Regeln der Implikation

Auch für die Implikation gibt es zwei Regeln. Die Elimination, klassisch bekannt als *modus ponens*, erlaubt uns aus den Voraussetzungen $G$ und $G \Rightarrow H$ die Schlussfolgerung $H$ zu ziehen, formal

$$G, G \Rightarrow H \vdash H \qquad \text{bzw} \qquad \dfrac{G \quad G \Rightarrow H}{H}$$

Umgekehrt ermöglicht es die Introduktionsregel, aus einem Beweis von $H$ spezifisch unter der Voraussetzung $G$, und evtl. unter weiteren vorher eingeführten Voraussetzungen, die Herleitbarkeit von $G \Rightarrow H$ ohne die spezifische Voraussetzung $G$ zu folgern. Während bisher die Prämissen direkt zu verwendet waren, handelt es sich hier um eine Regel mit „Fernwirkung“.

Ein Beispiel soll dies illustrieren: in Beispiel 7.3 haben wir einen Beweis in mehreren Schritten präsentiert, dass aus $H \wedge G$ die Formel $G \wedge H$ herleitbar ist, also $H \wedge G \vdash G \wedge H$. Dies erlaubt uns, die Voraussetzung in die Schlussfolgerung einzubeziehen, ohne die Zwischenschritte:

$$\vdash H \wedge G \Rightarrow G \wedge H$$

Dies als Regel zu formulieren erfordert eine neuartige Notation.

**Definition 7.4.** Die Regeln der Implikation lauten:

$$G, G \Rightarrow H \vdash H \ (\Rightarrow e) \qquad \text{sowie} \qquad [G] \ldots H \vdash G \Rightarrow H \ (\Rightarrow i)$$

oder in Baumform

$$\frac{G \quad G \Rightarrow H}{H}\ (\Rightarrow e) \qquad \text{sowie} \qquad \frac{\begin{array}{c}[G]\\ \vdots \\ H\end{array}}{G \Rightarrow H}\ (\Rightarrow i)$$

Bei der $\Rightarrow$-Introduktion steht der Klammer-Ausdruck links bzw. oben immer für eine ausgewähle Prämisse, hier $G$, eines formalen Beweis-Teils, dessen Ergebnis, hier $H$, nach den Punkten folgt. Dieser Beweis-Teil darf auch vorangegangene Ergebnisse von außerhalb verwenden. Aber nachdem die $\Rightarrow$-Introduktion$H$ ausgeführt worden ist, sind die „lokalen Variablen“ dieses markierten Beweis-Teils nicht mehr zugänglich! Bei zeilenweiser Schreibweise wird der durch Punkte angedeutete Gültigkeitsbereich (scope) der ausgewählten Prämisse in einen Kasten eingeschlossen. Das rechtfertigt den Namen *Kasten-Prämisse*; sie bedarf keiner externen Begründung. Achtung: es ist nicht nötig, dass die Kasten-Prämisse $G$ tatsächlich bei der Herleitung von $H$ verwendet wird. In der Baumdarstellung kann das zu Irritationen führen.

**Beispiel 7.5.** Wir beweisen die Herleitbarkeit der Sequenz

$$F \Rightarrow G, F \Rightarrow H \vdash F \Rightarrow G \wedge H$$

**Beweis:**

| | | |
|---|---|---|
| 1 | $F \Rightarrow G$ | Prämisse |
| 2 | $F \Rightarrow H$ | Prämisse |
| 3 | $F$ | **Kasten-Prämisse** |
| 4 | $G$ | $(\Rightarrow e), 1, 3$ |
| 5 | $H$ | $(\Rightarrow e), 2, 3$ |
| 6 | $G \wedge H$ | $(\wedge i), 4, 5$ |
| 7 | $F \Rightarrow G \wedge H$ | $(\Rightarrow i), 3-6$ |

Man beachte, dass in der letzten Zeile neben $(\Rightarrow i)$ alle Zeilen angegeben werden, über die sich der relevante Kasten erstreckt.

In Baumform läßt sich der Kasten schlecht realisieren:

$$\dfrac{\dfrac{[F] \quad F \Rightarrow G}{G}\ (\Rightarrow e) \qquad \dfrac{[F] \quad F \Rightarrow H}{H}\ (\Rightarrow e)}{\dfrac{G \wedge H}{F \Rightarrow G \wedge H}\ (\Rightarrow i)}\ (\wedge i)$$

Dafür erkennt man deutlicher, dass die Kasten-Prämisse (in eckigen Klammern) mehrfach verwendet wird.

**Hinweis:**

- ▷ Keine Zeile, die in einem Kasten liegt, darf außerhalb des Kastens (als Begründung) benutzt werden. In der Baum-Darstellung gilt dies für die Zweige von der relevanten Introduktionsregel bis zu den durch eckige Klammern markierten Auftreten der Kastenprämisse.
- ▷ Kästen dürfen hierarchisch verschachtelt werden. Innerhalb eines Kastens darf ein anderer eröffnet werden, dieser muss allerdings vor dem äußeren wieder geschlossen werden. D.h. die Linien der Kästen dürfen sich nicht kreuzen. Das ist in Baumform nur schwer darstellbar, wir werden es mit Farbe versuchen.

**Beispiel 7.6.**

$$F \wedge G \Rightarrow H \vdash F \Rightarrow (G \Rightarrow H)$$

*Beweis.*

| | | |
|---|---|---|
| 1 | $F \wedge G \Rightarrow H$ | Prämisse |
| 2 | $F$ | Kasten-Prämisse |
| 3 | $G$ | Kasten-Prämisse |
| 4 | $F \wedge G$ | $(\wedge i), 2, 3$ |
| 5 | $H$ | $(\Rightarrow e), 1, 4$ |
| 6 | $G \Rightarrow H$ | $(\Rightarrow i), 3-5$ |
| 7 | $F \Rightarrow (G \Rightarrow H)$ | $(\Rightarrow i), 2-6$ |

und als Baum:

$$\dfrac{\dfrac{\dfrac{[F] \quad [G]}{F \wedge G}\,(\wedge i) \qquad F \wedge G \Rightarrow H}{H}\,(\Rightarrow e)}{G \Rightarrow H}\,(\Rightarrow i)}{F \Rightarrow (G \Rightarrow H)}\,(\Rightarrow i)$$

**Beispiel 7.7.** Wir wollen die Gültigkeit der Sequenz $F \Rightarrow G \wedge H \vdash (H \Rightarrow G) \Rightarrow F$ beweisen:

| | | |
|---|---|---|
| 1 | $F \Rightarrow G \wedge H$ | Prämisse |
| 2 | $H \Rightarrow G$ | Kasten-Prämisse |
| 3 | | |
| 4 | ??? | |
| 5 | | |
| 6 | $F$ | |

Wie kann man von $H \Rightarrow G$ zu $F$ gelangen? Das scheint nicht zu funktionieren. In der Tat ist die Sequenz $F \Rightarrow G \wedge H \models (H \Rightarrow G) \Rightarrow F$ nicht gültig. Dies zeigt die Belegung $\alpha(F) = 0, \alpha(G) = 1 = \alpha(H)$, für die $\widehat{\alpha}(F \Rightarrow G \wedge H) = 1$ gilt, aber auch $\widehat{\alpha}(H \Rightarrow G) = 0$. Da sich unser Kalkül als korrekt herausstellen wird, kann man keine ungültigen Sequenzen beweisen. Bei Schwierigkeiten beim Beweis sollte man also durchaus überprüfen, ob die zu beweisende Sequenz evtl. ungültig ist.

## 7.iv Einschub: Natürliche Deduktion in der Praxis

Formale Beweise der Natürlichen Deduktion, die die Gültigkeit einer Sequenz nachweisen, operieren nach so strengen formalen Kriterien, dass im Prinzip ein Rechner diese nachvollziehen kann. In dieser Hinsicht ist die Baum-Notation klar im Nachteil gegenüber der tabellarischen Notation.

Ein formaler Beweis der Sequenz $\Gamma \vdash F$ ist, in Tabellenform, eine durchnummerierte Liste von Formeln, wobei $F$ in der letzten Zeile steht. Die Liste ist in hierarchische organisierten Kästen unterteilt, so dass die letzte Zeile zu keinem Kasten gehört. In jeder Zeile des Beweises steht eine Formel, die

- entweder in $\Gamma$ liegt (also eine Prämisse ist), oder
- frei wählbar als erste Zeile eines Kastens auftritt (Kasten-Prämisse), oder
- eine Begründung trägt, wie diese Zeile aus bestimmten davorliegenden Zeilen durch Anwendung einer spezifischen Regel folgt.

  Bezieht sich die Begründung der Zeile $i$ u.a. auf eine Zeile, die in einem Kasten liegt, selbst aber keine Kastenprämisse ist, dann muss auch Zeile $i$ in demselben Kasten liegen.

Allein schon die präzise Spezifikation eines formalen Beweises in Baumform wäre sehr mühsam. Andererseits spiegeln Bäume die innere Struktur eines Beweies recht gut wider, und man sieht besser, ob gewisse Prämissen mehrfach verwendet werden. Insofern kann die Baumform zumindest am Anfang beim Auffinden von Beweisen in Listenform nützlich sein. Für theoretische Überlegungen ist sie ohnehin vorzuziehen.

## 7.v Regeln der Disjunktion

Im Gegensatz zur Implikation ist die Introduktion für die Disjunktion klar, dafür ist die Elimination etwas schwieriger. Wenn $G$ oder $H$ zu den Voraussetzung gehört, soll $G \vee H$ herleitbar sein. Das ergibt zwei Varianten der Introduktionsregel.

Bei der Elimination nehmen wir an, dass $G \vee H$ zur Menge der Prämissen gehört. Welche Formeln $K$ sollen dann aus $\Gamma$ herleitbar sein? Dazu reicht es, einen Beweis von $K$ mit (Kasten-)Prämisse $G$ zu besitzen, sowie einen zweiten mit (Kasten-)Prämisse $H$, die jeweils nicht begründet werden müssen. Analog zur Introduktion der Implikation liefert dies eine Regel mit Fernwirkung. diesmal sogar in zweifacher Hinsicht:

**Definition 7.8.** Die Regeln der Disjunktion sind:

$$G \vdash G \vee H \ (\vee i) \quad \text{or} \quad H \vdash G \vee H \ (\vee i) \qquad \text{bzw.} \qquad \frac{G}{G \vee H}\,(\vee i) \quad \text{oder} \quad \frac{H}{G \vee H}\,(\vee i)$$

sowie

$$G \vee H, [G] \ldots K, [H] \ldots K \vdash K \qquad \text{bzw.} \qquad \frac{G \vee H \quad \begin{array}{c}[G]\\ \vdots \\ K\end{array} \quad \begin{array}{c}[H]\\ \vdots \\ K\end{array}}{K}\,(\vee \mathrm{e})$$

**Beispiel 7.9.** Die Implikation

$$G \wedge H \Rightarrow G \vee H$$

ist eine Tautologie. In der Tat, hier ist ein Beweis (ohne Prämissen):

| | | |
|---|---|---|
| 1 | $G \wedge H$ | Kasten-Prämisse |
| 2 | $G$ | $(\vee e), 1$ |
| 3 | $G \vee H$ | $(\vee i), 2$ |
| 4 | $G \wedge H \Rightarrow G \vee H$ | $(\Rightarrow i) 1 - 3$ |

In Baumform haben wir

$$\dfrac{\dfrac{\dfrac{[G \wedge H]}{G}\,(\wedge e)}{G \vee H}\,(\vee i)}{G \wedge H \Rightarrow G \vee H}\,(\Rightarrow i)$$

Dieser Baum hat nur einen Zweig, daher ist der Unterschied zur ersten Version gering.

**Beispiel 7.10.** Wir beweisen, dass $\vee$ kommutativ ist:

$$G \vee H \vdash H \vee G$$

| | | |
|---|---|---|
| 1 | $G \vee H$ | |
| 2 | $G$ | Kasten-Prämisse |
| 3 | $H \vee G$ | $(\vee i), 2$ |
| 4 | $H$ | Kasten-Prämisse |
| 5 | $H \vee G$ | $(\vee i), 4$ |
| 6 | $H \vee G$ | $(\vee e), 1, 4 - 5, 2 - 3$ |

In Baumform:

$$\dfrac{G \vee H \quad \dfrac{[G]}{H \vee G}\,(\vee i) \quad \dfrac{[H]}{H \vee G}\,(\vee i)}{H \vee G}\,(\vee e)$$

Während in der linearen Darstellung alle Prämissen außerhalb von Kästen links des Herleitungssymbols $\vdash$ auftreten, und die letzte Zeile als Schlussfolgerung erscheint, sind es in der Baumdarstellung die Blätter ohne eckige Klammern bzw. die Wurzel.

**Beispiel 7.11.** Wir beweisen eines der Distributivgesetze:

$$F \vee (G \wedge H) \vdash (F \vee G) \wedge (F \vee H)$$

| | | |
|---|---|---|
| 1 | $F \vee (G \wedge H)$ | Prämisse |
| 2 | $F$ | Kasten-Prämisse |
| 3 | $F \vee G$ | $(\vee i), 2$ |
| 4 | $G \wedge H$ | Kasten-Prämisse |
| 5 | $G$ | $(\wedge e), 4$ |
| 6 | $F \vee G$ | $(\vee i), 5$ |
| 7 | $F \vee G$ | $(\vee e), 1, 2-3, 4-6$ |
| 8 | $F$ | Kasten-Prämisse |
| 9 | $F \vee H$ | $(\vee i), 8$ |
| 10 | $G \wedge H$ | Kasten-Prämisse |
| 11 | $H$ | $(\wedge e), 10$ |
| 12 | $F \vee H$ | $(\vee i), 11$ |
| 13 | $F \vee H$ | $(\vee e), 1, 8-9, 10-12$ |
| 14 | $(F \vee G) \wedge (F \vee H)$ | $(\wedge i), 7, 13$ |

Der entsprechende Baum ufert etwas aus:

$$\dfrac{\dfrac{F \vee (G \wedge H) \quad \dfrac{[F]}{F \vee G}(\vee i) \quad \dfrac{\dfrac{[G \wedge H]}{G}(\wedge e)}{F \vee G}(\vee i)}{F \vee G}(\vee e) \quad \dfrac{F \vee (G \wedge H) \quad \dfrac{[F]}{F \vee H}(\vee i) \quad \dfrac{\dfrac{[G \wedge H]}{H}(\wedge e)}{F \vee H}(\vee i)}{F \vee H}(\vee e)}{(F \vee G) \wedge (F \vee H)}(\wedge i)$$

## 7.vi ND-Regeln der Negation und Absurdität

Die Introduktion der Negation $\neg G$ ist simpel, beruht aber wieder auf Fernwirkung: wir nehmen an, dass es uns gelungen ist, aus $G$ die Absurdität ($\bot$) herzuleiten, das erlaubt uns eine Herleitung von $\neg G$.

Wird die Negation eliminiert? Falls $G$ sowie $\neg G$ Prämissen in $\Gamma$ sind, ist $\Gamma$ nicht erfüllbar und erlaubt daher die Herleitung der Absurdität. Insofern fallen die Elimination von $\neg$ und die Introduktion von $\bot$ zusammen. Die Elimination von $\bot$ orientiert sich an der Tautologie $\bot \Rightarrow G$ für beliebiges $G$, die unter dem arg verkürzten Namen „ex falso quodlibet“ bekannt ist, frei übersetzt „aus Falschem läßt sich Beliebiges herleiten“, wobei „sequitur“ unter den Tisch gefallen ist.

Leider genügt dies noch nicht, um doppelte Negationen zu eliminieren, das erfordert eine Extra-Regel, die leider das bisherige Schema der paarweisen Introduktions- und Eliminationsregeln für die Junktoren bricht.

**Definition 7.12.** Die Regeln für Negation und Absurdität lauten

$$[G]\dots\bot \vdash \neg G \ (\neg i) \text{ bzw. } \dfrac{\begin{array}{c}[G]\\ \vdots\\ \bot\end{array}}{\neg G}\ (\neg i)$$

sowie

$$G, \neg G \vdash \bot \ (\neg e) \qquad \text{bzw.} \qquad \dfrac{G \quad \neg G}{\bot}\ (\neg e)$$

und schließlich

$$\neg\neg G \vdash G \ (\neg\neg e) \quad \text{bzw.} \quad \dfrac{\neg\neg G}{G}\ (\neg\neg e) \qquad \text{sowie} \qquad \bot \vdash G \ (\bot e) \quad \text{bzw.} \quad \dfrac{\bot}{G}\ (\bot e)$$

**Beispiel 7.13.** Wir beweisen die Gültigkeit der Sequenz

$$F \Rightarrow G \models \neg G \Rightarrow \neg F$$

**Beweis:**

| | | |
|---|---|---|
| 1 | $F \Rightarrow G$ | Prämisse |
| 2 | $\neg G$ | Kasten-Prämisse |
| 3 | $F$ | Kasten-Prämisse |
| 4 | $G$ | $(\Rightarrow e), 1, 3$ |
| 5 | $\bot$ | $(\neg e), 4, 2$ |
| 6 | $\neg F$ | $(\neg i), 3-5$ |
| 7 | $\neg G \Rightarrow \neg F$ | $(\Rightarrow i), 2-6$ |

Der äußere Kasten dient der Regel $(\Rightarrow i)$, der innere der Regel $(\neg i)$.

Im zugehörigen Baum sind die Scoping-Regeln besonders unübersichtlich:

$$\dfrac{\dfrac{[\neg G] \qquad \dfrac{[F] \quad F \Rightarrow G}{G}\ (\Rightarrow e)}{\bot}\ (\neg e)}{\dfrac{\bot}{\neg F}\ (\neg i)}\ (\Rightarrow i)$$

**Beispiel 7.14.** Obwohl $(\neg\neg e)$ eine separate Regel ist, brauchen wir für $(\neg\neg i)$ keine eigene Regel, denn die Herleitbarkeit von $\neg\neg G$ aus $G$ können wir beweisen:

| | | |
|---|---|---|
| 1 | $G$ | Prämisse |
| 2 | $\neg G$ | Kasten-Prämisse |
| 3 | $\bot$ | $(\neg e), 1, 2$ |
| 4 | $\neg\neg G$ | $(\neg i), 2-3$ |

oder in Baumform

$$\dfrac{\dfrac{G \quad [\neg G]}{\bot}\ (\neg e)}{\neg\neg G}\ (\neg i)$$

**Beispiel 7.15.** Wir zeigen

$$\neg(G \vee H) \vdash \neg G \wedge \neg H$$

| | | |
|---|---|---|
| 1 | $\neg(G \vee H)$ | Prämisse |
| 2 | $G$ | Kasten-Prämisse |
| 3 | $G \vee H$ | $(\vee i), 2$ |
| 4 | $\bot$ | $(\neg e), 3, 1$ |
| 5 | $\neg G$ | $(\neg i) 2 - 4$ |
| 6 | $H$ | Kasten-Prämisse |
| 7 | $G \vee H$ | $(\vee i), 6$ |
| 8 | $\bot$ | $(\neg e), 7, 1$ |
| 9 | $\neg H$ | $(\neg i), 6 - 8$ |
| 10 | $\neg G \wedge \neg H$ | $(\wedge i), 5, 9$ |

Hier ist derselbe Beweis in Baumform:

$$\dfrac{\dfrac{\neg(G \vee H) \quad \dfrac{[G]}{G \vee H}\,(\vee i)}{\dfrac{\bot}{\neg G}\,(\neg i)} (\neg e) \qquad \dfrac{\neg(G \vee H) \quad \dfrac{[H]}{G \vee H}\,(\vee i)}{\dfrac{\bot}{\neg H}\,(\neg i)} (\neg e)}{\neg G \wedge \neg H}\,(\wedge i)$$

**Beispiel 7.16.** Für eine der De Morganschen-Regeln benötigen wir neben Beispiel 7.15 noch einen Beweis der umgekehrten Richtung

$$\neg G \wedge \neg H \vdash \neg(G \vee H)$$

| | | |
|---|---|---|
| 1 | $\neg G \wedge \neg H$ | Prämisse |
| 2 | $G \vee H$ | Kasten-Prämisse |
| 3 | $G$ | Kasten-Prämisse |
| 4 | $\neg G$ | $(\wedge e), 1$ |
| 5 | $\bot$ | $(\neg e), 3, 4$ |
| 6 | $H$ | Kasten-Prämisse |
| 7 | $\neg H$ | $(\wedge e), 1$ |
| 8 | $\bot$ | $(\neg e), 6, 7$ |
| 9 | $\bot$ | $(\vee e), 2, 3 - 5, 6 - 8$ |
| 10 | $\neg(G \vee H)$ | $(\neg i), 2 - 9$ |

oder in Baumform

$$\dfrac{\dfrac{[G\vee H] \quad \dfrac{[G]\quad \dfrac{\neg G\wedge\neg H}{\neg G}\,(\wedge e)}{\bot}\,(\neg e) \qquad \dfrac{[H]\quad \dfrac{\neg G\wedge\neg H}{\neg H}\,(\wedge e)}{\bot}\,(\neg e)}{\bot}\,(\vee e)}{\dfrac{\bot}{\neg(G\vee H)}\,(\neg i)}$$

**Beispiel 7.17.** Eine Herleitung von $\neg(F\wedge G)\vdash\neg F\vee\neg G$ erhält man unter Verwendung des Beweises in Beispiel 7.15 wie folgt:

| | | |
|---|---|---|
| 1 | $\neg(F\wedge G)$ | |
| 2 | $\neg(\neg F\vee\neg G)$ | Kasten-Prämisse |
| 3 | $\neg F$ | Kasten-Prämisse |
| 4 | $\neg F\vee\neg G$ | $(\vee i), 3$ |
| 5 | $\bot$ | $(\neg e), 4, 2$ |
| 6 | $\neg\neg F$ | $(\neg i) 3-5$ |
| 7 | $F$ | $(\neg\neg e), 6$ |
| 8 | $\neg G$ | Kasten-Prämisse |
| 9 | $\neg F\vee\neg G$ | $(\vee i), 8$ |
| 10 | $\bot$ | $(\neg e), 9, 2$ |
| 11 | $\neg\neg G$ | $(\neg i), 8-10$ |
| 12 | $G$ | $(\neg\neg e), 11$ |
| 13 | $F\wedge G$ | $(\wedge i), 7, 12$ |
| 14 | $\bot$ | $(\neg e), 13, 1$ |
| 15 | $\neg\neg(\neg F\vee\neg G)$ | $(\neg i), 2-14$ |
| 16 | $\neg F\vee\neg G$ | $(\neg\neg e), 15$ |

Die Zeilen 2–13 entsprechen im Wesentlichen dem Beweis von 7.15 mit $\neg F$ anstelle von $G$ und $\neg G$ anstelle von $H$, ausgenommen die Zeilen 7 und 12, wo die doppelte Negation sofort eliminiert wird, statt sich später mit $\neg\neg F\wedge\neg\neg G$ herumzuplagen. Letzteres in $F\wedge G$ umzuwandeln erfordert mehr Aufwand, als die Bausteine gleich zu behandeln. Hier ist der zugehörige Baum:

$$\dfrac{\dfrac{\neg(F\wedge G) \qquad \dfrac{\dfrac{\dfrac{[\neg(\neg F\vee\neg G)] \quad \dfrac{[\neg F]}{\neg F\vee\neg G}\,(\vee i)}{\bot}\,(\neg e)}{\neg\neg F}\,(\neg i)}{F}\,(\neg\neg e) \qquad \dfrac{\dfrac{\dfrac{[\neg(\neg F\vee\neg G)] \quad \dfrac{[\neg G]}{\neg F\vee\neg G}\,(\vee i)}{\bot}\,(\neg e)}{\neg\neg G}\,(\neg i)}{G}\,(\neg\neg e)}{F\wedge G}\,(\wedge i)}{\bot}\,(\neg e)}{\dfrac{\bot}{\neg\neg(\neg F\vee\neg G)}\,(\neg i)}{\neg F\vee\neg G}\,(\neg\neg e)$$

Grundsätzlich ist man an der Wiederverwendbarkeit natürlicher Deduktionsbeweise interessiert, modulo geeigneter Substitutionen. Das gilt für beide Arten der Darstellung. Streamlining kann später erfolgen.

**Beispiel 7.18.** Wir zeigen $(F \vee G) \wedge (F \vee H) \vdash F \vee (G \wedge H)$

| | | |
|---|---|---|
| 1 | $(F \vee G) \wedge (F \vee H)$ | Prämisse |
| 2 | $F \vee G$ | $(\wedge e), 1$ |
| 3 | $F \vee H$ | $(\wedge e), 2$ |
| 4 | $\neg(F \vee (G \wedge H))$ | Kasten-Prämisse |
| 12 | $\vdots$ | (8 Zeilen aus Beispiel 7.15) |
| 13 | $\neg F \wedge \neg(G \wedge H)$ | |
| 14 | $\neg F$ | $(\wedge e), 13$ |
| 15 | $\neg(G \wedge H)$ | $(\wedge e), 13$ |
| 29 | $\vdots$ | (14 Zeilen aus Beispiel 7.17) |
| 30 | $\neg G \vee \neg H$ | |
| 31 | $\neg G$ | Kasten-Prämisse |
| 32 | $F$ | Kasten-Prämisse |
| 33 | $G$ | Kasten-Prämisse |
| 34 | $\bot$ | $(\neg e), 33, 31$ |
| 35 | $F$ | $(\bot e), 34$ |
| 36 | $F$ | $(\vee e), 2, 32, 33 - 35$ |
| 37 | $\neg H$ | Kasten-Prämisse |
| 38 | $F$ | Kasten-Prämisse |
| 39 | $H$ | Kasten-Prämisse |
| 40 | $\bot$ | $(\neg e), 39, 37$ |
| 41 | $F$ | $\bot, 40$ |
| 42 | $F$ | $(\vee e), 3, 38, 39 - 41$ |
| 43 | $F$ | $(\vee e), 30, 31 - 35, 37 - 42$ |
| 44 | $\bot$ | $(\neg e), 43, 14$ |
| 45 | $\neg\neg(F \vee (G \wedge H))$ | $(\neg i), 4 - 46$ |
| 46 | $F \vee (G \wedge H)$ | $(\neg\neg e), 45$ |

Aus offensichtlichen Gründen ersparen wir uns hier den Baum.

# 7.vii Zusammenfassung der ND-Regeln

**Zusammenfassung der Regeln der Natürlichen Deduktion**

| | Introduktion | Elimination |
|---|---|---|
| Konjunktion | $\dfrac{G \quad H}{G \wedge H}$ | $\dfrac{G \wedge H}{G}$ sowie $\dfrac{G \wedge H}{H}$ |
| | $G, H \vdash G \wedge H$ | $G \wedge H \vdash G \ , \ G \wedge H \vdash H$ |
| Implikation | $\dfrac{\begin{array}{c}[G] \\ \vdots \\ H\end{array}}{G \Rightarrow H}$ | $\dfrac{G \quad G \Rightarrow H}{H}$ |
| | $[G] \ldots H \vdash G \Rightarrow H$ | $G, G \Rightarrow H \vdash H$ |
| Disjunktion | $\dfrac{G}{G \vee H}$ sowie $\dfrac{H}{G \vee H}$ | $\dfrac{G \vee H \quad \begin{array}{c}[G] \\ \vdots \\ K\end{array} \quad \begin{array}{c}[H] \\ \vdots \\ K\end{array}}{K}$ |
| | $G \vdash G \vee H \ , \ H \vdash G \vee H$ | $G \vee H, [G] \ldots K, [H] \ldots K \vdash K$ |
| Negation | $\dfrac{\begin{array}{c}[G] \\ \vdots \\ \bot\end{array}}{\neg G}$ | $\dfrac{G \quad \neg G}{\bot}$ und $\dfrac{\neg\neg G}{G}$ und $\dfrac{\bot}{G}$ |
| | $[G] \ldots \bot \vdash \neg G$ | $G, \neg G \vdash \bot \ , \ \neg\neg G \vdash G \ , \ \bot \vdash G$ |

**Praktische Aspekte der Natürlichen Deduktion**

- ▷ Bei der natürlichen Deduktion werden Formeln rein syntaktisch als Zeichenketten behandelt, die nach den obigen 12 Methoden transformiert werden dürfen. Es dürfen keine semantischen Äquivalenzen verwendet werden, also darf man z.B. $F \Rightarrow G$ nicht direkt durch $G \vee \neg F$ ersetzen; stattdessen muß man $F \Rightarrow G \vdash G \vee \neg F$ ebenfalls syntaktisch herleiten. (Einmal hergeleitet, darf man dieses Ergebnis natürlich später wiederverwenden.)
- ▷ Der häufigste Fehler bei der natürlichen Deduktion ist die Verwendung „lokalen Variablen“ außerhalb ihres Gültigkeitsbereichs (scope). Letzterer ist in der Baumdarstellung gelegentlich schlecht auszumachen, dafür zeigt die Kastendarstellung die Struktur des Beweises weniger deutlich. Die Kombination beider Methoden zur Konstruktion eines Beweises dürfte am erfolgversprechendsten sein, aber letztendlich bleibt die Methode Ihrer Wahl Geschmackssache.

▷ In der Klausur dürfen nur die obigen Regeln verwendet werden, also keine Regeln, die in der Vorlesung oder Übung daraus abgeleitet worden sind. Letztere sind bei Bedarf neu abzuleiten. Auch kann in der Klausur das Endformat, Baum oder Kastendarstellung, vorgeschrieben sein.

**Definition 7.19.** Eine Formel $F$ ist aus einer Menge $\Gamma$ von Formeln **herleitbar**, wenn eine Herleitung von $F$ existiert, deren sämtliche Voraussetzungen zu $\Gamma$ gehören. (Solche in eckigen Klammern, d.h., Kasten-Prämissen, gehören nicht dazu.) $F$ heißt **Theorem**, falls $\vdash F$ gilt.

## 7.viii Korrektheit und Vollständigkeit der Natürlichen Deduktion

Wir wenden uns nun den zu Beginn angesprochenen Fragen zu, die den Zusammenhang zwischen $\models$ und $\vdash$ betreffen. Da die Herleitbarkeit $\vdash$ durch die Existenz einer Herleitung und somit rekursiv definiert ist, bietet sich für den Nachweis der Korrektheit ein Induktionsbeweis an.

**Satz 7.20** (Korrektheit)**.** *Aus* $\Gamma \vdash F$ *folgt* $\Gamma \models F$*.*

*Beweis.* Wir verwenden die Baumdarstellung der Herleitungen und überprüfen, ob die obigen Regeln Wahrheit erhalten.

▷ Hat der Herleitungsbaum nur einen Knoten, so gehört $F$ zwangsläufig zu $\Gamma$, so dass automatisch $\Gamma \models F$ folgt.

($\wedge i$) $R$ bzw. $S$ seien Ableitungsbäume für $G$ bzw. $H$, und für alle Mengen $\Omega$ bzw. $\Delta$, die die Prämissen dieser Bäume enthalten, gelte $\Omega \models G$ und $\Delta \models H$.

$\Gamma$ sei eine Formelmenge, zu der sämtliche Prämissen des entsprechenden ($\wedge i$)-Baums $T$ gehören.

Wählt man für $\Omega$ und $\Delta$ die minimalen Formelmengen, d.h., genau die Mengen der Prämissen von $R$ bzw. $S$, so folgt $\Gamma \supseteq \Omega \cup \Delta$.

Nach Induktionsvoraussetzung gilt $\Gamma \models G$ sowie $\Gamma \models H$. Jede $\Gamma$ erfüllende Belegung $\alpha$ erfüllt sowohl $\Omega$ als auch $\Delta$, folglich gilt $\widehat{\alpha}(G) = \widehat{\alpha}(H) = 1$, also auch $\widehat{\alpha}(G \wedge H) = 1$. Daraus schließen wir $\Gamma \models G \wedge H$.

($\wedge e$) $T$ sei ein Ableitungsbaum für $G \wedge H$ und jede Menge $\Delta$, die alle Prämissen von $T$ enthält, erfülle $\Delta \models G \wedge H$.

Der zugehörige Ableitungsbaum für $G$ hat dieselben Prämissen wie $T$. Sind diese in $\Gamma$ enthalten, so gilt für jede $\Gamma$ erfüllende Belegung $\alpha$, dass $\widehat{\alpha}$ sowohl $G \wedge H$ als auch $G$ und $H$ auf 1 abbildet. Also gilt $\Gamma \models G$ sowie $\Gamma \models H$.

($\Rightarrow i$) $T$ sei ein Herleitungsbaum für $H$, so dass für jede Formelmenge $\Delta$, die alle Prämissen des Baums enthält, $\Delta \models H$ gilt.

Der zugehörige Ableitungsbaum $S$ für $G \Rightarrow H$ hat bis auf evtl. $G$ dieselben Prämissen wie $T$. $\Gamma$ sei eine Formelmenge, die die Prämissen von $S$ enthält. Betrachte eine $\Gamma$ erfüllende Belegungen $\alpha$, die auch für $G$ paßt.

- ▷ Falls $\widehat{\alpha}(G) = 1$, erfüllt $\alpha$ alle Prämissen von $T$. Nach Voraussetzung gilt dann $\widehat{\alpha}(H) = 1$, folglich auch $\widehat{\alpha}(G \Rightarrow H) = 1$.
- ▷ Falls $\widehat{\alpha}(G) = 0$, gilt für jeden Wert $\widehat{\alpha}(H)$ immer $\widehat{\alpha}(G \Rightarrow H) = 1$.

In jedem Fall gilt also $\Gamma \models G \Rightarrow H$.

($\Rightarrow e$) $R$ bzw. $S$ seien Ableitungsbäume für $G$ bzw. $G \Rightarrow H$, und für alle Mengen $\Omega$ bzw. $\Delta$, die die Prämissen dieser Bäume enthalten, gelte $\Omega \models G$ und $\Delta \models G \Rightarrow H$.

$\Gamma$ sei eine Formelmenge, zu der sämtliche Prämissen des entsprechenden ($\Rightarrow e$)-Baums $T$ gehören.

Wählt man für $\Omega$ und $\Delta$ die minimalen Formelmengen, d.h., genau die Mengen der Prämissen von $R$ bzw. $S$, so folgt $\Gamma \supseteq \Omega \cup \Delta$.

Nach Induktionsvoraussetzung gilt $\Gamma \models G$ sowie $\Gamma \models G \Rightarrow H$. Jede $\Gamma$ erfüllende Belegung $\alpha$ erfüllt sowohl $\Omega$ als auch $\Delta$, folglich gilt $\widehat{\alpha}(G) = \widehat{\alpha}(G \Rightarrow H) = 1$, also auch $\widehat{\alpha}(H) = 1$. Daraus schließen wir $\Gamma \models G \Rightarrow H$.

Die Argumente in den restlichen Fällen sind ähnlich und dem Leser überlassen. □

Aufgrund dieses Ergebnisses sind Theoreme, also ohne Voraussetzungen herleitbare Formeln (siehe Definition 7.19), automatisch Tautologien. Kontraposition erlaubt uns auch nachzuweisen, das manche Formeln keine Theoreme sind, einfach indem man zeigt, dass sie keine Tautologien sind.

**Beispiel 7.21.** Die Formel

$$T_0 := G \Rightarrow (H \Rightarrow G)$$

ist eine Tautologie. Hier ist der Beweis für $\models T_0$:

| | | |
|---|---|---|
| 1 | $G$ | Kasten-Prämisse |
| 2 | $H$ | Kasten-Prämisse |
| 3 | $G \wedge G$ | $(\wedge i), 1, 1$ |
| 4 | $G$ | $(\wedge e), 3$ |
| 5 | $H \Rightarrow G$ | $(\Rightarrow i), 2-4$ |
| 6 | $G \Rightarrow (H \Rightarrow G)$ | $(\Rightarrow i), 1-5$ |

**Beispiel 7.22.** Die Formel

$$T_1 := (F \Rightarrow (G \Rightarrow H)) \Rightarrow ((F \Rightarrow G) \Rightarrow (F \Rightarrow H))$$

ist ebenfalls eine Tautologie:

| | | |
|---|---|---|
| 1 | $F \Rightarrow (G \Rightarrow H)$ | Kasten-Prämisse |
| 2 | $F \Rightarrow G$ | Kasten-Prämisse |
| 3 | $F$ | Kasten-Prämisse |
| 4 | $G$ | $(\Rightarrow e), 2, 3$ |
| 5 | $G \Rightarrow H$ | $(\Rightarrow e), 1, 3$ |
| 6 | $H$ | $(\Rightarrow e), 5, 4$ |
| 7 | $F \Rightarrow H$ | $(\Rightarrow i), 3 - 6$ |
| 8 | $(F \Rightarrow G) \Rightarrow (F \Rightarrow H)$ | $(\Rightarrow i), 2 - 7$ |
| 9 | $(F \Rightarrow (G \Rightarrow H)) \Rightarrow ((F \Rightarrow G) \Rightarrow (F \Rightarrow H))$ | $(\Rightarrow i), 1 - 8$ |

**Beispiel 7.23.** Die Tautologie **„Beweis durch Widerspruch“:**

$$T_2 := (\neg G \Rightarrow \neg H) \Rightarrow (H \Rightarrow G)$$

Ihr Beweis ist analog zum Beweis in Beispiel 7.11.

Die Umkehrung des Korrektheitssatzes ist schwieriger zu beweisen, siehe z.B. [HR09, Satz 1.4.4].

**Satz 7.24** (Vollständigkeit)**.** *Aus* $\Gamma \models F$ *folgt* $\Gamma \vdash F$. □

# 7.ix Hilberts Axiomatisierung

Wie in der Einführung erwähnt, hat David Hilbert zu Beginn des letzten Jahrhunderts einen anderen Stil für formale Beweise in der Aussagenlogik entwickelt: Er benutzt nur eine Regel, den Modus Ponens $(\Rightarrow e)$, in Verbindung mit einer Reihe sogenannter *Axiome*. Dabei handelt es sich um vorgegebene Formeln bzw. Formel-Schemata, die intuitiv als gültig angesehen werden. Aus diesen Axiomen können alle anderen Tautologien mittels $(\Rightarrow e)$ hergeleitet werden. Sein System benutzte nur die Junktoren $\Rightarrow$ und $\neg$. Diese sind adäquat (siehe Definition 3.9). Wie in Beispiel 3.10 gesehen, lassen sich Konjunktion und Disjunktion wie folgt darstellen:

$$F \vee G \equiv \neg F \Rightarrow G \qquad \text{sowie} \qquad F \wedge G \equiv \neg(F \Rightarrow \neg G)$$

Hilbert verwendete nur drei Axiome, nämlich die Tautologien $T_0$, $T_1$, $T_2$ aus den vorangegangenen Beispielen 7.21–7.23.

**Satz 7.25.** *Hilberts Kalkül ist vollständig: Jede Tautologie* $F$ *hat einen Beweis der Sequenz*

$$T_0, T_1, T_2 \vdash F$$

*der nur Modus Ponens benutzt.*

# 8. Hornlogik

## 8.i Hornformeln

Die Hornlogik beschränkt sich auf eine Untermenge der Formeln der Aussagenlogik in KNF, für die die Resolutionsmethode effizient funktioniert. Diese Formeln sind zudem für die Logik-Programmierung wichtig und werden etwa von der Programmiersprache PROLOG verwendet. Erinnern wir uns, dass ein Literal entweder positiv ist, d.h., eine atomare Aussage, oder die Negation einer atomaren Aussage.

**Definition 8.1** (vergl. Alfred Horn, 1951)**.** Eine **Hornformel** ist eine Formel in KNF, die in jeder Klausel höchstens ein positives Literal und kein Literal doppelt enthält.

Wir unterscheiden folgende Typen von Klauseln:

(a) **Tatsachen** ohne negative Literale, die folglich atomar sind;

(b) **Regeln** mit genau einem positiven Literal, die somit zu Implikationen äquivalent sind, etwa $B \vee \neg A_0 \vee \neg A_2 \vee \cdots \vee \neg A_{n-1} \equiv A_0 \wedge A_2 \wedge \ldots \wedge A_{n-1} \Rightarrow B$;

(c) **Frageklauseln** ohne positives Literal, die zur Negation einer Co-Klausel äquivalent sind, etwa $\neg A_0 \vee \neg A_2 \vee \cdots \vee \neg A_{n-1} \equiv \neg(A_0 \wedge A_2 \wedge \ldots \wedge A_{n-1})$. Die Gültigkeit dieser Co-Klausel ist letztendlich Gegenstand der zugrundeliegenden Frage. Sie ist gegeben, wenn die Gesamtformel mit der Negation der Co-Klausel *nicht* erfüllbar ist.

**Definition 8.2.** Die sogenannte **Hornlogik** hat dieselbe Sprache wie die Aussagenlogik, mit einer auf Hornformeln eingeschränkten Syntax. Die Semantik entspricht der Semantik der Aussagenlogik.

**Beispiel 8.3.** Die Formel $F = Q \wedge (Q \wedge S \Rightarrow P) \wedge (Q \wedge R \Rightarrow P) \wedge (\neg R \vee \neg S)$ ist strenggenommen keine Hornformel, aber sie ist zu der Hornformel

$$Q \wedge (\neg Q \vee \neg S \vee P) \wedge (\neg Q \vee \neg R \vee P) \wedge (\neg R \vee \neg S)$$

äquivalent. Diese können wir als eine Frage nach der Gültigkeit von $R \wedge S$ verstehen, unter der Voraussetzung, dass $Q$ eine Tatsache und somit atomar ist, und die Implikationen $Q \wedge S \Rightarrow P$ sowie $Q \wedge R \Rightarrow P$ gelten. Die Formel $F$ ist tatsächlich erfüllbar: für $\alpha(P) = \alpha(Q) = 1,\ \alpha(S) = \alpha(R) = 0$ hat sie Wert 1. Daher ist die Frage nach der Gültigkeit von $R \wedge S$ negativ zu beantworten.

**Beispiel 8.4.** An einen Wintertag (W) wurde festgestellt, dass es stark regnet (R). Jedesmal, wenn der Damm voll ist (D) und es stark regnet, ereignet sich eine Katastrophe (K). Und im Winter ist der Damm immer voll. Wir wissen auch, dass der Dammwächter nur im Winter im Urlaub (U) ist. Folgt daraus, dass es zu einer Katastrophe kommen wird?
Die Tatsachen sind $W$ und $R$, und die Regeln sind

$$D \wedge R \Rightarrow K \quad , \quad W \Rightarrow D \quad , \quad U \Rightarrow W$$

Die Frage $K$, ob eine Katastrophe eintritt, hat genau dann die Antwort „ja“, wenn die offenbar zu einer Horn-Formel äquivalente Formel

$$W \wedge R \wedge (D \wedge R \Rightarrow K) \wedge (W \Rightarrow D) \wedge (U \Rightarrow W) \wedge \neg K$$

unerfüllbar ist. Aus $W$ folgt $D$ und aus $D \wedge R$ folgt $K$. Dies bedeutet, dass die Formel in der Tat unerfüllbar ist, und die Antwort lautet: ja, $K$ gilt.

**Bemerkung 8.5.** In der Logik-Programmierung befaßt man sich mit der Bearbeitung von Horn-Klauseln, typischerweise mit der Programmiersprache PROLOG. In der wird Tatsache A mit der Syntax

$$\mathrm{A.} \quad , \quad \mathrm{B} :- \mathrm{A}_0, \mathrm{A}_2, \cdots, \mathrm{A}_{n-1}. \quad , \quad ?- \mathrm{A}_0, \mathrm{A}_2, \cdots, \mathrm{A}_{n-1}.$$

für die Tatsache $A$, die Regel $A_0 \wedge A_2 \wedge \ldots \wedge A_{n-1} \Rightarrow B$, bzw. die Frage $\neg A_0 \vee \neg A_2 \vee \cdots \vee \neg A_{n-1}$. Dabei soll $:-$ eine Implikation $\Leftarrow$ andeuten.

**Beispiel 8.6.** Das PROLOG-Programm

```
A:- B, C.
A:- D.
B:- A, D.
C.
D:- E.
? - D.
```

entspricht der folgenden Hornformel:

$$(A \vee \neg B \vee \neg C) \wedge (A \vee \neg D) \wedge (B \vee \neg A \vee \neg D) \wedge C \wedge (D \vee \neg E) \wedge \neg D$$

**Bemerkung 8.7.** Jede Hornformel ohne Fragen ist erfüllbar: Wir können einfach alle positiven Literale mit 1 belegen.

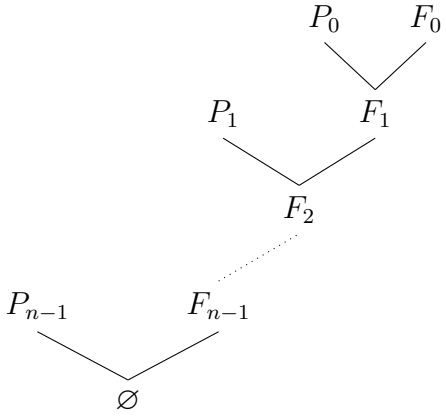
## 8.ii SLD-Resolutionsmethode

Tatsachen und Regeln werden begrifflich zu sogenannten *positiven Klauseln* zusammengefaßt (weil sie ein positives Literal enthalten), während die Fragen und $\varnothing$ *negative Klauseln* sind.

Die Resolventenbildung benötigt also zwingend eine positive Klausel. Die Resolvente zweier positiver Klauseln ist wieder positiv, da nur eines der beiden vorkommenden positiven Literale verschwindet, während die Resolvente einer positiven mit einer negativen Klausel negativ sein muß: das einzige positive Literal verschwindet. Damit sind Resolventen zweier positiver Klauseln bei der Suche nach $\varnothing$ irrelevant.

**Algorithmus 8.8.** Unter dem Begrriff **SLD-Resolution** versteht man eine Einschränkung der Resolutionsmethode, die für Hornformeln gut funktioniert. SLD ist die Abkürzung für „linear resolution with unrestricted selection for definite clauses“ (eigentlich also „LSD“; im Netz findet man die Abkürzung „SDL“ aber viel häufiger).

Man beginnt mit einer positiven Klausel $P_0$ und einer Frage $F_0$ und versucht eine Resolvente $F_1$ zu formen – die negativ sein muß. Existiert $F_1$, versucht man mit Hilfe einer positiven Formel $P_1$ der ursprünglichen Hornformel eine Frage $F_2$ als Resolvente zu bilden, usw.

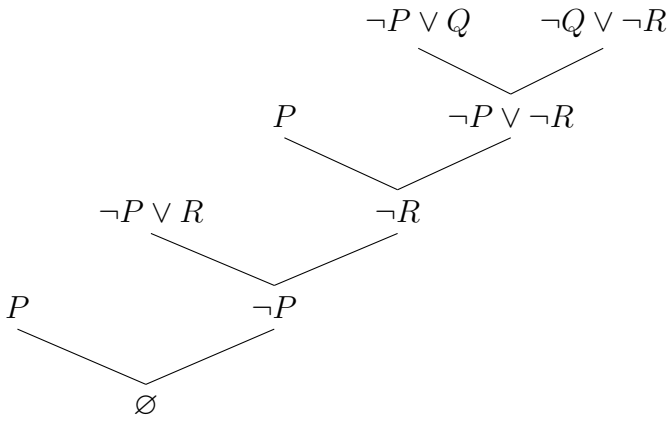



Läßt sich auf diese Weise die Resolvente $\varnothing$ erzeugen, ist die Hornformel unerfüllbar, andernfalls ist sie erfüllbar.

**Beispiel 8.9.** Die Formel

$$F := P \wedge (\neg P \vee Q) \wedge (\neg Q \wedge \neg R) \wedge (\neg P \vee R)$$

hat folgende SLD-Resolution:

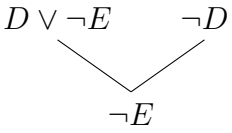



Deswegen ist sie unerfüllbar.

**Beispiel 8.10.** Für das vorige PROLOG-Beispielprogramm 8.6 mit der Hornformel

$$(A \vee \neg B \vee \neg C) \wedge (A \vee \neg D) \wedge (B \vee \neg A \vee \neg D) \wedge C \wedge (D \vee \neg E) \wedge \neg D$$

gibt es nur die folgende SLD-Resolution:




Deswegen ist die Formel erfüllbar.

**Satz 8.11.** *Für Hornformeln ist die SLD-Resolution ist korrekt und vollständig: Eine Hornformel ist genau dann erfüllbar, wenn keine SLD-Resolvente leer ist.*
Einen Beweis findet man z.B. in Kreuzer, Kühling „Logik für Informatiker“ (Pearson Studium 2006), Satz 3.17.

## 8.iii Markierungsalgorithmus

**Algorithmus 8.12.** Der **Markierungsalgorithmus** ist eine alternative Methode, die entscheidet, ob eine Hornformel $F$ erfüllbar ist. Sie arbeitet mit der Markierung bestimmter Variablen:

(M0) Markiere jedes Vorkommen jeder Tatsache $A$.

(M1) (Rekursiv) Für jede Regel $B \vee \neg A_0 \vee \neg A_2 \vee \cdots \vee \neg A_{n-1}$, in der die Variablen aller negativen Literale markiert sind, markiere jedes Vorkommen der Variable $B$. Falls keine neuen Variablen markiert werden können, gehe zu Schritt (M2).

(M2) Besteht eine Frage nur aus Literalen mit markierten Variablen, erfolgt die Ausgabe „$F$ unerfüllbar“, sonst wird „$F$ erfüllbar“ ausgegeben.

**Beispiel 8.13.** In Beispiel 8.9 kann man in

$$F := P \wedge (\neg P \vee Q) \wedge (\neg Q \wedge \neg R) \wedge (\neg P \vee R)$$

zunächst das Atom $P$ markieren (einzige Tatsache)

$$F := P \wedge (\neg P \vee Q) \wedge (\neg Q \wedge \neg R) \wedge (\neg P \vee R)$$

während Schritt (M1) die Literale $Q$ und $R$ markiert werden.

$$F := P \wedge (\neg P \vee Q) \wedge (\neg Q \wedge \neg R) \wedge (\neg P \vee R)$$

Das liefert eine vollständig markierte Frage, also ist $F$ nicht erfüllbar.

**Satz 8.14.** *Der Markierungsalgorithmus ist korrekt und braucht Zeit in $O(n^2)$ für Hornformeln mit $n$ Literalen (einschließlich Wiederholungen).*

*Beweis.*

(a) **Der Markierungsalgorithmus ist korrekt:** Wir zeigen erst, dass jede die Hornformel $F$ erfüllende Belegung $\alpha$ alle markierten Variablen auf 1 abbildet. Für Schritt (M0) ist das trivial, denn falls $A$ eine Tatsache und somit eine Klausel ist, muss $\alpha(A) = 1$ gelten. Beim rekursiven Schritt (M1) erinnern wir uns an die Implikationsform $A_0 \wedge A_1 \wedge \ldots A_{n-1} \Rightarrow B$ der Regeln: sind alle $A_i$ markiert und gilt damit nach Voraussetzung $\alpha(A_i) = 1$, $i < n$, so folgt $\widehat{\alpha}(\bigvee_{i<n} A_i) = 1$. Wegen $\widehat{\alpha}(F) = 1$ muß aber auch die Implikation von $\widehat{\alpha}$ auf 1 abgebildet werden, gemäß der Semantik von $\Rightarrow$ also auch $B$, was nach Voraussetzung ebenfalls markiert ist.

In Schritt (M2) kann in diesem Fall keine Frageklausel gefunden werden, in der alle Variablen markiert sind. Die Ausgabe „erfüllbar“ ist damit richtig. Umgekehrt verhindert eine Frage mit lauter markierten Variablen die Existenz einer Belegung $\alpha$ mit $\widehat{\alpha}(F) = 1$, insofern ist die Ausgabe “unerfüllbar” korrekt.

Lassen sich schließlich für keine Frage alle Variablen markieren, so setzen wir

$$\alpha(A) := \begin{cases} 1 & \text{falls } A \text{ markiert} \\ 0 & \text{sonst} \end{cases}$$

Damit haben alle positiven Klauseln den Wert 1. Für Tatsachen ist das klar, während bei Regeln entweder alle Variablen markiert sind, insbesondere die im positiven Literal, oder mindestens eine Variable in einem negativen Literal nicht markiert ist, was dem Literal den Wert 1 zuweist. Dieses letzte Argument greift auch bei Fragen. Damit werden alle Klauseln von $\widehat{\alpha}$ auf 1 abgebildet, also auch $F$.

(b) **Beweis der Zeitkomplexität:** Bei $n$ Literalen treten höchstens $n$ Variablen auf, also auch höchstens $n$ Tatsachen. Also braucht Schritt (M0) Zeit $O(n)$ und Schritt (M1) hat höchstens $n$ Durchläufe, die jeweils eine atomare Aussage abarbeiten. Ein Durchgang in Schritt (M1) läuft in $O(n)$ Schritten über alle Literale der Formel, weshalb Schritt (M1) als ganzes in der Zeitklasse $O(n^2)$ liegt. Schritt (M2) benötigt $O(n)$ Zeit. Insgesamt ergibt sich eine Zeitkomplexität von $O(n + n^2 + n) = O(n^2)$. □

**Zusammenfassung** In der Aussagenlogik stellen wir Fragen des Typs:

- Ist eine Formel erfüllbar?
- Ist eine Formel eine Tautologie?
- Folgt eine Formel aus gegebenen Formeln?

Diese drei Probleme sind äquivalent: siehe dazu die Sätze 2.20 und 6.4.

Wir haben verschiedenen Algorithmen vorgestellt, die diese Fragen beantworten: die Berechnung der Wahrheitstabelle, die Resolutionsmethode und die Natürliche Deduktion. Keiner dieser Algorithmen ist effizient: alle brauchen Zeit exponentiell in der Länge der Formel. Und es ist eher unwahrscheinlich, dass je ein Polynomialzeit-Algorithmus

dafür gefunden wird, da die Erfüllbarkeit ein NP-vollständiges Problem ist (vergl. Theoretische Informatik 2).

Die Situation für Hornformeln ist wesentlich besser: wir verfügen über den Markierungsalgorithmus mit Zeitkomplexität $O(n^2)$ sowie eine effiziente Variante der Resolutionsmethode.

# Teil II

# Prädikatenlogik

# 9. Syntax der Prädikatenlogik

## 9.i Einleitung

Die Aussagenlogik ist in allen Bereichen anwendbar, wo wahre und falsche Aussagen formuliert und kombiniert werden können, sei es die Herstellung von Würstchen oder die Theorie der Quantengraviation. Aber sie ist nicht in der Lage, auf die spezifischen Besonderheiten eines Bereichs einzugehen. Dazu bedarf es der Prädikatenlogik (auch als „Logik der ersten Stufe“ bekannt). Diese ist viel ausdrucksstärker als die Aussagelogik, denn sie ermöglicht es, Aussagen über die Objekte eines spezifischen „Universums“ zu formulieren, genauer, über

> Eigenschaften solcher Objekte,
>
> Relationen zwischen derartigen Objekten, und
>
> Funktionen, die auf Teilmengen solcher Objekte definiert sind.

Falls eine Menge $X$ von Objekten gegeben wird, kann eine Eigenschaft, oder „Prädikat“, formal als eine Untermenge $P \subseteq X$ (all derjeniger Objekte in $X$, die die Eigenschaft haben) verstanden werden. Analog wird eine Relation zwischen Paaren von Objekten als eine Untermenge $P \subseteq X \times X$ (aller Paare von Objekten die in Relation stehen) formalisiert; wir sprechen auch über „2-stellige Prädikate“. Falls zwei Objekte $x, y \in X$ in der Relation $P$ stehen, schreiben wir $P(x, y)$ oder alternativ $xPy$. Beispiel: die Relation „größer als“wird durch ”>”bezeichnet und statt $>(x, y)$ schreiben wir meist $x > y$. Dreistellige Prädikate sind Untermengen $P \subseteq X \times X \times X$ und formalisieren Relationen zwischen Tripeln von Objekten (z.B. kann $P(x, y, z)$ in einem geeigneten „Universums“ bedeuten: $x$ liegt zwischen $y$ und $z$).

**Definition 9.1.** Gegeben sei eine Menge $A$.

(a) Für jede natürliche Zahl $n \in \mathbb{N}$ bezeichent $A^n$ die Menge der geordneten $n$-Tupel mit Komponenten in $A$, genauer, die Menge aller Funktionen $n = \{0, 1, \ldots, n-1\} \longrightarrow A$. Dabei haben wir die Zahl $n$ mit der Menge ihrer Vorgänger identifiziert.

Zwei geordnete $n$-Tupel $\langle a_i : i < n\rangle$ und $\langle b_i : i < n\rangle$ stimmen genau dann überein, wenn sie komponentenweise gleich sind, d.h., $a_i = b_i$, $i < n$.[1]

[1] Dies ist die *Spezifikation* für geordnete $n$-Tupel, sie kann auf verschiedene Weise *realisiert* werden, aber das interessiert uns hier nicht.

Speziell im Fall $n = 2$ schreibt man häufig $A \times A$ anstelle von $A^2$ und spricht vom **cartesischen Produkt**. Weiter besitzt für $n = 0 = \varnothing$ die Menge $A^0$ genau ein Element, die Inklusion der leeren Menge in $A$.

(b) Ein **$n$-stelliges Prädikat** auf $A$ ist eine Untermenge $R$ von $A^n$, dem $n$-fachen kartesischen Produkt der Menge $A$ mit sich. Die Tatsache, dass ein $n$-Tupel $\vec{x} \in A^n$ zur Relation $R \subseteq A^n$ gehört, wird üblicherweise in der Logik durch $R(\vec{x})$ ausgedrückt anstelle von $\vec{x} \in R$; im binären Frall verwendet man gelegentlich auch die Infix-Schreibweise $x\,R\,y$ statt $R(x, y)$, etwa $x < y$.

Speziell im Fall $n = 0$ gibt es genau zwei Teilmengen der einelementigen Menge $A^0$: die leere Menge $\varnothing$ und die nichtleere Menge $A^0$. Diese lassen sich mit den **Wahrheitswerten** 0 und 1 identifizieren

Einstellige Prädikate entsprechen Teilmengen von $A$, oder **Eigenschaften**.

(c) Eine **$n$-stellige Funktion** ist eine Abbildung $A^n \xrightarrow{f} A$, also eine Teilmenge $f \subseteq A^{n+1}$, die, aufgefaßt als binäre Relation $f \subseteq A^n \times A$, zwei Eigenschaften hat:

- ▷ $f$ ist **total**, d.h., zu jedem $\vec{x} \in A^n$ existiert ein $y \in A$, das zu $\vec{x}$ in Relation steht;
- ▷ $f$ ist **einwertig**, d.h., stehen $y$ und $z$ aus $A$ zu $\vec{x} \in A^n$ in Relation, so gilt $y = z$.

Folglich steht zu jedem $\vec{x} \in A^n$ genau ein $y \in A$ in Relation, das konventionell mit $f(\vec{x})$ bezeichent wird.

Weil $A^0$ ein-elementig ist, können wir 0-stellige Funktionen auf $A$ mit Elementen von $A$ identifizieren. Man spricht dann auch von **Konstanten**.

Wir wollen anhand eines praxisorientierten Beispiels den sinnvollen Einsatz von Prädikaten und Funktionen demonstrieren.

**Beispiel 9.2** (politisch korrekte Version)**.** Sagt eine Frau, die ein Porträt beobachtet: „Geschwister habe ich keine. Und die Mutter dieser Frau ist meiner Mutter Tochter.“ Wer ist da abgebildet?
Wir führen zwei Konstanten für die beiden Frauen im Beispiel ein:

$a =$ die abgebildete Frau

$b =$ die Beobachterin.

Die erste Aussage arbeitet mit dem zweistelligen Prädikat „Geschwister“

$$G(x, y) \quad \text{steht für} \quad \text{„}x \text{ und } y \text{ sind Geschwister“}.$$

Die Beobachterin $b$ behauptet hier: $G(b, y)$ ist immer, d.h. für jedes $y$, falsch. Das schreiben wir in der Form

$$\forall y : \neg G(b, y) \tag{9.1}$$

wobei der sogenannte **Quantor** $\forall$ die Bedeutung „für alle“ hat.

Für die zweite Aussage brauchen wir das zweistellige Prädikat „Tochter von“

$$T(x, y) \quad \text{steht für} \quad „x \text{ ist Tochter von } y“.$$

Wir könnten auch „Mutter von“ als Prädikat nehmen, aber wir haben etwas besseres: eine Funktion $m(x)$, die jeder Person $x$ ihre (eindeutig bestimmte) Mutter $y = m(x)$ zuweist. Die zweite Aussage der Beobachterin formulieren wir nun als

$$T(m(a), m(b)) \quad \text{oder effizienter als} \quad m(m(a)) = m(b) \tag{9.2}$$

Die alternative Aussage benutzt noch ein wichtiges zweistelliges Prädikat: die Gleichheit (=). Sie nimmt eine Sonderstellung ein und wird als immer verfügbares Prädikat mit fester Interpretation verwendet.

Im Beispiel nicht explizit erwähnt, aber implizit zur Aufgabe gehörig, ist die Tatsache, dass jemand (Variable $z$) ohne Geschwister das einzige Kind ihrer Mutter ist:

$$\big(\forall y : \neg G(z, y)\big) \Rightarrow \forall u : (m(u) = m(z) \Rightarrow u = z) \tag{9.3}$$

Was folgt aus den Prämissen (9.1), (9.2) und (9.3)? Wir können für die Variable $z$ in (9.3) die Konstante $b$ substituieren und bekommen

$$\forall y : \neg G(b, y) \Rightarrow \forall u : (m(u) = m(b) \Rightarrow u = b)$$

Dies, zusammen mit (9.1), ergibt (wegen modus ponens)

$$\forall u : (m(u) = m(b) \Rightarrow (u = b))$$

Ließt man dies als verallgemeinerte Konjunktion, so müssen im Falle ihrer Wahrheit alle individuellen Instanzen wahr sein. Substituieren wir speziell $m(a)$ für $u$, so gilt

$$m(m(a)) = m(b) \Rightarrow m(a) = b.$$

Modus ponens liefert aufgrund von (9.2) $m(a) = b$. D.h., $a$ ist die Tochter der Beobachterin $b$.

Wir haben hier mit konkreten zweistelligen Relationen $G$ (Geschwister) und $=$ (Gleichheit) gearbeitet (die Tochter-Relation $T$ wurde nur übergangsweise verwendet). Dazu kamen 0-stellige Funktionen oder Konstanten $a$ und $b$ und die einstellige Mutter-Funktion $m(x)$. Wichtig ist, dass jeder Name ein eindeutig bestimmtes Objekt (Relation bzw. Funktion) auf der konkret spezifizierten Grundmenge (hier: alle Personen) beschreibt.

Man könnte aber auch auf einer ganz anderen Menge, z.B. $\mathbb{N}$, eine zweistellige Relation $G$ (und ggf. auch $T$) definieren, dazu eine einstellige Funktion $m$ und Konstanten $a$ und $b$, dann Aussagen formen und deren Wahrheitsgehalt überprüfen. Dabei ist allerdings nicht gewährleistet, dass die Formeln (9.1), (9.2) und (9.3) in dieser neuen Situation gelten. Das hängt von den konkret gewählten Interpretation der Daten in $\mathbb{N}$ ab. Es mag Formeln geben, die in beiden, oder in nur einer, oder in keiner der beiden Interpretionen gelten. Dagegen gilt $x = x$ in jeder möglichen Interpretation, und $\neg(x = x)$ in keiner.

Um abstrakt über unspezifische Mengen sprechen zu können, auf denen bestimmte Funktion und Prädikate definiert werden sollen, verwenden wir den syntaktischen Begriff der **Signatur**. Diese enthält Platzhalter, oder Namen, **symbolischer Natur**, die erst nach Spezifikation einer Menge $\mathcal{A}$, an der man konkret interessiert ist, als echte Funktionen bzw. Relationen auf $\mathcal{A}$ interpretiert werden können (siehe Abschnitt 10). Genauer:

**Definition 9.3.** Eine **Signatur** $\Sigma$ besteht aus

- einer Liste von Funktionssymbolen (meist mit kleinen Buchstaben bezeichnet) mit gegebenen Stelligkeiten $0, 1, 2, 3, \ldots$; die 0-stelligen Funktionssymbole heißen auch **Konstanten**.
- einer Liste von Prädikatensymbolen (meist mit großen Buchstaben bezeichnet) mit gegebenen Stelligkeiten $0, 1, 2, 3, \ldots$; die 0-stelligen Prädikatssymbole wollen wir **$\Sigma$-Propositionen** nennen und in $\Sigma$-***Prop*** zusammenfassen, sie werden eine besondere Rolle spielen.

Jede dieser Listen darf auch leer sein; die immer verfügbare Gleichheit braucht hier nicht explizit erwähnt werden.

**Beispiel 9.4.** Im Beispiel 9.2 haben wir die Signatur $\Sigma = \{m, a, b; G, T\}$ benutzt. Die Prädikatssymbole $G$ und $T$ sind zweistellig, $m$ ist ein einstelliges Funktionssymbol und $a, b$ sind als Konstanten 0-stellig. Allerdings war die Signatur in Beispiel 9.2 bereits *interpretiert* worden: die auftretenden Variablen waren Platzhalter für Menschen, $G$ und $T$ waren konkrete Relationen zwischen Menschen, und $m$ war eine konkrete einstellige Funktion. Eine derartige $\Sigma$ *Struktur* ist der erste Schritt Richtung Semantik, vergl. Definition 10.1.

**Bemerkung 9.5.** Eine notationelle Differenzierung zwischen den Symbolen in einer Signatur und ihren Interpretationen wird in Definition 10.1 zwar vorgeschlagen, ist in der Praxis aber nicht immer durchzuhalten. Es liegt also in der Verantwortung der Leserin, kontextsensitiv $G$ mal als Platzhalter für eine 2-stellige Relation, oder als konkrete 2-stellige Relation zu lesen.

Im Hinblick auf die intendierte Semantik betrachten wir weitere Beispiele.

**Beispiel 9.6.** Die Signatur $\Sigma = \{F\}$ bestehe aus einem einzigen einstelligen Prädikat. Kann es ein $x$ geben, so dass, wenn $x$ nicht die Eigenschaft $F$ hat, auch kein anderes Element die Eigenschaft $F$ haben kann, d.h., ob die Implikation $\neg F(x) \Rightarrow \forall y : \neg F(y)$ gilt? Mit Hilfe eines weiteren Quantors für die Existenz von Objekten ist die Wahrheit von

$$\exists x : (\neg F(x) \Rightarrow \forall y : \neg F(y))$$

zu untersuchen. Die Antwort ist (überraschenderweise?) positiv, denn bei Interpretation dieser Formel als verallgemeinerte Disjunktion genügt es, eine wahre Instanz zu finden. Unter der Voraussetzung, dass die Trägermenge unseres „Universums“ nicht leer ist, finden wir entweder ein Objekt $x_0$ mit Eigenschaft $F$, das wir für $x$ substituieren, um die wahre Implikation $\neg F(x_0) \Rightarrow \forall y : \neg F(y)$ zu erhalten (da die Voraussetzung falsch ist). Oder es existiert kein $x$ mit Eigenschaft $F$, so dass die Aussage $\forall y : \neg F(y)$

wahr ist, und damit auch alle Instanzen der Implikation $\neg F(x) \Rightarrow \forall y : \neg F(y)$. Achtung: bei leerer Grundmenge gibt es keine derartigen Instanzen, und das Argument funktioniert nicht.

**Beispiel 9.7.** In der Arithmetik benutzen wir Funktionen

$$+ \quad \text{und} \quad \cdot \quad \text{(beide zweistellig)}$$

das Prädikatensymbol

$$< \quad \text{(zweistellig)}$$

und die Konstanten 0 und 1. Die zugehörige Signatur hat die Form

$$\Sigma = \{<, +, \cdot, 0, 1\}$$

mit den entsprechenden Stelligkeiten. Ist folgende Aussage wahr?

$$\forall x : (x \cdot x > 0) \vee (x \cdot x = 0)$$

Das hängt von dem konkreten Universum ab, in dem wir die Signatur interprtieren. Falls $x$ aus dem Bereich der natürlichen Zahl stammt, ist die Aussage wahr. Aber wenn $x$ auch eine komplexe Zahl sein darf, lautet die Antwort „nein“. Der Wahrheitsgehalt einer Formel hängt also vom Kontext ab, dies werden wir im nächsten Abschnitt präzise formalisieren.

**Beispiel 9.8.** In der Theorie der gerichteten Graphen arbeitet man mit einem zweistelligen Prädikatensymbol $R$; wir interpretieren $R(x, y)$ als Existenz einer „gerichteten Kante“ von $x$ nach $y$, häufig dargestellt als $x \longrightarrow y$. Loops, also Kanten von einem Knoten zu sich selbst, sind zulässig, aber es kann höchstens eine Kante zwischen den Knoten $x$ und $y$ geben. Die Liste der Funktionssymbole ist leer. Folgende Formel charakterisiert alle gerichteten Graphen ohne 2-Zyklen

$$\forall x : \forall y : (R(x, y) \wedge R(y, x) \Rightarrow (x = y))$$

Falls man dagegen verlangt, die Relation $R$ möge symmetrisch sein:

$$\forall x : \forall y : (R(x, y) \Rightarrow R(y, x))$$

gehört zu jeder Kante eine Kante in der Gegenrichtung, $x \rightleftharpoons y$, was man als $x \longrightarrow\!\!\!\!\!- \; y$ abkürzen kann. Diese Bedingung charakterisiert also **ungerichtete** Graphen, deren Kanten auch als nichtleere Teilmengen der Knotenmenge mit höchstens zwei Elementen beschrieben werden können. Offenbar ist der Begriff des gerichteten Graphen der fundamentalere, der des ungerichteten Graphen ist ein Spezialfall. Es ist unklar, ob ungerichtete Graphen allein mit einer Signatur ohne weitere Formeln (sogenannte **Axiome**) beschreiben werden können; ein Prädikatssymbol genügt jedenfalls nicht.

## 9.ii Das Alphabet der Prädikatenlogik

In der Prädikatenlogik formen wir Ausdrücke wie $\forall x : Fx$ die sagen, dass für jeden Wert der Variable $x$ die Aussage $F$ gilt. Dies werden wir in späteren Abschnitten präzisieren, in denen die Semantik der Prädikatenlogik erläutert wird. Zuvor benötigen wir aber eine Syntax, und die erfordert eine präzise Festlegung der zugrunde liegenden Sprache. Dabei geht es um mehr als nur die Einführung des **All-Quantors** $\forall$ und seines Gegenstücks, des **Existenz-Quantors** $\exists$, als neue formale Symbole. Die Signatur ist ebenso zu berücksichtigen, und es werden weitere Hilfssymbole gebraucht.

**Definition 9.9.** Das **Alphabet** der Prädikatenlogik besteht aus

- ▷ $\mathcal{V}$, einer abzählbar unendlichen Menge von **Variablen** $x_0, x_2, x_3, \ldots$,
- ▷ einer Signatur $\Sigma$,
- ▷ den Quantoren $\forall$ und $\exists$
- ▷ und den Hilfssymbolen Klammern „)“ , „(“ , Komma „,“ und Doppelpunkt ”:”

**Bemerkung 9.10.** Für die Variablen schreiben wir oft $x, y, z$, in bestimmten Kontexten nutzen wir auch andere Symbole wie $\epsilon$ oder $\delta$. Dabei ist aber zu beachten, dass keine Namen aus der Signatur $\Sigma$ verwendet werden.

Für zweistellige Funktionssymbole, z.B. $+$, verwendet man neben $+(x, y)$ die In-Fix-Notation $x + y$, vergl. Bemerkung 2.5. Analog schreibt man auch für zweistellige Relationssymbole $R$ häufig $xRy$ anstelle von $R(x, y)$.

## 9.iii Die Syntax der Prädikatenlogik

Die Funktions- bzw. Relationssymbole einer Signatur $\Sigma$ sind für zwei völlig unterschiedliche rekursiv definierte syntaktische Kategorien verantwortlich, die *Terme* und die *Formeln*. Erstere haben einen algebraischen Charakter und dienen als abstrakte *Baupläne* für Elemente einer bisher unspezifischen Grundmenge. Mit den Formeln hingegen läßt sich Logik betreiben.

**Definition 9.11.** Die Menge aller $\Sigma$-**Terme** ist die kleinste Menge, die

- ▷ $\mathcal{V}$ umfaßt, d.h., jede Variable $x_0, x_1, x_2 \ldots$ ist ein Term;
- ▷ mit jedem $n$-stelliges Funktionssymbol $f$ und $n$ Termen $t_i$, $i < n$, auch $f(t_0, \ldots, t_{n-1})$ enthält. (Im Fall $n = 0$ erhält man so die Konstanten.)

Die Menge aller $\Sigma$-Terme deren Variablen in $\mathcal{M} \subseteq \mathcal{V}$ liegen bezeichnet man mit $\mathcal{T}_\Sigma(\mathcal{M})$.

**Beispiel 9.12.**

- Falls $\Sigma = \{s\}$ aus einer 1-stelligen Funktion $s$ besteht, sind die Terme genau die Variablen $x_i$ und die Ausdrücke der Form $s(x_i), s(s(x_i)), \ldots$

- Falls $\Sigma = \{+, 0\}$ aus einer 2-stelligen Funktion $+$ und einer Konstante 0 besteht, haben wir Terme

$$x,\ 0,\ x + y,\ x + 0,\ (x + 0) + (0 + 0),\ \ldots$$

**Bemerkung 9.13.** $\mathcal{T}_\Sigma(\varnothing)$ ist die Menge aller Terme ohne Variablen. Z.B. gehören in der Signatur in Beispiel 9.7 die Terme $1 + 0$, $1 + 1$, $1 \cdot (0 + 1)$ zu $\mathcal{T}_\Sigma(\varnothing)$, während in Beispiel 9.16 unten $m(0, 2)$, $f(a(0)), f(f(f(2)))$ Variablen-freie Terme sind. Für Signaturen ohne Konstanten gilt natürlich $\mathcal{T}_\Sigma(\varnothing) = \varnothing$.

Terme dienen zusammen mit den Relationssymbolen der Signatur als Bausteine für Formeln:

**Definition 9.14.**

▷ **Atomare Formeln** haben die Form

$$t_0 = t_2 \qquad \text{sowie} \qquad R(t_0, \ldots, t_{n-1})$$

wobei die Ausdrücke $t_i$ Terme sind und $R$ ein $n$-stelliges Prädikatssymbol der Signatur $\Sigma$ ist.

▷ Die **Formeln** der Prädikatenlogik bilden die kleinste Menge,

(a) die jede atomare Formel enthält,

(b) bezüglich der Junktoren $\neg$, $\wedge$ und $\vee$ abgeschlossen ist,

(c) mit jeder Formel $F$ und jeder Variable $x$ auch $(\forall x : F)$ und $(\exists x : F)$ enthält.

In Anlehnung an Definition 9.11 bezeichnen wir für eine Menge $\mathcal{M}$ von Variablen mit $\mathcal{A}_\Sigma(\mathcal{M})$ bzw. $\mathcal{F}_\Sigma(\mathcal{M})$ die Mengen der (atomaren) Formeln über $\mathcal{M}$

**Bemerkung 9.15.**

(a) Terme an sich haben keine logische, sondern eher algebraische Bedeutung. Die einzigen atomaren Formeln ohne Terme sind die 0-stelligen Relationssymbole (sofern in der Signatur vorhanden); diese entsprechen den atomaren Aussagen der Aussagenlogik. Selbst wenn die Signatur keine Funktionssymbole hat, sind alle Variablen Terme, die in passender Zahl in die Relationssymbole bzw. die Gleichheit eingesetzt werden können.

(b) Wie in der Aussagenlogik benutzen wir $F \Rightarrow G$ für $G \vee \neg F$ und $F \Leftrightarrow G$ für $(F \wedge G) \vee (\neg F \wedge \neg G)$.

(c) Wie formalisiert man Aussagen wie „Für alle $\epsilon$ größer 0 gilt XXX“? Der Ausdruck „$\forall \epsilon > 0 :$ XXX“ ist (leider?) keine syntaktisch korrekte Formel. Stattdessen kann man aber schreiben

$$\forall \epsilon : (\epsilon > 0 \Rightarrow \text{XXX})$$

mit der atomaren Aussage $\epsilon > 0$, aus der XXX folgen möge.

(d) Analog formalisiert man statt „Es gibt ein $\delta$ größer 0 mit YYY“ als

$$\exists\delta : (\delta > 0 \wedge \mathrm{YYY})$$

**Beispiel 9.16.** In der Analysis findet man die Aussage

„die Funktion $f(x)$ ist stetig in Punkt 2“

häufig wie folgt mit Hilfe von Quantoren leicht schlampig formuliert:

$$\forall\epsilon > 0 : \exists\delta > 0 : \forall x : (|x-2| < \delta \Rightarrow (|f(x) - f(2)| < \epsilon))$$

Hier sind $f(x)$ und der Betrag $a(x) = |x|$ jeweils 1-stellige Operationssymbole und die Funktion $m(x, y) = x - y$ ist zweistellig. Wir benutzen ferner zwei Konstanten 0 und 2 und ein zweistelliges Prädikat $>$: In der Signatur $\Sigma = \{>, m, a, f, 0, 2\}$ bekommt die obige Formel die korrekte aber leider etwas unübersichtlichere Form

$$\forall\varepsilon : \Big(\varepsilon > 0 \Rightarrow \exists\delta : \delta > 0 \wedge \forall x : \big(\delta > a(m(x,2)) \Rightarrow \epsilon > a(m(f(x), f(2)))\big)\Big)$$

die wir mit Hilfe der Klammergröße versucht haben, besser zu strukturieren. Auch hier besteht dringender Bedarf, den Wildwuchs an Klammern einzugrenzen.

**Beispiel 9.17.** Im Land Warling gibt es drei Stämme: Xur, Yzy und Poli. Jemand vom Stamm Xur vermählt sich nur mit jemand vom Stamm Xur oder vom Stamm Yzy. Polis vermählen sich nie, weil sie auf die entscheidende Frage immer mit „Nein“ antworten.

Um dies formal aufzuschreiben benötigen wir keine Operationen, sondern nur die folgenden Relationen:

1-stellig: $X$, $Y$ und $P$ für die drei Stämme

2-stellig: $V$ für „vermählt sein mit“

$J$ beide Partner beantworten die entscheidende Frage mit “Ja“.

Die erste Aussage ist

$$F_0 := (\forall x : (X(x) \Rightarrow (\forall y : (V(x,y) \Rightarrow (X(y) \vee Y(y))))))$$

Die zweite ist aus zwei Teilen zusammengesetzt:

$$F_2 := (\forall x : (P(x) \Rightarrow (\forall y : (\neg J(x,y)))))$$

und

$$F_3 := (\forall x : \forall y : (\neg J(x,y) \Rightarrow \neg V(x,y)))$$

Die komplette Aussage über Warling ist $F_0 \wedge F_2 \wedge F_3$.

**Notationelle Konvention 9.18.** Wir erweitern die **Bindungskonventionen** der Aussagenlogik: formal kann man $\forall x :$ und $\exists x :$ ebenso wie $\neg$ als einstellige Junktoren auffassen (wenn auch abzählbar unendlich viele), und diese binden stärker als alle 2-stelligen Junktoren. Es bleibt dabei, dass $\wedge$ und $\vee$ stärker binden als $\Rightarrow$ und $\Leftrightarrow$.

Dann dürfen wir schreiben:

$$\forall x : F \vee \exists y : G \Rightarrow \neg H$$

was dann folgendes bedeutet:

$$((\forall x : F) \vee (\exists y : G)) \Rightarrow \neg H$$

Diese Formel darf nicht mit den folgenden verwechselt werden

$$\forall x : (F \vee (\exists y : G \Rightarrow \neg H)) \qquad \text{oder} \qquad (\forall x : F) \vee (\exists y : (G \Rightarrow \neg H))$$

die etwas ganz anderes besagen. Ein praktisches Beispiel für das unterschiedliche Zusammenspiel verschiedener Quantoren bei einer 2-stelligen Relation findet sich unter Wikipedia unter dem Begriff "Loving relation".

**Beispiel 9.19.** „Es gab zwei Sorten von Studenten, die die volle Punktezahl erreichten: einige wussten alle Lösungen der Aufgaben und die anderen hatten das Glück, dass ihr bester Freund zur ersten Sorte gehört."

Dies wollen wir durch eine Formel ausdrücken. Sei $W(x)$ das einstellige Prädikat „$x$ weiß alle Lösungen" und $B(x, y)$ das zweistellige Prädikat „$y$ ist der beste Freund von $x$" – es kann natürlich mehrere $y$ je $x$ geben oder auch keinen. Die Formel für die volle Punktezahl ist dann:

$$P := W(x) \vee \exists y : (B(x, y) \wedge W(y))$$

**Bemerkung 9.20.** Im letzten Beispiel ist die Variable $x$ in der Formel $P$ **frei**, d.h., steht nicht im Einflußbereich (Scope) eines Quantors der sich auf $x$ bezieht: sie bezeichnet damit einfach abstrakt „einen Konsumenten akademischer Bildung". Dagegen ist $y$ durch den Ausdruck $\exists y :$ **gebunden**.

Diese Unterscheidung zwischen freien und gebundenen Variablen ist wichtig.

**Definition 9.21.** Die Menge Frei($F$) aller **freien Variablen** einer Formel $F$ ist rekursiv definiert

| Formel $F$ | Menge $Frei(F)$ | |
|---|---|---|
| $t_0 = t_2$ | alle Variablen, die in $t_0$ oder $t_2$ vorkommen | |
| $R(t_0, \ldots, t_{n-1})$ | alle Variablen, die in $t_0, \ldots, t_{n-1}$ vorkommen | |
| $F_0 \vee F_2$ | $Frei(F_0) \cup Frei(F_2)$ | |
| $F_0 \wedge F_2$ | $Frei(F_0) \cup Frei(F_2)$ | |
| $\neg G$ | $Frei(G)$ | d.h. es sind die selben Variablen frei wie in $G$ |
| $\forall x : G$ | $Frei(G) - \{x\}$ | alle freien Variablen von $G$ außer $x$ |
| $\exists x : G$ | $Frei(G) - \{x\}$ | alle freien Variablen von $G$ außer $x$ |

Eine Formel heißt **Satz relativ zu ihren $\Sigma$-Propositionen**, falls sie keine freien Variablen enthält. Treten auch keine $\Sigma$-Propositionen auf, sprechen wir von einem **Satz**.

**Beispiel 9.22.** In der obigen Formel $P$ aus Beispiel 9.19 ist $x$ eine freie und $y$ eine gebundene Variable. Es gilt: $\text{Frei}(F) = \{x\}$.

Zum Beispiel ist die Stetigkeitsformel in Beispiel 9.16 ein Satz: die Variablen $\epsilon$, $\delta$ und $x$ sind alle gebunden. Für Sätze ist die Frage sinnvoll, ob die Formel wahr oder falsch ist: im konkreten Beispiel hängt dies davon ab, ob die gegebene Funktion $f(x)$ an der Stelle 2 stetig ist oder nicht. Im Gegensatz dazu hängt der Wahrheitswert der obigen Formel $P$ von der Interpretation der freien Variablen $x$ ab. Dies wird im nächsten Kapitel erläutert.

**Achtung**: dieselbe Variable $x$ kann in einer Formel $F$ sowohl frei als auch gebunden auftreten: Betrachte

$$F = R(x, y) \wedge \exists x : x > 2$$

Das erste **Vorkommen** von $x$ ist frei, aber das zweite ist gebunden. Solch ein Problem läßt sich immer umgehen, wenn man die Namen der gebundenen Variablen etwas geschickter wählt. Die obige Problem-Formel wird sich als äquivalent herausstellen zu

$$G = R(x, y) \wedge \exists z : z > 2$$

Mit anderen Worten: gebundene Variable können *innerhalb ihres Gültigkeitsbereichs* beliebig umbenannt werden, vergl. Fakt 11.10 im übernächsten Abschnitt.

# 10. Semantik der Prädikatenlogik

Ist die Formel $\forall x : (x + 1 > 0)$ wahr? Die Antwort hängt vom „Universum“ ab, in dem $x$ **interpretiert** wird: die Formel ist z.B. wahr, falls $x$ auf die natürlichen Zahlen beschränkt wird und $+$, ! sowie $>$ ihre “normale” Interpretation haben, aber nicht, falls $x$ eine ganze Zahl darstellen kann. Wir müssen also zunächst spezifizieren, woher die Elemente stammen, die $x$ darstellt, man spricht auch von der *Trägermenge*, und anschließend, was die Symbole $+$, $>$ und 0 der Signatur in dieser Trägermenge konkret bedeuten sollen (hier waren schon suggestive Namen im Hinblick auf eine Interpretation in den Zahlen gewählt worden). Spezifikationen dieser Art heißen *Strukturen*:

**Definition 10.1.** Für eine Signatur $\Sigma$ besteht eine **$\Sigma$-Struktur $\mathcal{A}$** aus

- ▷ einer Menge $\mathcal{A}$ (dem **Träger** von $\mathcal{A}$), die nicht leer ist (warum? vergl. Beispiel 11.20(b)),
- ▷ einer konkreten $n$-stelligen Operation

  $$f^{\mathcal{A}} : \mathcal{A}^n \longrightarrow \mathcal{A}$$

  für jedes $n$-stellige Funktionssymbol $f$ in $\Sigma$, und
- ▷ einer konkreten $n$-stelligen Relation

  $$R^{\mathcal{A}} \subseteq \mathcal{A}^n$$

  für jedes $n$-stellige Prädikatssymbol $R$ in $\Sigma$.

Die konkreten Operationen $f^{\mathcal{A}}$ bzw. Relationen $R^{\mathcal{A}}$ sind die Instanziierungen der abstrakten Funktions- und Prädikatssymbole aus der Signatur $\Sigma$ für die Struktur $\mathcal{A}$. Gelegentlich, z.B. bei suggestiven Namen der Funktions- und Prädikatssymbole in $\Sigma$, lassen wir den oberen Index $\mathcal{A}$ auch weg.

Insbesondere wird jeder Konstante $c$ in $\Sigma$ eine Funktion $c^{\mathcal{A}} : \mathcal{A}^0 = 1 \longrightarrow \mathcal{A}$, d.h., ein Element $c^{\mathcal{A}} \in \mathcal{A}$ zugeordnet und jeder $\Sigma$-Proposition $R$ ein Wahreitswert $R^{\mathcal{A}} \in \{0, 1\}$. Da dieser Wert **frei wählbar**, läßt sich nicht nur ein begrenzter Teil der Aussagenlogik innerhalb der Prädikatenlogik für $\Sigma$ wiederzufinden, sondern auch “relative Semantik” bzgl. der $\Sigma$-Propositionen betreiben. Beides wird in dieser Vorlesung aber keine wichtige Rolle spielen.

**Beispiel 10.2.** Falls $\Sigma = \{R\}$ aus einem 2-stelligen Prädikat $R$ besteht, ist eine $\Sigma$-Struktur $\mathcal{A}$ genau ein (nichtleerer!) gerichteter Graph. Dabei ist $\mathcal{A}$ die Menge der Knoten von $\mathcal{A}$ ist, und $R^{\mathcal{A}} \subseteq A \times A$ die Menge der Kanten von $\mathcal{A}$.

**Beispiel 10.3.** Besteht $\Sigma$ aus einer Konstante 0, einem einstelligen Funktionssymbol $s$ und einem zweistelligen Prädikatssymbol $R$ können wir z.B. die $\Sigma$-Struktur $\mathcal{N}$ der natürlichen Zahlen betrachten:

Träger $\mathbb{N} = \{0, 1, 2, 3, \dots\}$

$0^{\mathcal{N}}$ ist die Konstante 0;

$s^{\mathcal{N}}$ ist die Nachfolgerfunktion, $s^{\mathcal{N}}(n) = n + 1$;

$R^{\mathcal{N}}(x, y)$ bedeutet $x < y$.

Eine andere $\Sigma$-Struktur $\mathcal{Z}$ verwendet als Träger die Menge $\mathbb{Z}$ der ganzen Zahlen, wobei $0^{\mathcal{Z}}$, $s^{\mathcal{Z}}$ und $R^{\mathcal{Z}}$ analog wie oben definiert sind. Eine dritte $\Sigma$-Struktur $\mathcal{A}$ habe den Träger $\{0, 1\}$, wobei $s^{\mathcal{A}}$ die beiden Werte vertauscht, $0^{\mathcal{A}} = 0$ gilt und $R^{\mathcal{A}} = \varnothing$ das leere Prädikat ist.

Auch wenn man häufig für eine Signatur $\Sigma$ eine bestimmte $\Sigma$-Struktur im Auge hat, mit der man sich beschäftigen möchte (die *intendierte* $\Sigma$-Stuktur), so gibt es doch beliebig viele andere $\Sigma$ Strukturen, die man zunächst nicht ausschließen kann. In Beispiel 9.8 hatten wir allerdings gesehen, wie man durch die Forderung nach Gültigkeit *zusätzlicher Formeln* (sogenannter **Axiome**) die passenden $\Sigma$-Strukturen einschränken kann. Von besonderem Interesse ist es daher, eine vorzugsweise endliche Axiomenmenge zu finden, so dass nur noch die intendierte $\Sigma$-Struktur übrigbleibt (bis auf Umbenennung der Elemente), die diese Axiome erfüllt. Das Problem der (endlichen) Axiomatisierbarkeit werden wir hier allerdings nicht weiter behandeln.

Wir können jetzt das Problem angehen, auch Formeln mit freien Variablen einen Wahrheitswert zuzuweisen. Ähnlich wie in der Aussagenlogik, wo atomaren Formeln mittels *Belegungen* Wahrheitswerte zugewiesen wurden, woraufhin man die Belegungen auf zusammengesetzte Formeln erweitern konnte, wollen wir nun freien Variablen auf Elemente im Träger der aktuellen $\Sigma$-Struktur abbilden, und solche Abbildungen dann auf die Menge aller Terme ausdehnen. Damit erhalten die Interpretationen alle Term-basierten atomaren Formeln Wahrheitswerte, allerdings nicht die $\Sigma$-Propositionen.

**Definition 10.4.** (Belegungen von Variablen im Träger $\mathcal{A}$ einer $\Sigma$-Struktur)

(1) Gegeben eine Struktur mit Träger $A$, eine **Belegung** $\alpha$ ist eine Funktion, die jeder Variablen $x$ aus einer Menge $\mathcal{M} \subseteq \mathcal{V}$ ein Element aus dem Träger $\mathcal{A}$ zuordnet:

$$\alpha : \mathcal{M} \longrightarrow \mathcal{A}$$

(2) Für jede auf $\mathcal{M}$ definierte Belegung $\alpha$, jede Variable $x$ und jeden Wert $a \in \mathcal{A}$ im Träger bezeichnet der **Update** von $x$ in $\alpha$ auf $a$ die (potentiell modifizierte) Belegung

$$\alpha[{}^{a}/_{x}] : \mathcal{M} \cup \{x\} \longrightarrow A$$

die alle Variablen $y \in \mathcal{M} - \{x\}$ genau wie $\alpha$ belegt, aber der Variable $x$ den Wert $a$ zuweist, formal:

$$\alpha[{}^{a}/_{x}](y) := \begin{cases} \alpha(y) & \text{falls } y \neq x \\ a & \text{sonst} \end{cases}$$

(Wir lesen den Ausdruck $[^a/_x]$ als "$a$ für $x$".)

**Definition 10.5.** Jede Belegung $\alpha : \mathcal{M} \longrightarrow \mathcal{A}$ wird rekursiv auf alle Terme in Variablen aus $\mathcal{M}$ erweitert. Wir definieren die Funktion $\overline{\alpha} : \mathcal{T}_\Sigma(\mathcal{M}) \longrightarrow \mathcal{A}$ wie erwartet:

$$\overline{\alpha}(x) = \alpha(x)$$

für alle Variablen $x \in \mathcal{M}$ und

$$\overline{\alpha}(f(t_0, \ldots, t_{n-1})) = f^{\mathcal{A}}(\overline{\alpha}(t_0), \ldots, \overline{\alpha}(t_{n-1}))$$

für alle Funktionssymbolde der Stelligkeit $n$ und alle Terme $t_0, \ldots, t_{n-1}$ in $\mathcal{T}_\Sigma(\mathcal{M})$, für die $\overline{\alpha}(t_i)$ schon definiert wurde.

**Beispiel 10.6.** In der Signatur $\Sigma = \{s, 0, R\}$ des obigen Beispiels 10.3 besteht $\mathcal{T}_\Sigma(\{x\})$ aus allen Termen $x, s(x), s(s(x)), \ldots$, sowie $0, s(0), s(s(0)), \ldots$
Gegeben sei eine Belegung

$$\alpha : \{x\} \longrightarrow \mathbb{Z} \quad \text{mit} \quad \alpha(x) = -7$$

Dann haben wir

$$\begin{array}{ll} \overline{\alpha}(x) = -7 & \widehat{\alpha}(0) = 0 \\ \overline{\alpha}(s(x)) = -6 & \overline{\alpha}(s(0)) = 1 \\ \overline{\alpha}(s(s(x))) = -5 & \overline{\alpha}(s(s(0))) = 2 \\ \vdots & \vdots \end{array}$$

Wir wollen jetzt die Semantik einer Formel $F$ in einer $\Sigma$ Struktur $\mathcal{A}$ definieren. Sofern $F$ freie Variable enthält, hängt der Wahrheitswert von $F$ in $\mathcal{A}$ natürlich von den zu $F$ **passenden** Belegungen $\alpha : \mathcal{M} \longrightarrow \mathcal{A}$ ab, das sind diejenigen Belegungen, die $\mathrm{Frei}(F) \subseteq \mathcal{M}$ erfüllen. Treten andererseits $\Sigma$-Propositionen auf, so sind diesen im Rahmen der $\Sigma$ Struktur schon Wahrheitswerte zugeordnet.

**Definition 10.7** (**Semantik der Prädikatenlogik**)**.** Gegeben: Eine Signatur $\Sigma$, eine $\Sigma$ Struktur $\mathcal{A}$ mit Träger $\mathcal{A}$, eine Formel $F$ mit $\mathrm{Frei}(F) \subseteq \mathcal{M}$ und eine Belegung $\alpha : \mathcal{M} \longrightarrow \mathcal{A}$. Wir definieren $\widehat{\alpha}(F)$ in $\mathcal{A}$ wie folgt:

- atomare Formeln:

$$\begin{array}{rlll} \widehat{\alpha}(t_0 = t_1) = 1 & \text{g.d.w.} & \overline{\alpha}(t_0) = \overline{\alpha}(t_1) & \text{gilt} \\ \widehat{\alpha}(R(t_0, \ldots, t_{n-1})) = 1 & \text{g.d.w.} & R^{\mathcal{A}}(\overline{\alpha}(t_0), \ldots, \overline{\alpha}(t_{n-1})) & \text{gilt} \end{array}$$

- zusammengesetzte Formeln

$$\begin{array}{rll} \widehat{\alpha}(\neg G) = 1 & \text{g.d.w.} & \widehat{\alpha}(G) = 0 \\ \widehat{\alpha}(G \wedge H) = 1 & \text{g.d.w.} & \widehat{\alpha}(G) = 1 \text{ und } \widehat{\alpha}(H) = 1 \\ \widehat{\alpha}(G \vee H) = 1 & \text{g.d.w.} & \widehat{\alpha}(G) = 1 \text{ oder } \widehat{\alpha}(H) = 1 \\ \widehat{\alpha}(\forall x : G) = 1 & \text{g.d.w.} & \text{für alle } a \in \mathcal{A} \text{ gilt } \widehat{\alpha[^a/_x]}(G) = 1 \\ \widehat{\alpha}(\exists x : G) = 1 & \text{g.d.w.} & \text{es gibt ein } a \in \mathcal{A} \text{ mit } \widehat{\alpha[^a/_x]}(G) = 1 \end{array}$$

**Bemerkung 10.8.**

(a) Für 0-stellige Prädikatssymbole oder Σ-Propositionen $R$ gilt $R^{\mathcal{A}}$ genau dann, wenn $R^{\mathcal{A}}$ der Wahrheitswert 1 ist. Die Zuweisung solcher Wahrheitswerte im Rahmen der Σ-Struktur unterliegt keinen Einschränkungen, ist also **frei wählbar**.

(b) Für 1-stellige Prädikatssymbole $R$ gilt die Formel $R(t)$ genau dann in $\mathcal{A}$ unter $\alpha$, wenn die Interpretation $\overline{\alpha}(t)$ des Terms die Eigenschaft $R^{\mathcal{A}}$ hat – d.h. wenn $R^{\mathcal{A}}(\overline{\alpha}(t))$ gilt. Die Semantik einer 2-stelligen Relation $R^{\mathcal{A}}$ unter $\alpha$ macht $R(t_0, t_1)$ ist genau dann wahr, wenn die Elemente $\overline{\alpha}(t_0)$ und $\overline{\alpha}(t_1)$ in der Relation $R^{\mathcal{A}}$ stehen.

(c) Bei der Interpretation von $\forall x : G$ bzw. $\exists x : G$ ist zu beachten, dass bei passenden Belegungen $\alpha$ der Wert $\alpha(x)$, sofern er existiert, irrelevant für den Wahrheitswert der quantifizierten Formel ist, da $x$ dort nicht frei vorkommt. Insofern kann die Funktion $\alpha$ an der Stelle $x$ beliebig abgewandelt werden, was durch die „Updates“ $\alpha[{}^{a}/_{x}]$ realisiert wird. Je nach Quantor ist entscheidend, ob der Wert von $G$ unter allen oder mindestens einem derartigen Update den Wert 1 annimmt.

(d) In Analogie zu Bemerkung 2.11 lassen sich die Wahrheitswerte der Formeln auch anders (und nützlicher) ausdrücken:

$$\begin{aligned}
\widehat{\alpha}(\neg G) &= 1 - \widehat{\alpha}(G) \\
\widehat{\alpha}(G \wedge H) &= \inf\{\widehat{\alpha}(g), \widehat{\alpha}(H)\} \\
\widehat{\alpha}(G \vee H) &= \sup\{\widehat{\alpha}(g), \widehat{\alpha}(H)\} \\
\widehat{\alpha}(\forall x : G) &= \inf\{\, \widehat{\alpha[{}^{a}/_{x}]}(G) \,:\, a \in \mathcal{A} \,\} \\
\widehat{\alpha}(\exists x : G) &= \sup\{\, \widehat{\alpha[{}^{a}/_{x}]}(G) \,:\, a \in \mathcal{A} \,\}
\end{aligned}$$

**Beispiel 10.9.** In (nichtleeren) gerichteten Graphen $\mathcal{A}$ als Σ-Strukturen (vgl. Beispiel 10.2) besagt die Gültigkeit der Formel

$$\forall x : \forall y : (R(x, y) \Rightarrow R(y, x))$$

dass der Graph $\mathcal{A}$ symmetrisch oder ungerichtet ist: zu jeder gerichteten Kante gibt es eine Kante in der entgegengesetzten Richtung. Die entsprechende Formel ohne Quantoren, also

$$R(x, y) \Rightarrow R(y, x)$$

hat keinen festen Wahrheitswert: Wir müssen die freien Variablen $x$ und $y$ zuerst als feste Knoten $a, b$ interpretieren, d.h., Belegungen $\alpha$ mit $\alpha(x) := a$ und $\alpha(y) := b$ verwenden. Dann verlangt die Formel: Wenn eine Kante von $a$ nach $b$ führt, dann gibt es auch eine Kante in der Gegenrichtung. Der Wahrheitswert hängt hier von der gewählten Interpretation, genauer, von den ausgewählten Knoten $a$ und $b$ ab. Sie kann für bestimmte Knoten richtig, für andere dagegen falsch sein.

**Beispiel 10.10.**

(a) In der $\Sigma$-Struktur $\mathcal{N}$ der natürlichen Zahlen aus Beispiel 10.3 ist die Formel

$$\forall x : \exists y : x < y$$

wahr, man kann z.B. $y = s(x) = x + 1$ setzen. In Ermangelung einer größten natürlichen Zahl liefert die Vertauscung der Quantoren eine falsche Formel:

$$\exists y : \forall x : x < y$$

(b) Die Formel $\forall x : (0 < x \vee 0 = x)$, die 0 als kleinstes Element charakterisiert, ist wahr in $\mathcal{N}$ aber nicht in $\mathcal{Z}$.

**Satz 10.11.** *Für eine Signatur $\Sigma$ und eine $\Sigma$ Struktur $\mathcal{A}$ mit Träger $\mathcal{A}$ hängt der Wahrheitswert einer Formel $F$ in $\mathcal{A}$ nur von der Interpretation ihrer freien Variablen ab: Für alle zu $F$ passenden Belegungen $\alpha$ und $\beta$ gilt*

$$\alpha(x) = \beta(x) \textit{ für alle } x \in \mathrm{Frei}(F) \quad \textit{impliziert} \quad \widehat{\beta}(F) = \widehat{\alpha}(F)$$

*Beweis.* Wir stellen zunächst fest, dass die Interpretation der $\Sigma$-Propositionen bereits durch die $\Sigma$-Struktur vorgegeben ist.

Aufgrund von Definition 10.5 folgt sofort, dass für zu $F$ passende Belegungen $\alpha$ und $\beta$, die Erweiterungen $\bar{\alpha}$ und $\bar{\beta}$ auf der Term-Menge $\mathrm{Frei}(F)$ übereinstimmen.

Zwecks struktureller Induktion über den Aufbau von $F$ unterscheiden wir die Fälle:

(a) $F$ sei atomar. Dann sind alle auftretenden Variablen von $F$ frei und $\widehat{\alpha}(F)$ ist genau dann wahr wenn $\widehat{\beta}(F)$ wahr ist, denn $\overline{\alpha}$ und $\overline{\beta}$ stimmen gemäß der Vorüberlegung bei allen auftretenden Termen überein. Treten keine Terme auf, so handelt es sich bei $F$ um eine $\Sigma$-Proposition, und deren Wert ist durch die $\Sigma$-Struktur bereits bestimmt.

(b) $F$ resultiert aus aussagenlogischer Verknüpfung, hat etwa die Form $G \wedge H$, und die Behauptung gelte für $G$ und $H$. Dann folgt wegen $\mathrm{Frei}(G), \mathrm{Frei}(H) \subseteq \mathrm{Frei}(F)$

$$\begin{aligned}\widehat{\alpha}(F) = \widehat{\alpha}(G \wedge H) &= \inf\{\widehat{\alpha}(G), \widehat{\alpha}(H)\} \\ &= \inf\{\widehat{\beta}(G), \widehat{\beta}(H)\} = \widehat{\beta}(G \wedge H) = \widehat{\beta}(G \wedge H)\end{aligned}$$

Analog argumentiert man für $F = G \vee H$ und $F = \neg H$.

(c) $F$ habe die Form $\forall x : G$ und die Behauptung gelte für $G$. In diesem Fall gilt wegen $\mathrm{Frei}(G) \subseteq \mathrm{Frei}(F) + \{x\}$

$$\begin{aligned}\widehat{\alpha}(F) &= \inf\{\, \widehat{\alpha[^a/_x]}(G) \,:\, a \in \mathcal{A} \,\} \\ &= \inf\{\, \widehat{\beta[^a/_x]}(G) \,:\, a \in \mathcal{A} \,\} = \widehat{\beta}(F)\end{aligned}$$

Analog argumentiert man für $\exists x : F$. □

**Notationelle Konvention 10.12.** Der Wahrheitswert eines Satzes $F$ in einer $\Sigma$-Struktur $\mathcal{A}$ ist unabhängig von Variablenbelegungen. Wir sagen, $F$ **gilt in** $\mathcal{A}$, wenn der Wahrheitswert unter jeder Variablenbelegung den Wert 1 annimmt.

**Beispiel 10.13.** Die Formel $\forall x : \forall y : (R(x, y) \Rightarrow R(y, x))$ aus 10.9 gilt genau in allen symmetrischen oder ungerichteten Graphen.

# 11. Logische Äquivalenz

Wir fahren fort, Begriffsbildungen aus der Aussagenlogik in die Prädikatenlogik zu übertragen.

**Definition 11.1.** Zwei Formeln $F$ und $G$ der Prädikatenlogik heißen **äquivalent**, in Symbolen

$$F \equiv G$$

falls für jede $\Sigma$-Struktur $\mathcal{A}$ und jede zu $F$ und $G$ passende Belegung $\alpha : \mathcal{M} \longrightarrow \mathcal{A}$ gilt: $\widehat{\alpha}(F)$ ist genau dann wahr, wenn $\widehat{\alpha}(G)$ wahr ist. Kurz:

$$\widehat{\alpha}(F) = \widehat{\alpha}(G).$$

**Bemerkung 11.2.** Jede logische Äquivalenz der Aussagenlogik ist auch in der Prädikatenlogik gültig. Z.B., die De Morgansche Regel:

$$\neg(F \wedge G) \equiv \neg F \vee \neg G$$

Der Beweis aus Satz 3.2 kann wörtlich übernommen werden.

Um neue Äquivalenzen zu finden, sind die neuen Junktoren $\forall x :$ und $\exists x :$ in Betracht zu ziehen. Aber zuvor ist noch die Verträglichkeit von $\equiv$ mit diesen neuen Junktoren zu prüfen.

**Satz 11.3.** *Die Äquivalenz ist auch hinsichtlich der neuen Junktoren $\forall x :$ und $\exists x :$ eine Kongruenzrelation.*

*Beweis.* Aus $G \equiv H$ folgt sofort für jede $\Sigma$-Struktur $\mathcal{A}$, jedes Element $a$ des Trägers $\mathcal{A}$ und jede passende Belegung $\alpha$

$$\widehat{\alpha[^a/_x]}(G) = \widehat{\alpha[^a/_x]}(H)$$

und somit

$$\widehat{\alpha}(\forall x : G) = \widehat{\alpha}(\forall x : H)$$

Den Existenz-Quantor behandelt man analog. □

**Beispiel 11.4.** Die Aussage

Nicht alle Vögel können fliegen.

ist eine weniger direkte Variante der Aussage

Es gibt einen Vogel, der nicht fliegen kann.

Da es nur endlich viele Vögel gibt, liegt ein Vergleich mit den endlich iterierten Versionen der Junktoren $\wedge$ und $\vee$ nahe, vergl. Bemerkung 3.5. Für diese war in Bemerkung 3.6 bereits eine Verallgemeinerung der de Morgan'schen Regeln erfolgt, die nun auch folgenden Satz motiviert:

**Satz 11.5.** *Für jede Formel $F$ und jede Variable $x$ gilt*

$$\neg\forall x : F \equiv \exists x : \neg F \qquad \textit{und} \qquad \neg\exists x : F \equiv \forall x : \neg F$$

*Beweis.* Für jede passende Belegung $\alpha$ gilt

$$\begin{aligned}\widehat{\alpha}(\neg(\forall x : F)) &= 1 - \widehat{\alpha}(\forall x : F)\\ &= 1 - \inf\{\, \widehat{\alpha[^a/_x]}(F) \,:\, a \in \mathcal{A} \,\}\\ &= \sup\{\, 1 - \widehat{\alpha[^a/_x]}(F) \,:\, a \in \mathcal{A} \,\}\\ &= \widehat{\alpha}(\exists x : \neg F)\end{aligned}$$

Das zweite Ergebnis folgt analog, oder auch durch Substitution von $\neg F$ für $F$ und anschließende Negation:

$$\neg\forall x : \neg F \equiv \exists x : F \quad \text{und somit} \quad \forall x : \neg F \equiv \neg\exists x : F \qquad \square$$

**Folgerung 11.6.** *Für die Prädikatenlogik sind die Junktoren $\wedge$ und $\neg$ zusammen mit dem All-Quantor $\forall$ adäquat (siehe Definition 3.9): Alle Formeln der Prädikatenlogik lassen sich mit $\wedge$, $\neg$ and $\forall$ ausdrücken.*

**Proposition 11.7.** *Es gelten die folgenden verallgemeinerten Assoziativ-Gesetze*

$$\forall x : (F \wedge G) \equiv (\forall x : F) \wedge (\forall x : G) \quad \textit{und} \quad \exists x : (F \vee G) \equiv (\exists x : F) \vee (\exists x : G)$$

*Beweis.* Für jede passende Belegung $\alpha$ gilt

$$\begin{aligned}\widehat{\alpha}(\forall x : (F \wedge G)) &= \inf\{\, \widehat{\alpha[^a/_x]}(F \wedge G) \,:\, a \in \mathcal{A} \,\}\\ &= \inf\{\, \inf\{\widehat{\alpha[^a/_x]}(F), \widehat{\alpha[^a/_x]}(G)\} \,:\, a \in \mathcal{A} \,\}\\ &= \inf\{\inf\{\, \widehat{\alpha[^a/_x]}(F) \,:\, a \in \mathcal{A} \,\}, \inf\{\, \widehat{\alpha[^a/_x]}(G) \,:\, a \in \mathcal{A} \,\}\,\}\\ &= \inf\{\widehat{\alpha}(\forall x : F), \widehat{\alpha}(\forall x : G)\}\\ &= \widehat{\alpha}(\forall x : F \wedge \forall x : G)\end{aligned}$$

Das zweite Ergebnis kann man analog herleiten, oder auch aus dem ersten Teil mit Hilfe der de Morgan'schen Regeln (siehe Satz 11.5)

$$\begin{aligned}\exists x : (F \vee G) &\equiv \neg\forall x : \neg(F \vee G)\\ &\equiv \neg\forall x : (\neg F \wedge \neg G)\\ &\equiv \neg((\forall x : \neg F) \wedge (\forall x : \neg G))\\ &\equiv (\neg(\forall x : \neg F)) \vee (\neg(\forall x : \neg G))\\ &\equiv (\neg(\neg\exists x : F)) \vee (\neg(\neg\exists x : G))\\ &\equiv (\exists x : F) \vee (\exists x : G)\end{aligned}$$ □

**Bemerkung 11.8.** Genausowenig, wie zwischen Konjunktion $\wedge$ und Disjunktion $\vee$ eine gemische Assoziativität gilt

$$(F \wedge G) \vee H \not\equiv F \wedge (G \vee H)$$

ist auch der All-Quantor mit $\vee$ bzw. der Existenz-Quantor mit $\wedge$ unverträglich:

$$\forall x : (F \vee G) \not\equiv (\forall x : F) \vee (\forall x : G)$$

Aufpassen! Oberflächlich betrachtet sieht dieses Beispiel den beiden vorigen ähnlich (nicht wirklich, wenn man sich klarmacht, dass der All-Quantor eine verallgemeinerte Konkunktion und der Existenz-Quantor eine verallgemeinerte Disjunktion ist), doch hier sind die beiden Seiten **nicht äquivalent**. Links steht, dass für jedes $x$ eine der beiden Aussagen zutrifft. Dagegen steht rechts, dass eine der Aussagen für jedes $x$ zutrifft. Wenn $F$ nur auf eine nichtleere Teilmenge des Trägers zutrifft und $G$ auf den nichtleeren Rest, dann ist die linke Seite wahr; – die rechte aber nicht. Dies tritt etwa bei der Signatur $\mathcal{N}$ aus Beispiel 10.3 bei den folgenden Formeln auf:

$$\forall x : (3 < x \vee x < 4)$$

gilt, während

$$(\forall x : 3 < x) \vee (\forall x : x < 4)$$

nicht gilt.

Wie bereits am Ende von Abschnitt 9.iii erwähnt, dürfen gebundene Variable umbenannt werden. Zunächst erweitern wir die von den Updates her bekannte Schreibweise zum Zweck der **Substitution** freier Variablen:

**Definition 11.9.** Für jede Formel $F$, jede Variable $x \in \mathrm{Frei}(F)$ und jeden Term $t \in \mathfrak{T}_\Sigma(\mathcal{V})$ bezeichnet $F[{}^{t}/_{x}]$ diejenige Formel, die entsteht, wenn jedes freie Vorkommen von $x$ durch $t$ substituiert wird.

**Fakt 11.10. Umbennung gebundener Variablen**
Für jede Variable $y$, die in $F$ nicht vorkommt, gilt

$$\forall x : F \equiv \forall y : F[{}^{y}/_{x}] \quad \text{sowie} \quad \exists x : F \equiv \exists y : F[{}^{y}/_{x}]$$

**Bemerkung 11.11.** In der Formel $\forall x : x < 3 \Rightarrow R(x)$ ist die Variable $x$ links gebunden und rechts frei (vergl. Bindungskonvention 9.18). Dies ist verwirrend. Aufgrund von Fakt 11.10 können wir aber die gebundene Variable umbenennen und bekommen die nicht mehr verwirrende Formel $\forall y : y < 3 \Rightarrow R(x)$.

**Notationelle Konvention 11.12.** Wenn eine Variable in einer Formel sowohl gebunden wie frei vorkommt, benennen wir die gebundene Variable um.

**Beispiel 11.13.** In der Formel

$$W(y) \vee \exists y : (B(x, y) \wedge W(y))$$

aus Beispiel 9.19 ist die Variable $y$ sowohl gebunden als frei. Wir können die gebundene Variable umbenennen:

$$W(y) \vee \exists t : (B(x, t) \wedge W(t))$$

Für jede neue Variable $t$ (verschieden von $x$, $y$) ist diese neue Formel logisch äquivalent zu der vorigen.

**Definition 11.14.** Eine Formel liegt in **bereinigter Form** vor, wenn

(a) keine ihrer Variablen frei und gebunden zugleich vorkommt, und

(b) alle Quantoren immer unterschiedliche Variablen binden.

**Beispiel 11.15.** Die Formel

$$\forall x : \forall y : (x < y \vee \forall x : x = 0) \vee \exists y : y < u$$

lässt sich wie folgt bereinigen

$$\forall x : \forall y : (x < y \vee \forall z : z = 0) \vee \exists t : t < u$$

**Folgerung 11.16.** *Jede prädikatenlogische Formel ist zu einer Formel in bereinigter Form äquivalent. Diese ergibt sich durch ggf. mehrmalige Anwendung von Fakt 11.10.*

**Bemerkung 11.17.**

(a) Die Reihenfolge von Quantoren desselben Typs spielt keine Rolle:

$$\forall x : \forall y : F \equiv \forall y : \forall x : F$$

sowie

$$\exists x : \exists y : F \equiv \exists y : \exists y : F$$

Dies ergibt sich eindeutig aus der Definition der Semantik: mehrfaches Bilden von Infima bzw. Suprema ist unabhängig von der Reihenfolge.

(b) Aber aufpassen! Wie schon mehrfach gesehen dürfen wir im Allgemeinen $\forall$ und $\exists$ nicht vertauschen.

**Beispiel 11.18.**

$$\forall x : \exists y : F \not\equiv \exists y : \forall x : F$$

In der Signatur aus 10.3 gilt in $\mathcal{N}$ die Formel

$$\forall x : \exists y : x < y$$

aber nicht die Formel

$$\exists y : \forall x : x < y$$

da es in $\mathbb{N}$ kein größtes Element gibt.

Wir brauchen noch Gegenstücke für die Begriffe der Tautologie und der Erfüllbarkeit in der Aussagenlogik.

**Definition 11.19.** Eine Formel $F$ der Prädikatenlogik heißt

(a) **allgemeingültig** falls sie in jeder $\Sigma$ Struktur unter jeder passenden Belegung wahr ist;

(b) **erfüllbar** falls es eine $\Sigma$-Struktur und eine passende Belegung gibt, sodass $F$ wahr ist.

**Beispiel 11.20.**

(a) Jede Tautologie der Aussagenlogik ist allgemeingültig, z.B. $F \Leftrightarrow \neg\neg F$

(b) Folgende Formel ist allgemeingültig:

$$\forall x : F \Rightarrow \exists x : F$$

Ist $\mathcal{A}$ eine Struktur und $\alpha$ eine Belegung mit $\widetilde{\alpha}(\forall x : F) = 1$, wählen wir ein Element $a \in \mathcal{A} \neq \varnothing$ aus. Dann gilt

$$\widehat{\alpha[{}^{a}/_{x}]}(F) = 1$$

und dies bedeutet $\widehat{\alpha}(\exists x : F) = 1$. Diese Argument setzt voraus, dass der Träger einer $\Sigma$-Struktur nichtleer ist und wird gern zur Rechtfertigung dieser Einschränkung herangezogen.

(c) Die Sätze 2.26 und 11.5 implizieren die Allbemeingültigkeit von

$$\forall x : \neg F \Leftrightarrow \neg\exists x : F$$

(d) Die Äquivalenz $\exists x : R(x) \Leftrightarrow \forall x : R(x)$ ist nicht allgemeingültig: falls $\mathcal{A}$ die Struktur mit Träger $\{0, 1\}$ ist und $R^{\mathcal{A}} = \{0\}$, dann gilt $\exists x : R(x)$ in $\mathcal{A}$ aber $\forall x : R(x)$ gilt nicht. Die Äquivalenz ist aber erfüllbar: Sei $\mathcal{A}$ eine Struktur mit nur einem Element im Träger, dann erfüllt $\mathcal{A}$ die Formel.

(e) Die Formel $\exists x : W(x) \wedge \forall x : \neg W(x)$ ist nicht erfüllbar.

# 12. Normalformen

**Definition 12.1.** Eine Formel, deren Quantoren vollständig am Anfang stehen, liegt in **Pränex-Normalform** oder **PNF** vor, etwa

$$\mathsf{Q}_0 x : \mathsf{Q}_2 y : \ldots \mathsf{Q}_{n-1} z : F$$

wobei $\mathsf{Q}_i \in \{\forall, \exists\}$ gilt und $F$ keine Quantoren enthält.

Alternieren zudem All- und Existenz-Quantoren, so spricht man von **alternierender Pränex-Normalform**, oder **aPNF**. Solche Formeln haben die Form

$$\forall x_0 : \exists x_1 : \forall x_2 : \ldots \mathsf{Q}_{n-1} x_{n-1} : F$$

**Beispiel 12.2.** $F \vee \forall x : G$ liegt nicht in PNF vor. Falls $x \notin \mathrm{Frei}(F)$, speziell falls die Ausgangsformel bereinigt ist, finden wir eine äquivalente Formel in PNF, indem wir $F$ „unter den Quantor ziehen“

$$F \vee \forall x : G \equiv \forall x : (F \vee G)$$

Das ist zulässig, da nach Voraussetzung $x \notin \mathrm{Frei}(F)$ gilt und durch das Verschieben in den Gültigkeitsbereich des Quantors keine freie Variable von $F$ gebunden wird. Speziell gilt $\widehat{\alpha[^a/_x]}(F) = \widehat{\alpha}(F)$ für jede zu $F$ passende Belegung und jedes $a \in \mathcal{A}$. Folglich

$$\begin{aligned}
\widehat{\alpha}(F \vee \forall x : G) &= \sup\{\widehat{\alpha}(F), \inf\{\, \widehat{\alpha[^a/_x]}(G) \,:\, a \in \mathcal{A}\,\}\} \\
&= \inf\{\, \sup\{\widehat{\alpha}(F), \widehat{\alpha[^a/_x]}(G)\} \,:\, a \in \mathcal{A}\,\} \\
&= \inf\{\, \sup\{\widehat{\alpha[^a/_x]}(F), \widehat{\alpha[^a/_x]}(G)\} \,:\, a \in \mathcal{A}\,\} \\
&= \inf\{\, \{\widehat{\alpha[^a/_x]}(F \vee G)\} \,:\, a \in \mathcal{A}\,\} \\
&= \widehat{\alpha}(\forall x : (F \vee G))
\end{aligned}$$

**Beispiel 12.3.** Analog dazu gelten für $x \notin \mathrm{Frei}(F)$ die folgenden Äquivalenzen:

$$\begin{aligned}
F \vee \exists x : G &\equiv \exists x : (F \vee G) \\
F \wedge \forall x : G &\equiv \forall x : (F \wedge G) \\
F \wedge \exists x : G &\equiv \exists x : (F \wedge G)
\end{aligned}$$

**Algorithmus 12.4** (PNF)**.**
Zu einer Formel $F$ der Prädikatenlogik wollen wir eine äquivalente Formel

$$\mathsf{PNF}(F) = \mathsf{Q}_0 x_0 : \ldots \mathsf{Q}_{n-1} x_{n-1} : F^*$$

berechnen, wobei $F^*$ keine Quantoren enthält.

ObdA möge $F$ bereits in bereinigter Form vorliegen.

0. $F$ atomar: wir setzen
$$\mathsf{PNF}(F) := F$$

1. $F = \neg G$ mit $\mathsf{PNF}(G) = \mathsf{Q}_0 x_0 : \ldots \mathsf{Q}_{n-1} x_{n-1} : G^*$. Wir setzen
$$\mathsf{PNF}(F) := \bar{\mathsf{Q}}_0 x_0 : \ldots \bar{\mathsf{Q}}_{n-1} x_{n-1} : \neg G^*$$
wobei $\bar{\forall} = \exists$ und $\bar{\exists} = \forall$ gilt.

2. $F = G \wedge H$ mit $\mathsf{PNF}(G)$ wie oben und $\mathsf{PNF}(H) = \mathsf{Q}'_0 y_0 : \ldots \mathsf{Q}'_{m-1} y_{m-1} : H^*$. Wir setzen
$$\mathsf{PNF}(F) := \mathsf{Q}_0 x_0 : \ldots \mathsf{Q}_{n-1} x_{n-1} : \mathsf{Q}'_0 y_0 : \ldots \mathsf{Q}'_{m-1} y_{m-1} : (G^* \wedge H^*)$$
(Da $F$ bereinigt ist, treten in $G$ und $H$ unterschiedliche gebundene Variablen auf. Weiter sind keine freien Variablen von $H$ in $G$ gebunden und umgekehrt.)

3. $F = G \vee H$: Analog zu 2. mit $\vee$ anstelle von $\wedge$.

4. $F = \forall x : G$ mit $\mathsf{PNF}(G)$ wie oben. Wir setzen
$$\mathsf{PNF}(F) = \forall x : \mathsf{Q}_0 x_0 : \ldots \mathsf{Q}_{n-1} x_{n-1} : G^*$$

5. $F = \exists x : G$ mit $\mathsf{PNF}(G)$ wie oben. Wir setzen
$$\mathsf{PNF}(F) = \exists x : \mathsf{Q}_0 x_0 : \ldots \mathsf{Q}_{n-1} x_{n-1} : G^*$$

**Satz 12.5.** *Der Algorithmus ist korrekt: die konstruierte Pränex-Normalform $\mathsf{PNF}(F)$ ist äquivalent zur Ausgangsformel $F$.*

*Beweis.* Fall 1 folgt aus Satz 11.5. Die Fälle 2 und 3 folgen aus Beispiel 12.3, während Satz 11.3 die Fäll 4 und 5 impliziert. □

**Beispiel 12.6.** Wir betrachten
$$F = \forall x : \exists y : x < y \wedge \neg \forall z : x + z = z$$
Die Variable $x$ ist im hinteren Teil der Formel frei, deshalb wird am Anfang statt $x$ die Variable $t$ verwendet. Das liefert die bereingte Form
$$F \equiv \forall t : \exists y : t < y \wedge \neg \forall z : x + z = z$$
Die Anwendung von Fall 1 ergibt nun:
$$F \equiv \forall t : \exists y : t < y \wedge \exists z : \neg(x + z = z)$$
Jetzt verwenden wir Fall 2 und erhalten
$$F \equiv \forall t : \exists y : \exists z : (t < y \wedge \neg(x + z = z))$$
Wir wollen demnächst beweisen, dass man für die Entscheidungen der Erfüllbarkeit von Formeln ohne den Existenz-Quantor auskommen kann. Zunächst ein Beispiel:

**Beispiel 12.7.** Die Stetigkeitsformel aus Beispiel 9.16 wollen wir ohne Existenz-Quantor umschreiben: da die Existenz-Quantifizierung von $\delta$ *nach* der universellen Quantifizierung von $\epsilon$ erfolgt („zu jedem $\varepsilon$ existiert ein $\delta$“), also $\delta$ von $\epsilon$ *abhängt*, denken wir uns für jedes $\epsilon$ ein spezifisches $\delta$ mit der gewünschten Eigenschaft *ausgewählt*.

Bezeichnet man die Auswahl mit $\delta = h(\epsilon)$, kann man die Signatur $\Sigma = \langle m, a, f, 0, 2; <\rangle\}$ um ein neues 1-stelliges Funktionssymbol $h$ zu $\Sigma'$ erweitern. In der neuen Signatur läßt sich die Stetigkeit dann wie folgt formalisieren:

$$\forall \epsilon : (\epsilon > 0 \Rightarrow (h(\epsilon) > 0 \wedge \forall x : (h(\epsilon) > am(x, 2) \Rightarrow \epsilon > am(f(x), f(2)))))$$

Im Weiteren werden wir sehen, wie man diese Idee anwenden kann, um den Quantor $\exists$ vollständig zu eliminieren.

**Definition 12.8.** Eine Pränex-Normalform ohne Existenz-Quantor heißen auch **Skolemsche Normalform**.

**Beispiel 12.9.** Um in der folgenden Formel in Pränex-Normalform

$$\exists x : \forall y : \exists z : x < z < y$$

den führenden Existenz-Quantor zu eliminieren, können wir die Signatur $\Sigma = \{<\}$ um eine neue Konstante $c$ erweitern und schreiben

$$\forall y : \exists z : c < z < y$$

Das von $y$ abhängige $z$ läßt sich wie oben mit Hilfe eines Funktonssymbols $f$ ausdrücken

$$\forall y : c < f(y) < y$$

Dabei benutzen wie die neue Signatur $\langle c, f; <\rangle$

**Bemerkung 12.10.** Die Formeln

$$F := \exists x : \forall y : \exists z : x < z < y \qquad \text{bzw.} \qquad S_F := \forall y : c < f(y) < y$$

sind **nicht äquivalent** (warum?), aber sie haben die Eigenschaft, dass $F$ erfüllbar ist genau dann wenn $S_F$ erfüllbar ist.

In der Tat, falls $F$ in einer Struktur $\mathcal{A}$ erfüllbar ist, haben wir noch die Freiheit $c$ und $f$ in $\mathcal{A}$ zu bestimmen. Wir wählen für $c$ eine Interpretation der Variable $x$, für die restliche Formel $\forall y : \exists z : x < z < y$ wahr ist. Und $f^{\mathcal{A}}$ definieren wir so, dass für alle $y$ die Formel $c < f^{\mathcal{A}}(y) < y$ gilt. Umgekehrt: Falls $S_F$ erfüllbar ist, ist es auch $F$. Denn $x$ in $F$ kann durch die Konstante $c$ in $S_F$ interpretiert werden, und für $z$ kann man immer das Element wählen, das durch $f(y)$ bestimmt wird.

**Algorithmus 12.11.** Für eine Signatur $\Sigma$ wollen wir einer jeden Formel in Pränex-Normalform

$$F := \mathsf{Q}_0 x_0 : \ldots \mathsf{Q}_{n-1} x_{n-1} : F^*$$

eine Formel $S_F$ für eine potentiell größere Signatur $\Sigma'$ zuordnen, die in Skolemscher Normalform vorliegt und genau dann erfüllbar ist, wenn $F$ erfüllbar ist.

Wir entfernen die Existenz-Quantoren von links nach rechts.

Sind alle Quantoren in $F$ All-Quantoren, so setzen wir

$$S_F := F$$

Andernfalls treten $n \in \mathbb{N}$ All-Quantoren vor dem ersten Existenz-Quantor auf, also

$$F := \forall x_0 : \forall x_1 : \ldots \forall x_{n-1} \exists x_n : G$$

wobei $G$ weitere Quantoren enthalten kann. Da $F$ in PNF vorliegt, ist $G$ automatisch bereinigt. Wir erweitern die Signatur um ein neues $n$-stelliges Funktionssymbol $h$ und setzen

$$S_F := \forall x_0 : \forall x_1 \ldots \forall_{x_{n-1}} : S_G[^{h(x_0,x_1,\ldots,x_{n-1})}/_{x_3}]]$$

**Satz 12.12.** *Der Algorithmus ist korrekt: für jede pränexe Normalform $F$ ist $S_F$ eine Skolemsche Normalform, die genau dann erfüllbar ist, wenn $F$ es ist.*
Den Beweis findet der Leser z.B. in Kreuzer, Kühling „Logik für Informatiker“ (Pearson Studium 2006), Satz 4.22.

# 13. Herbrandsche Modelle und abstrakte Datentypen

Wir haben mit $\mathcal{T}_\Sigma(\mathcal{M})$ die Menge aller Terme mit Variablen aus $\mathcal{M}$ bezeichnet. Ein Spezialfall ist die Menge

$$\mathcal{T}_\Sigma(\varnothing)$$

aller Terme ohne Variablen, der sogenannten **Grundterme**. Diese haben eine Bewertung in *jeder* $\Sigma$ Struktur.

**Beispiel 13.1.** Falls $\Sigma$ aus einer Konstante, 0, und einem einstelligen Operationssymbol, $s$, besteht, ist $\mathcal{T}_\Sigma(\varnothing)$ die Menge $\{0, s(0), s(s(0)), \ldots\}$. In jeder $\Sigma$-Struktur $(\mathcal{A}, 0^\mathcal{A}, s^\mathcal{A})$ ist die Bewertung des Termes $s^n(0)$ ($n$-fache Anwendung des Funktionssymbols $s$ auf die Konstante 0) gleich dem Element $(s^\mathcal{A})^n(0^\mathcal{A})$ von $\mathcal{A}$, das durch $n$-fache Anwendung der Funktion $s^\mathcal{A}$ auf die Konstante $0^\mathcal{A}$ entsteht.
Für die Signatur $\Sigma' = \{0, s, <\}$, die $\Sigma$ um ein zweistelliges Prädikatensymbol $<$ erweitert, haben wir dieselben Grundterme:

$$T_{\Sigma'}(\varnothing) = \mathcal{T}_\Sigma(\varnothing)$$

**Beispiel 13.2.** Für $\Sigma = \{+, 0, 1\}$ mit einer zweistelligen Operation $+$ und Konstanten 0 und 1 besteht $\mathcal{T}_\Sigma(\varnothing)$ aus Termen wie $0, 1, 0+1, 0+0, 0+(0+1), (1+1)+(0+1), \ldots$.
In jeder $\Sigma$-Struktur haben wir eine offensichtliche Bewertung all dieser Grundterme.

**Beispiel 13.3.** Für jede Signatur $\Sigma$ ohne Konstanten gilt $\mathcal{T}_\Sigma(\varnothing) = \varnothing$

**Definition 13.4.** Eine **Herbrandsche $\Sigma$ Struktur** ist eine $\Sigma$-Struktur $\mathcal{A}$, deren Träger $\mathcal{A}$ syntaktisch (oder „frei“) durch Anwendung der Funktionssymbole auf Grundterme erzeugt ist, d.h., $\mathcal{A}$ ist die kleinste Menge, die

- $\mathcal{T}_\Sigma(\varnothing)$ umfaßt, d.h., jeder Grundterm ist Element von $\mathcal{A}$;
- mit jedem $n$-stelliges Funktionssymbol $f$ und $n$ Elementen $t_i \in \mathcal{A}$, $i < n$, auch $f(t_0, \ldots, t_{n-1})$ enthält.

Die Funktionen $f^\mathcal{A}$ sind entsprechend syntaktisch definiert:

$$f^\mathcal{A}(t_0, \ldots, t_{n-1}) := f(t_0, \ldots, t_{n-1})$$

Die Prädikate können beliebig definiert sein.

**Beispiel 13.5.**

(a) Für $\Sigma = \{s, 0\}$ in Beispiel 13.1 gibt es nur eine Herbrandsche Struktur: der Träger besteht aus allen Iterationen $s^n(0)$, $n \in \mathbb{N}$ des Funktionssymbols auf die Konstante 0, und die Operation $s^{\mathcal{A}}$ ordnet jeden Term $s^n(0)$ den Term $s^{n+1}(0)$ zu.

(b) Für die Signatur $\Sigma' = \{s, 0, <\}$ gibt es viele Herbrandsche Strukturen. Z.B. $\mathcal{A}$ mit Elementen und Funktionen wie in (a) und mit $<^{\mathcal{A}}$ definiert durch

$$\left(s^{\mathcal{A}}\right)^n(0) <^{\mathcal{A}} \left[\left(s^{\mathcal{A}}\right)^m(0) \quad \text{g.d.w} \quad n < m\right.$$

Oder die Struktur $\mathcal{B}$ mit denselben Elementen und Funktonen wie $\mathcal{A}$, aber mit $<^{\mathcal{B}} = \varnothing$.

**Bemerkung 13.6.**

(a) Wenn wir die Erfüllbarkeit von Formeln in Skolemscher Normalform untersuchen wollen, z.B. von

$$\forall x : \forall y : x + y > a$$

können wir uns bei der Suche auf Herbrandsche Strukturen beschränken: falls ein Modell dieser Formel existiert, können wir aufgrund von Satz 13.8 annehmen, dass seine Elemente ausschließlich die Form $a, a + a, a + (a + a), (a + a) + a, \ldots$ haben und seine zweistellige Funktion die syntaktische Zusammensetzung $\langle s, t\rangle \longmapsto t + s$ ist.

(b) Ist die Formel

$$F := \forall x : \neg(x > a \vee a > x) \wedge \forall x : (x > b \Rightarrow x > a) \wedge a > b$$

erfüllbar? Zunächst benötigen wir $F$ in bereinigter Form, dann in PNF, was in diesem Fall auch schon Skolemsche Normalform ist:

$$\begin{aligned} F &\equiv \forall x : \neg(x > a \vee a > x) \wedge \forall y : (y > b \Rightarrow y > a) \wedge a > b \\ &\equiv \forall x : \forall y : \big(\neg(x > a \vee a > x) \wedge (y > b \Rightarrow y > a) \wedge a > b\big) \end{aligned}$$

Alle Herbrand'schen Strukturen haben die Trägermenge $\mathfrak{T}_\Sigma(\varnothing) = \{a, b\}$. Gibt es eine Relation $>^{\mathcal{A}}$ auf der Menge $\{a, b\}$, die $F$ erfüllt? Die letzte Klausel, $a >^{\mathcal{A}} b$, ergibt durch Substitution $a/y$ in der zweiten, dass $a >^{\mathcal{A}} a$ gilt, was der ersten Klausel widerspricht. Daraus folgt gemäß Satz 13.8, dass $F$ nicht erfüllbar ist.

**Notation 13.7.** Gegeben eine Formel $F$. Wir bezeichnen mit

$$\Sigma_F$$

die kleinste Signatur mit einer Konstanten, so dass $F$ eine passende Formel ist.

**Satz 13.8.** *Für jede Skolemsche Normalform $F$ gilt: falls $F$ erfüllbar ist, gilt sie in einer Herbrandschen Struktur des Signatur $\Sigma_F$.*

Den Beweis kann der Leser in „Logik für Informatiker“ von U. Schöning finden.

**Bemerkung 13.9.** Für Signaturen $\Sigma$ ohne Prädikatensymbole ist $\mathcal{T}_\Sigma(\varnothing)$ eine $\Sigma$ Struktur. Solche kann man als ein Modell **abstrakter Datentypen** auffassen: in einem Datentyp nehmen wir an, dass es eine Liste von Operationen gibt, die mit den Daten durchgeführt werden können. Diese bilden dann eine Signatur ohne Prädikatensymbole. Der abstrakte Datentyp ist dann das Modell, das durch Anwendung der Operationen auf die Grundterme entsteht, und für das auch keine nichttrivialen Gleichungen gelten. (Mathematisch gesprochen handelt es sich um „freie $\Sigma$-Algebren“ über der leeren Menge $\varnothing$.) Ein paar Beispiele:

**Beispiel 13.10** (Abstrakte Datentypen)**.**

(a) Natürliche Zahlen: Wir gehen von einer Konstante 0 und einer einstelligen Operation $s$ aus. Für $\Sigma = \{s, 0\}$ haben wir schon gesehen, dass $\mathcal{T}_\Sigma(\varnothing)$ die Menge aller Terme $0, s(0), s(s(0)), \ldots$ ist, die die natürlichen Zahlen $0, 1, 2, \ldots$ codieren

(b) Listen. Um die Menge $A^*$ aller Listen in Alphabet $A$ zu formen, brauchen wir eine Konstante $\mathsf{nil}$ und für jedes Symbol $a \in A$ die einstellige Operation, die zu jeder Liste $a$ am Anfang hinzufügt. Wir betrachten die Signatur

$$\Sigma = A \cup \{\mathsf{nil}\}$$

in der jedes Symbol $a \in A$ ein einstelliges Funktionssymbol ist. Hier besteht $\mathcal{T}_\Sigma(\varnothing)$ aus den Termen

$$\mathsf{nil}, a_0(\mathsf{nil}), a_1(a_0(\mathsf{nil})), \ldots (a_i \in A)$$

die die Listen $\mathsf{nil}, a_0, a_0a_0, \ldots$ codieren.

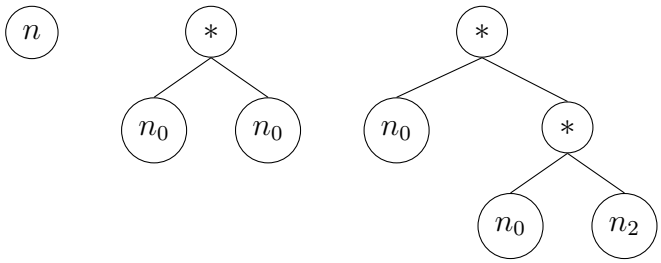
(c) Binäre Bäume: Die Signatur

$$\Sigma = \{*\} \cup \mathbb{N}$$

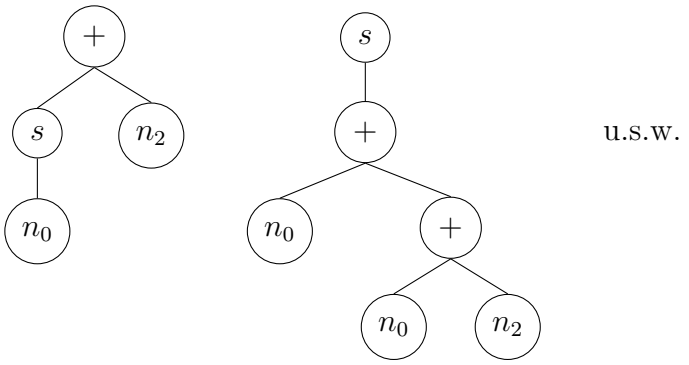
mit einem zweistelligen Operationssymbol $*$ und jeder natürlichen Zahl $n$ als Konstante hat Terme

$$n, *(n_0, n_0), *n_0(*(n_0, n_2)), \ldots$$

die alle binären Syntaxbäume codieren, deren Blätter von links nach rechts durch Listen natürlicher Zahlen markiert werden:



**Beispiel 13.11.** Die natürlichen Zahlen wollen wir jetzt nicht nur mit der Operation $s$ (Nachfolger) und der Konstante 0 versehen, sondern auch mit der Addition + und der Konstante 1. Die ensprechende Signatur $\Sigma = \langle +, s, 0, 1\rangle$ hat als abstrakten Datentyp allerdings **nicht** die natürliche Zahlen, sondern Bäume des Types




u.s.w.

Ähnich wie im Fall der gerichteten/ungerichteten Graphen (Beispiel 10.9) müssen wir hier geeignete Gleichungsaxiome hinzufügen, um die möglichen Strukturen stärker einzugrenzen. Dazu betrachten wir die Formeln, die fordern, dass + kommutativ und assoziativ ist, dass 0 die Einheit und $s$ die Nachfolgerfunktion ist:

$$\forall x : \forall y : x + y = y + x$$
$$\forall x : \forall y : \forall z : (x + y) + z = (x + y + z)$$
$$\forall x : s(x) = x + 1$$
$$\forall x : x = x + 0$$

Hier betrachten wir den abstrakten Datentyp als eine Struktur, die (wieder) aus nichts als den Konstanten entsteht und für die nur die Gleichung, die aus den 4 Formeln ableitbar sind, gelten. In unserem Beispiel liefert dies wieder eine Codierung der natürlichen Zahlen $\mathbb{N}$.

# 14. Resolutionsmethode der Prädikatenlogik

Wir wollen entscheiden, ob eine Formel $F$ der Prädikatenlogik erfüllbar ist. Wir dürfen uns aufgrund von Algorithmus 12.11 auf Formeln in Skolemscher Normalform beschränken:

$$F = \forall x_0 : \ldots \forall x_{n-1} : F^* \qquad\qquad F^* \text{ ohne Quantoren}$$

Darüber hinaus dürfen wir annehmen, dass $F^*$ in KNF vorliegt, was analog wie in der Aussagenlogik definiert ist:

**Definition 14.1.** Ein **Literal** ist eine atomare Formel oder ihre Negation. Eine **Klausel** ist eine Disjunktion von Literalen, z.B. $\neg R(x, y)$. Eine **KNF**ist eine Konjunktion von Klauseln.

**Satz 14.2.** *Für jede Formel $F$ der Aussagenlogik gibt es eine Skolemsche Normalform $S_F$, die genau dann erfüllbar ist, wenn $F$ es ist, und diese hat die Form*

$$S_F := \forall x_0 : \ldots \forall x_{n-1} : G \qquad\qquad G \textit{ ohne Quantoren in KNF}$$

In der Tat haben wir einen Algorithmus, der jeder Formel $F$ eine derartige Skolemsche Normalform $S_F$ zuordnet. Zuerst benutzten wir Algorithmus 12.11 und erhalten die Skolemsche Normalform $S'_F := \forall x_0 : \ldots \forall x_{n-1} : G'$ Anschließend wenden wir den KNF-Algorithmus 4.15 der Aussagenlogik auf $G'$ an.

**Beispiel 14.3.** Ist die folgende Formel erfüllbar?

$$F := \neg P(c) \wedge \forall x : (P(f(x)) \wedge \neg P(x))$$

Wir bringen sie erst in Skolemsche Normalform:

$$S_F := \forall x : (\neg P(c) \wedge P(f(x)) \wedge \neg P(x))$$

Speziell kann von $x$ auch den Wert $c$ annehmen: Falls $F$ (und $S_F$) erfüllbar ist, dann gilt dies auch für

$$\neg P(c) \wedge P(f(c)) \wedge \neg P(c)$$

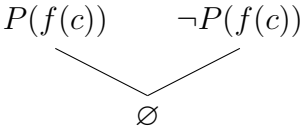
Analog dürfen wir $x$ auch mit $f(c)$ ersetzen: Ist $F$ erfüllbar, ist es auch die Formel:

$$\neg P(c) \wedge P(f(f(c))) \wedge \neg P(f(c))$$

Damit impliziert die Erfüllbarkeit von $F$ auch die der Konjunktion

$$\neg P(c) \wedge P(f(c)) \wedge \neg P(c) \;\wedge\; \neg P(c) \wedge P(f(f(c))) \wedge \neg P(f(c))$$

Nun ist die Antwort aber klar: Diese letzte Formel ist unerfüllbar aufgrund der Resolution




**Bemerkung 14.4.** Das obige Beispiel ist typisch. Wir haben in der Formel $F$ eine Konstante $c$ benutzt und die durch $\forall$ quantifizierte Variable $x$ haben wir mit $c$ oder $f(c)$ ersetzt. Dies dürfen wir, weil der All-Quantor auch jene Elemente des Trägers erfasst, die für die Konstante oder für $f(c)$ stehen.

Dadurch haben sich mehr und mehr komplizierte Formeln ohne freie Variablen ergeben. Auf diese können wir die Resolutionsmethode anwenden.

Falls eine Formel **keine** Konstante enthält, müssen wir extra eine in der Signatur einführen:

**Bemerkung 14.5.** Für eine Skolemsche Normalform

$$G := \forall x_0 : \ldots \forall x_{n-1} : F \qquad\qquad F \text{ ohne Quantoren in } \mathsf{KNF}$$

können wir neue Klauseln ohne Variablen formen, indem wir in den Klauseln von $F$ jede Variable $x_i$ durch einen Grundterm der Signatur $\Sigma_F$ (siehe 9.13) ersetzen. Wenn $G$ erfüllbar ist, ist jede KNF mit diesen Klauseln auch erfüllbar. Der folgende Resolutionssatz besagt die Umgekehrung: Wenn jede solche KNF erfüllbar ist, dann ist $G$ erfüllbar.

**Beispiel 14.6.** Ist die Formel

$$\forall x : \exists y : R(x, y)$$

erfüllbar? Wir nehmen eine 1-stellige Operation $f$ und bekommen die Skolemsche Normalform

$$G := \forall x : R(x, f(x))$$

Diese hat keine Konstanten, also nehmen wir eine neue Konstante $c$ hinzu:

$$\Sigma_G = \{R, f, c\}$$

Dadurch formen wir die entsprechende Menge aller Terme

$$\mathcal{T}_{\Sigma_G}(\varnothing) = \{c, f(c), f(f(c)), \ldots\}$$

Nun können wir Elemente aus $\mathcal{T}_{\Sigma_G}(\varnothing)$ in die Formel $G$ einsetzen und erhalten:

$$\begin{aligned}
&R(c, f(c))\\
&R(c, f(c)) \wedge R(f(c), f(f(c)))\\
&R(c, f(c)) \wedge R(f(c), f(f(c))) \wedge R(f(f(c)), f(f(f(c))))\\
&\quad\vdots
\end{aligned}$$

Diese Formeln sind alle erfüllbar, weil sie keine Resolventen haben. Daraus folgt, dass $F$ erfüllbar ist:

**Satz 14.7** (Resolutionssatz der Prädikatenlogik)**.** *Gegeben sei eine Skolemsche Normalform*

$$G := \forall x_0 : \ldots \forall x_{n-1} : (F_0 \wedge ldots \wedge F_{n-1})$$

*wobei $F_i$ Klauseln sind. Die Formel $G$ ist genau dann erfüllbar, wenn jede aussagenlogische Formel, deren Klauseln die Form*

$$F_i[{}^{t_0}/_{x_0}, \ldots, {}^{t_{n-1}}/_{x_{n-1}}] \qquad \text{mit } t_0, \ldots, t_{n-1} \text{ in } T_{\Sigma_G}(\varnothing)$$

*haben, erfüllbar ist. (In der Klausel $F_i$ wird die $k$-te Variable durch einem Term $t_k$ ohne Variablen ersetzt.)*

**Algorithmus 14.8.** Die **Resolutionsmethode** der Prädikatenlogik ist eine Methode zur Entscheidung, ob eine geschlossene Formel erfüllbar ist.

(PR0) Berechne eine Skolemsche Normalform

$$G := \forall x_0 : \ldots \forall x_{n-1} : F \qquad F \text{ ohne Quantoren und in } \mathsf{KNF}$$

die genau dann erfüllbar ist, wenn es $H$ ist.

(PR1) Berechne $\mathcal{T}_{\Sigma_F}(\varnothing) = \{t_0, t_2, t_3, \ldots\}$.

(PR2) Setze $n = 1$. Ersetze alle Variablen in F durch den Term $t_0$. Wende die Resolutionsmethode der Aussagenlogik auf die entstandene Formel (ohne Variablen) an. Ist das Ergebnis, dass diese Formel unerfüllbar ist, gib aus:

unerfüllbar

und halte. Andernfalls setze $n = 1$. Ersetze alle Variablen in $F$ durch $t_0$ oder $t_1$ und forme mit all diesen Klauseln eine $\mathsf{KNF}$. Wende die Resolutionsmethode der Aussagenlogik auf die dadurch entstandene Formel an. Ist das Ergebnis, dass diese Formel unerfüllbar ist, gib aus:

$G$ unerfüllbar

und halte. Andernfalls setze $n = 3$. Ersetze alle Variablen in F entweder durch $t_0$, $t_1$ oder $t_2$ und forme mit all diesen Klauseln eine $\mathsf{KNF}$; usw.

**Beispiel 14.9.** Ist die Formel

$$G := (\forall x : R(x) \Rightarrow \exists y : S(y)) \Rightarrow \exists x : (R(x) \Rightarrow S(x))$$

allgemeingültig? Wie auch bei der Aussagenlogik ist das genau dann der Fall, wenn die Negation nicht erfüllbar ist:

$$\begin{aligned}
\neg G &:= \neg((\forall x : R(x) \Rightarrow \exists y : S(y)) \Rightarrow \exists x : (R(x) \Rightarrow S(x))) \\
&\equiv \neg(\neg(\forall x : R(x) \Rightarrow \exists y : S(y)) \vee \exists x : (R(x) \Rightarrow S(x))) \\
&\equiv (\forall x : R(x) \Rightarrow \exists y : S(y)) \wedge \neg\exists x : (R(x) \Rightarrow S(x)) \\
&\equiv (\forall x : R(x) \Rightarrow \exists y : S(y)) \wedge \forall x : \neg(R(x) \Rightarrow S(x)) \\
&\equiv (\neg\forall x : R(x) \vee \exists y : S(y)) \wedge \forall x : (R(x) \wedge \neg S(x)) \\
&\equiv (\exists y : S(y) \vee \exists x : \neg R(x)) \wedge \forall z : (R(z) \wedge \neg S(z))
\end{aligned}$$

Dies ist bereits bereinigt, deswegen

$$\neg G \equiv \exists y : \exists x : \forall z : ((S(y) \vee \neg R(x))) \wedge R(z) \wedge \neg S(z)$$

Die Skolemsche Normalform benutzt zwei neue Konstanten $c$ und $d$, die $\exists y$ und $\exists x$ entsprechen:

$$S_{\neg G} = \forall z : (S(c) \vee \neg R(d)) \wedge R(z) \wedge \neg S(z)$$

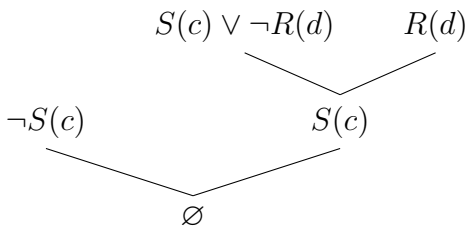
Wir haben keine Funktionssymbole außer $c$ und $d$, daher

$$\Sigma_{\neg G} = \{c, d\}$$

Die einzigen Terme ohne Variablen sind $c$ und $d$. Die Resolutionsmethode ergibt die Klauseln

$$S(c) \vee \neg R(d), R(c), \neg S(c), R(d), \neg R(d)$$

Dies führt zur Resolution




Wir sehen, dass $\neg G$ nicht erfüllbar ist. Daher ist $G$ allgemeingültig.

**Bemerkung 14.10.** In der Aussagenlogik ist die Resolutionsmethode ein entscheidbarer Algorithmus: Nach endlich vielen Schritten wissen wir, ob die gegebene Formel erfüllbar ist. Der Algorithmus der Prädikatenlogik terminiert jedoch nicht immer: Für erfüllbare Formeln bekommen wir nie in endlich vielen Schritten die Antwort. Dies ist kein Zufall, wie der nächste Satz belegt.

**Satz 14.11.** ***Churchscher Satz***
*Die Prädikatenlogik ist nicht entscheidbar: Es gibt keinen Algorithmus, der für jede Formel $F$ in endlich vielen Schritten entscheidet, ob $F$ erfüllbar ist.*
Die Unentscheidbarkeit wird intuitiv klar, wenn wir zeigen, dass es Formeln gibt, die

- erfüllbar sind,
- aber in keiner endlichen Struktur erfüllt werden.

Denn dann benötigen wir eine unendliche Struktur, um die Erfüllbarkeit zu demonstrieren.

**Beispiel 14.12.** Wir arbeiten mit einer zweistelligen Relation $<$. Diese ist transitiv $x < y \wedge y < z \Rightarrow x < z$ und antireflexiv $\neg(x < x)$. Zusätzlich verwenden wir die 1-stellige Funktion $f$, die die Formel $x < f(x)$ erfüllt:

$$F := \forall x : \forall y : \forall z : ((x < y \wedge y < z \Rightarrow x < z) \wedge \neg(x < x) \wedge x < f(x))$$

Diese Formel ist sicher erfüllbar: in der Struktur $\mathcal{N}$ mit dem Träger $\mathbb{N} = \{0, 1, 2, \dots\}$, wobei $<^{\mathcal{N}}$ die übliche Ordnung der natürlichen Zahlen und $f^{\mathcal{N}}$ die Nachfolgerfunktion ist.

Aber $F$ gilt in keiner endlichen Struktur $\mathcal{A}$. Betrachten wir eine Struktur, in der $F$ gilt. Wir wählen ein beliebiges Element $a_0$ des Trägers und setzen $\wedge_{n+1} = f^{\mathcal{A}}(a_n)$. Für jedes $x$ gilt $x < f(x)$, deswegen $a_0 < a_1 < a_2 \ldots$. Dies beweist, dass die Elemente $a_n$ paarweise verschieden sind: aus $a_i = a_j$ folgt $i = j$ denn jedes $x$ erfüllt $\neg(x < x)$ (und die Transitivität ergibt $a_i < a_j$ oder $a_j < a_i$ falls $i \neq j$).

# Literaturverzeichnis


[BS57] Friedrich Ludwig Bauer and Klaus Samelson. Verfahren zur automatischen Verarbeitung von kodierten Daten und Rechenmaschine zur Ausübung des Verfahrens. Patent submission, March 30 1957. Patent DE1094019, published December 1, 1960, granted August 12, 1971.

[BWW54] Arthur W. Burks, Don W. Warren, and Jesse B. Wright. An analysis of a logical machine using parenthesis-free notation. *MTAC*, 8(46):53–57, April 1954.

[Cal11] James Caldwell. Teaching natural deduction as a subversive activity. http://www.cs.uwyo.edu/ jlc/papers_chronological.html, June 2011. Presented at the Third International Congress on Tools for Teaching Logic, 1-4 June, 2011,Salamanca, Spain.

[D'A05] Marcello D'Agostino. Classical natural deduction. In Sergei N. Artëmov, Howard Barringer, Artur S. d'Avila Garcez, Luís C. Lamb, and John Woods, editors, *We Will Show Them! (1)*, pages 429–468. College Publications, 2005.

[Gen34] Gerhard Gentzen. Untersuchungen über das logische Schließen. I. *Mathematische Zeitschrift*, 39(2):176–210, 1934.

[Gen35] Gerhard Gentzen. Untersuchungen über das logische Schließen. II. *Mathematische Zeitschrift*, 39(3):405–431, 1935.

[Ham57] Charles Leonard Hamblin. An addressless coding scheme based on mathematical notation. Technical report, N.S.W. University of Technology, May 1957. (typescript).

[Hil99] David Hilbert. *Grundlagen der Geometrie*. Teubner, 1999. Erstausgabe 1899.

[HR09] Michael Huth and Mark Ryan. *Logic in Computer Science*. Cambridge University Press, 2009.

[Jaś34] Stanisław Jaśkowski. On the rules of suppositions in formal logic. *Studia Logica*, 1:5–32, 1934.

[Kle52] Stephen Cole Kleene. *Introduction to Metamathematics*. Noth Holland, 1952.