Christoph Lüth, Serge Autexier

Einführung in die Formale Logik

Sommersemester 2026

Vorlesungsbegleitende Unterlagen

Universität Bremen

dfki Deutsches Forschungszentrum für Künstliche Intelligenz
German Research Center for Artificial Intelligence

Stand vom 29. April 2026.

Inhaltsverzeichnis

Kapitel 1

Einführung

Basierend auf einem Vortrag “Computer und die Suche nach der Wahrheit” im Haus der Wissenschaft am 11.12.2021.

- Motivation aus der täglichen Arbeit an Methoden um die Korrektheit/Sicherheit von Computer zu garantieren.
- Wie können wir mit Hilfe von Computern die Korrektheit von Computerprogrammen beweisen? Was heißt überhaupt “beweisen”?
- Die Antwort führt uns 2500 Jahre in die Vergangenheit. . .

1.1 Die Anfänge

- Bei den Griechen war Mathematik eher eine Geheimwissenschaft.
- “Wissen ist Macht”, Methoden wie Lösungen von Gleichungen wurden unter Verschluss gehalten (Pythagoreische Bruderschaft, ca 500 BC).
- Euklid (ca. 300 BC) war einer der ersten, die eine “moderne” Herangehensweise an abstrakte (geometrische) Figuren hatten– oder von denen es überliefert wird:
  - Grundlegende Axiome
  - Schlussregeln
  - Definition — Satz — Beweis.

1.2 Die Moderne

- In den folgenden Jahrhunderten wurde graduell das symbolische Rechnen verfeinert (Analysis, Zahlentheorie; Leibniz und Newton in scharfen Disput über die Urheberschaft der Differential- und Integralrechnung, ca. 1680).

- Fermat (ca. 1640) und Kepler (1611) formulierten Aussagen, von denen keiner sagen konnte, ob sie wahr oder falsch waren.
- Damit wurde die Frage immer drängender: was heißt überhaupt *wahr*? Was ist ein *Beweis*?

1.3 Die Moderne

- Gottlob Frege (1848 – 1925) hat als erster versucht, die Mathematik zu "formalisieren", d. h. Beweise so aufzuschreiben, dass man sie mechanisch nachvollziehen kann. Das war die Geburt der formalen (symbolischen) Logik. Frege wandt sich später Verschwörungstheorien zu (allerdings nicht öffentlich).
- Georg Cantor (1845 - 1918) hat die moderne Mengenlehre als Grundstein der Mathematik erfunden, und systematisch über "unendliche Mengen" geredet. Dafür wurde er stark angefeindet. Er wurde später wahnsinnig.
- David Hilbert (1862 - 1943) hat den Anspruch formuliert, die Mathematik widerspruchsfrei und nachvollziehbar von Grund auf aufzubauen. *Hilberts Programm* sollte ein Formalismus mit folgenden Eigenschaften sein, in dem die ganze Mathematik erfasst (und bewiesen) werden konnte:
  - konsistent,
  - vollständig, und
  - entscheidbar.
- Henri Poincaré (1854 - 1912) wandte sich gegen Hilbert (und Cantor).
  - Auch er formulierte eine berühmte, bis vor kurzem unbewiesene Behauptung.
- Aus dem von Hilbert und anderen vertretenen *Formalismus* auf der einen, und dem von Brouwer und später Poincaré auf der anderen Seite vertretenen *Intuitionismus* erwuchs der *Grundlagenstreit* der Mathematik in den 1920er Jahren.

1.4 Russel's Paradox

- Bertrand Russel (1872- 1970) unternahm einen ernsthaften Versuch, Hilberts Programm zu implementieren und die Mathematik von Grund auf zu formalisieren *(„*Principia Mathematica", zusammen mit Alfred Whitehead). Dabei stieß er auf *Russel's Paradox* (1903), welches zeigt, dass die einfache Mengenlehre inkonsistent ist.
- Sei R die Menge aller Mengen, die sich nicht selbst enthalten: Wenn R in R ist, dann ist R nicht in R, also ist R in R, also ...
- Der Barbier rasiert alle Männer, die sich nicht selbst rasieren. Aber wer rasiert den Barbier?
- Das Paradox zeigte auch, dass Frege's Formalisierung ("Grundlagen der Arithmetik") inkonsistent ist — kurz bevor der zweite Band in Druck gehen sollte.

1.5 Gödel

- Kurt Gödel (1906 - 1978) konnte zeigen, dass Systeme einer gewissen Mächtigkeit (konkret: in denen ich das Rechnen mit Zahlen ausdrücken kann)
  - zum einen unvollständig sind – es gibt immer Aussagen, die weder bewiesen noch widerlegt werden können –
  - und zum anderen kann nicht in diesem System gezeigt werden, dass es konsistent ist.
- Dazu hat Gödel das formale Beweisen in das Rechnen mit Zahlen übersetzt.
- Gödel hat der Legende nach bei seiner Einbürgerung in den USA dem Beamten gegenüber bewiesen, dass die Verfassung der USA inkonsistent ist.
- Gödel ist verhungert, weil er am Ende seines Lebens (als seine Frau im Krankenhaus war) so misstrauisch gegenüber Fremden wurde, dass er aus Angst vergiftet zu werden nicht mehr genug aß.

1.6 Turing

- Alan Turing (1912 -1954) hat ein Berechnungsmodell entwickelt, dass das "menschliche Rechnen" formalisiert (Turing-Maschine). Interessanterweise wurden die ersten Computer basierend auf diesem Modell gebaut (John von Neumann). Er konnte zeigen, dass in einem hinreichend starken System für einen beliebigen Satz ϕ nicht mechanisch entscheidbar ist, ob ϕ gilt oder nicht.
- Turing's Maschinen waren nicht die einzigen (oder ersten) Berechnungsmodelle, aber alle diese sind äquivalent mit Turing-Maschinen:
  - Primitiv rekursive Funktionen (wie Gödel sie verwendet) gehen auf Dedekind und Peano zurück.
  - Alonzo Church (1903 – 1995) erfand den Lambda-Kalkül.
  - Moses Ilyich Schönfinkel (1888– 1942) und Haskell B. Curry (1900 – 1982) erfanden kombinatorische Logik.
- Deshalb wurde die Church-Turing-These formuliert:

  Formale Berechenbarkeit $\cong$ intuitive Berechenbarkeit

- Sie ist nicht beweisbar, weil der Begriff "intuitive Berechenbarkeit" *eo ipso* nicht formal definiert werden kann.
- Turing wurde wegen „gross indecency" (Homosexualität) verurteilt, seine Sicherheitszertifzierung wurde ihm entzogen, und er beging (wahrscheinlich) Selbstmord. Er wurde 2016/17 rehabilitiert.
- Das war allerdings nicht das Ende von Hilberts Programm, sondern eher der Anfang. Es bedeutet nämlich, dass wir immer nur die Wahrheit *in Bezug* auf andere, "grundlegendere" Wahrheiten (Axiome) beweisen können. Diese müssen wir dann "glauben" bzw. genauer gesagt uns darauf verständigen. Mathematik bleibt insbesondere ein soziales Konstrukt. Es löst damit auch in klassischer Dialektik den Grundlagenstreit– beide hatten recht.

1.7 Computerbeweise in der Praxis

- Chip-Verifikation für Infineon und Intel
- Softwareverifikation: L4, Luft-/Raumfahrt
- Thomas Hales (1958) bewies 1998 die Kepler-Vermutung mit Hilfe von Software, welche die vielen Fälle ... Die Gutachter konnten sich nicht einigen, ob der Beweis richtig war, konnten aber auch keinen Fehler finden („99% sicher“). 2005 wurde der Beweis trotzdem veröffentlicht. Thomas Hales begann daraufhin das Flyspeck-Projekt, in dem er den gesamten Beweis in einem Theorembeweiser formalisieren, d.h. in elementarer Form abbilden wollte (s. Formal Logik oben), so dass er von einem Computer geprüft werden kann. Dieses Projekt wurde 2015 abgeschlossen, und ist das erste, welches für forschungsrelevante Mathematik in einem Theorembeweiser benutzte.

Kapitel 2

Aussagenlogik

Vorlesung vom 08.04.2026: Aussagenlogik I

2.1 Logische Systeme

Definition 2.1 *Ein* logisches System *besteht aus:*

*(1) einer formalen Sprache (*Syntax*) mit einem Alphabet von Zeichen, und Regeln, die daraus erlaubte Worte (*Formeln*) bilden;*

(2) einer Semantik*, welche Formeln eine Bedeutung zuordnet;*

(3) Schlussregeln, welche ein oder mehrere Formeln in eine andere Formel umformen.

Am Anfang werden alle unsere Formeln entweder wahr (1) oder falsch (0) sein; später werden wir das etwas aufweichen.

Beispiele für logische Systeme: Programmiersprachen?

2.2 Aussagen

Aussagenlogik beruht auf atomaren Aussagen, die wir mit Konnektiven zu komplexeren Formeln kombinieren.

Definition 2.2 *Eine (logische)* Aussage *ist etwas, was entweder wahr oder falsch sein kann. Eine logische Aussage ist* atomar*, wenn sie nicht in weitere Aussagen zerlegt werden kann.*

Welches der folgenden sind Aussagen? Welche sind atomar?

(1) Säugetiere sind Warmblüter.

*Atomare Aussage (*1*).*

(2) Die Sonne umkreist die Erde.
Atomare Aussage (0)

(3) Jede Sekunde verbrennt die Sonne 4 Millionen Tonnen ihrer Masse zu Energie.
Atomare Aussage (1, anscheinend)

(4) Fiete ist größer als Kalle.
Atomare Aussage

(5) Kalle ist kleiner als Fiete.
Atomare Aussage, (fast) äquivalent zu der davor.

(6) Ist Fiete schlauer als Kalle?
Keine Aussage— keine Wahrheitswert.

(7) Kalle hat doch gar kein Auto.
Atomare Aussage.

(8) Kalle hat doch gar kein Auto?
Aussage wird zur Frage — keine Aussage.

(9)
$$\vec{v}(t) = \frac{\delta \vec{s}(t)}{\delta t}$$
Atomare Aussage, formal hingeschrieben.

(10) Geschwindigkeit ist die Ableitung des Ortsvektors nach der Zeit.
Atomare Aussage. Definition.

(11) Bitte nicht bewegen.
Keine Aussage

(12) Kalle kommt mit und Fiete kommt mit.
Komplexe Aussage.

(13) Kalle und Fiete kommen mit.
Komplexe Aussage.

(14) Fiete kommt mit, wenn Kalle auch kommt.
Komplexe Aussage.

(15) Kommen Kalle oder Fiete mit?
Keine Aussage

(16) Sie kommen nicht mit.
Wer sind sie? Aussage mit einer Variablen.

(17) Kalle und Fiete heiraten.
Aussage. Mehrdeutig — wenn sei einander *heiraten atomar, sonst komplex.*

(18) Der König ist tot, lang lebe der König!
Der erste Teil ist eine atomare Aussage, der zweite Teil ist keine Aussage.

(19) Solange die Sonne scheint, regnet es nicht.
Komplexe Aussage.

(20) Wenn der Kuchen spricht schweigen die Krümel.
Aussage, kann aber auch eine Aufforderung sein.

Folgendes sind also keine Aussagen:

- Fragen,
- Aufforderungen,
- Interjektionen (spontante Ausrufe).

Komplexe Aussagen werden aus atomaren Aussagen durch *Konnektive* verbunden. In den Aussagen haben wir verschiedene Konnektive:

- "und" (Konjunktion),
- "oder" (Disjunktion),
- "wenn. . . dann" (Implikation),
- "wenn", "solange" (zeitliche Abfolge).

Umgangssprachliche Konnektive sind mehrdeutig. Hier noch ein paar Beispiele:

(1) Heinz fuhr weiter und fuhr einen Fußgänger an.

(2) Heinz fuhr einen Fußgänger an und fuhr weiter.

(3) Wenn ich das Fenster öffne haben wir Frischluft.

(4) Wenn ich das Fenster öffne kreist die Erde um die Sonne.

(5) Wenn die Sonne um die Erde kreist dann haben wir Frischluft.

(6) Hannes arbeitet, oder er ist in der Kneipe.

(7) Euclid war Grieche oder Mathematiker.

- In (1), (2) haben wir eine implizite zeitliche Ordnung.
- (3) ist ein kausaler Zusammenhang. (4) und (5) kommen uns widersinnig vor, (5) sogar falsch; (5) könnte eine rhetorische Figur sein ("Wenn die Sonne um die Erde kreist dann bin ich Kaiser von China!").
- In (6) und (7) wird "oder" exklusiv gelesen; (6) erscheint richtig (es sei denn Hannes arbeitet in der Gastronomie), (7) falsch.

Zusammenfassend hat natürliche Sprache mehrere Probleme: sie ist *mehrdeutig* und enthälft oft versteckte (implizite) *Annahmen*.

2.3 Aussagenlogik

Aussagenlogik kombiniert atomare Aussage mit Konnektiven, die eine feste Bedeutung haben. Dabei betrachten wir keine zeitlichen Ablauf oder sonstige Modalitäten (das wäre dann Temporallogik oder Modallogik).

2.3.1 Sprache

Wir definieren erst unsere formale Sprache der Aussagenlogik:

Definition 2.3 *Gegeben eine abzählbar unendliche Menge P von atomaren Aussagen, dann ist Prop die kleinste Menge so dass mit $\phi, \psi \in Prop$:*

$$\begin{aligned} P &\subseteq Prop \\ \bot &\in Prop \\ \neg\phi &\in Prop \\ \phi \wedge \psi &\in Prop \\ \phi \vee \psi &\in Prop \\ \phi \longrightarrow \psi &\in Prop \end{aligned}$$

Einfacher geschrieben:

$$p ::= q \in P \mid \bot \mid \neg p \mid p_1 \wedge p_2 \mid p_1 \vee p_2 \mid p_1 \longrightarrow p_2$$

Syntaktisch gelten folgende *Präzedenzen:* $\neg$ vor $\wedge$ vor $\vee$ vor $\longrightarrow$

Bei gleichen Konnektoren wird von rechts nach links geklammert, i.e., $A \longrightarrow B \longrightarrow C$ klammert wie folgt: $A \longrightarrow (B \longrightarrow C)$

Übung 2.1 *Wie sehen folgende Ausdrücke voll geklammert aus?*

- $A \wedge B \vee C \longrightarrow C \vee X$
- $\neg(B \vee C) \wedge X \vee Y$
- $A \longrightarrow B \vee \neg C \wedge D$

2.3.2 Bedeutung

Was ist die *Bedeutung* einer aussagenlogischen Formal $p \in Prop$? Sie ist entweder wahr oder falsch. Wenn wir wahr mit 1 kodieren und falsch mit 0, dann ist $\mathbb{B} = \{0,1\}$ die Menge aller boolschen Werte. (Offensichtlich ist $\mathbb{B} \subset \mathbb{N}$.) Die Semantik ist dann eine Abbildung $sem : Prop \to \mathbb{B}$ (die wir allerdings mit "semantischen Klammern" $[\![\cdot]\!]$ schreiben). Das ist nicht so ganz perfekt — was ist mit atomaren Aussagen? Deren Bedeutung können wir *festlegen*; sie sind quasi Parameter der Bedeutungsfunktion — wir wissen zwar, dass die Erde um die Sonne kreist, aber bspw. nicht ob Kalle oder Fieter größer ist. (Wir wissen ja nicht mal, wer die beiden sind.)

$\wedge$	0	1
0	0	0
1	0	1

$\vee$	0	1
0	0	1
1	1	1

$\neg$	
0	1
1	0

$\longrightarrow$	0	1
0	1	1
1	0	1

Abbildung 2.1: Wahrheitstabellen für die vier Konnektive

Wir definieren die Funktion *induktiv* über der Struktur von *Prop*. (Das ist wie in Haskell.) Dafür benötigen wir alle vier Konnektive eine *Bedeutung*.

Für $\wedge$ heißt das beispielsweise, dass wir die Bedeutung von $p \wedge q$ aus der Kombination der möglichen Bedeutungen von p und q definieren müssen. Für jeden der beiden ist dies entweder $\top$ oder $\bot$, was uns zu vier Möglichkeiten führt.

Die mögliche Bedeutung läßt sich schnell an der Aussage "Serge und Christoph sind in der Vorlesung." verdeutlichen:

Serge	Christoph	Serge und Christoph
da	da	da
nicht da	da	nicht da
da	nicht da	nicht da
nicht da	nicht da	nicht da

Für die Disjunktion ergibt sich schnell eine ähnliche Tabelle:

Serge	Christoph	Serge oder Christoph
da	da	da
nicht da	da	da
da	nicht da	da
nicht da	nicht da	nicht da

Die Negation überlassen wir dem geneigten Leser, aber was mit der Implikation? "Wenn Serge in der Vorlesung ist, ist Christoph auch da." Eine Wahrheitstabelle:

Serge	Christoph	Wenn Serge da ist, dann ist Christoph da
da	da	wahr
nicht da	da	?
da	nicht da	falsch
nicht da	nicht da	wahr

In allen Fällen ist der Wahrheitsgehalt evident, bis auf die zweite Zeile. Ist die Aussage jetzt wahr oder nicht? Wir *definieren* sie in dem Fall als wahr — das könnte man auch anders machen, aber so ist die resultierende Theorie eleganter (Ableitungsregeln, Äquivalenzen, usw.).

Diese Wahrheitstabellen stellt man traditionellerweise etwas kompakter dar (vgl. Abb. 2.1).

Der Wahrheitswert einer beliebigen Formel $\phi \in \mathit{Prop}$ ist abhängig von den Wahrheitswerten der atomaren Aussagen (*Atome*). Dazu müssen wir nicht alle Atome mit einem Wert belegen, sondern nur die in ϕ enthaltenen. Wir definieren:

Definition 2.4 *Sei $\phi, \psi \in Prop$ eine Aussage, dann ist die Menge der in ϕ, ψ enthaltenen Atome induktiv definiert als*

$$\begin{aligned}
\text{atoms}(q) &= \{q\} \quad (\textit{für } q \in P)\\
\text{atoms}(\bot) &= \emptyset\\
\text{atoms}(\neg\phi) &= \text{atoms}(\phi)\\
\text{atoms}(\phi \wedge \psi) &= \text{atoms}(\phi) \cup \text{atoms}(\psi)\\
\text{atoms}(\phi \vee \psi) &= \text{atoms}(\phi) \cup \text{atoms}(\psi)\\
\text{atoms}(\phi \longrightarrow \psi) &= \text{atoms}(\phi) \cup \text{atoms}(\psi)
\end{aligned}$$

Dabei ist atoms : $Prop \to \mathbb{P}(P)$ *wobei* $\mathbb{P}(P)$ *für die Potenzmenge von P steht, d. h.* atoms(ϕ) *liefert eine* Menge *von Atomen.*

Für eine Aussage $\phi \in Prop$ ist eine *Valuation* oder *Belegung* (der Variablen) eine Funktion $v : \text{atoms}(\phi) \to \mathbb{B}$.

Definition 2.5 *Für eine Formel $\phi \in Prop$ und eine Belegung $v : \text{atoms}(\phi) \to \mathbb{B}$ ist der Wert $[\![\phi]\!]_v$ von ϕ unter v induktiv definiert als*

$$\begin{aligned}
[\![p]\!]_v &= v(p) \quad (\textit{für } p \in P)\\
[\![\bot]\!]_v &= 0\\
[\![\neg\phi]\!]_v &= 1 - [\![\phi]\!]_v\\
[\![\phi \wedge \psi]\!]_v &= \min([\![\phi]\!]_v, [\![\psi]\!]_v)\\
[\![\phi \vee \psi]\!]_v &= \max([\![\phi]\!]_v, [\![\psi]\!]_v)\\
[\![\phi \longrightarrow \psi]\!]_v &= \begin{cases} 0 & \textit{wenn } [\![\phi]\!]_v = 1 \textit{ und } [\![\psi]\!]_v = 0\\ 1 & \textit{sonst} \end{cases}
\end{aligned}$$

Für eine gegebene Belegung v ist die Semantik eindeutig definiert, d. h. wenn v und w zwei Belegungen für ϕ sind und $v(p) = w(p)$ für alle $p \in \text{atoms}(\phi)$, dann ist $[\![\phi]\!]_v = [\![\phi]\!]_w$. Der Beweis erfolgt durch Induktion über die Struktur von ϕ.

Vorlesung vom 14.04.26: Aussagenlogik II

Übung 2.2 (Aufwärmübung) *Formalisiert folgende Aussagen. Welches sind die Atome, und wie sieht die Formel $\phi \in Prop$ aus?*

"Wenn ich alle Übungsblätter abgebe und die Prüfung bestehe bekomme ich den Schein. Wenn ich keinen Schein habe, habe ich also entweder nicht alle Übungsblätter abgegeben oder die Prüfung nicht bestanden."

"Wenn es regnet, werde ich naß. Wenn ich aber einen Schirm habe, dann werde ich nicht naß, wenn es regnet."

"Wenn gegessen habe, bin ich satt, aber ich bin hungrig, also habe ich nicht gegessen."

Sind diese Aussagen wahr? Was heißt das?

2.3.3 Tautologien

Damit können wir jetzt darüber reden, ob eine Formel (immer) "wahr" ist:

Definition 2.6 *Gegeben eine Formel $\phi \in Prop$, dann ist ϕ eine* Tautologie*, wenn $[\![\phi]\!]_v = 1$ für alle Belegungen v. Wir schreiben dafür auch $\models \phi$, und sagen ϕ ist* semantisch gültig.

Ist eine Formel ϕ für *keine* Belegung erfüllt, ist sie *unerfüllbar* (oder ein Widerspruch). Wie man leicht sieht, ist ϕ unerfüllbar gdw. $\neg\phi$ semantisch gültig ist (weil $[\![\neg\phi]\!]_v = 1$ gdw. $[\![\phi]\!]_v = 0$). Darüber hinaus gibt es noch Formeln, die für einige, aber nicht alle Belegungen, erfüllbar sind.

Wie finden wir heraus, ob ϕ eine Tautologie ist? Das wird uns den Rest der Veranstaltung beschäftigen... aber eine erste, einfache Möglichkeit sind *Wahrheitstabellen*: wir zählen einfach alle möglichen Belegungen auf.

Beispiele:

- $A \wedge B \longrightarrow A$

A	B	A	$\wedge$	B	$\longrightarrow$	A
0	0	0	0	0	1	0
0	1	0	0	1	1	0
1	0	1	0	0	1	1
0	1	1	1	1	1	1

- $\bot \longrightarrow A$ (*Ex falso quodlibet*)

A	0	$\longrightarrow$	$\neg A$
0	0	1	1
1	0	1	0

- Gegenbeispiel: $A \vee B \longrightarrow A$

A	B	A	$\vee$	B	$\longrightarrow$	A
0	0	0	0	0	1	0
0	1	0	1	1	0	0
1	0	1	1	0	1	1
0	1	1	1	1	1	1

- Längeres Beispiel: $(A \longrightarrow (B \longrightarrow C) \longrightarrow (A \wedge B \longrightarrow C)$ (und umgekehrt)

A	B	C	$(A$	$\longrightarrow$	$(B$	$\longrightarrow$	$C))$	$\longrightarrow$	$(A$	$\wedge$	B	$\longrightarrow$	$C)$
0	0	0	0	1	0	1	0	1	0	0	0	1	0
0	0	1	0	1	0	1	1	1	0	0	0	1	1
0	1	0	0	1	1	0	0	1	0	0	1	1	0
0	1	1	0	1	1	1	1	1	0	0	1	1	1
1	0	0	1	1	0	1	0	1	1	0	0	1	0
1	0	1	1	1	0	1	1	1	1	0	0	1	1
1	1	0	1	0	1	0	0	1	1	1	1	0	0
1	1	1	1	1	1	1	1	1	1	1	1	1	1

Übung 2.3 *Weitere Beispiele als Aufgabe:*

(1) $A \vee \neg A$ *(Tertium non datur, Satz des ausgeschlossenen Dritten)*

(2) $(\neg A \vee B) \longrightarrow (A \longrightarrow B)$ *und umgekehrt.*

Diese Art von Beweisen skaliert ganz klar nicht: die Anzahl der möglichen Belegungen für eine Formel $\phi \in Prop$ ist 2^n mit $n = |\text{atoms}(\phi)|$, d.h. die Anzahl der Atome in ϕ. Exponentielles Wachstum fällt immer unter "schlechte Nachrichten", was irgendeine Art von Automatisierung im Größeren betrifft. (Wir werden später noch weiter über Aufwand reden.)

2.3.4 Semantische Folgerung

Wenn Logik tatsächlich die "Suche nach der Wahrheit" ist sollte man annehmen, dass Tautologien unser starkes Interesse wecken müssten— denn sie sind ja immer wahr (unter jeder Belegung). Dem ist aber tatsächlich gar nicht so;eben weil Tautologien immer wahr sind, haben sie wenig Aussagekraft.

Was uns wirklich interessiert in der Logik ist die *Suche*, nicht so sehr die *Wahrheit*. Technischer ausgedrückt, wir sind stärker daran interessiert wie man aus bekannten Tatsachen neue Fakten herleiten kann. In der Sprache der Aussagenlogik ausgedrückt: wir haben eine Menge $\Gamma = \{\phi_1, \ldots, \phi_n\}$ von Aussagen, und wollen wissen ob eine neue Aussage ψ daraus folgt. Das erlaubt es uns, von einer (kleinen) Menge von Annahmen auf immer größere Mengen von Fakten zu schließen.

Folgende Definition formalisiert den Folgerungsbegriff auf semantischer Ebene:

Definition 2.7 (Semantische Folgerung) *Sei* $\Gamma = \{\phi_1, \ldots, \phi_n\}$ *eine Menge von Aussagen, und* ψ *eine Aussage, dann ist* $\Gamma \models \psi$ *gdw. folgendes gilt: für alle Valuationen v mit* $[\![\phi_1]\!]_v = 1, \ldots, [\![\phi_n]\!]_v = 1$ *ist auch* $[\![\psi]\!]_v = 1$.

Das folgende Theorem zeigt, dass semantische Folgerung und syntaktische Implikation übereinstimmen. Das ist eine erste Aussage über den Zusammenhang von Syntax und Semantik:

Theorem 2.1 (Deduktionstheorem)

(1) $\phi \models \psi$ *gdw.* $\models \phi \longrightarrow \psi$

(2) $\models \phi \wedge \psi$ *gdw.* $\models \phi$ *und* $\models \psi$

(3) $\Gamma \models \psi$ *gdw.* $\models \phi_1 \wedge \ldots \wedge \phi_n \longrightarrow \psi$

Beweis. (1) und (2) folgen direkt aus der Definition von $\models$. Wir zeigen (1):

$$\begin{aligned} \phi \models \psi &\Longleftrightarrow \text{für alle } v\text{: wenn } [\![\phi]\!]_v = 1 \text{ dann } [\![\psi]\!]_v = 1 \\ &\Longleftrightarrow \text{für alle v: } [\![\phi \longrightarrow \psi]\!]_v \\ &\Longleftrightarrow \models \phi \longrightarrow \psi \end{aligned}$$

Aus (1) und (2) folgt direkt (3). □

2.3.5 Äquivalenz

Äquivalenz ist Folgerung in beiden Richtungen. Wie bei der Folgerung definieren wir die Äquivalenz auf semantischer und syntaktischer Ebene, und zeigen dann dass beides dasselbe (also äquivalent, ähem) ist.

Definition 2.8 (Semantische Äquivalenz) *Zwei Aussagen $\phi, \psi \in Prop$ heißen* semantisch äquivalent $\phi \approx \psi$, *wenn $[\![\phi]\!]_v = [\![\psi]\!]_v$ für alle Valuationen v ist.*

Auf syntaktischer Ebene führen wir den Äquivalenzoperator $\longleftrightarrow$ ein, auch als Bi-Implikation bezeichnet.

Definition 2.9 (Äquivalenz) *Die Äquivalenz ist definiert als*

$$A \longleftrightarrow B \text{ gdw } (A \longrightarrow B) \wedge (B \longrightarrow A)$$

Mit der Definition ergibt sich folgende Wahrheitstabelle:

$\longleftrightarrow$	0	1
0	1	0
1	0	1

Syntaktisch bindet $\longleftrightarrow$ am schwächsten von allen Konnektiven.

Wir wollen jetzt zeigen, dass die syntaktische und semantische Äquivalenz der Gleichheit entsprichen.

Lemma 2.2 *Syntaktische Äquivalenz ist semantische Gleichheit:*

$$\models \phi \longleftrightarrow \psi \text{ gdw. } \phi \approx \psi. \tag{2.1}$$

Beweis. Für Gleichung (2.1) müssen wir für alle Belegungen v zeigen, dass

$$[\![\phi \longleftrightarrow \psi]\!]_v = 1 \text{ gdw. } [\![\phi]\!]_v = [\![\psi]\!]_v \tag{2.2}$$

Zum Beweis nutzen wir folgende Gleichung. Für alle Belegungen v gilt nämlich auch:

$$[\![\phi \longrightarrow \psi]\!]_v = 1 \text{ gdw. } [\![\phi]\!]_v \leq [\![\psi]\!]_v \tag{2.3}$$

Diese Gleichung folgt direkt aus der Definition von $[\![\phi \longrightarrow \psi]\!]_v$; man kann sie sogar als Definition dafür nehmen. Damit folgt jetzt: wenn $[\![\phi \longleftrightarrow \psi]\!]_v = 1$, dann $[\![\phi \longrightarrow \psi]\!]_v = 1$ und $[\![\phi \longrightarrow \psi]\!]_v = 1$ (nach Definition von $\longleftrightarrow$), mit (2.3) dann $[\![\phi]\!]_v \leq [\![\psi]\!]_v$ und $[\![\psi]\!]_v \leq [\![\phi]\!]_v$, und damit $[\![\phi]\!]_v = [\![\psi]\!]_v$ wie gewünscht. □

Lemma 2.2 erweitert das Deduktionstheorem 2.1 um das neue Konnektiv $\longleftrightarrow$. Es zeigt, dass zwei äquivalente Formeln die gleiche Semantik haben müssen.

Um mit der Äquivalenz $\approx$ wie mit Gleichheit in der Algebra rechnen zu können, müssen wir Transitivität und Symmetrie zeigen, mit anderen Worten, dass die Äquivalenz eine *Äquivalenzrelation* ist:

$$\phi \approx \phi \tag{2.4}$$

$$\text{Wenn } \phi \approx \psi \text{ dann } \psi \approx \phi \tag{2.5}$$

$$\text{Wenn } \phi \approx \psi \text{ und } \psi \approx \sigma \text{ dann } \psi \approx \sigma \tag{2.6}$$

$$\tag{2.7}$$

Beweis. Aus der Definition von $\approx$ folgt dass $A \approx B$ gdw. $[\![\phi]\!]_v = [\![\psi]\!]_v$ für alle Belegungen v. Da die Gleichheit $=$ transitiv, reflexiv und symmetrisch ist, folgt dies auch für $\approx$. □

Vorlesung vom 15.04.2026: Aussagenlogik III

Übung 2.4 (Aufwärmübung) *Wir haben über das exklusive Oder schon gesprochen. Welche Wahrheitstabelle kann der entsprechende Operator $\dot\vee$ haben? Und wie können wir $\phi \dot\vee \psi$ mit den anderen Operatoren ausdrücken?*

In der Algebra beweisen wir oft durch wiederholtes Umformen von Ausdrücken, wie man es in der Schule lernt. Hier ein typischer Beweis:

$$\begin{aligned}
& x^2 - 8\cdot x + 12 = 0 \\
\iff\ & x^2 - 2\cdot 4\cdot x + 16 - 4 = 0 \\
\iff\ & (x-4)^2 - 4 = 0 \\
\iff\ & (x-4)^2 = 4 \\
\iff\ & (x-4) = 2 \vee (x-4) = -2 \\
\iff\ & x = 6 \vee x = 2
\end{aligned}$$

Warum funktioniert dieser Beweis?

- Der wesentliche Schluss ist, dass $x = 6$ oder $x = 2$ die Anfangsgleichung erfüllen. Damit das gilt, muss die Gleichheit $=$ *transitiv* sein.
- Wir rechnen vorwärts (von der Anfangsgleichung ausgehend), aber sind am Schluss rückwärts interessiert (wenn x die berechneten Werte hat, dann ist es eine Lösung); dazu muss die Relation $=$ zwingend *symmetrisch* sein.
- Wir ersetzen *innerhalb* der Gleichungen, bspw. in dem $12 = 16 - 4$ nutzen, die Assoziativität der Addition, die binomische Formel usw.. Dazu muss die Gleichheit $=$ *substitutiv* sein, d.h. wenn $s = t$ dann gilt auch $s' = t'$ wenn ich in s und t gleiches durch gleiches ersetze.

Unsere logische Äquivalenz $\approx$ ist eine Äquivalenzrelation, also transitiv, symmetrisch und reflexiv. Es fehlt noch die *Substitutivität*, d. h. wir müssen in einer aussagenlogischen Formel gleiches durch gleiches ersetzen können. Aber was heißt überhaupt ersetzen?

2.3.6 Ersetzung

Wir können atomare Aussagen als Variablen auffassen, die wir durch andere *Aussagen* ersetzen. Wir schreiben die Substitution als $\phi[\psi/p]$ für ein Atom p, gelesen als "in ϕ ersetzen wir ψ für p". Syntaktisch bindet sie schwächer als alle Operatoren.

Definition 2.10 (Substitution) *Für $\phi \in Prop$, $p,q \in P$ und $\psi \in Prop$ definieren wir die Ersetzung von q in ϕ durch ψ, geschrieben $\phi[\psi/q]$, rekursiv über der Struktur von ϕ wie folgt:*

$$\begin{aligned}
p[\psi/q] &= \begin{cases} \psi & p = q \quad (p \in P) \\ p & p \neq q \quad (p \in P) \end{cases} \\
\bot[\psi/q] &= \bot \\
(\neg\phi)[\psi/q] &= \neg(\phi[\psi/q]) \\
(\phi_1 \wedge \phi_2)[\psi/q] &= (\phi_1[\psi/q]) \wedge (\phi_2[\psi/q]) \\
(\phi_1 \vee \phi_2)[\psi/q] &= (\phi_1[\psi/q]) \vee (\phi_2[\psi/q]) \\
(\phi_1 \longrightarrow \phi_2)[\psi/q] &= (\phi_1[\psi/q]) \longrightarrow (\phi_2[\psi/q]) \\
(\phi_1 \longleftrightarrow \phi_2)[\psi/q] &= (\phi_1[\psi/q]) \longleftrightarrow (\phi_2[\psi/q])
\end{aligned}$$

Warum fordern wir eigentlich nicht, dass $p \in \text{atoms}(\phi)$? Es stellt sich heraus, dass das gar nicht notwendig ist. Es gilt:

$$p \notin \text{atoms}(\phi) \Longleftrightarrow \phi[\psi/p] = \phi \text{ für } \psi \in Prop, \psi \neq p$$

Wenn p also nicht in ϕ vorkommt, dann ist die Substitition von p in ϕ die Identität. Der Beweis ist eine Induktion über der Struktur von ϕ.

So wie die Gleichheit auf $\mathbb{B}$ das semantische Gegenstück für die Äquivalenz $\longleftrightarrow$ ist, können wir auch für die Substitution ein semantisches Gegenstück angeben. Die Semantik einer Formel $\phi[p/\psi]$ in der das Atom p durch eine Formel *psi* ersetzt wird, ist die Semantik der Formel ϕ unter einer Belegung v', in der p auf die Semantik von ψ unter der Belegung v belegt wird.

Für eine partielle Funktion $f : A \rightharpoonup B$ und $a \in A$, $b \in B$ schreiben wir $f[a \mapsto b]$ für die Funktion, welche a auf b und alles andere wie f abbildet:

$$(f[a_1 \mapsto b])(a_0) = \begin{cases} b & a_1 = a_0 \\ f(a_0) & a_1 \neq a_0 \end{cases} \tag{2.8}$$

Bevor wir unsere wesentlichen Ergebnisse formulieren, üben wir diese Konzepte anzuwenden:

Übung 2.5 *Gegeben die Formel* $\rho = (A \longrightarrow \neg B \wedge C) \vee (C \vee \neg A)$, *die Belegung* $v = \langle A \mapsto 1, B \mapsto 0, C \mapsto 1 \rangle$ *und* $\sigma = (A \longrightarrow B \longleftrightarrow \neg B \wedge C)$.

(i) *Berechne* $\phi = \rho[\sigma/A]$.

(ii) *Berechne* $t_1 = [\![\phi]\!]_v$.

(iii) *Berechne* $s = [\![\sigma]\!]_v$.

(iv) *Berechne* $w = v[A \mapsto s]$.

(v) *Berechne* $t_2 = [\![\rho]\!]_w$.

Jetzt können wir unser Lemma formulieren:

Lemma 2.3 (Substitutionslemma) *Für alle Formeln* $\phi, \psi \in Prop$ *und Belegungen* v *gilt:*

$$[\![\phi[\psi/p]]\!]_v = [\![\phi]\!]_{v[p \mapsto [\![\psi]\!]_v]}$$

Beweis. Durch Induktion über der Struktur von ϕ.

Die Induktionsbasis sind die zwei Fälle $\phi = q$ mit $q \in P$ und $\phi = \bot$. Für $\phi = q$ gibt es zwei Fälle, nämlich $p = q$ oder $p \neq q$. Für den ersten Fall haben wir

$$[\![p[\psi/p]]\!]_v = [\![\psi]\!]_v = v[p \mapsto [\![\psi]\!]_v](p) = [\![p]\!]_{v[p \mapsto [\![\psi]\!]_v]},$$

und für den zweiten Fall

$$[\![q[\psi/p]]\!]_v = [\![q]\!]_v = v(q) = v[p \mapsto [\![\psi]\!]_v](q) = [\![q]\!]_{v[p \mapsto [\![\psi]\!]_v]}.$$

Für $\phi = \bot$ gilt einfacher $[\![\bot]\!]_w = 0$ für alle Belegungen w, und $[\![\bot[\psi/q]]\!]_v = [\![\bot]\!]_v$.

Wir zeigen den Induktionsschritt für die Negation, die anderen Fälle sind analog:

$$[\![\neg\phi[\psi/p]]\!]_v = 1 - [\![\phi[\psi/p]]\!]_v = 1 - [\![\phi]\!]_{v[p \mapsto [\![\psi]\!]_v]} = [\![\neg\phi]\!]_{v[p \mapsto [\![\psi]\!]_v]}$$

Der zweite Schritt ist hier die Induktionsvoraussetzung. □

Damit können wir folgendes Theorem zeigen:

Theorem 2.4 (Substitutionstheorem) *Wenn* $\models \phi_1 \longleftrightarrow \phi_2$, *dann* $\models \psi[\phi_1/p] \longleftrightarrow \psi[\phi_2/p]$.

Beweis. Nach Definition gilt $\psi[\phi_1/p] \approx \psi[\phi_2/p]$ gdw. $[\![\psi[\phi_1/p]]\!]_v = [\![\psi[\phi_2/p]]\!]_v$, und genauso $\phi_1 \approx \phi_2$ gdw. $[\![\phi_1]\!]_v = [\![\phi_2]\!]_v$ (jeweils für beliebige v). Jetzt können wir rechnen:

$$[\![\psi[\phi_1/p]]\!]_v = [\![\psi]\!]_{v[p\mapsto[\![\phi_1]\!]_v]} = [\![\psi]\!]_{v[p\mapsto[\![\phi_2]\!]_v]} = [\![\psi[\phi_2/p]]\!]_v$$

Der erste und letzte Schritt folgen aus dem Substitutionslemma. □

2.3.7 Boolsche Algebren

Jetzt können wir rechnen. Dazu brauchen wir aber erstmal eine Handvoll von Äquivalenzen, von denen wir ausgehen können. Ein Beispiel in der Algebra sind die Gruppenaxiome, aus denen sich dann eine reichhaltige Theorie ergibt, und die es uns erlauben, Gleichungen wie oben beschrieben umzuformen; leider sind Aussagen keine Gruppen, sondern sogenannte Boolsche Algebren.

Theorem 2.5 *Es gelten folgende Äquivalenzen für* $\wedge, \vee$ *und* $\neg$*:*

$(\phi \wedge \psi) \wedge \sigma \approx \phi \wedge (\psi \wedge \sigma)$	$(\phi \vee \psi) \vee \sigma \approx \phi \vee (\psi \vee \sigma)$	*(Assoziativität)*
$\phi \wedge \psi \approx \psi \wedge \phi$	$\phi \vee \psi \approx \psi \vee \phi$	*(Kommutativität)*
$\phi \wedge (\psi \vee \sigma) \approx (\phi \wedge \psi) \vee (\phi \wedge \sigma)$	$\phi \vee (\psi \wedge \sigma) \approx (\phi \vee \psi) \wedge (\phi \vee \sigma)$	*(Distributivität)*
$\neg(\phi \wedge \psi) \approx \neg\phi \vee \neg\psi$	$\neg(\phi \vee \psi) \approx \neg\phi \wedge \neg\psi$	*(De Morgan)*
$\phi \wedge \phi \approx \phi$	$\phi \vee \phi \approx \phi$	*(Idempotenz)*
	$\neg\neg\phi \approx \phi$	*(Doppelnegation)*
$\phi \wedge \bot \approx \bot$	$\phi \vee \bot \approx \phi$	*(Falsum)*
$(\neg\phi \vee \phi) \approx \top$	$(\neg\phi \wedge \phi) \approx \bot$	*(Inverse)*

Die meisten dieser Eigenschaften folgen aus entsprechenden Eigenschaften der semantischen Funktion. So gilt $(\phi \wedge \psi) \wedge \sigma \approx \phi \wedge (\psi \wedge \sigma)$ weil $\min(x, \min(y,z)) = \min(\min(x,y),z)$ oder $\neg\neg\phi \approx \phi$, weil $1-(1-x))=x$. Etwas schwieriger sind die Distributivitätsgesetze, hier muss man zeigen dass $\min(x, \max(y,z)) = \max(\min(x,y), \min(x,z))$ (durch eine Unterscheidung aller sechs Fälle). Die letzten beiden folgen aus $\min(x,0)=0$ und $\max(x,0)=x$, für $x \geq 0$.

Wie man sieht, fehlt hier noch ein neutrales Element für $\wedge$. Was kann das sein, und wie definieren wir es?

Definition 2.11 (Verum) *Wir definieren True ("Verum") als*

$$\top \longleftrightarrow \neg\bot$$

Damit folgt $[\![\top]\!]_v = 1$, und es gilt

$$\phi \wedge \top \approx \phi \qquad\qquad \phi \vee \top = \top$$

$$\models \phi \text{ gdw } \phi \approx \top$$

Die ersten beiden Äquivalenzen folgen aus $\min(x,1) = x$ und $\max(x,1) = 1$ für $x \leq 1$. Die dritte Gleichung folgt aus $[\![\phi]\!]_\nu = 1$ wenn $\models \phi$.

Weitere Gleichungen ([2, Theorem 1.3.4]) zeigen, wie wir Operatoren aus anderen definieren können:

$$(\phi \longleftrightarrow \psi) \approx (\phi \longrightarrow \psi) \wedge (\psi \longrightarrow \phi) \tag{2.9}$$
$$(\phi \longrightarrow \psi) \approx (\neg\phi \vee \psi) \tag{2.10}$$
$$\phi \vee \psi \approx \neg\phi \longrightarrow \psi \tag{2.11}$$
$$\phi \wedge \psi \approx \neg(\neg\phi \vee \neg\psi) \tag{2.12}$$
$$\phi \vee \psi \approx \neg(\neg\phi \wedge \neg\psi) \tag{2.13}$$
$$\neg\phi \approx (\phi \longrightarrow \bot) \tag{2.14}$$
$$\bot \approx (\phi \wedge \neg\phi) \tag{2.15}$$

Beweise:

- (2.9): nach Definition von $\longleftrightarrow$.
- (2.10): nach Wahrheitstabellen beider Seiten.
- (2.11): $\phi \vee \psi \approx \neg(\neg\phi) \vee \psi \approx \neg\phi \longrightarrow \psi$
- (2.12): $\phi \wedge \psi \approx \neg(\neg(\phi \wedge \psi)) \approx \neg(\neg\phi \vee \neg\psi)$
- (2.13): $\phi \vee \psi \approx \neg(\neg(\phi \vee \psi)) \approx \neg(\neg\phi \wedge \neg\psi)$
- (2.14): $\neg\phi \approx \neg(\phi \wedge \top) \approx \neg\phi \vee \neg\top \approx \neg\phi \vee \bot \approx \phi \longrightarrow \bot$
- (2.15): $\bot \approx \neg\top \approx \neg(\neg\phi \vee \phi) \approx \neg\neg\phi \wedge \neg\phi \approx \phi \wedge \neg\phi$

□

Wie man hier sieht haben wir auf der linken Seite der Äquivalenz eine Formal $\phi \square \psi$, wobei $\square$ auf der rechten Seite der Äquivalenz nicht auftaucht. Durch wiederholte Anwendung dieser Umformung können wir jede Formel in eine überführen, in der der Operator $\square$ nicht mehr enthalten ist. ($\square$ kann hier für $\longleftrightarrow$, $\longrightarrow$, $\wedge$, $\vee$ stehen.)

2.3.8 Beispiele für Umformungen

Wir können $\models \sigma \longleftrightarrow \tau$ zeigen, in dem wir $\sigma \approx \tau$ durch Umformung zeigen, und wir können $\models \sigma$ zeigen, in dem wir $\sigma \approx \top$ durch Umformung zeigen.

- $\models \phi \longrightarrow \psi \longleftrightarrow \neg\psi \longrightarrow \neg\phi$:

$$\begin{aligned}\phi \longrightarrow \psi &\approx \neg\phi \vee \psi\\ &\approx \psi \vee \neg\phi\\ &\approx \neg(\neg\psi) \vee \neg\phi\\ &\approx \neg\psi \longrightarrow \neg\phi\end{aligned}$$

- $\models \phi \longrightarrow \psi \longrightarrow \phi$

$$\begin{aligned}\phi \longrightarrow \psi \longrightarrow \phi &\approx \neg\phi \vee (\neg\psi \vee \phi)\\ &\approx (\neg\phi \vee \phi) \vee \neg\psi\\ &\approx \top \vee \neg\psi \approx \top\end{aligned}$$

Übung 2.6 *Für die ausschließende Disjunktion gibt es zwei mögliche Formulierungen durch andere Operatoren:*

$$\phi \dot{\vee} \psi \longleftrightarrow (\neg(\phi \longleftrightarrow \psi)) \tag{2.16}$$
$$\phi \dot{\vee} \psi \longleftrightarrow (\phi \vee \psi) \wedge (\neg\phi \vee \neg\psi) \tag{2.17}$$

Zeige durch Umformen, dass diese äquivalent sind.

Übung 2.7 *Zeige durch Umformen:*

$$\models \phi \longrightarrow (\psi \longrightarrow \sigma) \longleftrightarrow \phi \wedge \psi \longrightarrow \sigma$$

Vorlesung vom 21.04.2026: Aussagenlogik IV

Übung 2.8 (Aufwärmübung) *Kann ich folgende Formel in eine äquivalente Formel umformen, die nur $\wedge$ und $\neg$ als Konnektive enthält?*

$$(A \longrightarrow \neg B) \wedge A$$

2.3.9 Kernsprachen

Aus den Gleichungen (2.9) bis (2.15) undTheorem 2.5 folgt, dass sich die Aussagenlogik nur mit folgenden Mengen von Operatoren definieren lässt:

$$\{\vee, \neg\} \quad \{\longrightarrow, \neg\} \quad \{\wedge, \neg\} \quad \{\longrightarrow, \bot\} \tag{2.18}$$

Die anderen Operatoren werden dann jeweils als abgeleitete Konnektive definiert, wie wir das mit Äquivalenzoperator gemacht haben. Warum machen wir das nicht? Es würde das Beweisen sehr erschweren, weil wir immer erst alle Operatoren durch die repräsentierenden ersetzen müssen, dadurch werden die Terme (und der Suchraum) sehr groß. (Schließlich programmieren wir ja auch nicht in minimalen Programmiersprachen wie SUBLEQ.)

Wie beweisen wir das? Man kann es sich einfach machen: “einfach die Äquivalenzen solange anwenden wie es geht”. Das ist allerdings nicht präzise genug. Man muss dazu dieses “Anwenden solange es geht” präziser machen. Wir zeigen eine etwas vereinfachte Ausage, die wir später brauchen:

Lemma 2.6 *Sei $Prop' \subseteq Prop$ die Menge aller aussagenlogischen Formlen, die nur Atome und die Operatoren $\wedge, \vee, \neg$ enthalten. Dann gibt es zu jeder aussagenlogischen Formel $\phi \in Prop$ gibt es eine äquivalente Formal $\phi' \in Prop'$, die nur Atome und die Operatoren $\wedge, \vee, \neg$ enthält.*

Beweis. Der Beweis ist über der *Struktur* von ϕ:

- $\phi \equiv p \in P$: Induktionsbasis, $p \in Prop'$.
- $\phi \equiv \neg\phi_1$: Nach Induktionsvoraussetzung gibt es $\phi_1' \in Prop'$ mit $\phi_1 \approx \phi_1'$, dann ist $\phi' \stackrel{def}{=} \neg\phi_1' \in Prop'$ und $\phi' \equiv \phi$.
- $\wedge, \vee$: Analog.
- $\phi \equiv \phi_1 \longrightarrow \phi_2$: Nach Induktionsvoraussetzung gibt es $\phi_1', \phi_2' \in Prop'$ mit $\phi_1 \approx \phi_1', \phi_2 \approx \phi_2'$. Dann ist $\phi' \stackrel{def}{=} \neg\phi_1' \vee \phi_2' \approx \phi_1' \longrightarrow \phi_2' \approx \phi_1 \longrightarrow \phi_2 \approx \phi$ wie gefordert.
- $\phi \equiv \phi_1 \longleftrightarrow \phi_2$: Analog.

□

2.3.10 Der Sheffer-Strich

Man braucht nicht unbedingt zwei Operatoren, es reicht ein einziger: die Wahrheitstabelle des Sheffer-Strich ist definiert als

$\mid$	0	1
0	1	1
1	1	0

Offensichtlich ist das die negierte Konjunktion (NAND), und es gilt

$$\phi \mid \psi \approx \neg(\phi \wedge \psi)$$
$$\neg\phi \approx \phi \mid \phi$$
$$\phi \wedge \psi \approx (\phi \mid \psi) \mid (\phi \mid \psi)$$

Da wir wissen, dass $\{\neg, \wedge\}$ alle anderen Operatoren ausdrücken kann (2.18), reicht also auch der Sheffer-Strich alleine.

Wir können uns beliebig weitere Operatoren ausdenken und anhand ihrer Wahrheitstabelle definieren. Können wir diese auch mit den Operatoren aus (2.18), oder äquivalent dem Sheffer-Strich, ausdrücken? Ja, das geht, aber wir müssen das erstmal beweisen.

Theorem 2.7 (Functional completeness) *Sei $\$$ ein n-stelliger Operator, definiert durch die Auswertungsfunktion $f_\$$ (bspw. notiert als Wahrheitstabelle), d. h. $[\![\$(p_1,\ldots,p_n)]\!]_v = f_\$(v(p_1),\ldots,v(p_n))$ (mit Atomen $p_1,\ldots,p_n$). Dann gibt es eine Aussage $\tau \in Prop$, die lediglich Atome $p_1\ldots,p_n$, $\wedge$ und $\neg$ enthält, so dass $\models \tau \longleftrightarrow \$(p_1,\ldots,p_n)$.*

Beweis. Induktion über n. Siehe [2, Theorem 1.3.6] (dort wird die funktionale Vollständigkeit von $\vee$, $\neg$ gezeigt, aber das ist äquivalent zu unserer Behauptung, da wird $\vee$ durch $\wedge$ und $\neg$ ausdrücken können). □

Wir sagen, dass der Sheffer-Strich, die Menge $\{\vee, \neg\}$ oder die anderen Mengen von Operatoren aus (2.18) *funktional vollständig* (functionally complete) sind.

Übung 2.9 *Wir haben bereits gesehen, wie wir $\neg$ und $\wedge$ mit dem Sheffer-Strich ausdrücken; drücke auch alle anderen Operatoren damit aus.*

2.4 Beweisverfahren

Dieses ganze Umformerei oben ist ja etwas planlos. Wir haben eine ungefähre Idee, wie wir eine *gegebene* Formel beweisen, aber das ist noch keine allgemeines Rezept.

2.4.1 Normalformen

Kommen wir noch einmal auf die quadratischen Gleichungen vom Anfang zurück. Wie lösen wir eine *beliebige* quadratische Gleichungen wie

$$(2x+7)x+19 = 15x+13?$$

Die Antwort ist, wir bringen die Gleichung in eine Normalform:

$$\begin{aligned}(2x+7)x-19 &= 15x-9\\ 2x^2+7x-15x-19+9 &= 0\\ 2x^2-8x-10 &= 0\\ x^2-4x-5 &= 0\end{aligned}$$

Diese Normalform können wir dann schematisch lösen, und erhalten $x = -\frac{4}{2} \pm \sqrt{\frac{p^2}{4}+5} = 2 \pm 3$, also $x = -1 \vee x = 5$.

Normalformen für quadratische (und allgemeiner polynomiale) Gleichungen haben die Form $ax^2+bx+c=0$ (oder $\sum_{i=0}^{n} a_i x^i = 0$). Für Aussagenlogik gibt es etwas ähnliches. Wir können nämlich Lemma 2.6 anwenden, und unsere Aussagen in eine ähnliche Normalenform bringen.

Definition 2.12 (Literale) *Ein* Literal *hat die Form p oder $\neg p$ für $p \in P$. Für ein Literal (das entweder die Form p oder $\neg p$ für ein Atom $p \in P$ hat) ist $\bar{L}$ definiert als $\bar{p} = \neg p$ und $\bar{(\neg p)} = p$.*

Definition 2.13 (Normalformen) *Ein Ausdruck $\phi \in Prop$ ist in* konjunktiver Normalform *(conjunctive normal form, CNF) wenn ϕ die Form*

$$\phi = \phi_1 \wedge \phi_2 \wedge \cdots \wedge \phi_n$$

und jedes ϕ_i hat die Form

$$\phi_i = \psi_{i,1} \vee \psi_{i,2} \vee \cdots \vee \psi_{i,m_i}$$

hat, wobei $\psi_{i,j}$ ein Literal ist.

Ein Ausdruck $\phi \in Prop$ ist in disjunktiver Normalform *(disjunctive normal form, DNF) wenn ϕ die Form*

$$\phi = \phi_1 \vee \phi_2 \wedge \cdots \vee \phi_n$$

und jedes ϕ_i hat die Form

$$\phi_i = \psi_{i,1} \wedge \psi_{i,2} \wedge \cdots \wedge \psi_{i,m_i}$$

hat, wobei $\psi_{i,j}$ ein Literal ist.

Mit anderen Worten, konjunktive Normalformen sind Konjunktionen von Disjunktionen von Literalen, und disjunktive Normalformen sind Disjunktionen von Konjunktionen von Literalen.

Für die verallgemeinerte Konjunktion und Disjunktion nutzen wir folgende Schreibweisen:

$$\bigwedge_{i=1,\ldots,n} \phi_i = \phi_1 \wedge \phi_2 \wedge \cdots \wedge \phi_n$$

$$\bigvee_{i=1,\ldots,n} \phi_i = \phi_1 \vee \phi_2 \vee \cdots \vee \phi_n$$

Damit können wir eine Formel in CNF schreiben als $\phi \equiv \bigwedge_i \bigvee_j L_{i,j}$, wobei alle $L_{i,j}$ Literale sind.

Die Negation einer Formel in CNF ergibt mit den deMorgan-Gesetzen eine Formel in DNF, und umgekehrt:

$$\neg(\bigwedge_i \bigvee_j L_{i,j}) \approx (\bigvee_i \bigwedge_j \overline{L_{i,j}} \tag{2.19}$$

$$\neg(\bigvee_i \bigwedge_j L_{i,j}) \approx (\bigwedge_i \bigvee_j \overline{L_{i,j}} \tag{2.20}$$

Theorem 2.8 *Zu jeder aussagenlogischen Formel $\phi \in Prop$ gibt es eine Formel $\rho \in Prop$ in CNF, so dass $\phi \approx \rho$, und $\sigma \in Prop$ in DNF, so dass $\phi \approx \sigma$.*

Beweis. Der Beweis ist ähnlich wie Lemma 2.6 durch strukturelle Induktion über ϕ. Hierfür nutzen wir zuerst die Äquivalenzen (2.9) und (2.10), um die Operatoren $\longleftrightarrow$ und $\longrightarrow$ zu ersetzen, so dass wir nur die Operatoren $\wedge$, $\vee$, $\neg$ und $\bot$ betrachten müssen.

Die Induktionsbasis sind also Atome $p \in P$, die sowohl in CNF als auch DNF sind, und $\bot$; $\bot$ ist die leere Disjunktion, und damit sowohl in CNF als auch DNF.

Für die Negation ist es wichtig, dass man beide Teile des Theorems gleichzeitig beweist, nicht separat. Mit $\phi \equiv \neg\phi_1$ haben wir nach Induktionsannahme ρ in CNF und σ in DNF mit $\phi_1 \approx \rho \approx \sigma$. Dann sind aber mit (2.19) und (2.20) $\neg(\rho)$ äquivalent zu einer Formel in DNF und $\neg(\sigma)$ äquivalent zu einer Formel in CNF.

Für die Disjunktion ist $\phi \equiv \phi_1 \vee \phi_2$. Um eine äquivalente Formel in DNF zu finden, wenden wir einfach die Induktionsvoraussetzung auf ϕ_1 und ϕ_2 and, und erhalten Formeln σ_1, σ_2 in DNF, so dass $\sigma_1 \vee \sigma_2 \approx \phi_1 \vee \phi_2 \approx \phi$ auch in DNF ist. Interessanter ist der Fall, eine CNF für ϕ zu finden. Hierzu nutzen wir das Distributivgesetz. Wir haben

$$(\bigvee_i A_i) \vee B \approx (\bigvee_i (A_i \vee B))$$
$$(\bigvee_i A_i) \vee (\bigvee_j B_j) \approx \bigvee_{i,j} A_i \vee B_j$$

Mit anderen Worten, die Disjunktion zweier verallgemeinerter Disjunktionen $\bigvee_i A_i$ und $\bigvee_j B_j$ ist die Konjunktion aller Kombinationen von Disjunktionen $A_i \vee B_j$. Damit lässt sich hier für ϕ_1 und ϕ_2 jeweils eine CNF ρ_1 und ρ_2 finden, so dass $\rho_1 \vee \rho_2$ äquivalent zu einer CNF ist. □

Übung 2.10 *Ist die CNF/DNF eindeutig?*

Ein Ausdruck wird also durch wiederholte Anwendung der Regeln in CNF/DNF überführt. Leider ist das sehr aufwändig, und die Formel wird sehr groß. Eine andere Möglichkeit ist, die CNF/DNF aus der Wahrheitstabelle abzulesen. Wir stellen das an einem Beispiel dar.

Gegeben die Formel $\phi = (A_3 \vee \neg A_1) \longrightarrow (A_2 \longleftrightarrow A_3)$ mit folgender Wahrheitstabelle:

A_1	A_2	A_3	ϕ
0	0	0	1
0	0	1	0
0	1	0	0
0	1	1	1
1	0	0	1
1	0	1	0
1	1	0	1
1	1	1	1

Hier können wir die DNF direkt ablesen. Die Formel ist wahr gdw. einer der Zeilen in denen der letzte Eintrag 1 ist zutrifft:

$$\phi_D = (\neg A_1 \wedge \neg A_2 \wedge \neg A_3) \vee (A_1 \wedge \neg A_2 \wedge \neg A_3) \vee (\neg A_1 \wedge A_2 \wedge A_3) \vee (A_1 \wedge A_2 \wedge \neg A_3) \vee (A_1 \wedge A_2 \wedge A_3)$$

Die CNF ergibt sich aus den anderen Zeilen. Die Formel ist wahr, wenn sie nicht falsch ist, d.h. wenn keine der Zeilen in denen der letzte Eintrag 0 ist, zutrifft, und dafür muss mindestens eine der Variablen vorne mit 1 belegt sein:

$$\phi_C = (A_1 \vee A_2 \vee \neg A_3) \wedge (A_1 \vee \neg A_2 \vee A_3) \wedge (\neg A_1 \vee A_2 \vee \neg A_3)$$

Vorlesung vom 22.04.2026: Aussagenlogik V

2.4.2 Erfüllbarkeit und SAT

Übung 2.11 (Aufwärmübung) *Gegeben folgende Menge von Formeln:*

$$\Psi = \{B \longrightarrow A, C \longrightarrow B, \neg C \longrightarrow A \vee D, \neg D\}$$

1. Kann *im Kontext* Ψ *die Aussage A gelten?*
2. *Gilt sie im Kontext* Ψ immer*?*

Die Frage, ob eine gegebene aussagenlogische Formel $\phi \in \mathit{Prop}$ erfüllbar ist (d. h. gibt es eine Belegung v der Atome in ϕ, so dass $[\![\phi]\!]_v = 1$), ist das Erfüllbarkeitsproblem (SAT). Es ist aus zwei Gründen prominent: zum einen lassen sich viele praktische Probleme aus dem Schaltkreisentwurf und der Programmverifikation auf Erfüllbarkeitsprobleme abbilden und damit lösen, und zum anderen war das Erfüllbarbeitsproblem das erste (soweit ich weiß), dessen NP-Vollständigkeit nachgewiesen wurde:

Theorem 2.9 (Cook) *Das Erfüllbarkeitsproblem für aussagenlogische Formeln ist NP-vollständig.*

NP-vollständig heißt zum einen, dass SAT NP-hart ist, d. h. nicht besser als NP lösbar, und darüber hinaus, dass jedes Problem in NP in polynomialier Zeit in ein SAT-Problem überführt werden kann. Das macht SAT zu einer beliebten Technik, um NP-Vollständigkeit zu beweisen, indem man das Problem auf SAT reduziert; man muss dazu nur die Problemstellung in Aussagenlogik formulieren (was wesentlich einfacher ist als es in das Wortproblem einer Turingmaschine zu übersetzen).

Heutzutage gibt es eine Reihe von Werkzeugen, die SAT lösen (SAT-Solver, wie Minisat oder Chaff). Richtig mächtig sind Kombinationen von SAT mit sogenannten Hintegrundtheorien, wie beispielsweise lineare Ungleichheiten über natürlichen Zahlen (*satisfiability modulo theory*, SMT), wie sie Werkzeugen wie Z3, Yices, CVC oder Alt-Ergo implementiert sind. Moderne SAT-Solver können problemlos Instanzen mit mehreren Millionen Klauseln lösen.

Das typische Eingabeformat für einen SAT-Solver ist CNF, kodiert im DIMACS-Format; SMT-Solver lesen Eingaben im smtlib-Format, und übersetzen intern in CNF (wenn nötig).

SAT-Solving geht von einer Formel ϕ in CNF aus:

$$\phi = \bigwedge_{i=1}^{n} \bigvee_{i=1}^{m_n} L_{i,j}$$

Die einzelnen Glieder der Konjunktionen nennen wir *Klauseln*. Im folgenden wird sowohl die Konjunktion als eine Menge von Klauseln, und die einzelnen Klauseln als eine Menge von Literalen betrachtet (weil die Reihenfolge hier Banane ist). Das erlaubt uns Dinge zu schreiben wie $L \in \phi_i$ wenn eine Klausel ϕ_i das Literal L enthält, oder $\phi_1 \cup \phi_2$ für die Vereinigung von Klauselmengen (was der Konjunktion entspricht). Eine Klausel mit einem einzigen Literal ist eine *Unit-Klausel*.

Um eine Formel ϕ wie oben zu erfüllen, müssen alle Klauseln erfüllt sein. Das machen wir, indem wir Stück für Stück alle Atome $a \in \text{atoms}(\phi)$ belegen. Das Problem ist, da gibt es sehr viele Möglichkeiten — nämlich exponentiell viele.

Trotzdem ist SAT NP-vollständig, also besser als exponentiell. Es hat die "klassische" Struktur von NP-vollständigen Problemen: einen sehr großen (exponentiellen) Suchraum, aber eine effiziente Funktion zur Überprüfung (polynomial oder besser; hier ist das die Auswertung der Klauselmenge unter einer gegebenen Belegung).[1]

Der klassische Algorithmus nach Davis-Putnam-Logemann-Loveland (DPLL) ist ein Backtracking-Algorithmus und funktioniert wie folgt: wähle ein beliebiges Atom $a \in \phi$. Wähle eine Belegung $a \mapsto 1$ und propagiere die Belegung durch alle Klauseln in ϕ. Wenn dabei eine unerfüllbare Klausel entsteht, brechen wir diesen Versuch ab, ansonsten geht der Versuch mit dem nächsten Literal weiter. Beim Abbruch macht das Backtracking mit der Belegung $a \mapsto 0$ weiter.

Eine einfache Optimierung besteht darin, vor der Auswahl eines Literals alle Unit-Klauseln zu propagieren. Dazu wird für jede Unit-Klausel p die Belegung $p \mapsto 1$, und für $\neg p$ die Belegung $p \mapsto 0$ auf die Klauselmenge angewandt. Wenn durch die Belegung ein Literal zu 1 auswertet, dann wird diese Klausel gestrichen (weil $1 \vee A \approx 1$ ist), und wenn in einer Klausel ein Literal zu $\bot$ auswertet, wird dieses aus der Klausel gestrichen (weil $0 \vee A \approx A$).

Hier ist ein einfaches Beispiel. Gegeben folgende Formel in CNF:

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

Wir berechnen eine Belegung für die Atome A, B, C, D:

	$A \vee B$	$B \vee C \vee \neg D$	$\neg A \vee B \vee C$	$A \vee \neg B$
$A \mapsto 0$	$0 \vee B$	$B \vee C \vee \neg D$	$0 \vee B \vee C$	$0 \vee \neg B$
	B	$B \vee C \vee \neg D$	$B \vee C$	$\neg B$
$UP : B \mapsto 1$	1	$1 \vee C \vee \neg D$	$1 \vee C$	0 ↯
$A \mapsto 1$	$1 \vee B$	$B \vee C \vee \neg D$	$0 \vee B \vee C$	$1 \vee \neg B$
	1	$B \vee C \vee \neg D$	$B \vee C$	1
$B \mapsto 0$	1	$0 \vee C \vee \neg D$	$0 \vee C$	1
	1	$C \vee \neg D$	C	1
$UP : C \mapsto 1$	1	$1 \vee \neg D$	1	1
	1	1	1	1

Die Belegung ist also $\sigma = \langle A \mapsto 1, B \mapsto 0, C \mapsto 0 \rangle$. D wird hier gar nicht belegt; das bedeutet, die Formel wird mit σ unter allen (beiden) Belegungen von D wahr.

Man beachte, wie die *unit propagation* (UP) die Berechnung deutlich beschleunigt.

Übung 2.12 *Gegeben folgende Formelmenge:*

$$\psi = \{A \wedge B \longrightarrow C, D \longrightarrow B, D \longrightarrow B \vee C\}$$

- *Berechne die Klauselmenge für ψ: bilde die Konjunkt aller Elemente, $\phi = \bigwedge \psi$, und berechne CNF.*
- *Berechne eine erfüllende Belegung mit dem Algorithmus von oben.*

[1] Eine andere wichtige Meta-Eigenschaft von NP-vollständigen Problemen, die aus dieser Struktur folgt, ist dass wir sie mit guten Heuristiken oft erstaunlich gut lösen können. Das bedeutet auch, dass für ein interessantes NP-vollständiges Problem die Suche nach guten Heuristiken ein lohnenswertes Unterfangen ist.

2.4.3 Resolution

Erfüllbarkeit ist bis jetzt eine rein semantische Eigenschaft. Intuitiv ist eine Formel ϕ erfüllbar, genau dann wenn aus ihr kein Widerspruch folgt (sie ist *konsistent*, aber das definieren wir später noch genau), und sie ist nicht erfüllbar genau dann wenn sie inkonsistent ist:

$$\phi \text{erfüllbar} gdw. \phi \not\models \bot$$

Wir werden jetzt einen syntaktischen Kalkül einführen, der es uns erlaubt aus einer Formelmenge andere Formeln zu schließen, mit dem Ziel, den Widerspruch herzuleiten. Damit hätte man dann gezeigt, dass $\phi \models \bot$ wie oben. Damit kann man auch zeigen, ob die Formel ϕ eine Tautologie ist (indem man $\neg\phi$ zum Widerspruch führt), oder ob ϕ aus Ψ folgt, $\Psi \models \phi$ (indem man $\Psi \wedge \neg\phi$ zum Widerspruch führt).

Generell betrachten wir Formeln in CNF, die wir als *Mengen* von *Klauseln* behandeln; eine Klausel ist dabei ein Disjunktion von Literalen, auch als Menge behandelt. Eine Menge von Formeln in CNF ist dann einfach die Konjunktion der Formeln in CNF, oder alternativ die Vereinigung der Mengen von Klauseln.

Resolution basiert auf folgender Idee: wenn wir Klauseln $A \vee P$ und $B \vee \neg P$ haben, dann ist die Belegung von P eigentlich irrelevant, weil beide gelten, wenn $A \vee B$ gilt. Mit anderen Worten, Literale die in einer Klausul positiv (nicht negiert) und einer negiert auftreten, können wir elminieren.

Wir fassen das formal. Gegeben drei Klauseln ϕ_1, ϕ_2, ϕ_3, dann ist ϕ_3 die *Resolvente* von ϕ_1 und ϕ_2, wenn

- es ein Literal L gibt, so dass $L \in \phi_1$ und $\bar{L} \in \phi_2$, und
- ϕ_3 von der Form $\phi_1 \setminus \{L\} \cup \phi_2 \setminus \{\bar{L}\}$ ist.[2]

Wir resolvieren immer über einem Literal L. Gibt es mehrere $L_1, \ldots, L_N$ in ϕ_1, ϕ_2 welche die Voraussetzung erfüllen, gibt es auch mehrere Resolventen. Welches sind alle Resolventen der Formeln $\phi_1 = A \vee \neg B \vee X \vee D$ und $\phi_2 = \neg A \vee B \vee Y \vee \neg D$?

Lemma 2.10 (Resolutionslemma) *Sei F eine Klauselmenge, mit $\phi_1, \phi_2 \in F$ und ϕ_3 eine Resolvente von ϕ_1 und ϕ_2. Dann ist F erfüllbar gdw. $F \cup \{\phi_3\}$ erfüllbar ist.*

Beweis. Siehe [1, Kapitel 1.5, S. 40]. □

Der Resultionsalgorithmus fügt solange Resolventen hinzu, bis keine neuen mehr zu finden sind. Formal:

Definition 2.14 (Resolution (Res)) *Für eine Klauselmenge F*

$$\begin{aligned} \mathrm{Res}(F) &= F \cup \{\phi_3 \mid \phi_1, \phi_2 \in F, \phi_3 \textit{ ist Resolvente von } \phi_1 \textit{ und } \phi_2\} \\ \mathrm{Res}^0(F) &= F \\ \mathrm{Res}^{n+1}(F) &= \mathrm{Res}(\mathrm{Res}^n(F)) \\ \mathrm{Res}^*(F) &= \bigcup_{0 \le i} \mathrm{Res}^i(F) \end{aligned}$$

Wichtig ist folgende Eigenschaft des Algorithmus: wenn wir bei der Resolution irgendwann durch Resolution eine leere Klausel finden, dann ist F nicht erfüllbar.

[2] Diese Mengennotation ist korrekt, weil wir die Klauseln als Mengen (von Literalen) betrachten.

Theorem 2.11 (Resolutionssatz) *Eine Klauselmenge F ist unerfüllbar genau dann wenn $\emptyset \in \mathrm{Res}^*(F)$.*

Beweis. Dieser Satz ist nicht ganz so trivial zu beweisen, insbesondere die Vollständigkeit (Richtung von links nach rechts), d. h. wenn die Formelmenge unerfüllbar ist finden wir auch immer eine leere Klausel. Siehe [1, Kapitel 1.5, S. 41] □

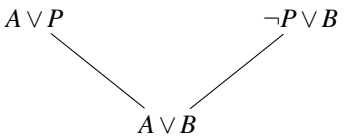
Resolution ist zum einen ein mächtiges Beweisverfahren, welches auch für die Prädikatenlogik funktioniert (im Gegensatz zum SAT-Solving, welches nur für die Aussagenlogik funktioniert), und zum anderen dient es als Ausführungsmodell bei der Programmiersprache Prolog. In Prolog sind Programme Klauseln in spezieller Form: eine *Hornklausel* ist eine Klausel $\phi = \{L_1, \ldots, L_n\}$ wobei es höchstens ein positives L_i mit $L_i = p_i$ (und alle anderen $L_j = \neg p_j$) gibt. Eine Hornklausel $\{\neg p_1, \ldots, \neg p_n, q\}$ ist offensichtlich äquivalent zu einer Implikation der Form $p_1 \wedge \ldots p_n \longrightarrow q$.

Übung 2.13 *Betrachte folgende Klauselmenge:*

$$F = \{\neg B \vee A, B \vee A, \neg A \vee \neg C, \neg B \vee C, B \vee C\}$$

Können wir die Unerfüllbarkeit zeigen, indem wir mittels Resolution die leere Klausel herleiten?

Resolution wird gerne graphisch dargestellt:



Das wird bei längeren Resolutionsbeweisen schwierig, deshalb nutzen wir eine lineare Schreibweise, in der wir die Klauseln untereinander schreiben, und jeder eine fortlaufende Nummer geben. Bei neuen Resolventen wird notiert, welche Klausel darüber resolviert wurden.

Hier ist ein ganz einfaches Beispiel. Gegeben folgende Klauselmenge:

$$F = \{A, \neg A \vee B, \neg B\}$$

Res^0	1:	A	
	2:	$\neg A \vee B$	
	3:	$\neg B$	
Res^1	4:	B	$\mathrm{Res}(1,3)$
	5:	$\neg A$	$\mathrm{Res}(2,3)$
Res^2	6:	$\bot$	$\mathrm{Res}(1,5)$
	7:	$\bot$	$\mathrm{Res}(3,5)$

Im nullten Schritt ist die Klauselmenge Res^0 die initiale Klauselmenge (hier F). Danach resolvieren wir alle Klauseln miteinander, und erhalten hier zwei neue Klauseln 4 und 5. Im nächsten Schritte versuchen wir alle "alten" Klauseln mit den neuen zu resolvieren, und erhalten weitere neue Klauseln— in diesem Fall zwei leere Klauseln, und wir können die Resolution beenden, weil die Inkonsistenz nachgewiesen wurde. Hinter jeden neu hinzu gekommenen Klausel vermerken wir, von welchen Klauseln sie die Resolvente ist; falls das nicht eindeutig ist (s. unten), dann vermerken wir auch das Literal, über dem resolviert wurde.

Etwas allgemeiner müssen wir initial (Stufe 1) alle Klauseln in Res^0 miteinander resolvieren; die entstehenden Resolventen bilden dann die Menge Res^1. Im $i+1$-ten Schritt ($i \geq 1$) resolvieren wir dann die Klauseln aus Res^{i-1} mit denen in Res^i neu hinzugekommenen.

Res^0	1 :	$\neg X \vee A$	
	2 :	$\neg A \vee \neg Y$	
	3 :	$\neg X \vee Y$	
	4 :	$Y \vee X$	
	5 :	$X \vee A$	
Res^1	6 :	$\neg X \vee \neg Y$	$\mathrm{Res}(1,2)$
	7 :	$A \vee Y$	$\mathrm{Res}(1,4)$
	8 :	A	$\mathrm{Res}(1,5)$
	9 :	$\neg A \vee \neg X$	$\mathrm{Res}(2,3)$
	10 :	$\neg A \vee X$	$\mathrm{Res}(2,4)$
	11 :	$X \vee \neg Y$	$\mathrm{Res}(2,5)$
	– :	$A \vee Y$	$\mathrm{Res}(3,5)$, redundant
...			

Tabelle 2.1: Resolution für die Klauselmenge $F = \{\neg X \vee A, \neg A \vee Y, \neg X \vee Y, Y \vee X, X \vee A\}$

Nicht jede Resolvente ist neu; mit zunehmender Dauer des Algorithmus werden immer mehr Resolventen erzeugt, die der Algorithmus schon "gesehen" hat. Diese Resolventen fügen wir nicht hinzu.

Hier ist ein längeres Beispiel für eine Resolution. Gegeben seien folgende Klauselmenge:

$$F = \{\neg X \vee A, \neg A \vee Y, \neg X \vee Y, Y \vee X, X \vee A\}$$

Der Resolutionsalgorithmus berechnet Res^i für $i = 1,2,3$ wie in Tabelle 2.1; im dritten Schritt erhalten wir direkt einen Widerspruch und terminieren.

Eine Beobachtung aus Tabelle 2.1 ist, dass eine Resolution mit Klauseln der Form $P \vee \neg P$ (mit P atomar) keine neuen Klauseln hervorbringt: sei die andere Klausel $X \vee P$ (mit X einem Literal), dann ist die Resolvente wieder $\{X,P\} \setminus \{P\} \cup \{P,\neg P\} \setminus \{\neg P\} = \{X,P\}$ (analog für Resolution mit $X \vee \neg P$); diese Klauseln brauchen wir also eigentlich nicht weiter zu betrachten.

Vorlesung vom 28.04.2026: Natürliches Schließen I

2.5 Natürliches Schließen

Übung 2.14 (Aufwärmübung) *Formalisiert folgende Aussagen:*

> *"Wenn ihr intelligent seid, und für die Prüfung lernt, dann werdet ihr auch die Prüfung bestehen. Wenn ihr diesen Kurs gewählt habt, dann seid ihr schon mal intelligent. Wenn ihr hier seid, dann habt ihr wohl diesen Kurs gewählt, und ihr seid offensichtlich hier. Lernen tut ihr auch. Daraus folgt, dass ihr die Prüfung bestehen werdet."*

Die Atome sollten sein:

- I — Ihr seid intelligent
- L — Ihr lernt für die Prüfung
- K — Ihr habt diesen Kurs gewählt

- A — Ihr seid hier (anwesend)
- P — Ihr besteht die Prüfung

Wir formalisieren den Sachverhalt in eine Menge Γ von Voraussetzungen, und eine Folgerung P — nämlich, dass ihr die Prüfung besteht:

$$\Gamma = \{I \wedge L \longrightarrow P, K \longrightarrow I, A \longrightarrow K, A, L\}$$

Zu zeigen ist $\Gamma \models P$ oder

$$(I \wedge L \longrightarrow P) \wedge (K \longrightarrow I) \wedge (A \longrightarrow K) \wedge A \wedge L \longrightarrow P$$

Wir könnten das mit den aus dem letzten Abschnitt bekannten Mitteln zeigen, die auf Wahrheitstabellen beruhen, aber auf die Dauer ist das nicht befriedigend, weil es auf einer externen Sicht von Wahrheit beruht. Was genau bedeutet es, wenn ein Atom "wahr" ist, oder nicht? In dem Beispiel oben, was bedeutet "intelligent"? Oder ein Beispiel aus der Mathematik, aus den Wahrheitstabellen folgt semantisch dass

$$\models R \vee \neg R$$

was für eine beliebige Aussage R heißt, dass eines von beiden gilt — aber was, wenn R nicht entscheidbar ist? Oder wir nicht wissen, welches von beiden gilt (z. B. für $P = NP$)?

Dieses Problem wird noch drängender, wenn unsere Aussagen nicht mehr atomar sind, sondern strukturiert, so dass wir beispielsweise mathematische Aussagen formalisieren können. Wir können dann einzelnen Aussagen nicht mehr einfach Wahrheitswerte zuweisen, sondern müssen strukturierte Beweise führen. Ein Beispiel hierfür sind wieder die Lösung quadratischer Gleichungen — um diese Art von Rechnung zu formalisieren reicht es sicherlich nicht, jeden Term $a = b$ als atomare Aussage zu behandeln.

Darüber hinaus zeigen die semantisch basierten Beweisverfahren aus den vorherigen Abschnitten für realistische Probleme (wenn wir über mehr reden als nur atomare Aussagen) eine überexponentielle Komplexität, oder sind gar nicht mehr entscheidbar. Um auch bei diesen Problemen — und dazu zählen typischerweise Probleme, die bei der Programmverifikation entstehen – behandeln zu können, müssen wir über *Beweise* reden.

Was bedeutet das alles? Anstatt eine Wahrheitstabelle für die oberen Aussagen zu machen, und zu folgern, dass P in allen Fällen gelten muss (wir überlassen das den geneigten Lesern als Übung), wollen wir folgende Beweisführung in Logik gießen — und zwar so, dass wir es hinterher mechanisch[3] überprüfen können:

1. Wir sind hier, und wenn wir hier sind, dann haben wir diesen Kurs gewählt, also haben wir den Kurs gewählt.
2. Wir haben den Kurs gewählt, und wenn man den Kurs wählt, ist man intelligent, also sind wir intelligent.
3. Wir haben für die Prüfung gelernt.
4. Wir sind also intelligent *und* haben für die Prüfung gelernt.
5. Wenn wir intelligent sind und für die Prüfung gelernt haben, bestehen wird die Prüfung.

[3] Idealerweise mit einem Computer — aber dazu später.

6. Also bestehen wir die Prüfung.

Noch einmal mit Atomen:

1. Wir haben A und es gilt $A \longrightarrow K$, also haben wir K;
2. Wir haben K und es gilt $K \longrightarrow I$, also haben wir I;
3. Wir haben L.
4. Wir haben I und L, also haben wir $I \wedge L$.
5. Wir haben $I \wedge L$, und es gilt $I \wedge L \longrightarrow P$, also haben wir P.

Wir sehen, dass unser Beweis aus einer Folge von Schlüssen besteht, die eine oder mehrere Voraussetzungen (Prämissen) haben, und eine Schlussfolgerung (Konklusion). Generell ist also eine Schlussregel von der Form

$$R \in Prop^n \times Prop$$

Wir können zwei Schlussregeln R und S komponieren, wenn die Konklusion von R einer Prämisse von S entspricht (ggf. müssen wir die Atome in R und S geeignet substituieren); dann entsteht eine neue Regel, deren Prämissen die von R und den von S übrigbleibenden Prämissen sind, deren Schlußfolgerung die von S ist.

Beispiel 2.1 (Eine Semantik-Freie Sprache) *Gegeben sei eine Sprache* $\mathscr{L} = \{\clubsuit, \spadesuit, \heartsuit, \diamondsuit\}$ *mit folgenden Schlußregeln:*

$$\frac{\diamondsuit}{\clubsuit}\,\alpha \qquad \frac{\diamondsuit}{\spadesuit}\,\beta \qquad \frac{\clubsuit \quad \spadesuit}{\heartsuit}\,\gamma \qquad \frac{}{\diamondsuit}\,\delta$$

Wie können wir jetzt $\heartsuit$ *ableiten?*

Wenn wir δ *mit* α *komponieren, erhalten wir eine Ableitung von* $\clubsuit$*; analog für* δ *und* β *eine Ableitung von* $\spadesuit$*. Diese beiden können wir mit* γ *komponieren, und erhalten eine Ableitung von* $\heartsuit$*.*

Graphisch stellen wir diese Ableitung wie folgt dar:

$$\dfrac{\dfrac{\dfrac{}{\diamondsuit}\,\delta}{\clubsuit}\,\alpha \qquad \dfrac{\dfrac{}{\diamondsuit}\,\delta}{\spadesuit}\,\beta}{\heartsuit}\,\gamma$$

Für das natürliche Schließen führen wir folgende Regeln für Konjunktion und Implikation ein. Diese Regeln sind *Axiome*, sie sind die Basis aller Ableitungen, und wir müssen sie — zu einem gewissen Grad — einfach glauben, genau wie wir die Wahrheitstafeln als gegeben hinnehmen mussten.[4]

$$\frac{A \quad B}{A \wedge B}\,\wedge I \qquad\qquad \frac{A \quad A \longrightarrow B}{B}\,mp$$

Damit können wir jetzt unseren Beweis formalisieren:

$$\dfrac{\dfrac{\dfrac{\dfrac{A \quad A \longrightarrow K}{K}\,mp \qquad K \longrightarrow I}{I}\,mp \qquad L}{I \wedge L}\,\wedge I \qquad I \wedge L \longrightarrow P}{P}\,mp$$

[4] Wir können allerdings zeigen, dass diese Regeln und die Wahrheitstafeln zueinander passen — mehr dazu später.

Wie wir sehen, können wir Konjunktionen zeigen und mit Implikationen arbeiten, aber können wir mit Konjunktionen arbieten, und beispielsweise aus $I \wedge L\, I$ folgern? Dazu brauchen wir eine weitere Regel — genauer gesagt zwei:

$$\frac{A \wedge B}{A}\ \wedge E_L \qquad\qquad \frac{A \wedge B}{B}\ \wedge E_R$$

Wir beginnen ein Muster zu erkennen: es gibt für $\wedge$ eine Regel mit $\wedge$ in der Konklusion, und zwei (weil $\wedge$ zwei Argumente hat) mit $\wedge$ in der Prämisse. Regeln der ersten Art heißen *Einführungsregeln*, Regeln der zweiten Art *Eliminationsregeln*.

Für die Implikation hatten wir schon eine Regel mit $\longrightarrow$ in der Konklusion, also eine Eliminationsregel für $\longrightarrow$, der gute alte Modus Ponens (hier aus Gründen der Konsistenz *mp* genannt). Wie sieht hier die Einführungsregeln aus? Das führt uns zum wesentlichen Feature des natürlichen Schließens, nämlich der Umgang mit Annahmen. Die Regel ist

$$\frac{\begin{array}{c}[A]\\ \vdots \\ B\end{array}}{A \longrightarrow B}\ \longrightarrow I$$

Wenn wir aus der Annahme A die Schlussfolgerung B herleiten können, dann haben wir die Implikation $A \longrightarrow B$ gezeigt (und können die Annahme A *schließen*).

Hier ist Beispiel 2.1 erweitert, um den Umgang mit Annahmen zu demonstrieren:

Beispiel 2.2 (Eine Semantik-Freie Sprache) *Gegeben sei die Sprache* $\mathscr{L} = \{\clubsuit, \spadesuit, \heartsuit, \diamondsuit\}$ *mit folgenden Schlußregeln:*

$$\frac{\diamondsuit}{\clubsuit}\ \alpha \qquad \frac{\clubsuit}{\spadesuit}\ \beta \qquad \frac{\begin{array}{c}[\diamondsuit]\\ \vdots \\ \spadesuit\end{array}}{\heartsuit}\ \gamma$$

Wie können wir jetzt $\heartsuit$ *ableiten?*

Wir können γ *mit* β *kombinieren. Damit bleibt die Annahme* $\diamondsuit$ *offen. Wenn wir diese kombinierte Regel noch einmal mit* α *kombinieren, können wir die Annahme schließen und erhalten die gewünschte Ableitung.*

Graphisch stellen wir diese Ableitung wie folgt dar:

$$\begin{array}{l}\dfrac{[\diamondsuit]}{\clubsuit}\ \alpha \\ \dfrac{\clubsuit}{\spadesuit}\ \beta \\ \dfrac{\spadesuit}{\heartsuit}\ \gamma\end{array}$$

Annahmen wie $\diamondsuit$ *oben* können *(müssen aber nicht) benutzt werden, um Annahmen zu schließen. Hätten wir zusätzlich eine Regel*

$$\frac{}{\spadesuit}\ \delta$$

könnten wir $\heartsuit$ *direkt durch die Kombination von* γ *und* δ *beweisen.*

Mit den Axiomen von oben erlauben uns diese Regel die Schlußfolgerung “Wenn ihr hier, seit ihr intelligent” (Glückwunsch!):

$$\dfrac{\dfrac{[A]^1 \quad A \longrightarrow K}{K}\, mp \qquad K \longrightarrow I}{\dfrac{I}{A \longrightarrow I}\, \longrightarrow I_1}\, mp$$

Jede Anwendung der $\longrightarrow I$-Regel *kann* eine oder *mehrere* Annahmen schließen. Damit wir wissen, welche Annahme geschlossen wird, notieren wir an der Regel und an der Annahme einen Index (hier 1).

Die Anwendung von Regeln definiert die syntaktische Ableitbarkeit:

Definition 2.15 (Ableitbarkeit) *Seit Γ eine Menge von Aussagen und $\phi \in Prop$ eine Aussagen, dann ist ϕ aus Γ ableitbar, geschrieben*

$$\Gamma \vdash \phi$$

genau dann wenn es eine Ableitung mit den Regeln des natürlichen Schließens gibt, so dass die offenen Annahmen der Ableitung alle in Γ enthalten sind. Für $\emptyset \vdash \phi$ schreiben wir $\vdash \phi$.

Der Begriff “Regeln des natürlichen Schließens” ist hier noch etwas offen gehalten, da wir noch nicht für alle Konnektiven Regeln definiert haben (siehe Definition 2.16 unten).

In einem Beweis des natürlichen Schließens (ND-Beweis, für *natural deduction*) $\Gamma \vdash \phi$ ist ein Baum aus Instanzen dieser Regeln, dessen offene Blätter nur Elemente von Γ sind; alle anderen Blätter müssen geschlossen werden. Aus Gründen der Lesbarkeit verzichten wir an dieser Stelle auf eine präzise mathematische Definition dieses Baumes ([2, Abschnitt 1.5] hat hier die technischen Details); die obige Darstellung und Definition 2.15 sind für unsere Zwecke präzise genug.

Der Beweis einer Tautologie ϕ ist eine ND-Beweis $\vdash \phi$, der keine offenen Annahmen hat. Wir betrachten zwei weitere Beispiele ([2, S. 32]):

- Das erste ist $\vdash A \wedge B \longrightarrow B \wedge A$:

$$\dfrac{\dfrac{\dfrac{[A \wedge B]^1}{B}\, \wedge E_R \qquad \dfrac{[A \wedge B]^1}{A}\, \wedge E_L}{B \wedge A}\, \wedge I}{A \wedge B \longrightarrow B \wedge A}\, \longrightarrow I_1$$

- Das zweite ist $\vdash (P \longrightarrow (Q \longrightarrow R)) \longrightarrow (P \wedge Q \longrightarrow R)$:

$$\dfrac{\dfrac{\dfrac{\dfrac{[P \wedge Q]^2}{Q}\, \wedge E_R \qquad \dfrac{\dfrac{[P \wedge Q]^2}{P}\, \wedge E_L \qquad [P \longrightarrow (Q \longrightarrow R)]^1}{Q \longrightarrow R}\, mp}{R}\, mp}{P \wedge Q \longrightarrow R}\, \longrightarrow I_2}{(P \longrightarrow (Q \longrightarrow R)) \longrightarrow (P \wedge Q \longrightarrow R)}\, \longrightarrow I_1$$

Übung 2.15 *Auf dem Landgut von Sir Archibald Fortescue Lord Netherworth-Middlington (seine Freunde nannten ihn Neddles) hat sich ein schreckliches Verbrechen ereignet: der Lord wurde morgens ermordet in Gewächshaus gefunden, wo er seine seltenen Rosen züchtet. Detective Constable Brian Quickthink von Scotland Yard ermittelt.*

Außer dem Lord waren zum Tatzeitpunkt (über Nacht) auf dem Landgut nur noch sein Butler und der Gärtner.[5] *Wenn es der Butler war, muss er im Gewächshaus gewesen sein. Allerdings war der Butler*

[5] Es ist ja heute nicht mehr so einfach Personal zu finden...

nicht im Gewächshaus (an seinen Schuhen war keine Spur von Dreck zu finden, und es hatte in der Nacht stark geregnet).

Formalisiert den Sachverhalt in Aussagenlogik, überführt den Mörder und beweist seine Schuld!

Wir haben Atome G (der Mörder war der Gärtner), B (der Mörder war der Butler), und C (der Butler war im Gewächshaus). Damit ist

$$\Gamma = \{G \vee B, B \longrightarrow C, \neg C\}$$

Wie kann DC Quickthink den Mörder überführen? Uns fehlen noch Regeln, mit Negation und Disjunktion umzugehen!

Wir führen diese ein:

$$\frac{A}{A \vee B}\ \vee I_L \qquad \frac{B}{A \vee B}\ \vee I_R \qquad \frac{A \vee B \quad \begin{array}{c}[A]\\ \vdots \\ C\end{array} \quad \begin{array}{c}[B]\\ \vdots \\ C\end{array}}{C}\ \vee E$$

$$\frac{\begin{array}{c}[A]\\ \vdots \\ \bot\end{array}}{\neg A}\ \neg I \qquad \frac{A \quad \neg A}{\bot}\ \neg E$$

$$\frac{\bot}{A}\ \mathit{false} \qquad \frac{\begin{array}{c}[\neg A]\\ \vdots \\ \bot\end{array}}{A}\ \mathit{raa}$$

Die beiden Einführungsregeln für die Disjunktion sind (hoffentlich) offensichtlich; wenn A gilt, dann auch $A \vee B$. Die Eliminationsregel müssen wir als *Fallunterscheidung* begreifen: um aus $A \vee B$ etwas zu folgern, müssen wir zum einen A und zum anderen B annehmen (also die beiden Fälle unterscheiden), und aus beiden jeweils das Gleiche folgern, dann gilt es auch für die Disjunktion. Die Regeln für die Negation werden schnell klar, wenn wir die Negation als Abkürzung $\neg A \equiv A \longrightarrow \bot$ einführen. Die Regeln für Falsum sind etwas erklärungsbedürftig. Die erste (eine Eliminationsregel für $\bot$) besagt semantisch "ex falso quodlibet", also wenn ich erst einmal einen Widerspruch habe, kann ich alles zeigen; die zweite Regel ist der klassische Widerspruchsbeweis: um A zu zeigen, führe ich $\neg A$ zum Widerspruch.

Mit diesen Regel können wir den Mörder überführen:

$$\frac{B \vee G \qquad \dfrac{\dfrac{[B]^1 \quad B \longrightarrow C}{C}\ mp \qquad \neg C}{\dfrac{\bot}{G}\ \mathit{false}}\ \neg E \qquad [G]^1}{G}\ \vee E_1$$

Der Mörder war also der Gärtner. Verhaften Sie die üblichen Verdächtigen.

Vorlesung vom 29.04.2026: Natürliches Schließen II

Es fehlen noch Regeln für die Äquivalenz ($\longleftrightarrow$), die aber recht offensichtlich sein sollten. Tabelle 2.2 zeigt die Regeln für das natürliche Schließen mit allen Konnektiven.

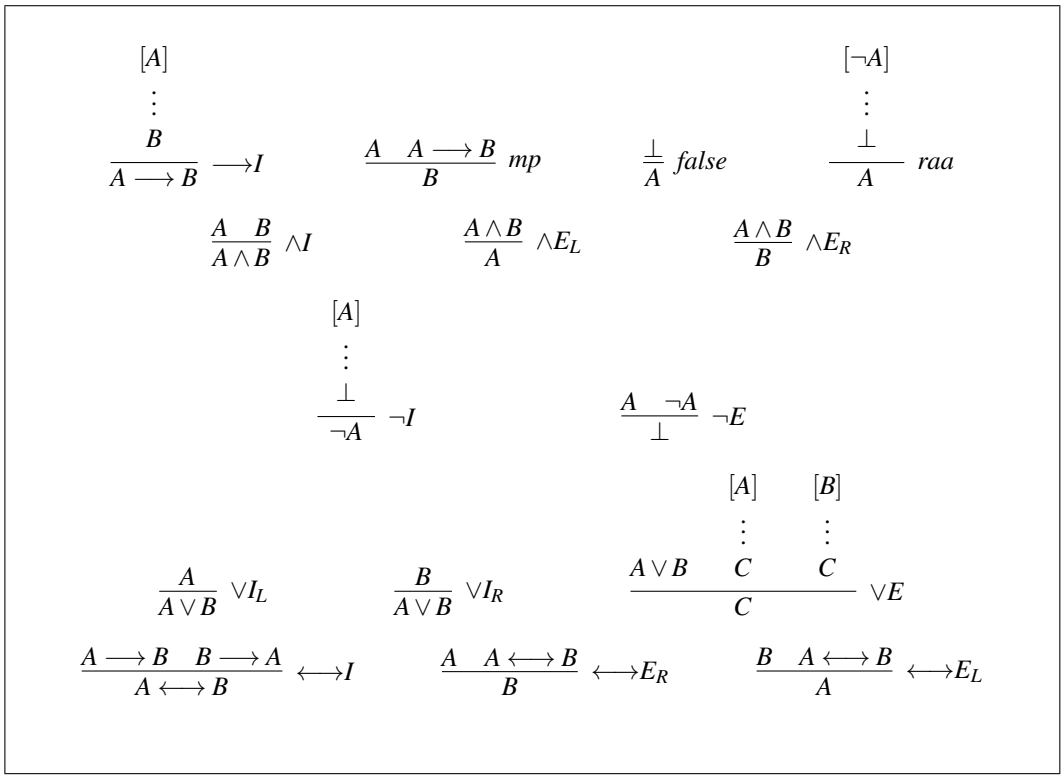


Tabelle 2.2: Regeln für das natürliche Schließen

Damit können wir jetzt die Definition 2.15 von syntaktischer Ableitbarkeit um die Definition der dafür nutzbaren Regeln des natürlichen Schließens erweitern:

Definition 2.16 (Regeln des natürlichen Schließens) *Tabelle 2.2 führt alle gültigen Regeln des natürlichen Schließens auf.*

Natürlich stellt sich sofort die Frage, wie $\Gamma \vdash \phi$ und $\Gamma \models \phi$ (syntaktische Herleitbarkeit und semantische Gültigkeit) zusammenhängen — tatsächlich ist das einer der Kernfragen der Logik. Wenn die syntaktische Herleitbarkeit die semantische Gültigkeit impliziert, sagen wir die Logik ist *korrekt*; wenn die semantische Gültigkeit die syntaktische Herleitbarkeit impliziert, wenn also alle semantisch gültigen Aussagen syntaktisch herleitbar sind, dann ist die Logik *vollständig*. Für die Aussagenlogik, soviel sei schon mal verraten, sind die beiden äquivalent; wir zeigen das im Detail später, erstmal betrachten wir die syntaktische Herleitbarkeit etwas mehr im Detail.

Noch ein einfaches Beispiel: $\vdash (A \wedge B \longrightarrow C) \longrightarrow (A \longrightarrow (B \longrightarrow C))$

$$\dfrac{\dfrac{\dfrac{\dfrac{[A]^2 \quad [B]^3}{A \wedge B}\,\wedge I \qquad [A \wedge B \longrightarrow C]^1}{C}\,mp}{B \longrightarrow C}\,\longrightarrow^3}{A \longrightarrow (B \longrightarrow C)}\,\longrightarrow I^2}{(A \wedge B \longrightarrow C) \longrightarrow (A \longrightarrow (B \longrightarrow C))}\,\longrightarrow I^1 \tag{2.21}$$

Dieses Beispiel demonstriert, wie wir ganz schematisch beweisen können. Das geht nicht immer so einfach, besonders wenn man Negation oder RAA benutzen muss.

Übung 2.16 *Leite folgende Theoreme her:*

$$\vdash (A \longrightarrow B) \longrightarrow ((B \longrightarrow C) \longrightarrow (A \longrightarrow C)) \tag{2.22}$$
$$\vdash (A \longrightarrow B) \wedge \neg B \longrightarrow \neg A \tag{2.23}$$
$$\vdash A \longrightarrow (B \longrightarrow A) \tag{2.24}$$
$$\vdash A \longrightarrow (\neg A \longrightarrow B) \tag{2.25}$$
$$\vdash \neg\neg A \longleftrightarrow A \tag{2.26}$$
$$\vdash (A \longrightarrow (B \longrightarrow C)) \longleftrightarrow (A \wedge B \longrightarrow C) \tag{2.27}$$
$$\vdash \bot \longleftrightarrow (A \wedge \neg A) \tag{2.28}$$

2.5.1 Eine Lineare Notation

Beweisbäume wie oben sind intuitiv, aber nicht einfach zu schreiben. Gerade wenn wir Beweise maschinell lesbar aufschreiben wollen, oder einfach nur in Markdown für die Übungsblätter, bietet sich eine lineare Notation an.

Um einen Baum linear zu notieren müssen wir die Baumstruktur gruppieren können, damit klar ist, welche Zeilen zu einem gegebenen Teilbaum gehören. Das kann auf drei Weisen passieren:

1. Wir können die Gruppierung explizit annotieren, mit Klammern (`{...}`) oder `begin`...`end`-Blöcken wie wir sie aus C oder Java kennen.
2. Wir können die Baumstruktur durch Einrückung sichtbar machen, wie wir das aus modernen Programmiersprachen wie Haskell oder Python kennen.
3. Wir können die Zeilen nummerieren, und die Referenzen explizit machen.

Wir entscheiden uns hier für die letzte Alternative (das Problem an Einrückungen ist, dass es während des Schreibens nicht sehr stabil ist, und Klammern sind einfach etwas unschön). In der linearen Notation hat jede Zeile ein Label (das kann, muss aber nicht, eine Zeilennummer sein).

Ein Beweis ist dann eine Folge von Zeilen. Jede Zeile besteht aus

- entweder einer Aussage $\phi \in \mathit{Prop}$ der Aussagenlogik, gefolgt von einem Regelnamen, dem Verweis auf die Prämissen der Regel in runden Klammern, und eventuell durch diesen Beweis eingeführte offene Annahmen (wie in den Regeln $\longrightarrow I$ oder $\vee E$);
- oder eine Ausage $\phi \in \mathit{Prop}$ in eckigen Klammern, gefolgt von einem Label, das entspricht einer Annahme, die hier genutzt wird (entweder aus einer Menge von Axiomen, oder durch die entsprechenden Regeln eingeführt).

Die Annahmen werden benannt; das können, wie in den Beispielen oben, Zahlen sein, oder beliebige Label, wie in dem Beispiel unten; der Namensraum der Zeilen und der Annahmen ist disjunkt. Man beachte, dass eine Annnahme öfter als ein Mal (oder auch gar nicht) benutzt werden können, aber *nur in den Zeilen, die den Prämissen und deren Prämissen* (also dem Teilbeweisbaum über der Regel, der die Annahmen einführt).

Hinweis: Die Anzahl der Regelanwendungen im Beweisbaum (oben) und die Anzahl der dabei eingeführten Prämissen sowie die Anzahl der Annahmen gibt einen Hinweis über die maximale Anzahl an Zeilennummern, die man brauchen wird.

Hier ist das Beispiel oben in linearer Markdown-Notation:

```
8: [A] b
7: [B] c
6: A & B          | conjI (7, 8)
5: [A & B --> C] a
4: C              | mp (5, 6)
3: B--> C         | impI (4) [c]
2: A--> (B--> C) | impI (3) [b]
1: (A & B --> C)--> (A--> (B--> C))  | impI (2) [a]
```

Die Zeilennummern gehen von unten, weil wir den Beweis von der Behauptung ausgehend schreiben. Ein Vorteil der Notation mit Zeilennummern ist auch, dass wir den Beweis genauso gut anders herum aufschreiben können:

```
1: (A & B --> C)--> (A--> (B--> C))  | impI (2) [a]
2: A--> (B--> C) | impI (3) [b]
3: B--> C         | impI (4) [c]
4: C              | mp (5, 6)
5: [A & B --> C] a
6: A & B          | conjI (7, 8)
7: [B] c
8: [A] b
```

Generell müssen wir bei so einem Beweis prüfen:

- Ist die Aussage $\phi \in \mathit{Prop}$ tatsächlich eine Instanz der Konklusion der Regel (d.h. können wir eine Substitution σ der Atome in der Konklusion finden, welche die Konklusion zu ϕ macht)?
- Sind die Ausssagen in den Zeilen der genannten Prämissen die gleichen wie die Prämissen der genannten Regel, wenn wir die Substitution σ anwenden? Stimmt die Anzahl der genannten Prämissen mit der Regel überein?
- Stimmt die Anzahl der genannten Annahmen mit den Annahmen der Regel überein?
- Werden alle Annahmen nur in dem Teilbaum oberhalb der Regel angewandt, die sie einführen, bzw. sind alle anderen Annahmen in der Menge der Axiome enthalten?

Die Markdown-Notation für die Aussagen sollte selbsterklärend sein. Tabelle 2.3 zeigt die Namen der Regeln in Markdown-Notation.

$\longrightarrow I$	`impI`	$\wedge I$	`conjI`	$\neg I$	`notI`	$\vee I_L$	`disjIL`	$\longleftrightarrow I$	`iffI`
mp	`mp`	$\wedge E_L$	`conjEL`	$\neg E$	`notE`	$\vee I_R$	`disjIR`	$\longleftrightarrow E_L$	`iffEL`
false	`false`	$\wedge E_R$	`conjER`			$\vee E$	`disjE`	$\longleftrightarrow E_R$	`iffER`
raa	`raa`								

Tabelle 2.3: Regelnamen in Markdown-Notation

Literaturverzeichnis

[1] U. Schöning. *Logik für Informatiker*. Spektrum Akademischer Verlag, 2000.

[2] D. van Dalen. *Logic and Structure*. Springer Verlag, 2004. Vierte Auflage.