Friedrich-Alexander-Universität
Technische Fakultät

Grundlagen der Logik in der Informatik

Lutz Schröder

Vorlesungsskript Wintersemester 2024/25
Friedrich-Alexander-Universität Erlangen-Nürnberg
Department Informatik, Lehrstuhl 8 (Theoretische Informatik)

https://www8.cs.fau.de/teaching/ws24/gloin/

Stand: 4. Dezember 2024

Überarbeitet von Thorsten Wißmann. Urpsrünglich aufbauend auf der Vorlesung von Lutz Schröder im Sommersemester 2012 mitgeschrieben von:

Johannes Schilling
Ulrich Rabenstein
Andreas Schieb
Dominik Paulus
Tobias Polzer

Inhaltsverzeichnis

1 Syntax & Semantik

In *Grundlagen der Logik in der Informatik* widmen wir uns logischem Argumentieren und dem formalen Aufschreiben logischer Beweise. Die Logiken, die wir betrachten, zerlegen sich in folgende Bestandteile:

- *Syntax*: „Was kann ich hinschreiben?“, „Was ist eine logische Aussage?“
- *Semantik*: „Was bedeutet das?“, „Wann ist eine logische Aussage wahr (bzw. falsch)?“
- *Beweismethoden*: „Wie ziehe ich daraus Schlüsse?“, „Wie beweise ich, dass eine logische Aussage wahr (bzw. falsch) ist?“

Bei den Beweismethoden, die wir betrachten, steigern wir schrittweise den Grad der Automatisierung:

1. *Formales Beweisen auf Papier*: Beweise folgen einfachen formalen Regeln.
2. *Beweisassistenten*: Beweise lassen sich in einer Programmiersprache (hier: Coq) aufschreiben und werden dann automatisiert auf Korrektheit geprüft.
3. *Resolutions-Algorithmen*: hier sucht ein Algorithmus geschickt nach einem Beweis.

In der Vorlesung hinterfragen wir diese Beweismethoden kritisch und überprüfen, dass sie auch das Konzept von *Wahrheit* korrekt abbilden. Zu Beginn betrachten wir einfache Aussagenlogik und verallgemeinern dann die Semantik und Beweismethoden auf Prädikatenlogik, die der modernen Mathematik zu Grunde liegt.

2 Aussagenlogik

Aussagenlogik redet über atomare Aussagen A, B, C, ... ohne Rücksicht auf deren innere Struktur.

Beispielsweise stellen wir eine Aussage wie

„Wenn es regnet, dann habe ich einen Schirm oder werde nass“

als Formel

$$R \to S \vee N$$

dar, um zu verdeutlichen, dass atomare Aussagen keine innere Struktur haben. Diese atomaren Aussagen können klassische Wahrheitswerte $\bot$ („falsch“) oder $\top$ („wahr“) annehmen, aus denen sich dann der Wahrheitsgehalt der logischen Formel ableitet.

2.1 Syntax

Wir definieren die Menge der aussagenlogischen Formeln ϕ, ψ *induktiv*, d.h. wir geben Regeln an, die besagen, dass bestimmte Dinge Formeln sind, wenn bereits andere Dinge Formeln sind, und als Formel akzeptieren wir dann solche Dinge, von denen sich durch endlich viele Regelanwendungen zeigen lässt, dass sie Formeln sind. Wir parametrisieren die Syntax über die Festlegung einer Menge $\mathscr{A}$ von *(propositionalen) Atomen* $A, B, C, \ldots$.

Hinweis Wir schreiben neu eingeführte Begriffe kursiv; im Tafelanschrieb werden solche Begriffe stattdessen unterstrichen. Eingeklammerte Teile eines Begriffs, hier das Wort „propositional“, verstehen wir als optional, d.h. sie können der Kürze halber weggelassen werden. Der Satz eben bedeutet also, dass wir die Elemente von $\mathscr{A}$ Atome nennen, nicht etwa, dass $\mathscr{A}$ Atome in einem schon vorher definierten Sinn enthält.

Definition 2.1. *Aussagenlogische Formeln* sind solche Objekte, die sich aus diesen Regeln bauen lassen:

- $\bot$ und $\top$ sind Formeln.
- Für jedes Atom $A \in \mathscr{A}$ ist A eine Formel.
- Wenn ϕ eine Formel ist, dann auch $(\neg\phi)$.
- wenn ϕ und ψ Formeln sind, dann auch $(\phi \wedge \psi)$, $(\phi \vee \psi)$, $(\phi \to \psi)$ und $(\psi \leftrightarrow \psi)$.

Wir schreiben $\mathscr{F}$ für die Menge aller Formeln.

Beispiel 2.2. Wir leiten her, dass $((A \wedge (\neg B)) \to \bot)$ eine Formel ist:

1. A ist eine Formel.
2. B ist eine Formel.
3. $(\neg B)$ ist eine Formel.
4. $(A \wedge (\neg B))$ ist eine Formel.
5. $\bot$ ist eine Formel.
6. $((A \wedge (\neg B)) \to \bot)$ ist eine Formel.

Man mache sich klar, in welcher Zeile jeweils welche Regel auf welche vorhergehenden Zeilen angewendet wird.

Wir können nach folgenden Konventionen Klammern weglassen: Die äußersten Klammern stets, und weitere Klammern nach den Operatorpräzedenzen:

$$\neg \text{ bindet stärker als } \wedge \text{ bindet stärker als } \vee \text{ bindet stärker als } \to \text{ und } \leftrightarrow.$$

Das ist zu verstehen wie Slogan „Punkt- vor Strichrechnung“, wo $\cdot$ stärker bindet als $+$. Damit ist z.B. $\neg A \wedge B \vee C \to D$ zu lesen als $(((\neg A) \wedge B) \vee C) \to D$. Ferner lesen wir $\wedge$, $\vee$ und $\to$ als *rechtsassoziativ*, d.h. $A \wedge B \wedge C$ steht für $A \wedge (B \wedge C)$ und $A \to B \to C$ für $A \to (B \to C)$.

Wir halten die Aussprache und intuitive Bedeutung der einzelnen Alternativen fest (die formale Definition der Bedeutung folgt im nächsten Abschnitt):

$\bot$	„Falsch“
$\top$	„Wahr“
A	„A ist wahr“, oder einfach „A“
$\neg\phi$	„nicht ϕ“
$\phi \wedge \psi$	„ϕ und ψ“
$\phi \vee \psi$	„ϕ oder ψ“
$\phi \to \psi$	„wenn ϕ, dann ψ“
$\phi \leftrightarrow \psi$	„genau dann ϕ, wenn ψ“

Hierbei ist „oder“ als *inklusives Oder* zu lesen, d.h. es dürfen auch beide Aussagen wahr sein; „wenn“-„dann“ ist eine *materielle Implikation*, d.h. wenn ϕ nicht gilt, ist $\phi \to \psi$ stets wahr – insbesondere auch dann, wenn ψ falsch ist! Ferner ist $\phi \to \psi$ stets wahr, wenn ψ gilt, ohne Rücksicht darauf, ob diese Tatsache etwas mit ϕ zu tun hat, oder ϕ auch nur wahr ist. Die Atome A bezeichnet man auch als *nichtlogische Symbole*, und entsprechend alle anderen Symbole ($\bot$, $\top$, $\neg$ etc.) als *logische Symbole*. Die nichtlogischen Symbole sind gewissermaßen der benutzerdefinierte Anteil der Formelsyntax; man erinnere sich an die Atome R für „es regnet“, S für „habe Schirm“ und N für „werde nass“ aus dem Beispiel oben.

Für die logischen Operatoren werden folgende sprachliche Bezeichnungen verwendet:

$\neg$	Negation
$\wedge$	Konjunktion
$\vee$	Disjunktion
$\to$	Implikation
$\leftrightarrow$	Biimplikation, Äquivalenz
$\top$	Wahrheit, konstant wahre Aussage, Verum
$\bot$	Falschheit, konstant falsche Aussage, Absurdität, Falsum

Ferner haben wir folgende Bezeichnungen für Bestandteile von zusammengesetzten Formeln:

- In einer Konjunktion $\phi \wedge \psi$ sind ϕ und ψ die *Konjunkte*;
- in einer Disjunktion $\phi \vee \psi$ sind ϕ und ψ die *Disjunkte*;
- in einer Implikation $\phi \to \psi$ ist ϕ das *Antezedens* und ψ das *Sukzedens*

Schreibweise 2.3. Das Gleichheitszeichen = zwischen Formeln bezeichnet syntaktische Gleichheit; z.B. $A \wedge B \neq B \wedge A$.

2.2 Natürliches Schließen

Systeme formalen Schließens dienen der rein syntaxbasierten *Herleitung* von Formeln, die einfach als zu manipulierende Zeichenketten angesehen werden. So ein Deduktionssystem besteht typischerweise aus *Axiomen*, also Formeln, die ohne weitere Voraussetzungen als hergeleitet

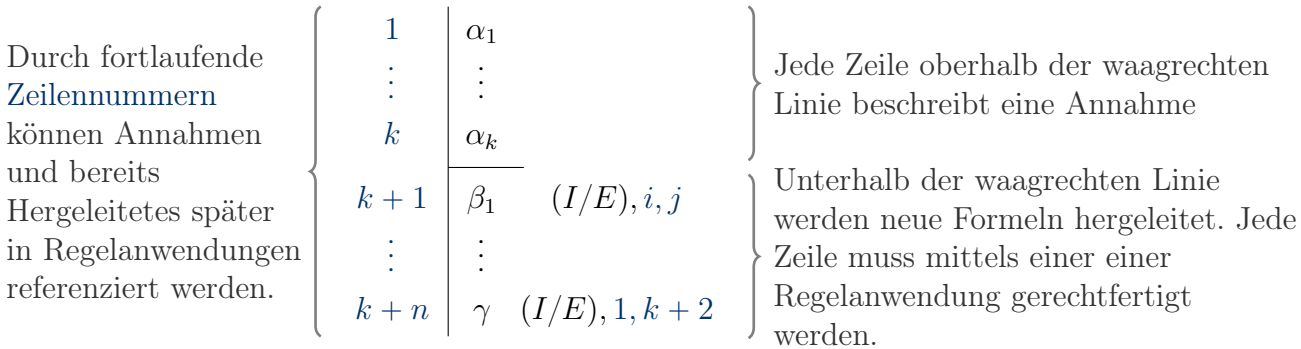


Abbildung 1: Fitch-Notation eines Beweises von $\{\alpha_1, \ldots, \alpha_k\} \vdash \gamma$

hingeschrieben werden können, sowie aus *Regeln*, die festlegen, wie man aus bereits hergeleiteten Formeln bzw. schon durchgeführten Herleitungen neue Formeln herleitet. Im einfachsten Fall hat eine Regel die Form

$$\frac{\text{Prämissen}}{\text{Konklusion}}.$$

Die *Prämissen* repräsentieren Formeln, die zur Anwendung der Regel bereits hergeleitet sein müssen; die Regel gestattet dann die Herleitung der *Konklusion*. In komplizierteren Fällen können die Prämissen statt Formeln auch ganze Herleitungen sein; dazu sehen wir später Beispiele. Zusätzlich kann eine Regel *Seitenbedingungen* haben, also außerhalb der logischen Syntax ausgedrückte Bedingungen für die Anwendbarkeit der Regel, idealerweise einfache Zusatzforderungen an die syntaktische Struktur wie etwa das Vorkommen oder Nichtvorkommen gewisser Symbole.

Oft sind die Regeln eines Kalküls so organisiert, dass man für jedes logische Konnektiv Regeln sowohl zur Einführung (I wie *Introduction*) als auch zur Entfernung (E wie *Elimination*) bereitstellt. Diese Struktur weist auch das folgende einführende Beispiel auf.

Beispiel 2.4 (Regeln für eine auf Konjunktion beschränkte Logik)**.**

$$(\wedge I)\ \frac{\phi \quad \psi}{\phi \wedge \psi} \qquad (\wedge E1)\ \frac{\phi \wedge \psi}{\phi} \qquad (\wedge E2)\ \frac{\phi \wedge \psi}{\psi}$$

Die Regel $(\wedge I)$ („Und-Einführung“) erlaubt es uns, sofern wir bereits ϕ und ψ hergeleitet haben, auch $\phi \wedge \psi$ herzuleiten; die Regeln $(\wedge E1)$ und $(\wedge E2)$ („Und-Elimination“) erlauben es, aus einer bereits hergeleiteten Konjunktion deren Konjunkte herzuleiten.

Definition 2.5. Für eine Formel ψ und eine Menge Φ von Formeln schreiben wir

$$\Phi \vdash \phi$$

wenn ϕ mittels der gegebenen Regeln (also insbesondere rein syntaktisch) aus Annahmen in Φ herleitbar ist. Für eine einelementige Formelmenge $\{\phi\}$ schreiben wir statt $\{\phi\} \vdash \psi$ einfach $\phi \vdash \psi$.

Beispiel 2.6. Es gilt $\{A, B\} \vdash A \wedge B$, schlicht per einmaliger Anwendung von $(\wedge I)$.

Je nach Deduktionssystem werden die Annahmen und bewiesenen Aussagen unterschiedlich organisiert (Beispiele in Anhang C), wir verwenden im folgenden ein System natürlichen Schließens in *Fitch-Notation*. Die allgemeine Struktur eines Beweises in Fitch ist in Abbildung 1 illustriert.

Beispiel 2.7. Ein konkreter Fitch-Beweis von $\phi \wedge \psi \vdash \psi \wedge \phi$ ist wie folgt:

1	$\phi \wedge \psi$	
2	ψ	$(\wedge E2)$ 1
3	ϕ	$(\wedge E1)$ 1
4	$\psi \wedge \phi$	$(\wedge I)$ 2, 3

Wir notieren in jedem Schritt die hergeleitete Formel (erste Spalte) sowie die verwendete Regel und die Prämissen (zweite Spalte). Die erste Zeile enthält eine Annahme; Annahmen werden von Schlüssen durch einen waagerechten Strich getrennt.

Ein Beweis $\phi \vdash \psi$ zeigt genau, dass sich ψ aus ϕ herleiten lässt. Im System natürlichen Schließens ist eine Herleitung von ψ aus ϕ genau das, was man tun muss, um eine Implikation $\phi \to \psi$ herzuleiten:

$$(\to I)\frac{\begin{array}{|l}\phi\\\hline\vdots\\\psi\end{array}}{\phi \to \psi} \qquad (\to E)\frac{\phi \to \psi \qquad \phi}{\psi}$$

Die Eliminationsregel $(\to E)$ ist auch als *modus ponens* bekannt. Die Regel $(\to I)$ hat keine Formel, sondern einen *Unterbeweis* als Prämisse. Unterbeweise haben *lokale* Annahmen, die wie in den bisherigen Beweisen durch einen waagerechten Strich abgetrennt werden. Die lokale Annahme verschwindet mit dem Ende des Unterbeweises (daher der Name), d.h. sie steht zum einen außerhalb des Unterbeweises nicht als Prämisse für Regelanwendungen zur Verfügung, zum anderen ist aber der äußere Beweis auch von ihr unabhängig, d.h. liefert Herleitungen mit entsprechend weniger Annahmen (also stärkere Gesamtaussagen).

Beispiel 2.8. Auf der linken Seite ist eine Herleitung von $A \to B \wedge C \ \vdash \ A \to B$ zu sehen:

1	$A \to B \wedge C$	
2	$\quad A$	
3	$\quad B \wedge C$	$(\to E)$ 1, 2
4	$\quad B$	$(\wedge E1)$ 3
5	$A \to B$	$(\to I)$ 2–4

1	A	
2	$\quad B$	
3	$\quad A \wedge B$	$(\wedge I)$ 1, 2
4	~~B~~	~~$(\wedge E1)$ 3~~ ↯

Der *fehlerhafte* Beweis auf der rechten Seite illustriert, was unter Lokalität zu verstehen ist: Die Annahme B in Zeile 2 sowie $A \wedge B$ in Zeile 3 sind nur innerhalb des Unterbeweises verfügbar und dürfen im äußeren Beweis in Zeile 4 nicht mehr verwendet werden. Das sollte intuitiv auch nicht der Fall sein; im dargestellten fehlerhaften Beweis hätte man zum Beispiel B aus der Annahme A hergeleitet, was ohne weitere Informationen über A und B sicher nicht möglich sein sollte.

Zu jedem Zeitpunkt ist es gestattet, einen Unterbeweis zu führen. Die rekursiv definierte Syntax von Formeln erlaubt das Verschachteln von Implikationen, und auf ähnliche Weise dürfen auch Unterbeweise verschachtelt sein. Daraus ergibt sich eine hierarchische Strukturierung von Beweisen.

Beispiel 2.9. Für den Beweis $\vdash A \to B \to A \wedge B$ verwenden wir zwei verschachtelte Unterbeweise:

1		
2	A	
3	B	
4	$A \wedge B$	$(\wedge I)$ 2, 3
5	$B \to A \wedge B$	$(\to I)$ 3–4
6	$A \to B \to A \wedge B$	$(\to I)$ 2–5

Die Beweisregeln für $\top$ und $\bot$ sind wieder etwas kürzer:

$$(\top I)\ \frac{}{\top} \qquad (\bot E)\ \frac{\bot}{\phi}$$

Für $\top$ gibt es also eine Einführungsregel $(\top I)$ ohne Prämissen, d.h. $\top$ ist ohne weitere Annahmen stets herleitbar. Da $\top$ keinerlei Information trägt, hat $\top$ keine Eliminationsregel. Genau umgekehrt verhält es sich für Falsum: $\bot$ hat keine eigene Einführungsregel (eine Regel, die $\bot$ herleitet, taucht allerdings weiter unten im Zusammenhang mit Negation auf). Die Eliminationsregel $(\bot E)$ für $\bot$ gestattet, aus $\bot$ eine beliebige Formel ϕ zu folgern:

aus einem Widerspruch folgt beliebiges (*ex falso quodlibet*).

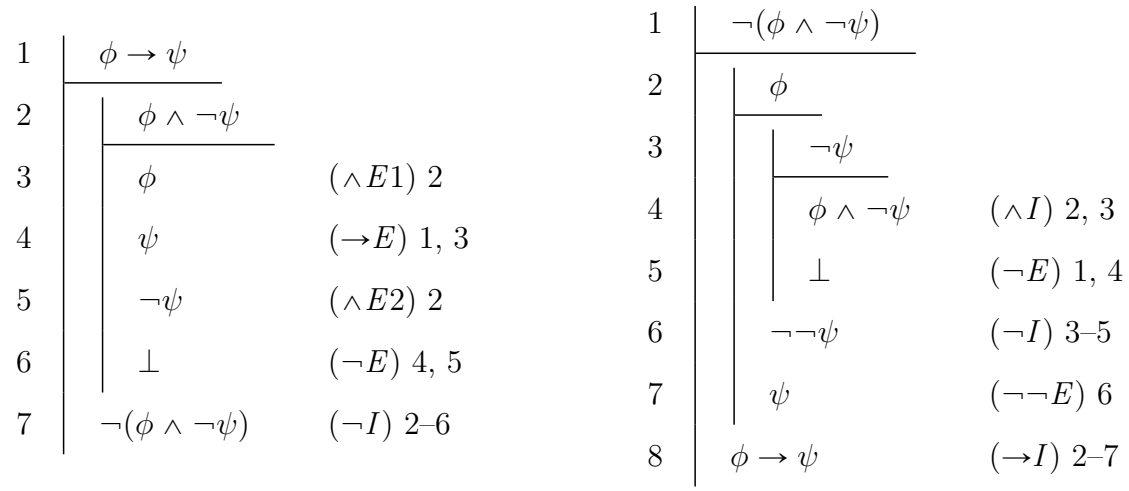
Wir können Negation $\neg\phi$ mittels Implikation und Falsum als $\phi \to \bot$ kodieren (wir haben noch keine Semantik, die eine solche Kodierung formal rechtfertigt, aber es erscheint auch zum aktuellen Zeitpunkt vielleicht plausibel, dass die Aussage „aus ϕ folgt Unsinn“ gleichbedeutend damit ist, dass ϕ eben nicht gilt). Aus der Kombination von Implikation und Falsum erhält man dann die Regeln für Negation:

$$(\neg I)\ \frac{\begin{array}{|l} \phi \\ \hline \vdots \\ \bot \end{array}}{\neg\phi} \qquad (\neg E)\ \frac{\phi \qquad \neg\phi}{\bot} \qquad (\neg\neg E)\ \frac{\neg\neg\phi}{\phi}$$

In Worten:

- um eine negierte Formel $\neg\phi$ zu beweisen, nimmt man in einem Unterbeweis ϕ an und leitet daraus einen Widerspruch her;
- um einen Widerspruch (verkörpert in der Formel $\bot$) herzuleiten, zeigt man für irgendeine Formel ϕ sowohl ϕ als auch $\neg\phi$;
- aus einer zweifach negierten Formel $\neg\neg\phi$ folgt ϕ (Doppelnegationselimination). Allein in dieser letzten Regel verbirgt sich das Prinzip vom ausgeschlossenen Dritten (*tertium non datur*, *excluded middle*).

Beispiel 2.10. Da wir mit Doppelnegationselimination $(\neg\neg E)$ arbeiten, könnte man eine Implikation $\phi \to \psi$ auch mittels $\neg(\phi \wedge \neg\psi)$ kodieren. Die Beweise in Abbildung 2 zeigen die beiden Richtungen der Äquivalenz zwischen diesen beiden Formeln. (**Achtung**: $\to$ ist üblicherweise *nicht* mittels $\neg$ und $\wedge$ kodiert, sondern elementar mittels $(\to I)$ und $(\to E)$ zu verwenden.)



(a) $\phi \to \psi \vdash \neg(\phi \wedge \neg\psi)$ (b) $\neg(\phi \wedge \neg\psi) \vdash \phi \to \psi$

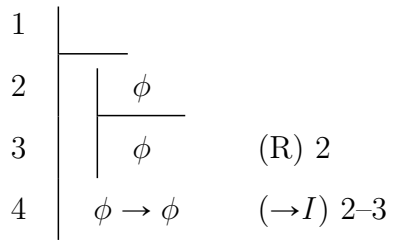
Abbildung 2: Kodierung von $\to$ mittels $\neg$ und $\wedge$

Aus technischen Gründen gönnen wir uns noch eine *Reiterationsregel*:

$$(R)\,\frac{\phi}{\phi}$$

Diese erlaubt es uns, Annahmen oder bereits Bewiesenes in die aktuelle Zeile zu kopieren, um einen Unterbeweis abzuschließen. Wir erwähnen Verwendung dieser Regel meist nicht ausdrücklich. Wie bei allen anderen Regeln zuvor darf auch bei Reiteration nur auf das Zugegriffen werden, das im aktuellen Geltungsbereich verfügbar ist.

Beispiel 2.11. Offensichtlich sollte $\vdash \phi \to \phi$ beweisbar sein – doch wie könnte man es aus den vorherigen Regeln (also ohne Reiteration) herleiten? Mit Reiteration kann man direkt zeigen:



Erweiterung auf Disjunktion Eine Disjunktion $\phi \vee \psi$ lässt sich auf zwei Arten herleiten: indem man ϕ herleitet *oder* indem man ψ herleitet. Darin spiegelt sich auch wider, dass wir $\vee$ als *inklusives* Oder verstehen: sobald wir eine Seite (ϕ oder ψ) herleiten können spielt die Herleitbarkeit der anderen Seite (also ψ bzw. ϕ) keine Rolle.

$$(\vee I1)\,\frac{\phi}{\phi \vee \psi} \qquad (\vee I2)\,\frac{\psi}{\phi \vee \psi} \qquad (\vee E)\,\frac{\begin{array}{|c|}\hline \phi \\ \vdots \\ \chi \\ \hline\end{array} \quad \begin{array}{|c|}\hline \psi \\ \vdots \\ \chi \\ \hline\end{array} \quad \phi \vee \psi}{\chi}$$

Die Eliminationsregel $(\vee E)$ verkörpert dabei das Prinzip der *Fallunterscheidung*: Um χ aus einer Disjunktion $\phi \vee \psi$ herzuleiten, zeigt man, dass man χ sowohl aus der Annahme ϕ als auch aus der Annahme ψ herleiten kann.

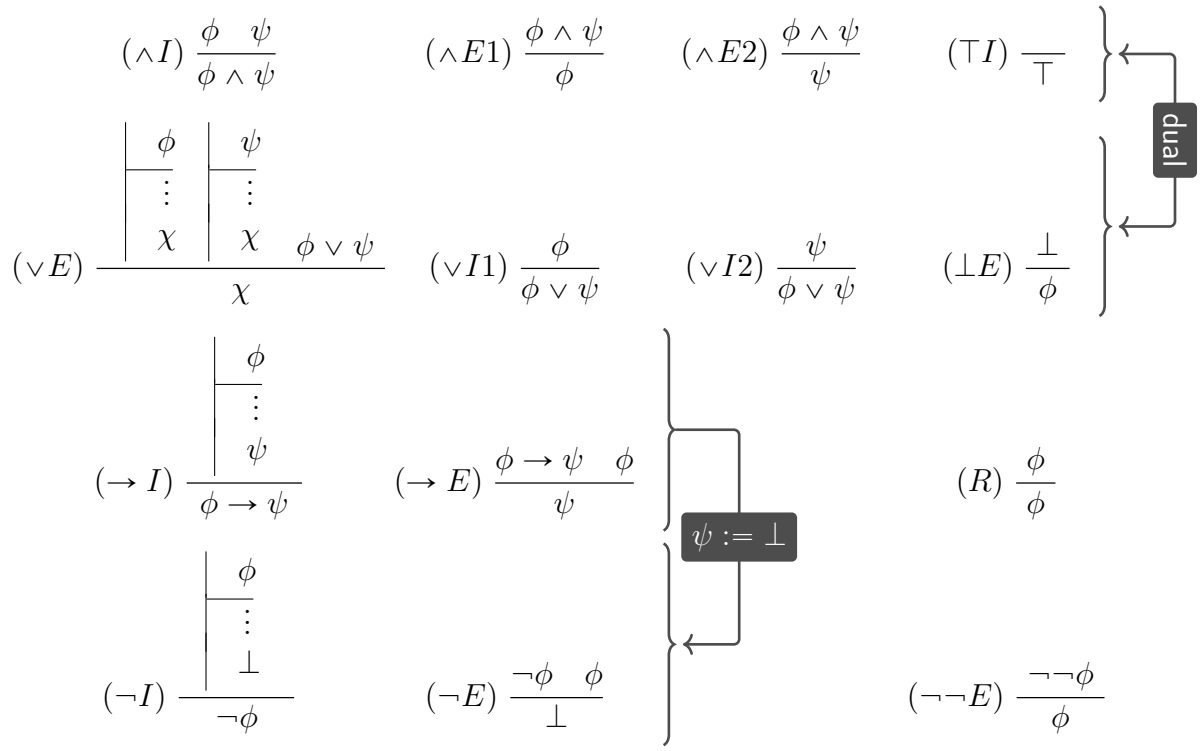


Abbildung 3: Übersicht aller Regeln natürlichen Schließens in Aussagenlogik

Beispiel 2.12. Würde man eine Implikation $\phi \to \psi$ mittels $\neg\phi \vee \psi$ kodieren, dann ließe sich $(\to E)$ mittels Fallunterscheidung $(\vee E)$ rekonstruieren:

1	ϕ	
2	$\neg\phi \vee \psi$	
3	$\quad\neg\phi$	
4	$\quad\bot$	$(\neg E)$ 1, 3
5	$\quad\psi$	$(\bot E)$ 4
6	$\quad\psi$	
7	$\quad\psi$	(R) 6
8	ψ	$(\vee E)$ 3–5, 6–7, 2

Das einzig noch verbleibende Konnektiv der Formelsyntax, Äquivalenz $\leftrightarrow$, werden wir stets mittels Implikation und Konjunktion kodieren:

$$(\phi \leftrightarrow \psi) \quad := \quad (\phi \to \psi) \wedge (\psi \to \phi)$$

Für $\leftrightarrow$ sehen wir daher keine eigenen Regeln vor.

Regelwerk Abschließend besteht also der *Kalkül natürlichen Schließens* oder kurz der *Fitch-Kalkül* für (klassische) Aussagenlogik aus den Regeln $(\wedge I)$, $(\wedge E1)$, $(\wedge E2)$, $(\top I)$, $(\bot E)$, $(\neg I)$,

$(\neg E)$, $(\neg\neg E)$, $(\vee I1)$, $(\vee I2)$, $(\vee E)$, $(\to E)$, $(\to I)$ und (R), zusammengefasst in Abbildung 3. Das Herleitbarkeitssymbol $\vdash$ bezieht sich bis auf Weiteres auf Herleitbarkeit in diesem Kalkül.

Beobachtung Die Regeln für $\wedge$, $\top$ sind *dual* zu den Regeln für $\vee$, $\bot$: Die Eliminationsregeln der einen ergeben sich aus den Einführungsregeln der anderen unter Vertauschung von Konklusion und Prämissen. Da jede Regel *genau eine* Konklusion haben muss, ergeben sich zwei Sonderfälle:

- Da $(\top I)$ keine Prämisse hat, also aus Beliebigem hergeleitet werden kann, ist die Konklusion von $(\bot E)$ eine beliebige Formel ϕ.
- Da $(\wedge I)$ zwei Prämissen hat, simulieren wir in $(\vee E)$ die zwei – als Alternativen zu denkenden – Konklusionen, indem wir die beiden Fälle ϕ und ψ auf einen gemeinsamen Nenner χ bringen, der dann die Konklusion von $(\vee E)$ ist.

2.3 Semantik

2.3.1 Wahrheitsbelegungen und Erfülltheit

Im Allgemeinen sind Formeln nicht schlechthin wahr oder falsch, sondern erhalten erst durch die Festlegung konkreter Wahrheitswerte für die propositionalen Atome einen Wahrheitswert. Formal stellt sich dies wie folgt dar:

Definition 2.13 (Wahrheitsbelegung)**.** Wir schreiben $2 = \{0, 1\} = \{\bot, \top\}$ für die Menge der *Wahrheitswerte*. Eine *Wahrheitsbelegung (WB)* ist eine Funktion

$$\kappa : \mathscr{A} \to 2,$$

legt also Wahrheitswerte für alle Atome fest. Wir schreiben $\mathscr{A} \to 2$ für die Menge aller Wahrheitsbelegungen.

Notation 2.14. Wir verwenden das Wort „Funktion“ wie in der Mathematik: $f\colon X \to Y$ bedeutet, dass f ein Objekt ist, das jedem Element von X genau ein Element von Y zuordnet; man schreibt dann $f(x)$ für das einem Element $x \in X$ zugeordnete Element von Y. Die Notation $X \to Y$ bezeichnet die Menge aller Funktionen von X nach Y.

Somit sind obige Wahrheitsbelegungen schlicht ein Spezialfall für $X = \mathscr{A}$ und $Y = 2$.

Beispiel 2.15. Für die Atome $\mathscr{A} = \{A, B, C, D, E, \ldots\}$ definiert

$$\kappa\colon \mathscr{A} \to 2 \qquad \kappa(x) = \begin{cases} \top & \text{falls } x = A \text{ oder } x = C \\ \bot & \text{andernfalls.} \end{cases}$$

eine Wahrheitsbelegung, die A sowie C auf wahr und alle anderen Atome auf falsch setzt.

Notation 2.16. Zwecks kompakterer Darstellung verwenden wir folgende Notation zur punktweisen Definition von Wahrheitsbelegungen (und ähnlichen Funktionen später). Für Elemente $y_1, \ldots, y_n \in Y$ und paarweise verschiedene $x_1, \ldots, x_n \in X$ bezeichnet

$$[x_1 \mapsto y_1, \ldots, x_n \mapsto y_n]\colon X \to Y$$

eine Funktion $X \to Y$, die jedes x_i auf y_i schickt. (Deshalb wird es verlangt, dass die x_i paarweise verschieden sind.) Falls X mehr als die n gelisteten Elemente haben sollte, dann definiert das strenggenommen die Funktion nicht vollständig, weil nicht eindeutig ist, wie sich die Funktion auf $X\backslash\{x_1, \ldots, x_n\}$ verhält. Deshalb werden wir die partielle Notation nur in Beispielen nutzen, und auch nur in solcher Weise, dass das Verhalten für alle *aktuell relevanten* Elemente von X definiert ist.

Beispiel 2.17. Wir schreiben kurz

$$\kappa = [A \mapsto \top, B \mapsto \bot, C \mapsto \top, D \mapsto \bot]$$

anstelle der Wahrheitsbelegung aus Beispiel 2.15, wenn im Kontext nur A, B, C und D relevant sind. (Beispielsweise über den Wahrheitswert $\kappa(E)$ ist dann nichts gesagt, was aber nicht schlimm ist, wenn E nirgends verwendet wird.)

Die Semantik von Formeln ist dann in Begriffen einer *Erfülltheitsrelation* $\models$ zwischen Wahrheitsbelegungen und Formeln definiert.

Definition 2.18. Allgemein ist eine *Relation zwischen X und Y* einfach eine Teilmenge $R \subseteq X \times Y$ des Kreuzprodukts $X \times Y = \{(x, y) \mid x \in X, y \in Y\}$, die eben angibt, zwischen welchen Elementen die Relation vorliegt; man schreibt dann kurz $x\ R\ y$ für $(x, y) \in R$.

Definition 2.19. Die *Erfülltheitsrelation* $\models$ ist eine Relation zwischen Wahrheitsbelegungen und Formeln:

$$\models \quad \subseteq \quad (\mathscr{A} \to 2) \times \mathscr{F}$$

(man erinnere sich, dass $\mathscr{A} \to 2$ die Menge der Wahrheitsbelegungen und $\mathscr{F}$ die Menge der Formeln ist). Wir schreiben dann $\kappa \models \phi$ für $(\kappa, \phi) \in \models$. Wir lesen $\kappa \models \phi$ als „κ erfüllt ϕ“; entsprechend schreiben wir $\kappa \not\models \phi$ („κ erfüllt ϕ nicht“) für das Gegenteil, d.h. $(\kappa, \phi) \notin \models$.

Wir definieren $\models$ nunmehr durch *Rekursion* über Formeln, wie folgt:

$$\begin{aligned}
\kappa &\not\models \bot \\
\kappa &\models \top \text{ stets} \\
\kappa \models A &\iff \kappa(A) = \top \\
\kappa \models \neg\phi &\iff \kappa \not\models \phi \\
\kappa \models \phi \wedge \psi &\iff (\kappa \models \phi \text{ und } \kappa \models \psi) \\
\kappa \models \phi \vee \psi &\iff (\kappa \models \phi \text{ oder (inklusiv) } \kappa \models \psi) \\
\kappa \models \phi \to \psi &\iff (\text{falls } \kappa \models \phi, \text{ so auch } \kappa \models \psi) \\
\kappa \models \phi \leftrightarrow \psi &\iff (\kappa \models \phi \text{ genau dann, wenn } \kappa \models \psi)
\end{aligned}$$

Man mag sich fragen, inwieweit so etwas tatsächlich eine Definition darstellt, weil ja das definierte Symbol $\models$ in seiner eigenen Definition wieder vorkommt. Man beachte aber, dass auf den jeweiligen rechten Seiten $\models$ nur für kleinere Formeln als auf der linken Seite verwendet wird; z.B. wird in der Zeile für $\phi \wedge \psi$ rechts nur Erfülltheit der echt kleineren Formeln ϕ und ψ getestet. Da eine Formel nur endlich oft echt kleiner werden kann, terminiert also die wiederholte Auffaltung der Definition auf einer gegebenen Formel stets in endlich vielen Schritten, letztlich per *Induktion* über die Größe der Formel gemäß der üblichen Induktion über natürliche Zahlen (wie in den Mathematikvorlesungen erläutert). Wir demonstrieren dies gleich noch an einem Beispiel. Für Interessierte findet sich noch eine ausführlichere Diskussion von Induktionsprinzipien in Abschnitt B des Anhangs.

Beispiel 2.20. Für die Wahrheitsbelegung $\kappa = [A \mapsto \top, B \mapsto \bot]$ ergibt obige Definition:

$$\kappa \vDash ((A \vee \neg B) \to B) \iff (\text{falls } \kappa \vDash (A \vee \neg B), \text{ so auch } \kappa \vDash B)$$

Es gilt $\kappa \vDash A \vee \neg B \iff (\kappa \vDash A \text{ oder } \kappa \vDash \neg B)$. Nun gilt $\kappa(A) = \top$, also $\kappa \vDash A$, somit $\kappa \vDash A \vee \neg B$. Es gilt aber $\kappa \nvDash B$ (da $\kappa(B) = \bot$), also $\kappa \nvDash (A \vee \neg B) \to B$.

Andere Wahrheitsbelegungen können die Formel aber erfüllen, z. B. gilt $\kappa_2 \vDash (A \vee \neg B) \to B$ für $\kappa_2 = [A \mapsto \top, B \mapsto \top]$, oder in der Tat für jedes κ_2 mit $\kappa_2(B) = \top$ (nie dagegen, wenn $\kappa_2(B) = \bot$).

Definition 2.21. Wir dehnen den Begriff der *Erfülltheit* auf Mengen $\Phi \subseteq \mathscr{F}$ von Formeln aus: Eine Wahrheitsbelegung κ *erfüllt* Φ genau dann, wenn κ alle Formeln ϕ erfüllt, die in Φ enthalten sind, d.h.

$$\kappa \vDash \Phi \ :\iff \ \forall\, \phi \in \Phi.\, (\kappa \vDash \phi).$$

$\kappa \models \Phi$ (Insbesondere gilt also $\kappa \models \varnothing$ stets.)

2.3.2 Erfüllbarkeit, Gültigkeit, logische Konsequenz

Aus der eben angegebenen Semantik ergeben sich verschiedene von einzelnen Wahrheitsbelegungen unabhängige, nur auf Formeln anzuwendende semantische Begriffe:

Definition 2.22. Seien $\phi, \psi \in \mathscr{F}$ Formeln und $\Phi \subseteq \mathscr{F}$ eine Menge von Formeln.

- Eine Menge Φ ist *erfüllbar*, wenn eine Wahrheitsbelegung κ mit $\kappa \models \Phi$ existiert; ϕ ist *erfüllbar*, wenn $\{\phi\}$ erfüllbar ist.
- Eine Formel ϕ ist *gültig*, wenn $\kappa \vDash \phi$ für *alle* Wahrheitsbelegungen κ; wir schreiben dann $\vDash \phi$.
- Eine Formel ψ ist eine *logische Folgerung* oder *logische Konsequenz* aus Φ, in Symbolen $\Phi \vDash \psi$, wenn für jede Wahrheitsbelegung κ mit $\kappa \vDash \Phi$ auch $\kappa \vDash \psi$ gilt. Für eine einelementige Formelmenge $\{\phi\}$ schreiben wir statt $\{\phi\} \vDash \psi$ einfach $\phi \vDash \psi$.
- Formeln ϕ und ψ sind *logisch äquivalent*, in Symbolen $\phi \equiv \psi$, wenn $\phi \leftrightarrow \psi$ gültig ist, d.h. wenn für jede Wahrheitsbelegung κ genau dann $\kappa \vDash \phi$ gilt, wenn $\kappa \vDash \psi$ gilt.

Logische Folgerung bezeichnet also das Verhältnis, das zwischen einer gegebenen Menge von Formeln, den *Annahmen*, und einer einzelnen Formel, der *Folgerung*, besteht, wenn immer dann, wenn die Annahmen erfüllt sind, auch die Folgerung erfüllt ist.

Achtung: Für $\Phi \vDash \psi$ wird ausdrücklich *nicht* verlangt, dass umgekehrt auch aus $\kappa \vDash \psi$ stets $\kappa \vDash \Phi$ folgt.

Beispiel 2.23. Betrachten wir für die obigen Konzepte folgende Beispiel-Formeln:

- $A \to \neg A$ ist erfüllbar, denn eine erfüllende Wahrheitsbelegung ist $\kappa(A) = \bot$.
- $A \wedge \neg A$ ist unerfüllbar.
- Sowohl $A \vee \neg A$ als auch $(A \wedge B) \to A$ sind gültig.
- $(A \vee \neg B) \to B$ und B sind logisch äquivalent

- Die logische Konsequenz $\{A \to B, A\} \models B$ gilt.

Wichtig sind folgende logische Äquivalenzen, die besagen, dass man mit $\bot$, $\neg$ und $\wedge$ die anderen logischen Operationen $\vee, \to, \leftrightarrow$ sowie $\top$ definieren kann:

$$\begin{aligned} \phi \vee \psi &\equiv \neg(\neg\phi \wedge \neg\psi) \\ \phi \to \psi &\equiv \neg\phi \vee \psi \\ \phi \leftrightarrow \psi &\equiv (\phi \to \psi) \wedge (\psi \to \phi) \\ \top &\equiv \neg\bot \end{aligned}$$

Wir werden daher oft annehmen, dass eine gegebene Formel nur $\neg$, $\wedge$ und $\bot$ verwendet.

Es ergibt sich unmittelbar aus den Definitionen, dass eine Formelmenge $\Phi \subseteq \mathscr{F}$ genau dann unerfüllbar ist, wenn $\Phi \models \bot$, und dass eine Formel ϕ genau dann gültig ist, wenn $\varnothing \models \phi$. Wir haben folgende weitere Beziehungen zwischen den einzelnen Begriffen:

Lemma 2.24. *Seien $\psi \in \mathscr{F}$ eine Formel und $\Phi \subseteq \mathscr{F}$ eine Menge von Formeln. Dann gilt*

1. *ψ ist genau dann gültig, wenn $\neg\psi$ unerfüllbar ist.*
2. *ψ ist genau dann erfüllbar, wenn $\neg\psi$ nicht gültig ist.*
3. *ψ ist genau dann logische Folgerung aus Φ, wenn die Vereinigung von Φ und der Negation von ψ unerfüllbar ist:*
$$\Phi \models \psi \iff (\Phi \cup \{\neg\psi\}) \models \bot.$$

Beweis. Teil 1. ergibt sich einfach durch Spezialisierung von 3. auf $\Phi = \varnothing$. Ferner erhält man 2. per Anwendung von 1. auf $\neg\psi$, mittels der logischen Äquivalenz $\neg\neg\psi \equiv \psi$. Für 3. zeigen wir zwei Implikationen:

„$\Longleftarrow$“: Sei $\kappa \models \Phi$; wir müssen $\kappa \models \psi$ zeigen. Beweis durch Widerspruch: wenn nicht $\kappa \models \psi$, dann per Definition $\kappa \models \neg\psi$, also $\kappa \models \Phi \cup \{\neg\psi\}$. Letzteres ist aber nach Annahme unerfüllbar, Widerspruch.

„$\Longrightarrow$“: Wir nehmen zwecks Widerspruchs an, es gäbe κ mit $\kappa \models \Phi \cup \{\neg\psi\}$. Dann gilt $\kappa \models \Phi$, aber nicht $\kappa \models \psi$, im Widerspruch zu $\Phi \models \psi$. □

2.4 Wahrheitstafeln

Eine Wahrheitstafel ist eine tabellarische Auflistung der Wahrheitswerte einer Formel in Abhängigkeit von den Wahrheitswerten der (endlich vielen!) in ihr vorkommenden Atome. Wahrheitstafeln liefern Entscheidungsalgorithmen für Erfüllbarkeit, Gültigkeit, logische Konsequenz und logische Äquivalenz in der Aussagenlogik; diese sind aber offenbar in der Praxis nicht skalierbar, da stets die gesamte (exponentiell große) Wahrheitstafel erzeugt werden muss. Konkret gilt für Formeln $\phi, \psi \in \mathscr{F}$ und eine Formelmenge $\Phi \subseteq \mathscr{F}$:

- ϕ ist gültig genau dann, wenn alle Werte für ϕ in der Wahrheitstafel $\top$ sind .
- ϕ ist erfüllbar genau dann, wenn $\top$ als Wert für ϕ in der Wahrheitstafel für ϕ vorkommt.
- $\phi \equiv \psi$ genau dann, wenn ϕ und ψ in der gemeinsamen (!) Wahrheitstafel in jeder Zeile den gleichen Wert haben.

A	B	$\neg A$	$\neg A \vee B$
$\bot$	$\bot$	$\top$	$\top$
$\bot$	$\top$	$\top$	$\top$
$\top$	$\bot$	$\bot$	$\bot$
$\top$	$\top$	$\bot$	$\top$

(a) Wahrheitstafel von $\neg A \vee B$

A	$\neg A$	$\neg A \vee A$
$\bot$	$\top$	$\top$
$\top$	$\bot$	$\top$

(b) $\neg A \vee A$ ist gültig.

A	$\neg A$	$A \to \neg A$
$\bot$	$\top$	$\top$
$\top$	$\bot$	$\bot$

(c) $A \to \neg A$ ist erfüllbar.

Abbildung 4: Beispiele für Wahrheitstafeln

A	B	$A \to B$	$\neg(A \to B)$
$\bot$	$\bot$	$\top$	$\bot$
$\bot$	$\top$	$\top$	$\bot$
$\top$	$\bot$	$\bot$	$\top$
$\top$	$\top$	$\top$	$\bot$

(a) $\neg(A \to B) \models A$

A	B	$A \leftrightarrow B$	$A \to B$
$\bot$	$\bot$	$\top$	$\top$
$\bot$	$\top$	$\bot$	$\top$
$\top$	$\bot$	$\bot$	$\bot$
$\top$	$\top$	$\top$	$\top$

(b) $(A \leftrightarrow B) \models (A \to B)$

A	B	$A \to B$	$A \wedge B$
$\bot$	$\bot$	$\top$	$\bot$
$\bot$	$\top$	$\top$	$\bot$
$\top$	$\bot$	$\bot$	$\bot$
$\top$	$\top$	$\top$	$\top$

(c) $A \to B \not\models A \wedge B$

Abbildung 5: Wahrheitstafeln für das Überprüfen logischer Konsequenzen

- $\Phi \models \phi$ (für Φ endlich) genau dann, wenn in der gemeinsamen Wahrheitstafel für Φ und ψ in jeder Zeile, in der alle $\phi \in \Phi$ den Wert $\top$ haben, die Formel ψ ebenfalls den Wert $\top$ hat.

Beispiel 2.25. Beispiele für Wahrheitstafeln finden sich in Abbildung 4. Wenn man logische Konsequenzen per Warheitstafel prüft, sind nur solche Zeilen von Relevanz, in denen alle Annahmen erfüllt sind; deshalb sind diese Zeilen in den Wahrheitstafeln in Abbildung 5 markiert. Um eine logische Konsequenz zu widerlegen, genügt es, eine beliebige Wahrheitsbelegung zu finden, in denen die Annahmen, aber nicht die (postulierte) Folgerung gilt (vgl. Abbildung 5c).

Am Beispiel $B \vee \neg B \equiv A \vee \neg A$: sieht man, dass zwei Formeln trotz unterschiedlicher verwendeter Atome äquivalent sein können. Wir geben eine formale Definition der Menge in einer Formel verwendeten Atome an:

Definition 2.26 (Atome einer Formel)**.** Die Menge $\mathrm{At}(\phi)$ der in einer Formel $\phi \in \mathscr{F}$ vorkommenden Atome ist rekursiv definiert durch

$$\begin{aligned} \mathrm{At}(A) &= \{A\} \\ \mathrm{At}(\top) &= \mathrm{At}(\bot) = \varnothing \\ \mathrm{At}(\neg\phi) &= \mathrm{At}(\phi) \\ \mathrm{At}(\phi \odot \psi) &= \mathrm{At}(\phi) \cup \mathrm{At}(\psi) \qquad \text{für } \odot \in \{\wedge, \vee, \to, \leftrightarrow\}. \end{aligned}$$

(Damit ist At eine Funktion $\mathrm{At}\colon \mathscr{F} \to \mathcal{P}(\mathscr{A})$.)

Wie bereits bei der Definition der Erfülltheit terminiert diese Definition, weil das Argument in jedem rekursiven Aufruf echt kleiner wird.

Beispiel 2.27. Wir berechnen $\mathrm{At}((A \wedge B) \wedge \neg A)$:

$$\begin{aligned}\mathrm{At}((A \wedge B) \wedge \neg A) &= \mathrm{At}(A \wedge B) \cup \mathrm{At}(\neg A)\\ &= \mathrm{At}(A) \cup \mathrm{At}(B) \cup \mathrm{At}(A)\\ &= \{A\} \cup \{B\} \cup \{A\} = \{A, B\}\end{aligned}$$

Das folgende Lemma ist die formale Legitimation dafür, dass Wahrheitstafeln sich auf die in ϕ vorkommenden Atome beschränken dürfen.

Lemma 2.28. *Für eine Formel $\phi \in \mathscr{F}$ hängt die Erfülltheit $\kappa \models \phi$ nur von den Wahrheitswerten der in ϕ vorkommenden Formeln, also von den Werten $\kappa(A)$ für $A \in \mathrm{At}(\phi)$, ab; d.h. wenn $\kappa, \kappa' \colon \mathscr{A} \to 2$ auf $\mathrm{At}(\phi)$ übereinstimmen, so erfüllt κ genau dann ϕ, wenn κ' dies tut. Formal: Wenn $\kappa, \kappa' \colon \mathscr{A} \to 2$ auf $\mathrm{At}(\phi)$ übereinstimmen, d.h. $\kappa(A) = \kappa'(A)$ für alle $A \in \mathrm{At}(\phi)$, dann gilt*

$$\kappa \models \phi \iff \kappa' \models \phi.$$

Beweis. Wie oben erläutert können wir annehmen, dass ϕ nur $\bot, \neg$ und $\wedge$ verwendet.

Wir verwenden dann Induktion über die Länge von ϕ, d.h. wir nehmen an, dass die Behauptung des Lemmas für alle Formeln, die echt kürzer sind als ϕ, schon richtig ist, und zeigen dann, dass die Behauptung für ϕ gilt. (Man erinnere sich hier an die in der Vorlesung vorgeführte boole-wertige rekursive Funktion, die sich als die konstant wahre Funktion entpuppt hat.) Wir unterscheiden Fälle nach der Form von ϕ:

$\phi = \bot$: Es gilt $\kappa \not\models \bot$ und $\kappa' \not\models \bot$.

$\phi = A \in \mathscr{A}$: Es gilt $\kappa \models A \iff \kappa(A) = \top \iff \kappa'(A) = \top \iff \kappa' \models A$.

$\phi = \neg\psi$: Es gilt $\mathrm{At}(\neg\psi) = \mathrm{At}(\psi)$, d.h. κ und κ' stimmen auf $\mathrm{At}(\psi)$ überein. Daher $\kappa \models \neg\psi \iff \kappa \not\models \psi \overset{\text{IV}}{\iff} \kappa' \not\models \psi \iff \kappa' \models \neg\psi$.

$\phi = \psi \wedge \chi$: Es gilt $\mathrm{At}(\psi \wedge \chi) \supseteq \mathrm{At}(\psi), \mathrm{At}(\chi)$, d.h. κ und κ' stimmen auf $\mathrm{At}(\psi)$ und auf $\mathrm{At}(\chi)$ überein. Daher $\kappa \models \psi \wedge \chi \iff (\kappa \models \psi \text{ und } \kappa \models \chi) \overset{\text{IV}}{\iff} (\kappa' \models \psi \text{ und } \kappa' \models \chi) \iff \kappa' \models \psi \wedge \chi$. □

Bemerkung 2.29. Die oben durchgeführte Induktion stellt sich etwas spezieller dar als *strukturelle Induktion über ϕ*, d.h. sie folgt den Regeln zur Erzeugung von Formeln gemäß Abschnitt 2.1: In jedem Schritt wird die Induktionsvoraussetzung genau für die Formeln verwendet, von denen bei der Erzeugung von ϕ angenommen wird, dass sie Formeln sind. Konkret sind dies

- keine weiteren Formeln bei $\phi = \bot$ oder $\phi = A \in \mathscr{A}$;
- die Formel ψ bei $\phi = \neg\psi$; sowie
- die Formeln ψ und χ bei $\phi = \psi \wedge \chi$.

Dieses Phänomen treffen wir häufig an. Strukturelle Induktion ist zwar grundsätzlich ein weniger mächtiges Prinzip als Induktion über die Formellänge, da wir im Induktionsschritt für weniger Formeln die Induktionsvoraussetzung zur Verfügung haben (eben nur für die direkten Bestandteile von zusammengesetzten Formeln statt für alle echt kürzeren Formeln), aber andererseits verkleinert ihre Verwendung den Suchraum bei der Durchführung des Beweises.

Die Semantik von ϕ ist also bestimmt durch endliche Tabellierung von $\kappa \models \phi$ für alle $\kappa : A_0 \to 2$ mit $A_0 \subseteq \mathscr{A}$ endlich, $\mathrm{At}(\phi) \subseteq A_0$. Dies ist die formale Rechtfertigung für das Wahrheitstafelverfahren; wir geben für den Fall der logischen Äquivalenz eine explizite Präzisierung an:

Lemma 2.30. *Für Formeln $\phi, \psi \in \mathscr{F}$ gilt $\phi \equiv \psi$ genau dann, wenn ϕ, ψ identische Wahrheitstafeln über $\mathrm{At}(\phi) \cup \mathrm{At}(\psi)$ haben.*

2.5 Logische Äquivalenzen

Im folgenden geben wir eine Übersicht wichtiger logischer Äquivalenzen, die man jeweils leicht mit der Wahrheitstafelmethode nachprüft:

$\neg\neg\phi \equiv \phi$ (Doppelnegationselimination)

$\neg(\phi \wedge \psi) \equiv (\neg\phi \vee \neg\psi)$ (de-morgansche Gesetze)
$\neg(\phi \vee \psi) \equiv (\neg\phi \wedge \neg\psi)$

$\phi \wedge (\psi \vee \chi) \equiv (\phi \wedge \psi) \vee (\phi \wedge \chi)$ (Distributivgesetze)
$\phi \vee (\psi \wedge \chi) \equiv (\phi \vee \psi) \wedge (\phi \vee \chi)$

$(\phi \wedge \psi) \wedge \chi \equiv \phi \wedge (\psi \wedge \chi)$ (Assoziativgesetze)
$\phi \vee (\psi \vee \chi) \equiv (\phi \vee \psi) \vee \chi$

$\chi \wedge \top \equiv \chi$ (Neutrale Elemente)
$\chi \vee \bot \equiv \chi$

$\phi \wedge \psi \equiv \psi \wedge \phi$ (Kommutativität)
$\phi \vee \psi \equiv \psi \vee \phi$

$\phi \wedge \phi \equiv \phi$ (Idempotenz)
$\phi \vee \phi \equiv \phi$

Man beachte, dass es zu jedem Gesetz ein *duales* Gesetz gibt, dass durch Vertauschen von $\wedge$ und $\vee$ entsteht. Das ist kein Zufall, sondern ein systematischer Effekt, der grob gesagt dadurch entsteht, dass sich bei Vertauschen der beiden Wahrheitswerte strukturell im wesentlichen nichts ändert, außer, dass eben $\wedge$ und $\vee$ die Rollen tauschen. Der formale Kern dieses Prinzips sind gerade die de-morganschen Gesetze.

2.6 Korrektheit und Vollständigkeit

Wir wollen nun natürlich, dass der Kalkül natürlichen Schließens sich vernünftig zu logischer Konsequenz $\models$ verhält; idealerweise fallen Herleitbarkeit $\vdash$ und logische Konsequenz $\models$ zusammen.

Satz 2.31 (Korrektheit)**.** *Für eine Formelmenge $\Phi \subseteq \mathscr{F}$ und eine Formel $\psi \in \mathscr{F}$ gilt*

$$\Phi \vdash \psi \implies \Phi \models \psi$$

Beweis. Wir verallgemeinern zunächst die Behauptung auf Aussagen ψ, die in Unterbeweisen gefolgert werden: Wir behaupten, dass für solche Aussagen $\Phi' \models \psi$ gilt, wobei Φ' aus allen im betreffenden Unterbeweis aktiven Annahmen besteht (inklusive der globalen Annahmen Φ). Wir beweisen die verallgemeinerte Behauptung per Induktion über die Länge n der Herleitung von ψ. Als Induktionsannahme verwenden wir, dass die Behauptung für alle Beweise echt kürzerer Länge als n gilt.

Wir unterscheiden dann nach der zuletzt angewendeten Regel, z.B. a) $\wedge I$ (hier im Skript ausgelassen), b) $(\neg I)$: Der Unterbeweis in der Prämisse bedeutet, dass $\Phi' \cup \{\phi\} \vdash \bot$, wobei Φ' die Menge der bei Anwendung der Regel aktiven Annahmen ist und im Unterbeweis eben die Annahme ϕ dazukommt. Da der Unterbeweis echt kürzer ist als der Gesamtbeweis, können wir auf ihn die Induktionsvoraussetzung anwenden, d.h. es folgt $\Phi' \cup \{\phi\} \models \bot$; mit anderen Worten, $\Phi' \cup \{\phi\}$ ist unerfüllbar. Damit folgt per Lemma 2.24 wie verlangt $\Phi' \models \neg\phi$. Für die anderen Regeln läuft der Beweis ähnlich. □

In Umkehrung zur Korrektheit gilt auch

Satz 2.32 (Vollständigkeit)**.** *Für eine Formelmenge $\Phi \subseteq \mathscr{F}$ und eine Formel $\psi \in \mathscr{F}$ gilt*

$$\Phi \models \psi \implies \Phi \vdash \psi$$

Annahme 2.33. Wir haben logische Äquivalenzen $\phi \vee \psi \equiv \neg(\neg\phi \wedge \neg\psi)$ und $\phi \to \psi \equiv \neg\phi \vee \psi$, die, wie man leicht nachrechnet, auch im System herleitbar sind (in dem Sinne, dass jeweils die rechte Formel aus der linken hergeleitet werden kann und umgekehrt). Für Zwecke des Vollständigkeitsbeweises nehmen wir daher ohne Einschränkung an, dass $\vee$ und $\to$ (und $\leftrightarrow$, das wir für Zwecke des natürlichen Schließens ohnehin als mittels $\to$ kodiert ansehen) nicht vorkommen.

Der Beweis der Vollständigkeit ist wesentlich schwieriger als der der Korrektheit, da uns die Annahme $\Phi \models \psi$ kein Objekt liefert, über das man Induktion treiben könnte (während der Beweis der Korrektheit ja einfach per Induktion über die Herleitung von ψ aus Φ lief). Stattdessen beweisen wir als Zwischenergebnis die Kontraposition für den Spezialfall $\bot$:

$$\Phi \nvdash \bot \implies \Phi \not\models \bot$$

Man bemerke, dass $\Phi \not\models \bot$ schlicht bedeutet, dass Φ erfüllbar ist. Für $\Phi \nvdash \bot$ hingegen definieren wir den Begriff der *Konsistenz*:

Definition 2.34. Eine Formelmenge $\Phi \subseteq \mathscr{F}$ heißt *konsistent*, wenn $\Phi \nvdash \bot$, d.h. wenn sich aus Φ kein Widerspruch herleiten lässt. Ferner ist Φ *maximal konsistent*, wenn Φ maximal bezüglich Mengeninklusion $\subseteq$ unter den konsistenten Mengen ist, d.h.

1. Φ ist konsistent, und
2. wenn $\Psi \subseteq \mathscr{F}$ konsistent ist und $\Phi \subseteq \Psi$, dann folgt $\Phi = \Psi$.

Mit dieser Definition lässt sich der Vollständigkeitsbeweis wie in Abbildung 6 dargestellt strukturieren. Wir vereinfachen die zu beweisende Aussage sukzessive:

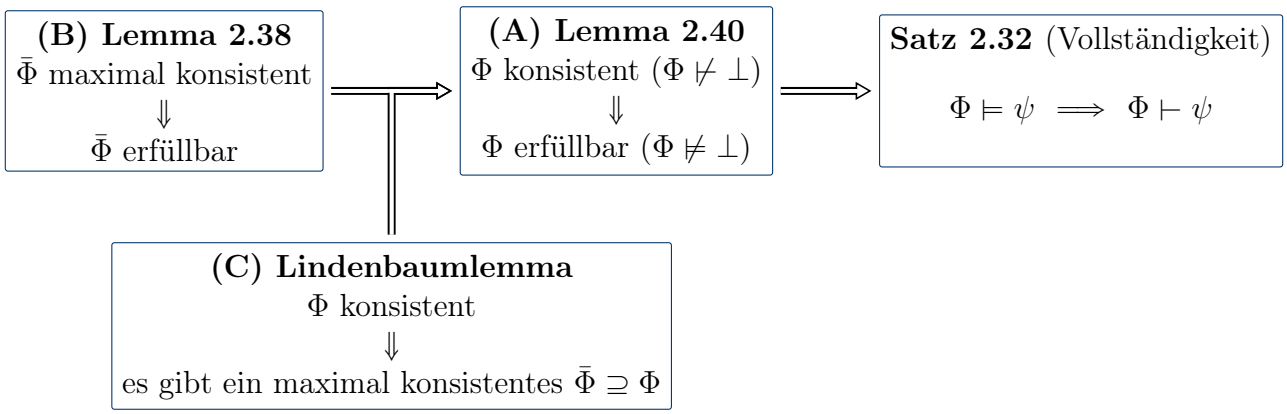


Abbildung 6: Struktur des Vollständigkeitssatzes 2.32

- Statt für allgemeines ψ betrachten wir zunächst den Spezialfall $\bot$ (A), wir nehmen also an, dass Φ konsistent ist.
- Hierfür nehmen wir zunächst sogar an, dass Φ sogar *maximal* konsistent ist.
- Um letztendlich (A) aus (B) zeigen zu können, konstruieren wir aus konsistenten Mengen maximal konsistente (C), und müssen für den allgemeinen Vollständigkeitssatz von $\bot$ zu ψ verallgemeinern.
- **Slogan:** Vollständigkeitsbeweise bestehen aus *Modellkonstruktionen.*

Mehrere Schritte dieses Programms verwenden folgende einfache Aussage:

Lemma 2.35 (Konsistenzlemma)**.** *Sei $\Phi \subseteq \mathscr{F}$ konsistent. Dann ist mindestens eine der Formelmengen $\Phi \cup \{\psi\}$ und $\Phi \cup \{\neg\psi\}$ konsistent.*

Beweis. Per Kontraposition: Seien $\Phi \cup \{\psi\} \vdash \bot$ und $\Phi \cup \{\neg\psi\} \vdash \bot$. Mittels $(\neg I)$ folgt $\Phi \vdash \neg\psi$ und $\Phi \vdash \neg\neg\psi$, also per $(\neg E)$ $\Phi \vdash \bot$, d.h. Φ ist inkonsistent. □

Wir arbeiten nun unser Programm gemäß Abbildung 6 ab, beginnend mit der schwächsten Aussage (B). Als Vorbereitung halten wir folgende Eigenschaften maximal konsistenter Mengen fest:

Lemma 2.36 (Abgeschlossenheit unter Herleitung)**.** *Sei $\Phi \subseteq \mathscr{F}$ maximal konsistent. Wenn $\Phi \vdash \psi$, dann $\psi \in \Phi$.*

Beweis. Nach den Annahmen ist $\Phi \cup \{\psi\}$ konsistent (denn aus $\Phi \cup \{\psi\} \vdash \bot$ und $\Phi \vdash \psi$ würde $\Phi \vdash \bot$ folgen), also per Maximalität enthalten in Φ. □

Lemma 2.37 (Hintikka-Eigenschaften)**.** *Sei Φ maximal konsistent. Dann gilt*

1. $\bot \notin \Phi$
2. $\neg\psi \in \Phi \iff \psi \notin \Phi$
3. $\phi \wedge \psi \in \Phi \iff \phi \in \Phi$ *und* $\psi \in \Phi$

Beweis. 1. Klar.

2. „⇒“: wäre $\psi \in \Phi$, so würde man $\Phi \vdash \bot$ mittels $(\neg E)$ erhalten, im Widerspruch zur Konsistenz von Φ.
   „⇐“: Sei $\psi \notin \Phi$. Dann erhalten wir die *echte* Obermenge $\Phi \cup \{\psi\} \supsetneq \Phi$. Da Φ maximal konsistent ist, ist $\Phi \cup \{\psi\}$ inkonsistent; nach Konsistenzlemma folgt, dass $\Phi \cup \{\neg\psi\}$ konsistent ist, und per maximaler Konsistenz von Φ folgt $\Phi \cup \{\neg\psi\} \subseteq \Phi$, also $\neg\psi \in \Phi$.

3. Per Abschlossenheit von Φ unter Herleitung (Lemma 2.36). □

Nun können wir (B) beweisen und aus einer maximal konsistenten Menge $\Phi \subseteq \mathscr{F}$ eine Wahrheitsbelegung konstruieren:

Lemma 2.38 (Wahrheitslemma)**.** *Für eine maximal konsistente Formelmenge Φ und die Wahrheitsbelegung*

$$\kappa\colon \mathscr{A} \to 2 \qquad \kappa(A) = \begin{cases} \top & \text{falls } A \in \Phi \\ \bot & \text{falls } A \notin \Phi \end{cases}$$

gilt für alle Formeln ψ:

$$\kappa \models \psi \iff \psi \in \Phi.$$

Beweis. Wir verwenden Induktion über ψ; dies ist die eine Stelle im Programm, wo wir die Annahme, dass $\vee$ und $\to$ in ψ nicht vorkommen (Annahme 2.33), tatsächlich verwenden (übrigens auch nur zur Arbeitsersparnis). Der Fall, dass ψ ein Atom A ist, ist gerade die Definition von κ; der Fall $\psi = \bot$ und die Induktionsschritte für Negation und Konjunktion verwenden die Hintikka-Eigenschaften. Z.B. haben wir

$$\kappa \models \neg\psi \iff \kappa \not\models \psi \overset{\text{IV}}{\iff} \psi \notin \Phi \overset{\text{Hintikka}}{\iff} \neg\psi \in \Phi.$$ □

Damit gilt wie verlangt $\kappa \models \Phi$, die Formelmenge Φ ist also erfüllbar.

Um (B) zu (A) zu verallgemeinern, konstruieren wir schrittweise aus konsistenten Formelmengen maximal konsistente:

Lemma 2.39 (Lindenbaumlemma)**.** *Wenn $\Phi \subseteq \mathscr{F}$ konsistent ist, dann existiert ein maximal konsistentes $\overline{\Phi} \subseteq \mathscr{F}$ mit $\Phi \subseteq \overline{\Phi}$.*

Beweis. Sei $\phi_1, \phi_2, \phi_3, \ldots$ eine Aufzählung aller Formeln. Wir konstruieren eine aufsteigende Kette $\Phi = \Phi_0 \subseteq \Phi_1 \subseteq \Phi_2 \subseteq \ldots$ konsistenter Formelmengen per

$$\Phi_{i+1} = \begin{cases} \Phi_i \cup \{\phi_i\} & \text{wenn konsistent} \\ \Phi_i \cup \{\neg\phi_i\} & \text{sonst (konsistent per Konsistenzlemma).} \end{cases}$$

Setze $\overline{\Phi} = \bigcup_{i=0}^{\infty} \Phi_i \supseteq \Phi$. Zu zeigen ist dann folgendes:

1. $\overline{\Phi}$ *ist konsistent:* Man nehme an, dass $\overline{\Phi} \vdash \bot$. Der Beweis ist endlich, verwendet also nur endlich viele Annahmen aus $\overline{\Phi}$. Jede dieser Annahmen kommt in einem Φ_i vor; durch Wahl des größten unter diesen endlich vielen Indizes i erhalten wir ein Φ_i, das *alle* verwendeten Annahmen enthält. Dann $\Phi_i \vdash \bot$, im Widerspruch zur Konsistenz von Φ_i.

2. $\overline{\Phi}$ *ist maximal:* Sei $\Psi \subseteq \mathscr{F}$ konsistent und $\overline{\Phi} \subseteq \Psi$. Zu zeigen ist dann $\Psi \subseteq \overline{\Phi}$. Sei also $\psi \in \Psi$. Es existiert n mit $\psi = \phi_n$. Aus $\Phi_n \subseteq \bar{\Phi} \subseteq \Psi$ und $\{\phi_n\} = \{\psi\} \subseteq \Psi$ folgt $\Phi_n \cup \{\phi_n\} \subseteq \Psi$. Teilmengen von konsistenten Mengen sind wieder konsistent, als ist $\Phi_n \cup \{\phi_n\}$ konsistent, und damit nach Konstruktion $\phi_n \in \Phi_{n+1} \subseteq \bar{\Phi}$. □

Somit können wir die Lücke zwischen (A) und (B) schließen:

Lemma 2.40. *Wenn $\Phi \subseteq \mathscr{F}$ konsistent ist, dann ist Φ erfüllbar.*

Beweis. Per Lindenbaumlemma (Lemma 2.39) existiert eine maximal konsistente Formelmenge $\overline{\Phi} \supseteq \Phi$. Für $\overline{\Phi}$ liefert das Wahrheitslemma (Lemma 2.38) eine erfüllende Wahrheitsbelegung κ für $\overline{\Phi}$; diese erfüllt natürlich auch die Teilmenge Φ von $\overline{\Phi}$. □

Nun können wir den Vollständigkeitsbeweis abschließen:

Beweis des Vollständigkeitssatzes (Satz 2.32). Sei $\Phi \models \psi$. Dann ist die Menge $\Phi \cup \{\neg\psi\}$ unerfüllbar. Per Kontraposition von Lemma 2.40 (A) ist $\Phi \cup \{\neg\psi\}$ inkonsistent, also

$$\Phi \cup \{\neg\psi\} \vdash \bot.$$

Per $(\neg I)$ folgt nun $\Phi \vdash \neg\neg\psi$ und per $(\neg\neg E)$ schließlich $\Phi \vdash \psi$. □

Eine unmittelbare Folgerung aus der Vollständigkeit ist die folgende Eigenschaft:

Korollar 2.41 (Kompaktheit)**.** *Sei Φ eine Formelmenge, so dass alle endlichen Teilmengen $\Phi_0 \subseteq \Phi$ erfüllbar sind (so eine Formelmenge heißt* endlich erfüllbar*). Dann ist Φ erfüllbar.*

Beweis. Nach Korrektheit und Vollständigkeit ist Erfüllbarkeit gleichbedeutend mit Konsistenz. Konsistenz hat aber offenbar die behauptete Eigenschaft: nach Negation beider Seiten ist zu zeigen, dass $\Phi \vdash \bot$ genau dann, wenn es eine endliche Teilmenge $\Phi_0 \subseteq \Phi$ gibt mit $\Phi_0 \vdash \bot$. Das ist aber klar, da ein Beweis von $\bot$ aus Φ nur endlich viele der Annahmen in Φ verwendet. □

2.7 Anwendungen des Kompaktheitssatzes

Kompaktheit ist eine durchaus überraschende Eigenschaft, die zahlreiche ebenfalls überraschende Anwendungen in der Theorie diskreter Strukturen hat. Wir diskutieren im folgenden ein Beispiel aus der Graphfärbungstheorie.

Definition 2.42. Ein (ungerichteter) Graph mit Knotenmenge V und Kantenmenge E heißt *k-färbbar*, wenn es eine Abbildung $c : V \to \{1, \ldots, k\}$ (*Colouring/Färbung*) gibt, so dass $c(v) \neq c(w)$ für jede Kante $\{v, w\} \in E$ gilt.

Der berühmte Vierfarbensatz besagt beispielsweise, dass jeder planare Graph 4-färbbar ist.

Satz 2.43. *Ein (möglicherweise unendlicher) Graph ist genau dann k-färbbar, wenn alle seine endlichen Untergraphen k-färbbar sind.*

Beweis. „Nur dann, wenn“ ist trivial; wir zeigen „wenn“.

Sei V die Knotenmenge und E die Kantenmenge des Graphen. Wir führen Atome $A_{v,i}$ für $v \in V$, $i \in \{1, \ldots, k\}$ ein, mit der Lesart „Knoten v hat Farbe i“ (also $c(v) = i$). Wir definieren Formelmengen Φ_1, Φ_2, Φ_3 wie folgt:

1. Jeder Knoten hat mindestens eine Farbe:

$$\Phi_1 = \{\textstyle\bigvee_{i=1}^{k} A_{v,i} \mid v \in V\}$$

2. Kein Knoten hat mehr als eine Farbe:

$$\Phi_2 = \{\neg(A_{v,i} \wedge A_{v,j}) \mid v \in V, i, j \in \{1, \ldots, k\}, i \neq j\}$$

3. Keine zwei adjazenten Knoten haben die gleiche Farbe:

$$\Phi_3 = \{\neg(A_{v,i} \wedge A_{w,i}) \mid \{v, w\} \in E, i \in \{1, \ldots, k\}\}$$

Wir setzen nun $\Phi = \Phi_1 \cup \Phi_2 \cup \Phi_3$. Dann ist Φ endlich erfüllbar: Sei $\Psi \subseteq \Phi$ endliche Teilmenge; dann ist die Menge $V_0 = \{v \mid \exists i.\, A_{v,i} \in \mathrm{At}(\Psi)\}$ endlich. Sei G_0 der von V_0 aufgespannte Untergraph. Nach Annahme ist G_0 k-färbbar; sei c eine entsprechende Färbung. Wir definieren eine Wahrheitsbelegung κ_0 durch

$$\kappa_0(A_{v,i}) = \top \text{ gdw. } c(v) = i$$

für $v \in V_0$ (und beliebig auf anderen Atomen). Dann gilt $\kappa_0 \models \Phi_0$.

Nach dem Kompaktheitssatz ist also Φ erfüllbar; sei $\kappa \models \Phi$. Dann existiert für jedes v genau ein i, so dass $\kappa(A_{v,i}) = \top$; wir erhalten eine k-Färbung c des Graphen, indem wir $c(v) = i$ setzen. □

Sehr ähnlich erhält man z.B. einfache Beweise von Königs Lemma (jeder endlich verzweigende unendliche Graph hat einen unendlichen Pfad) oder der Aussage, dass man, gegeben ein Satz K von quadratischen Kacheln mit gefärbten Kanten, genau dann die $\mathbb{N} \times \mathbb{N}$-Ebene mit Kacheln aus K farblich passend anschließend überdecken kann, wenn dies für jede $n \times n$-Fläche geht.

3 Normalformen und Resolution

Wir entwickeln nunmehr ein Entscheidungsverfahren für die Erfüllbarkeit aussagenlogischer Formeln, das zwar immer noch in schlechten Fällen in exponentieller Zeit läuft (und wegen der NP-Vollständigkeit des SAT-Problems, die im dritten Semester in BFS behandelt wird, wird sich das wahrscheinlich, d.h. wenn P≠NP, grundsätzlich nicht vermeiden lassen), aber in der Praxis wesentlich schneller ist als das bisher verwendete Wahrheitstafelverfahren: das sogenannte Resolutionsverfahren. Dieses Verfahren erwartet die Eingabeformel in einem besonderen Format, der sogenannten *konjunktiven Normalform*, die wir in zwei Schritten einführen.

3.1 Negationsnormalform (NNF)

Die ab jetzt betrachteten Normalformen sind dadurch definiert, dass sie eine bestimmte Reihenfolge der logischen Operatoren bei der Traversierung des Syntaxbaums einer Formel von der Wurzel zu den Blättern festlegen. Wir behandeln zunächst nur die Negation, von der wir verlangen, dass sie hierbei zuletzt kommt; da sich aufeinanderfolgende Negationen aufheben, läuft dies darauf hinaus, dass es Negationen nur unmittelbar vor Blättern, also Atomen, geben darf.

Wir wollen für Zwecke der Normalformbildung Implikation als mittels $\neg$ und $\vee$ (also per $\phi \to \psi \equiv \neg\phi \vee \psi$) kodiert ansehen, während wir $\top$ und $\vee$ weiterhin als eigenständige Operationen behandeln.

Definition 3.1 (Negationsnormalform)**.** Eine aus Atomen sowie $\neg, \wedge, \vee, \bot, \top$ gebildete Formel ϕ ist in *Negationsnormalform (NNF)* (oder *eine NNF*), wenn die Negation $\neg$ in ϕ nur direkt vor Atomen vorkommt. Eine NNF χ ist *NNF von* ϕ, wenn $\chi \equiv \phi$.

Beispiel 3.2 (Terme in NNF)**.** Die Formel $\neg A \vee (B \vee \neg C)$ ist in NNF, $\neg(A \vee \neg B)$ dagegen nicht.

Lemma 3.3. *Jede Formel hat eine NNF.*

Beweis. Auf eine gegebene Formel und deren Teilformeln wende man so lange wie möglich die Äquivalenzumformungen

$$\begin{aligned} \neg\neg\phi &\equiv \phi \\ \neg(\phi \wedge \psi) &\equiv \neg\phi \vee \neg\psi \\ \neg(\phi \vee \psi) &\equiv \neg\phi \wedge \neg\psi \\ \neg\top &\equiv \bot \\ \neg\bot &\equiv \top \end{aligned}$$

von links nach rechts an. □

Beispiel 3.4. Die Formel $\neg((A \vee \neg B) \wedge C)$ bringen wir nach der im vorigen Beweis angegebenen Methode folgendermaßen in NNF:

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

Man beachte dabei, dass wir im zweiten und dritten Schritt Teilformeln umformen (nämlich im zweiten Schritt die Teilformel $\neg(A \vee \neg B)$ und im dritten Schritt die Teilformel $\neg\neg B$).

3.2 Konjunktive Normalformen

Wir wollen nun zusätzlich darauf bestehen, dass bei Traversierung des Syntaxbaums einer Formel von der Wurzel zu einem Blatt stets die Konjunktionen vor den Disjunktionen kommen (und weiterhin zuletzt ggf. die Negation); etwas anders gesagt sollen also die Konjunktionen „oben“ liegen.

Formal fassen wir dies wie folgt:

Definition 3.5.

1. Ein *Literal* ist entweder ein Atom A oder die Negation $\neg A$ eines Atoms $A \in \mathscr{A}$.
2. Eine *Klausel* ist eine endliche Disjunktion

   $$L_1 \vee \cdots \vee L_n$$

   von Literalen $L_1, \ldots, L_n$, mit $n \geqslant 0$. Hierbei verstehen wir die leere Disjunktion ($n = 0$) als Falsum $\bot$.

3. Eine *konjunktive Normalform (CNF)* ist eine endliche Konjunktion

$$C_1 \wedge \cdots \wedge C_m$$

von Klauseln $C_1, \ldots, C_m$, mit $m \geqslant 0$. Hierbei verstehen wir die leere Konjunktion ($m = 0$) als Verum $\top$.

Eine CNF ist demnach eine Konjunktion von Disjunktionen von Literalen, hat also die allgemeine Form

$$\bigwedge_{i=1}^{n} (\bigvee_{j=1}^{k_i} L_{ij}),$$

wobei wir in Analogie zum Summenzeichen $\sum$ mit $\bigwedge$ und $\bigvee$ die Konjunktion bzw. Disjunktion aller Formeln im jeweiligen Indexbereich schreiben (z.B. ist $\bigvee_{i=1}^{n} \phi_i = \phi_1 \vee \cdots \vee \phi_n$, so wie in der Arithmetik $\sum_{i=1}^{n} x_i$ die Summe $x_1 + \cdots + x_n$ bezeichnet).

Meistens, insbesondere zum Zwecke der Repräsentation im Rechner, verwenden wir eine alternative Darstellung von CNFs als Mengen, d.h. wir abstrahieren in Konjunktionen und Disjunktionen von der Reihenfolge und Wiederholungen (unter Verwendung von Assoziativität, Kommutativität und Idempotenz von $\wedge$ und $\vee$):

Klauseln sind endliche Mengen von Literalen, d.h. die Disjunktion $L_1 \vee \cdots \vee L_n$ wird repräsentiert als die Menge $\{L_1, \ldots, L_n\}$. Die leere Klausel repräsentiert $\bot$ und wird auch als $\square$ notiert.

CNFs sind endliche Mengen von Klauseln, d.h. die Konjunktion $C_1 \wedge \cdots \wedge C_m$ wird repräsentiert als die Menge $\{C_1, \ldots, C_m\}$. Die leere CNF repräsentiert $\top$.

In dieser Schreibweise wird also Erfülltheit einer CNF ϕ durch eine Wahrheitsbelegung $\kappa \colon \mathscr{A} \to 2$ beschrieben durch

$$\kappa \models \phi \iff \forall C \in \phi.\, \exists L \in C.\, \kappa \models L.$$

Definition 3.6 (CNF einer Formel)**.** Sei ϕ eine Formel. Eine CNF ϕ' heißt *CNF von* ϕ, wenn $\phi \equiv \phi'$.

Lemma 3.7. *Jede Formel hat eine CNF.*

Beweis. Auf eine gegebene Formel, o.E. bereits in NNF, und ihre Teilformeln wende man so lange wie möglich die Äquivalenzumformung

$$(\phi \wedge \psi) \vee \chi \equiv (\phi \vee \chi) \wedge (\psi \vee \chi)$$

sowie die symmetrische Variante $\chi \vee (\phi \wedge \psi) \equiv (\chi \vee \phi) \wedge (\chi \vee \psi)$ an, jeweils von links nach rechts. $\square$

Bemerkung 3.8. Mit *Umformung* meinen wir wie angedeutet sowohl bei der NNF als auch bei der CNF eventuell auch Umformungsschritte tief in der Formel, wie $\phi \wedge \neg\neg\psi \equiv \phi \wedge \psi$. Wir haben die Frage, ob jede Sequenz von solchen Umformugsschritten terminiert, also in einer Normalform des jeweiligen Typs endet, in den obigen Beweisen größzügig übergangen. Im Fall der NNF ist dies relativ leicht zu sehen (wie?), im Fall der CNF eher nicht; in *Theorie der Programmierung* werden Methoden behandelt, die auch in diesem Fall einen relativ einfachen Terminierungsbeweis ermöglichen.

Das Problem an der CNF ist, dass, wenn die (genauer: eine) CNF von ϕ n Klauseln hat und die CNF von ψ k Klauseln, dann die CNF von $\phi \vee \psi$ im allgemeinen $n \cdot k$ Klauseln hat, was bei längeren Disjunktionen einen exponentiellen Zuwachs in der Größe bewirkt. Z.B. hat die CNF von

$$(A_1 \wedge B_1) \vee \ldots \vee (A_n \wedge B_n)$$

2^n Klauseln (dieses Phänomen ist vom Ausmultiplizieren von arithmetischen Ausdrücken her bekannt: Der Ausdruck $(x_1 + y_1) \cdot (x_2 + y_2) \cdot \ldots \cdot (x_n + y_n)$ hat nach Ausmultiplzieren 2^n Summanden). Durch Einführung zusätzlicher Atome lässt sich eine polynomielle Größe der CNF sicherstellen; dann ist aber eben wegen der zusätzlichen Atome die so gewonnene CNF einer Formel ϕ nur noch *erfüllbarkeitsäquivalent* zu ϕ, d.h. sie ist *erfüllbar* genau dann, wenn ϕ erfüllbar ist, aber i.a. nicht mehr logisch äquivalent zu ϕ.

Beispiel 3.9. Wir bilden die (genauer: eine) CNF von $(\neg B \wedge C) \vee (\neg A \wedge B)$ durch Umformung wie folgt (bei Verwendung von etwas mehr Klammern als nötig):

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

In Mengenschreibweise wird dies

$$\big\{\{\neg B, \neg A\}, \{\neg B, B\}, \{C, \neg A\}, \{C, B\}\big\}.$$

3.3 Resolution

Das Resolutionsverfahren ist ein Algorithmus zur Entscheidung der Erfüllbarkeit einer CNF, hier dargestellt als eine Klauselmenge ϕ. Er beruht auf der *Resolutionsregel*

$$(\text{Res}) \; \frac{C_1 \cup \{A\} \qquad C_2 \cup \{\neg A\}}{C_1 \cup C_2}. \tag{1}$$

Diese Regel wird im Resolutionsverfahren auf die Menge der Klauseln in einer CNF angewendet, in dem Sinne, dass, wenn in der Menge alle Prämissen, d.h. die Klauseln über dem Strich, enthalten sind, die Konklusion, also die Klausel $C_1 \cup C_2$, hinzugefügt werden darf; letztere heißt *Resolvente* von $C_1 \cup \{A\}$ und $C_2 \cup \{\neg A\}$.

Die Resolutionsregel ist im folgenden Sinn korrekt:

Lemma 3.10. *Für Klauseln C_1, C_2 und $A \in \mathscr{A}$ gilt $C_1 \cup \{A\}, C_2 \cup \{\neg A\} \models C_1 \cup C_2$.*

Beweis. Sei $\kappa \models C_1 \cup \{A\}, \kappa \models C_2 \cup \{\neg A\}$. Wir führen eine Fallunterscheidung nach $\kappa(A)$ durch:

$\kappa(A) = \top$: Dann $\kappa \models C_2$, also $\kappa \models C_1 \cup C_2$

$\kappa(A) = \bot$: Dann $\kappa \models C_1$, also $\kappa \models C_1 \cup C_2$ □

Auf der Resolutionsregel basiert der folgende Algorithmus zur Entscheidung der Erfüllbarkeit einer CNF:

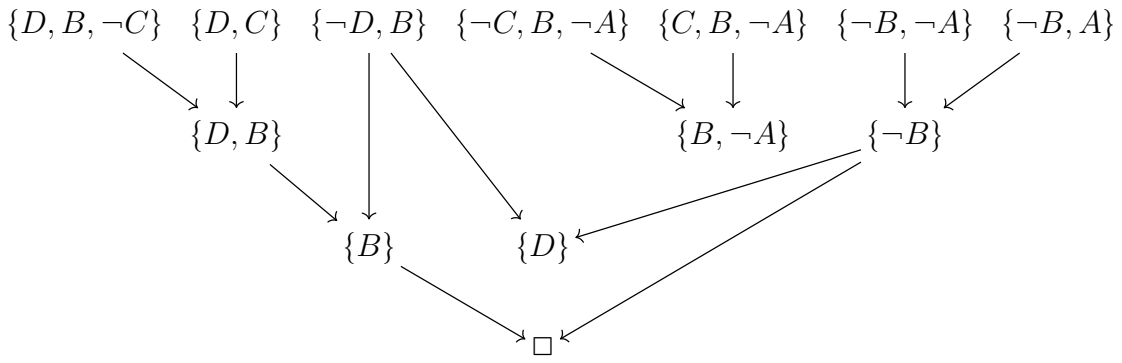


Abbildung 7: Mögliche Ausführung des Resolutions-Algorithmus

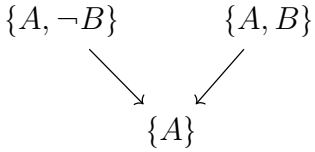
Algorithmus 3.11 (Resolutionsverfahren)**.** Eingabe: CNF ϕ_0. Ausgabe: „ja“, wenn ϕ erfüllbar, „nein“ sonst.

Verwende globale Variable ϕ vom Typ CNF:

1. Initialisiere $\phi := \phi_0$
2. If $\square \in \phi$ then return „nein“.
3. Suche $C_1 \cup \{A\}, C_2 \cup \{\neg A\} \in \phi$, $C_1 \cup C_2 \notin \phi$. Falls keine solchen C_1, C_2, A existieren, return „ja“.
4. $\phi := \phi \cup \{C_1 \cup C_2\}$.
5. Gehe zu Schritt 2.

Beispiel 3.12. Betrachte die folgenden Klauselmengen:

1. Für $\phi_0 = \big\{\{D, B, \neg C\}, \{D, C\}, \{\neg D, B\}, \{\neg C, B, \neg A\}, \{C, B, \neg A\}, \{\neg B, \neg A\}, \{\neg B, A\}\big\}$ ist in Abbildung 7 eine mögliche Ausführung des Resolutions-Algorithmus zu sehen. Mit dem Schritt, in dem die leere Klausel $\square$ erzeugt wird, gibt der Algorithmus die Antwort „nein“ zurück, die Klauselmenge ist also unerfüllbar. Ein paar Beobachtungen:
   - Da der Algorithmus immer nur Klauseln ergänzt, spielt die genaue zeitliche Abfolge keine Rolle. In der Visualisierung vermerken wir lediglich, aus welchen Klauseln sich die Resolventen ergeben.
   - Da der Algorithmus die zu resolvierenden Klauseln beliebig wählt, können manche (usprünglich gegebenen oder erzeugten) Klauseln unverwendet bleiben (z.B. $\{B, \neg A\}$).
   - Da die Klauseln selbst Mengen sind, erzeugt der erste Schritt links oben die Menge:
   $$\{D, B\} \cup \{D\} = \{D, B, D\} = \{D, B\}$$
2. Für $\phi_0 = \big\{\{A, \neg B\}, \{A, B\}\big\}$ kann der Resolutions-Algorithmus nur einen Schritt durchführen:



Nachdem nun keine miteinander resolvierbaren Klauseln mehr in der Menge $\phi = \big\{\{A\}, \{A, \neg B\}, \{A, B\}\big\}$ existieren, bricht der Algorithmus mit der Antwort ja ab. In der Tat ist die Formel $(A \vee \neg B) \wedge (A \vee B)$ erfüllbar, beispielsweise durch die Wahrheitsbelegung $\kappa = [A \mapsto \top, B \mapsto \bot]$.

Wir beweisen die totale Korrektheit des Algorithmus:

Satz 3.13 (Terminierung)**.** *Der Resolutions-Algorithmus terminiert für alle Eingaben.*

Beweis. Der Algorithmus arbeitet ausschließlich auf der Menge der in der Eingabe ϕ_0 vorkommenden Literale, d.h. die Anwendung der Regel führt nie neue Literale ein. Formaler: Für eine CNF ϕ in Mengendarstellung ist $\bigcup \phi$, also die Vereinigung der Klauseln in ϕ, einfach die Menge der in ϕ vorkommenden Literale. Man sieht leicht, dass die Eigenschaft $\bigcup \phi = \bigcup \phi_0$ eine Invariante des Algorithmus ist. In Mengendarstellung gibt es nur endlich viele unterschiedliche Klauseln über $\bigcup \phi_0$, nämlich $2^{|\bigcup \phi|}$. Insbesondere kann also nur endlich oft eine Klausel hinzugefügt werden, die nicht bereits in der aktuellen CNF enthalten ist. □

Satz 3.14 (Korrektheit)**.** *Falls der Algorithmus „nein“ antwortet, ist ϕ_0 unerfüllbar.*

Beweis. Per Korrektheit der Resolutionsregel (Lemma 3.10) folgen während der Ausführung des Algorithmus alle jeweils in ϕ enthaltenen Klauseln aus der Eingabe ϕ_0 (diese Eigenschaft ist also wiederum eine Invariante des Algorithmus). Wenn $\square$, also nach unseren Konventionen $\bot$, eine dieser Klauseln ist, ist somit die Eingabe unerfüllbar. □

Wie für natürliches Schließen ist auch bei Resolution der Vollständigkeitsbeweis aufwändiger, weil wir hier ein konkretes Modell, also eine erfüllende Wahrheitsbelegung, konstruieren müssen:

Satz 3.15 (Vollständigkeit)**.** *Falls der Algorithmus „ja“ antwortet, ist ϕ_0 erfüllbar.*

Beweis. Wir sagen, dass ϕ *resolutionsabgeschlossen (ra.)* ist, wenn mit $C_1 \cup \{A\}, C_2 \cup \{\neg A\} \in \phi$ stets $C_1 \cup C_2 \in \phi$ gilt. Der Algorithmus bricht also genau dann bei Schritt 3 ab, wenn ϕ ra. ist. Sei also ϕ ra., $\square \notin \phi$. Zu zeigen ist dann, dass ϕ erfüllbar ist. Wir verwenden Induktion über $|\operatorname{At}(\phi)|$:

Induktionsanfang: Ohne Atome kann ϕ als Klauselmenge nur leer sein, da die einzige Klausel ohne Atome, nämlich $\square$, nach Voraussetzung nicht in ϕ enthalten ist. Die leere Klauselmenge repräsentiert nach unseren Konventionen die Formel $\top$ und ist damit natürlich erfüllbar.

Induktionsschritt: Wähle $A \in \operatorname{At}(\phi)$. Sei

$$\phi/A = \{\, D \backslash \{\neg A\} \mid A \notin D \in \phi \,\}$$
$$\phi/\neg A = \{\, D \backslash \{A\} \mid \neg A \notin D \in \phi \,\}.$$

(Kommentar: Damit ist ϕ/A logisch äquivalent zu ϕ, wenn wir A annehmen, entsprechend für $\phi/\neg A$ und $\neg A$; formal: $A \models \phi \leftrightarrow \phi/A$ und $\neg A \models \phi \leftrightarrow \phi/\neg A$. Es gilt $\operatorname{At}(\phi/A) \not\ni A \notin \operatorname{At}(\phi/\neg A)$.)

Dann ist entweder $\square \notin \phi/A$ oder $\square \notin \phi/\neg A$: sonst $\{\neg A\} \in \phi \ni \{A\}$; da ϕ ra., würde folgen $\square \in \phi$, Widerspruch.

Ohne Einschränkung[1] sei $\square \notin \phi/A$.

[1] „ohne Einschränkung“ ist gleichbedeutend mit (aber kürzer als) „ohne Beschränkung der Allgemeinheit“. Konkret heißt es hier, dass der Beweis für den Fall $\square \notin \phi/\neg A$ genauso geführt werden kann und deshalb weggelassen wird.

Nun ist auch ϕ/A ra.: Sei $E_1 \cup \{B\}, E_2 \cup \{\neg B\} \in \phi/A$. Da dann $B \neq A$, haben die E_i nach Definition von ϕ/A die Form

$$E_1 = C_1 \backslash \{\neg A\}, \quad E_2 = C_2 \backslash \{\neg A\}$$

für geeignete C_1, C_2 mit $C_1 \cup \{B\}, C_2 \cup \{\neg B\} \in \phi$, $A \notin C_1 \cup \{B\}$ und $A \notin C_2 \cup \{\neg B\}$. Da ϕ ra. ist, folgt $C_1 \cup C_2 \in \phi$; da $A \notin C_1 \cup C_2$, folgt $E_1 \cup E_2 = (C_1 \cup C_2)\backslash\{\neg A\} \in \phi/A$.

Da $|\operatorname{At}(\phi/A)| < |\operatorname{At}(\phi)|$, folgt nach Induktionsvoraussetzung, dass ϕ/A erfüllbar ist, d.h. es existiert κ mit $\kappa \models \phi/A$. Wir behaupten nun, dass $\kappa[A \mapsto \top] \models \phi$, womit dann ϕ wie verlangt erfüllbar ist. Sei also $D \in \phi$; zu zeigen ist $\kappa[A \mapsto \top] \models D$.

Fall 1: $A \in D$ ✓

Fall 2:

$$\begin{aligned} A \notin D &\Rightarrow D\backslash\{\neg A\} \in \phi/A \\ &\Rightarrow \kappa \models D\backslash\{\neg A\} \\ &\Rightarrow \kappa[A \mapsto \top] \models D\backslash\{\neg A\} \qquad (A \notin \operatorname{At}(D\backslash\{\neg A\}), \text{ Lemma 2.28}) \\ &\Rightarrow \kappa[A \mapsto \top] \models D. \end{aligned}$$

□

Insgesamt haben wir die *totale Korrektheit* des Resolutionsalgorithmus gezeigt, d.h. er gibt stets eine Antwort, und die ist stets richtig.

4 Prädikatenlogik erster Stufe

Ein *Prädikat* ist eine Eigenschaft, die Individuen haben (oder nicht haben) können, d.h. eine Aussage mit Parametern, für die Individuen einzusetzen sind; Beispiel: $\mathsf{positiv}(n)$ oder $\mathsf{verheiratetMit}(\mathsf{x}, \mathsf{y})$. Diese Parametrisierung unterscheidet Prädikate von den Atomen der Aussagenlogik, die nur schlechthin wahr oder falsch sein können (wie z.B. $\mathsf{esRegnet}$). Logiken, die über Prädikate sprechen, heißen *Prädikatenlogiken*. Sie beinhalten typischerweise auch die Möglichkeit, Aussagen zu *quantifizieren*, etwa dahingehend, ob sie für *alle* möglichen Belegungen von Variablen oder für *mindestens eine* Belegung gelten. Wir beschäftigen uns hier nur mit dem Fall, in dem die betreffenden Variablen für Individuen stehen, also mit der Prädikatenlogik *erster Stufe*. (Es gibt auch Logiken, die Variablen für komplexere Gebilde, z.B. für Mengen von Individuen, zulassen; man spricht dann von Prädikatenlogik *höherer Stufe*, im Beispielfall *zweiter Stufe*.) Gelegentlich wird einfach der Begriff „Prädikatenlogik“ verwendet, womit meist die Prädikatenlogik erster Stufe gemeint ist. Letztere wird auf Englisch mit dem deutlich kürzeren Begriff *first order logic* bezeichnet, abgekürzt *FOL*; der Knappheit halber werden wir oft diese Abkürzung verwenden.

4.1 Signaturen

Die konkrete Syntax und Semantik der Prädikatenlogik hängt davon ab, über welche konkreten Prädikate Aussagen getroffen und bewiesen werden sollen. Analog zur Atommenge $\mathscr{A}$ in Aussagenlogik ist bei Prädikatenlogik die Signatur Σ der Vorrat an Symbolen, aus denen wir komplexere Aussagen bauen können:

Definition 4.1. Eine *Signatur* $\Sigma = (P_\Sigma, F_\Sigma, \mathrm{ar})$ besteht aus

- einer Menge P_Σ von *Prädikatensymbolen*;
- einer (der Einfachheit halber zu P_Σ disjunkten) Menge F_Σ von *Funktionssymbolen*;
- einer *Stelligkeitsfunktion* $\mathrm{ar}\colon P_\Sigma \cup F_\Sigma \to \mathbb{N}$, die jedem Funktions- oder Prädikatensymbol s eine endliche *Stelligkeit* $\mathrm{ar}(s) \geqslant 0$ zuordnet. Statt $s \in P_\Sigma \cup F_\Sigma$ mit $\mathrm{ar}(s) = n$ schreiben wir kurz $s/n \in \Sigma$.

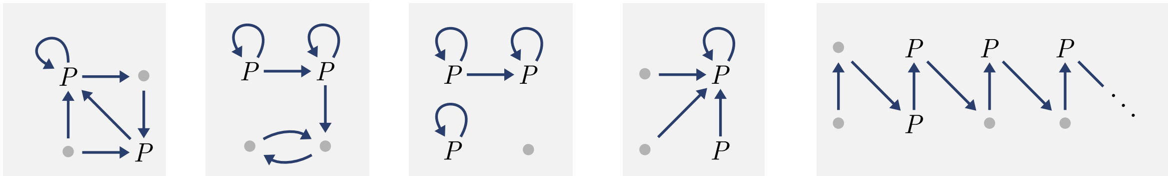
Wenn keine Gefahr von Missverständnissen besteht, bezeichnen wir Prädikats- und Funktionssymbole auch kurz als *Prädikate* bzw. *Funktionen*. Um die Unterscheidung zwischen den beiden Symbolarten sicherstellen, werden wir Funktionssymbole mit Kleinbuchstaben und Prädikatensymbole mit Großbuchstaben bezeichnen.

1. Ein n-stelliges Prädikat $P/n \in \Sigma$ beschreibt eine *Eigenschaft* von n Individuen (in gegebener Reihenfolge). Beispiele:
   - $\mathsf{Wertvoll}/1$ ist ein einstelliges Prädikat.
   - $\mathsf{Mag}/2$ ist ein zweistelliges Prädikat, das z. B. angibt, ob die erste Person die zweite mag.
   - Da jedes *nullstellige* Prädikat $\mathsf{A}/0$ eine Aussage über 0 Individuen trifft (z. B. $\mathsf{EsRegnet}$) und damit entweder schlicht war oder schlicht falsch ist, entsprechen die nullstelligen Prädikate gerade den Atomen der Aussagenlogik.
2. Ein n-stelliges Funktionssymbol $f/n \in \Sigma$ bezeichnet *Individuen*, die durch n Individuen (in gegebener Reihenfolge) bestimmt sind. Beispiele:
   - $\mathsf{chefin}/1$ ist eine einstellige Funktion, die z.B. jede Person auf ihre Vorgesetzte abbildet (ggf. auf sich selbst).
   - Ein nullstelliges Funktionssymbol $c/0 \in \Sigma$ bezeichnet also einfach ein Individuum und heißt auch *Konstante*.

Die realweltlichen Erklärungen dieser Beispiele dienen nur dem intuitiven Verständnis und sind keine eindeutigen Definitionen. Allgemein hängt der Wahrheitsgehalt (wissenschaftlicher ausgedrückt die Erfülltheit) von Formeln in Prädikatenlogik davon ab, wie man die Symbole in einem konkreten Modell *interpretiert*. Ein solches *Modell* besteht aus einer nichtleeren Menge an Individuen, über die wir sprechen (z. B. die natürlichen Zahlen oder die Menschen in der EU), und gibt zu allen Symbolen aus der Signatur eine konkrete Interpretation an (eine formale Definition folgt in Def. 4.11).

Beispiel 4.2. Für die Signatur aus einem einstelligen und einem zweistelligen Prädikat $\Sigma = \{P/1, R/2\}$ sind zwei Beispielmodelle in Abbildung 8 visualisiert. In diesen Modellen sind die Individuen als Knoten dargestellt und die Interpretation der Prädikatensymbole als Knotenannotation und Kanten. Die Interpretation des Prädikatensymbols $P/1$ in einem Modell besteht aus einer Teilmenge der Individuen, und die Interpretation von $R/2$ ist eine zweistellige Relation (durch gerichtete Kanten dargestellt).

Das Modell in 8a besteht aus vier Individuen; zwei davon erfüllen P, die anderen zwei nicht. Wenn man P als „prominent“ versteht und R als „hat fotografiert“, dann gilt in diesem Modell beispielsweise, dass jede schon einmal eine Prominente fotografiert hat. Dieselbe Aussage ist hingegen in den Modellen 8b und 8c nicht erfüllt, denn in beiden gibt es (mindestens) ein



(a) (b) (c) (d) (e)

Abbildung 8: Beispiel-Modelle für $\Sigma = \{P/1, R/2\}$

Individuum, das noch nie eine Prominente fotografiert hat. In 8d gibt es sogar eine Prominente, die von allen (inklusive sich selbst!) schon einmal fotografiert worden ist. Modelle müssen nicht zwingend endlich sein: in 8e sind die Individuen die natürlichen Zahlen $\mathbb{N}$ (inklusive 0), R setzt jede natürliche Zahl mit ihrem Nachfolger in Relation, und P bezeichnet die Primzahlen.

4.2 Syntax

Die Syntax der Prädikatenlogik bringt sowohl deutlich mehr definitorischen Aufwand als die der Aussagenlogik als auch etliche technische Subtilitäten mit sich.

Definition 4.3 (Σ-Terme)**.** Wir unterstellen einen festen Vorrat V an Variablen $x, y \dots$. Die Menge T_Σ der Σ-Terme ist induktiv gegeben durch folgende Regeln:

- Jede Variable $x \in V$ ist ein Term x.
- Wenn $f/n \in \Sigma$ ein Funktionssymbol ist und $E_1, \dots, E_n$ Terme sind, dann ist $f(E_1, \dots, E_n)$ ein Term.

Beispiel 4.4. Für die Signatur $\Sigma = \{c/0, f/1, g/2, P/1, R/2\}$ (also nach obiger Konvention Funktionen c, f, g und Prädikate P, R) erhalten wir beispielsweise diese Terme (mittels Variablen $x, y \in V$):

$$x \qquad y \qquad c \qquad f(f(x)) \qquad g(c,c) \qquad g(f(x),c) \qquad g(x,f(y)) \qquad g(f(c),f(f(c)))$$

Man beachte insbesondere, dass die Relationssymbole nicht in Termen verwendet werden können.

Definition 4.5 (Syntax der Prädikatenlogik erster Stufe)**.** Die Syntax von *Formeln*, oder genauer die Menge der Σ-*Formeln*, ist induktiv durch folgende Regeln definiert:

- Wenn E, D Terme sind, dann ist $E = D$ eine Formel, eine *Gleichung*.
- Wenn $E_1, \dots, E_n$ Terme sind und $P/n \in \Sigma$, dann ist $P(E_1, \dots, E_n)$ eine Formel, eine *Prädikatenanwendung*.
- Wenn ϕ und ψ Formeln sind, dann sind $\neg\phi$, $\phi \wedge \psi$, $\phi \vee \psi$, $\phi \to \psi$ und $\phi \leftrightarrow \psi$ Formeln.
- $\bot$ und $\top$ sind Formeln.
- Wenn ϕ eine Formel ist und $x \in V$, dann sind $\forall x.\,\phi$ und $\exists x.\phi$ Formeln.

Gleichungen und Prädikatenanwendungen sind *atomare Formeln*. Wir schreiben $\mathscr{F}_{\mathsf{FO}}(\Sigma)$ oder einfach $\mathscr{F}_{\mathsf{FO}}$ für die Menge der Σ-Formeln.

Wie in der Aussagenlogik können wir die Konnektive $\vee$, $\rightarrow$, $\leftrightarrow$ mittels $\neg$ und $\wedge$ definieren; außerdem wird sich (nach Einführung der Semantik) $\exists x.\, \phi$ als logisch äquivalent zu $\neg\forall x.\, \neg\phi$ erweisen. Anders als im Fall der Aussagenlogik können wir $\top$ und $\bot$ selbst dann durch die anderen Konnektive ausdrücken, wenn die Signatur leer ist (wie?). In der Tat führen wir rekursive Definitionen im folgenden nur für eine entsprechend eingeschränkte Syntax durch, die nur aus atomaren Formeln, $\neg$, $\wedge$ und $\forall$ beinhaltet; die sich durch Kodierung ergebende Erweiterung solcher Definitionen auf die volle Syntax ist zum Teil Gegenstand der Übungen.

Die Sprechweise für $\forall x.\, \phi$ ist „für alle x gilt ϕ“, und für $\exists x.\, \phi$ „es existiert ein x, so dass ϕ“ oder „es existiert ein x mit ϕ“.

Zur Einsparung von Klammern vereinbaren wir, dass der Geltungsbereich eines Quantors $\forall x$ oder $\exists x$ immer so weit nach rechts reicht wie möglich (also entweder bis zum Ende der Formel oder bis zur ersten schließenden Klammer, deren zugehörige öffnende Klammer links vom betreffenden Quantor liegt). Bei aufeinanderfolgenden Quantoren wie in $\forall x \exists y.\, \phi$ lassen wir Punkte zwischen den Quantoren weg. Mehrere aufeinanderfolgende Quantoren der gleichen Art fassen wir gelegentlich zu einem zusammen; z.B. schreiben wir auch $\forall x, y.\, \phi$ statt $\forall x \forall y.\, \phi$.

Beispiel 4.6 (Wissenswertes über Erlangen)**.** Der Satz „Jede:r Erlanger:in mag eine:n andere:n Erlanger:in“ wird in Logik erster Stufe ausgedrückt als

$$\forall x.\, E(x) \rightarrow \exists y.\, E(y) \wedge L(x, y)$$

(mit E für „Erlanger:in“ und L für „likes“). Man beachte hierbei, dass gemäß obiger Konvention die Geltungsbereiche beider Quantoren bis zum Ende der Formel reichen. Der etwas unwahrscheinlicher klingende Satz „Es gibt eine:n Erlanger:in, den/die alle Erlanger:innen mögen“ dagegen wird formalisiert als

$$\exists y.\, E(y) \wedge \forall x.\, E(x) \rightarrow L(x, y)$$

(auch hier reichen wieder die Geltungsbereiche beider Quantoren bis zum Ende der Formel). Diese Beispiele illustrieren zum einen die typische und meist allein sinnvolle Kombination von $\forall$ mit $\rightarrow$ und von $\exists$ mit $\wedge$ gemäß den klassischen *aristotelischen Formen*:

$$\begin{aligned} \text{Alle Ps sind Qs:} \quad & \forall x.\, P(x) \rightarrow Q(x) \\ \text{Einige Ps sind Qs:} \quad & \exists x.\, P(x) \wedge Q(x). \end{aligned}$$

Zum anderen sieht man hier, obwohl wir natürlich noch keinerlei formale Bedeutung von Formeln diskutiert haben, schon anhand der Lesart, dass man Quantoren $\forall x$ und $\exists y$ im allgemeinen keineswegs bedeutungserhaltend vertauschen kann.

Wir sehen die Variable x in $\forall x.\, \phi$ als durch den Quantor *gebunden* an, d.h. ihre Lebensdauer ist begrenzt auf die Unterformel ϕ. Wenn $\forall x.\, \phi$ Teil einer größeren Formel ist, die außerhalb von $\forall x.\, \phi$ die Variable x bereits erwähnt, so wird die äußere Verwendung von x durch den Quantor *verschattet*. Diese Phänomene sind letztlich ähnlich der Verwendung lokaler Variablen in üblichen Programmiersprachen.

Formal definieren wir die *freien Variablen* in einem Term oder einer Formel wie folgt.

Definition 4.7 (Freie Variable)**.** Die Mengen $FV(E)$ und $FV(\phi)$ der freien Variablen eines

Terms E bzw. einer Formel ϕ sind rekursiv definiert durch

$$\begin{aligned}
FV(x) &= \{x\} \\
FV(f(E_1, \ldots, E_n)) &= \bigcup_{i=1}^{n} FV(E_i) \\
FV(E = D) &= FV(E) \cup FV(D) \\
FV(P(E_1, \ldots, E_n)) &= \bigcup_{i=1}^{n} FV(E_i) \\
FV(\neg\phi) &= FV(\phi) \\
FV(\phi \wedge \psi) &= FV(\phi) \cup FV(\psi) \\
FV(\forall x.\, \phi) &= FV(\phi) \backslash \{x\}
\end{aligned}$$

(wobei wir wie angekündigt nur die Klauseln für die reduzierte Formelgrammatik angeben). Z.B. gilt $FV(P(x, y) \wedge \forall y.\, \forall z.\, Q(y, w, z)) = \{x, y, w\}$.

Definition 4.8. Eine Formel ϕ heißt *geschlossen*, wenn $FV(\phi) = \varnothing$, d.h. wenn alle in ϕ vorkommenden Variablen durch Quantoren gebunden sind.

Z.B. ist die Formel $\forall x.\, E(x) \to \exists y.\, E(y) \wedge L(x, y)$ aus unserem obigen Beispiel geschlossen.

Wir sehen die Namen gebundener Variablen als unwichtig an und betrachten daher zwei Formeln, die sich nur durch die Namen gebundener Variablen unterscheiden (also *α-äquivalent* sind) als im wesentlichen gleich; z.B. sind $\forall x.\, x = z$ und $\forall y.\, y = z$ (nicht aber $\forall z.\, z = z$) im wesentlichen dieselbe Formel.

Definition 4.9 (Substitution, Umbenennung)**.** Eine *Substitution* ist eine Abbildung σ, die jeder Variablen x einen Term $\sigma(x)$ zuordnet, so dass die Menge

$$\mathrm{Dom}(\sigma) = \{x \mid \sigma(x) \neq x\}$$

endlich ist, d.h. σ verändert nur eine endliche Anzahl an Variablen. $\mathrm{Dom}(\sigma)$ („Domain“, „Bereich“ von σ) ist die Menge aller Variablen, mit denen σ „etwas tut“, d.h. denen σ einen anderen Term zuordnet. Mit $E\sigma$ bezeichnen wir die Anwendung von σ auf den Term E. Dabei wird jede Variable wie in σ angegeben ersetzt: $x\sigma = \sigma(x)$, $f(E_1, \ldots, E_n)\sigma = f(E_1\sigma, \ldots, E_n\sigma)$. Die Anwendung $\phi\sigma$ einer Substitution σ auf eine Formel ϕ ist wegen eventuell gebundener Variablen komplizierter:

$$\begin{aligned}
(E = D)\sigma &= (E\sigma = D\sigma) \\
P(E_1, \ldots, E_n)\sigma &= P(E_1\sigma, \ldots, E_n\sigma) \\
(\neg\phi)\sigma &= \neg(\phi\sigma) \\
(\phi \wedge \psi)\sigma &= \phi\sigma \wedge \psi\sigma \\
(\forall x.\, \phi)\sigma &= \forall y.\, \phi\sigma',
\end{aligned}$$

wobei $\sigma'(x) = y$, $\sigma'(z) = \sigma(z)$ für jedes $z \neq x$, und wobei ferner y so gewählt ist, dass $y \notin FV(\sigma(z))$ für alle $z \in FV(\forall x.\, \phi)$ („y ist frisch“). (Falls x diese Bedingung bereits erfüllt, kann für y auch x gewählt werden.) (Wiederum geben wir hier nur die Klauseln für die reduzierte Formelgrammatik an.)

Eine Substitution σ heißt eine *Umbenennung*, wenn $\sigma(x)$ für alle x eine Variable ist.

Die Substitution $[E_1/x_1, \ldots, E_n/x_n]$ ist die Substitution σ mit

$$\sigma(x) = \begin{cases} E_i & \text{wenn } x = x_i \\ x & \text{sonst} \end{cases}$$

Die Identität wird denotiert durch die leere Substitution $[\,]$, also $E\,[\,] \equiv E$. Für Substitutionen σ, τ bezeichnet $\sigma\tau$ die Substitution mit $(\sigma\tau)(x) = (\sigma(x))\tau$, d.h. die Substitution, die entsteht, wenn man zuerst σ und dann τ ausführt. (Achtung: Das ist gerade andersherum als bei der üblichen Notation für Funktionskomposition, in der $f \circ g$ die Funktion bezeichnet, die entsteht, indem man zuerst g und dann f anwendet. Dies ist der Tatsache geschuldet, dass die Anwendung einer Substitution σ auf eine Formel ϕ in Postfixnotation, also als $\phi\sigma$, geschrieben wird.)

Beispiel 4.10. Mit $\sigma = [add(z, y)/x, suc(x)/y]$ haben wir $\mathrm{Dom}(\sigma) = \{x, y\}$ und z.B.

$$add(add(x, z), add(x, y))\sigma = add(add(add(z, y), z), add(add(z, y), suc(x))).$$

Dieses Beispiel illustriert insbesondere, dass die Ersetzung von x und y durch σ definitionsgemäß *gleichzeitig* stattfindet; wenn man zuerst die Substitution $[add(z, y)/x]$ und anschließend die Substitution $[suc(x)/y]$ anwendet, kommt etwas anderes heraus, da dann auch im für x eingesetzten Term $add(z, y)$ die Variable y durch $suc(x)$ ersetzt wird.

Als weiteres Beispiel betrachten wir die Formel $\phi = \forall x.\, x \leqslant y$, wobei $\leqslant$ ein in Infixnotation geschriebenes binäres Prädikat ist; wenn wir uns x, y etc. (*nur für dieses Beispiel!!*) als Zahlen aus einem gegebenen Bereich vorstellen und $\leqslant$ als die übliche kleiner-gleich-Relation interpretieren, ist dies die Aussage, dass y die größte Zahl des gegebenen Zahlbereichs ist. Ferner sei σ die Substitution $[z/x, a(x, y)/y]$, wobei a eine zweistellige Funktion ist. Die Seitenbedingung in der Definition der Anwendung $\phi\sigma$ von σ auf ϕ verbietet in diesem Fall gerade, dass wir den Namen der allquantifizierten Variablen x unverändert lassen, da x nicht frisch im Sinne der Seitenbedingung ist: x kommt frei im Term $a(x, y)$ vor, den σ für die freie Variable y in $\forall x.\, x \leqslant y$ substituiert. Wenn wir uns über diese Beschränkung hinwegsetzen, ergibt sich denn auch eine Formel, die offenbar nicht im beabsichtigten Sinne eine Instanz von ϕ ist (formaler wird es gerade noch nicht, da wir noch keine Semantik definiert haben): Mit $\sigma'(x) = x$ und $\sigma'(y) = \sigma(y) = a(x, y)$ erhielten wir hier als Resultat der Substitution

$$\forall x.\, (x \leqslant y)\sigma' = \forall x.\, x \leqslant a(x, y),$$

also eine gänzlich unverwandte Aussage, die z.B., wenn wir wir a als Addition interpretieren, besagt, dass y nichtnegativ ist.

Wählen wir dagegen wie vorgeschrieben eine frische Variable, beispielsweise z (das ist wirklich frisch im verlangten Sinn: z kommt zwar im Term $\sigma(x)$ vor, aber x ist nicht frei in ϕ), so erhalten wir mit $\sigma'(x) = z$, $\sigma'(y) = \sigma(y) = a(x, y)$ korrekterweise

$$\phi\sigma = (\forall x.\, x \leqslant y)\sigma = \forall z.\, (x \leqslant y)\sigma' = \forall z.\, z \leqslant a(x, y),$$

also die Aussage, dass $a(x, y)$ die größte Zahl im gegebenen Zahlbereich ist; diese Aussage ist in der erwarteten Form eine Instanz der ursprünglichen Aussage $\forall x.\, x \leqslant y$.

4.3 Natürliches Schließen in Prädikatenlogik

Wir erweitern nun das System des natürlichen Schließens um Regeln für die neuen Ausdrucksmittel $\exists$, $\forall$ und $=$. Die Regeln für Gleichheit sind

$$(= I)\ \frac{}{E = E} \qquad (= E)\ \frac{\phi[E/x] \qquad E = D}{\phi[D/x]}.$$

Es überrascht eventuell zunächst, dass das schon genügt; man beweist hiermit aber eben andere erwartete Eigenschaften von Gleichheit, etwa Symmetrie

1	$E = D$	
2	$E = E$	(=I)
3	$D = E$	(=E) 2,1

und Transitivität

1	$E = D$	
2	$D = F$	
3	$E = F$	(=E) 1,2

Die Regeln für $\forall$ formalisieren die erwarteten Schlussweisen: Die $\forall$-Eliminationsregel lautet

$$(\forall \mathrm{E})\ \frac{\forall x.\, \phi}{\phi[E/x]}$$

für beliebig gewählte Terme E, also umgangssprachlich „Wenn ϕ für alle x gilt, dann gilt ϕ auch für ein gegebenes E“.

Die Einführungsregel liest sich umgangssprachlich „Wenn man ϕ für ein beliebiges c zeigen kann, dann gilt ϕ für alle x“, formal

$$(\forall \mathrm{I})\ \frac{\begin{array}{|l} \boxed{c} \\ \hline \vdots \\ \phi[c/x] \end{array}}{\forall x.\, \phi}$$

wobei c eine *frische* Konstante ist, d.h. außer in dem betreffenden Unterbeweis nirgendwo vorkommt (dies wird durch die Box um das c angedeutet). Durch diese Einschränkung an c wird der informelle Ausdruck „beliebig“ eingefangen: c kann dann insbesondere nicht in irgendwelchen gerade aktiven Annahmen vorkommen, d.h. der Unterbeweis in der Prämisse zeigt $\phi[c/x]$, ohne über c irgendwelche unzulässigen Voraussetzungen zu machen.

Die Einführungsregel für $\exists$ ist ebenso intuitiv:

$$(\exists \mathrm{I})\ \frac{\phi[E/x]}{\exists x.\, \phi}$$

In Worten: Wenn ϕ für einen Term E gilt, dann gibt es ein x, für das ϕ gilt – nämlich eben gerade E.

Etwas gewöhnungsbedürftiger ist die Eliminationsregel für $\exists$:

$$\dfrac{\exists x.\,\phi \qquad \begin{array}{|l} \boxed{c}\ \phi[c/x] \\ \hline \vdots \\ \psi \end{array}}{\psi}\ (\exists\mathrm{E})$$

wiederum mit der Seitenbedingung, dass c frisch ist; **dies bedeutet hier insbesondere, dass c nicht in ψ vorkommen darf**. In Worten: Wenn es ein x gibt, für das ϕ gilt, und wenn ich für beliebiges c eine Aussage ψ daraus folgern kann, dass ϕ für c gilt, dann gilt ψ. In natürlichsprachlichen Beweisen entspricht dies folgendem Vorgehen: Man hat Existenz eines Elements mit einer gegebenen Eigenschaft ϕ, nimmt sich also ein solches Element c her und argumentiert damit weiter; eine so hergeleitete Aussage ψ, die c nicht mehr erwähnt, gilt dann als ohne die Annahme, dass c ϕ erfüllt, hergeleitet.

Herleitung der Regeln für $\exists$ Als erste Anwendung der Regeln für $\forall$ können wir die Regeln für $\exists$ bei Kodierung von $\exists$ per $\exists x.\,\phi \equiv \neg\forall x.\,\neg\phi$ herleiten:

Beweis von ($\exists I$)

1	$\phi[E/x]$	
2	$\quad\forall x.\,\neg\phi$	
3	$\quad\neg\phi[E/x]$	($\forall$E, 2)
4	$\quad\bot$	($\bot$I 1, 3)
5	$\neg\forall x.\,\neg\phi$	($\neg$I 1-4)

Beweis von ($\exists E$)

1	$\neg\forall x.\,\neg\phi$	
2	$\quad\neg\psi$	
3	$\quad\boxed{c}$	
4	$\quad\quad\phi[c/x]$	
5	$\quad\quad\vdots$	
6	$\quad\quad\psi$	(Unterbeweis aus Prämisse)
7	$\quad\quad\bot$	($\neg$E) 2,6
8	$\quad\neg\phi[c/x]$	($\neg$I) 4–7
9	$\quad\forall x.\,\neg\phi$	($\forall$I) 3–8
10	$\quad\bot$	($\bot$I) 1,9
11	$\neg\neg\psi$	($\neg$I) 2–10
12	ψ	($\neg\neg$E) 11

Man beachte im obigen Beweis insbesondere, dass c bei seiner Einführung in Zeile 3 in der Tat frisch ist, da wir bei der Formulierung von $(\exists E)$ darauf bestanden haben, dass c auch frisch für ψ ist.

Notation. Wie schon im Falle der Aussagenlogik (Definition 2.5) schreiben wir $\Phi \vdash \psi$, wenn ein Beweis von ψ aus Annahmen in Φ existiert.

Weitere Beispiele für Herleitungen, jetzt im vollen Kalkül (also mit den Regeln für $\exists$):

Folgerung der Existenz aus der Universalität $\forall x.\, P(x) \vdash \exists x.\, P(x)$

1	$\forall x.\, P(x)$	
2	$P(x)$	($\forall$E, 1)
3	$\exists x.\, P(x)$	($\exists$I, 2)

(Wir werden später sehen, dass diese Folgerung auch semantisch gerechtfertigt ist.)

Transitivität der Subsumption

1	$\forall x.\, P(x) \to Q(x)$	
2	$\forall z.\, Q(z) \to R(z)$	
3	$\boxed{d}$	
4	$P(d)$	
5	$P(d) \to Q(d)$	($\forall$E) 1
6	$Q(d)$	($\to$E) 3, 4
7	$Q(d) \to R(d)$	($\forall$E) 2
8	$R(d)$	($\to$E) 5, 6
9	$P(d) \to R(d)$	($\to$I) 4–8
10	$\forall x.\, P(x) \to R(x)$	($\forall$I) 3–9

(Nicht-)-Vertauschung von $\forall$ und $\exists$ Wir können aus $\exists x \forall y.\, P(x, y)$ die Formel $\forall y.\, \exists x.\, P(x, y)$ folgern. Die Umkehrung gilt nicht, wie man sich anhand der bisher nur informell beschriebenen Bedeutung der Formeln klarmacht; wir werden dies später auch formal zeigen.

1	$\exists x \forall y.\, P(x,y)$	
2	$\boxed{c}$ $\forall y.\, P(c,y)$	
3	$\boxed{d}$	
4	$P(c,d)$	($\forall$E) 2
5	$\exists x.\, P(x,d)$	($\exists$I) 4
6	$\forall y \exists x.\, P(x,y)$	($\forall$I) 3–5
7	$\forall y \exists x.\, P(x,y)$	($\exists$E) 1, 2–6

Noch mehr Beispiele:

$$\exists x.\, \phi \vee \psi \equiv \exists x.\, \phi \vee \exists x.\, \psi$$
$$\phi \wedge \exists x.\, \psi \equiv \exists x.\, \phi \wedge \psi \qquad (x \notin FV(\phi))$$
$$\phi \vee \exists x.\, \psi \equiv \exists x.\, \phi \vee \psi \qquad (x \notin FV(\phi))$$

4.4 Semantik

Wir legen nunmehr formal die *Bedeutung* prädikatenlogischer Formeln fest. Ein in der Logik verbreitetes Prinzip ist, dass Bedeutung hierbei die Gestalt eines Begriffs von *Erfülltheit* von Formeln relativ zu einem *Modell* annimmt, d.h. die Semantik ist eine binäre Relation $\models$ zwischen Modellen und Formeln. Im Falle der Aussagenlogik war ein Modell schlicht und einfach eine Wahrheitsbelegung κ. Abhängig von einer gegebenen Signatur Σ definieren wir nun den Begriff des Σ-*Modells* (weitere geläufige Begriffe: Σ-Struktur, Σ-Algebra, FO-Modell, FO-Struktur, Interpretation). Ein solches Modell muss offenbar hinreichend Struktur für die Interpretation von Funktionen- und Prädikatensymbolen zur Verfügung stellen.

Definition 4.11 (Σ-Modell)**.** Ein Σ-*Modell* $\mathfrak{M}$ besteht aus

- einer nichtleeren (!) Menge M (dem *Universum*, *(Grund)bereich* oder *Träger*);
- einer *Interpretation* in $\mathfrak{M}$ für jedes n-stellige Funktionssymbol $f/n \in \Sigma$, gegeben durch eine Funktion $\mathfrak{M}[\![f]\!] : M^n \to M$
- einer *Interpretation* in $\mathfrak{M}$ für jedes n-stelliges Prädikatensymbol $P/n \in \Sigma$, gegeben durch eine Teilmenge $\mathfrak{M}[\![P]\!] \subseteq M^n$.

Eine *Belegung* η (in $\mathfrak{M}$) ist eine Abbildung $\eta : V \to M$, ordnet also jeder Variablen $v \in V$ einen Wert $\eta(v)$ im Grundbereich M zu.

Da Konstanten einfach nullstellige Funktionssymbole sind, werden sie durch Abbildungen $M^0 \to M$ interpretiert. Da es genau ein leeres Tupel () gibt, d.h. $M^0 = \{()\}$, läuft dies einfach auf die Auswahl eines Elements von M hinaus; wir werden daher im folgenden so tun, als würden Konstanten einfach durch Elemente von M interpretiert, wenn dies die Schreibweise vereinfacht.

Definition 4.12 (Interpretation von Termen, Erfülltheit)**.** Für ein gegebenes Modell $\mathfrak{M}$ und eine Belegung η definieren wir die *Interpretation*

$$\mathfrak{M}[\![E]\!]\eta \in M$$

$\mathfrak{M}[\![E]\!]\eta$ eines Terms E rekursiv durch

$$\mathfrak{M}[\![x]\!]\eta = \eta(x)$$
$$\mathfrak{M}[\![f(E_1, \ldots, E_n)]\!]\eta = \mathfrak{M}[\![f]\!]\,(\mathfrak{M}[\![E_1]\!]\eta, \ldots, \mathfrak{M}[\![E_n]\!]\eta)$$

Die *Erfülltheit* einer Formel ϕ in einem Modell $\mathfrak{M}$ und einer Belegung η, geschrieben $\mathfrak{M}, \eta \models \phi$ (sprich „$\mathfrak{M}, \eta$ erfüllt ϕ“) ist dann rekursiv definiert durch

$$\begin{aligned}
\mathfrak{M}, \eta \models (E = D) &\iff \mathfrak{M}[\![E]\!]\eta = \mathfrak{M}[\![D]\!]\eta \\
\mathfrak{M}, \eta \models P(E_1, \ldots, E_n) &\iff (\mathfrak{M}[\![E_1]\!]\eta, \ldots, \mathfrak{M}[\![E_n]\!]\eta) \in \mathfrak{M}[\![P]\!] \\
\mathfrak{M}, \eta \models \forall x.\, \phi &\iff \text{für alle } m \in M \text{ gilt } \mathfrak{M}, \eta[x \mapsto m] \models \phi \\
\mathfrak{M}, \eta \models \exists x.\, \phi &\iff \text{für (mindestens) ein } m \in M \text{ gilt } \mathfrak{M}, \eta[x \mapsto m] \models \phi
\end{aligned}$$

und durch die erwarteten Klauseln für die booleschen Fälle. Dabei bezeichnet $\eta[x \mapsto m]$ die Belegung mit

$$\eta[x \mapsto m](y) = \begin{cases} m & (y = x) \\ \eta(y) & (\text{sonst}). \end{cases}$$

Beispiel 4.13. Wir betrachten $\Sigma = \{z/0, s/1, O/1\}$, also eine Signatur, die aus einer Konstante z (*zero*), einem einstelligen Funktionssymbol s (*successor*) und einem einstelligen Prädikat O (*odd*) besteht, und suchen nach Σ-Modellen, die die geschlossene Formel

$$\phi = (\neg O(z) \wedge \forall x.\, O(s(x)) \leftrightarrow \neg O(x))$$

erfüllen. (Hierfür spielt die Belegung offenbar zunächst keine Rolle; wir beweisen dies weiter unten auch formal.)

- $M = \mathbb{N}$, $\mathfrak{M}[\![z]\!] = 0$, $\mathfrak{M}[\![s]\!](n) = n + 1$, $\mathfrak{M}[\![O]\!] = \{2n + 1 \mid n \in M\}$.
- $M = \{0, \ldots, 2n - 1\}$, $\mathfrak{M}[\![z]\!] = 0$, $\mathfrak{M}[\![s]\!](x) = x + 1$ für $x < 2n - 1$, $\mathfrak{M}[\![s]\!](2n - 1) = 0$, $\mathfrak{M}[\![O]\!] = \{2i - 1 \mid 1 \leqslant i \leqslant n\}$.
- $M = \mathbb{N} \times \mathbb{N}$, $\mathfrak{M}[\![z]\!] = (0, 0)$, $\mathfrak{M}[\![s]\!](n, k) = (n + 1, k)$, $\mathfrak{M}[\![O]\!] = \{(n, k) \mid n + k \text{ ungerade}\}$.

Nun wollen wir zeigen, dass der Begriff der freien Variable das leistet, was wir haben wollten, als wir ihn definiert haben, nämlich dass eine Formel höchstens von den in ihr vorkommenden freien Variablen abhängt. („Höchstens“, weil es z.B. auch Tautologien gibt, die von ihren freien Variablen nicht abhängen, wie etwa $\forall x.\, x = y \vee \neg(x = y)$.)

Lemma 4.14. *Sei $\eta_1(x) = \eta_2(x)$ für alle $x \in FV(\phi)$; dann gilt $\mathfrak{M}, \eta_1 \models \phi \iff \mathfrak{M}, \eta_2 \models \phi$.*

Beweis. Da in Formeln Terme vorkommen, müssen wir zunächst eine entsprechende Aussage für Terme beweisen, nämlich, dass für alle Terme E und alle Belegungen η_1, η_2, wenn $\eta_1(x) = \eta_2(x)$ für alle $x \in FV(E)$ gilt, dann

$$\mathfrak{M}[\![E]\!]\eta_1 = \mathfrak{M}[\![E]\!]\eta_2. \tag{2}$$

Der Beweis per Induktion über E wird dem Leser überlassen.

Wir beweisen nun die Aussage des Lemmas durch Induktion über ϕ: Atomare Formeln werden per (2) abgehandelt, z.B. $\mathfrak{M}, \eta_1 \models E = D \iff \mathfrak{M}[\![E]\!]\eta_1 = \mathfrak{M}[\![D]\!]\eta_1 \overset{(2)}{\iff} \mathfrak{M}[\![E]\!]\eta_2 = \mathfrak{M}[\![D]\!]\eta_2 \iff \mathfrak{M}, \eta_2 \models E = D$. Die Booleschen Fälle sind trivial. Es bleibt der Allquantor:

$$\begin{aligned} \mathfrak{M}, \eta_1 \models \forall x.\, \psi &\iff \text{für alle } m \in M \text{ gilt } \mathfrak{M}, \eta_1[x \mapsto m] \models \psi \\ &\overset{\text{IV}}{\iff} \text{für alle } m \in M \text{ gilt } \mathfrak{M}, \eta_2[x \mapsto m] \models \psi \\ &\iff \mathfrak{M}, \eta_2 \models \forall x.\, \psi \end{aligned}$$

Zur Anwendung der Induktionsvoraussetzung brauchen wir hierbei, dass

$$\eta_1[x \mapsto m](y) = \eta_2[x \mapsto m](y) \text{ für alle } y \in FV(\psi) \subseteq FV(\forall x.\, \psi) \cup \{x\}.$$

Dies zeigt man durch Fallunterscheidung über y: Wenn $y \in FV(\forall x.\, \psi)$, dann gilt $\eta_1[x \mapsto m](y) = \eta_1(y)$, da dann $y \neq x$, entsprechend für η_2, und $\eta_1(y) = \eta_2(y)$ gilt nach Voraussetzung. Falls $y = x$, so gilt $\eta_1[x \mapsto m](y) = m = \eta_2[x \mapsto m](y)$. □

Damit ist für eine *geschlossene* Formel ϕ (die ja keine freien Variablen hat) $\mathfrak{M}, \eta \models \phi$ von η unabhängig; wir schreiben in diesem Fall kurz $\mathfrak{M} \models \phi$ für $\mathfrak{M}, \eta \models \phi$.

Analog wie schon für aussagenlogische Formeln schreiben wir für eine Menge Φ von Formeln $\mathfrak{M}, \eta \models \Phi$, wenn $\mathfrak{M}, \eta \models \phi$ für alle $\phi \in \Phi$. Logische Konsequenz, Erfüllbarkeit, Gültigkeit und logische Äquivalenz werden analog wie bisher definiert, bis auf Ersetzung der Wahrheitsbelegung κ durch $\mathfrak{M}, \eta$. Z.B. gilt per Definition $\Phi \models \psi$ (ψ ist logische Folgerung aus Φ) dann, wenn aus $\mathfrak{M}, \eta \models \Phi$ stets $\mathfrak{M}, \eta \models \psi$ folgt. Wie bisher gilt $\Phi \models \psi$ genau dann, wenn $\Phi \cup \{\neg\psi\}$ unerfüllbar ist.

Beispiel 4.15 (Logische Konsequenz)**.** $\exists x \forall y.\, L(y, x) \models \forall y \exists x.\, L(y, x)$

Beweis. Aus der Annahme folgt, dass $m \in M$ existiert mit

$$\mathfrak{M}, [x \mapsto m] \models \forall y.\, L(y, x). \tag{3}$$

Sei $n \in M$; zu zeigen ist $\mathfrak{M}, [y \mapsto n] \models \exists x(L(y, x))$. Wegen (3) gilt $\mathfrak{M}, [y \mapsto n, x \mapsto m] \models L(y, x)$, woraus die Behauptung folgt. □

Andererseits gilt $\forall y \exists x(L(y, x)) \not\models \exists x \forall y.\, L(y, x)$: Ein Gegenmodell ist z.B. $M = \{0, 1\}$, $\mathfrak{M}[\![L]\!] = \{(1, 0), (0, 1)\}$.

In Beispielen wie dem obigen ist eine Sichtweise hilfreich, in der man die Erfülltheit einer Formel über das Gewinnen eines Spiels zwischen zwei Spielern, *Eloise* ($\exists$) und *Abaelard* ($\forall$), definiert. Eloise versucht zu zeigen, dass eine Formel erfüllt ist, und Abaelard, dass sie nicht erfüllt ist. Spielpositionen sind Paare (η, ϕ), wobei η eine Belegung in $\mathfrak{M}$ und ϕ eine Formel in NNF (also mit Negation nur vor atomaren Formeln; Existenz einer NNF für jede Formel wird weiter unten behandelt) ist; das Spiel ist beendet, wenn eine atomare Formel oder deren Negation erreicht ist (für die man dann unmittelbar im Modell nachsehen kann, wer gewonnen hat). Regeln des Spiels sind z.B.

- $\phi \wedge \psi$: $\forall$ ist dran und wählt ϕ oder ψ als neue Formel

- $\phi \vee \psi$: $\exists$ ist dran und wählt ϕ oder ψ als neue Formel
- $\forall x.\, \phi$: $\forall$ ist dran und wählt ein Element des Grundbereichs als neuen Wert für x (in η)
- $\exists x.\, \phi$: $\exists$ ist dran und wählt ein Element des Grundbereichs als neuen Wert für x

Atomare Formeln und deren Negationen sind Gewinnpositionen für einen der Spieler, je nachdem, ob die Formel eben gerade gilt oder nicht; z.B. gewinnt Eloise die Position $(\eta, L(x,y))$, wenn $\mathfrak{M}, \eta \models L(x,y)$ (d.h. wenn $(\eta(x), \eta(y)) \in \mathfrak{M}[\![L]\!]$). Der entscheidende Unterschied zwischen $\forall y \exists x.\, L(y,x)$ und $\exists x \forall y.\, L(y,x)$ ist in dieser Sichtweise, dass im Erfülltheitsspiel für $\forall y \exists x.\, L(y,x)$ Abaelard den ersten Zug macht, in dem für $\exists x \forall y.\, L(y,x)$ dagegen Eloise.

→ Welche Spielregeln müsste man für nichtatomare Negation einführen?

Beispiel 4.16 (Logische Äquivalenz)**.**

$$\neg \forall x.\, \phi \equiv \exists x.\, \neg\phi \qquad \neg \exists x.\, \phi \equiv \forall x.\, \neg\phi$$

Mittels der Äquivalenzen im Beispiel lässt sich die Transformation in Negationsnormalform (NNF) auf prädikatenlogische Formeln erweitern. Dazu müssen natürlich sowohl $\forall$ als auch $\exists$ als vollberechtigte Bestandteile der Grammatik von Formeln angesehen werden (ebenso wie schon im aussagenlogischen Fall sowohl $\wedge$ als auch $\vee$); eine NNF ist dann eine Formel in diesem Sinne, die Negation nur direkt vor atomaren Formeln (also Gleichungen und Prädikatenanwendungen) enthält. Wir werden später noch weitergehende Normalformresultate behandeln.

Leicht zu zeigen, aber zentral ist

Lemma 4.17 (Substitutionslemma)**.** *Es gilt*

$$\mathfrak{M}, \eta \models \phi\sigma \iff \mathfrak{M}, \eta_\sigma \models \phi,$$

wobei $\eta_\sigma(x) = \mathfrak{M}[\![\sigma(x)]\!]\eta$ *für* $x \in V$*.*

(Man beachte, dass $\eta_\sigma(x) = \eta(x)$ für $x \notin \mathrm{Dom}(\sigma)$: dann haben wir nämlich $\mathfrak{M}[\![\sigma(x)]\!]\eta = \mathfrak{M}[\![x]\!]\eta = \eta(x)$.)

Beweis. Induktion über ϕ. Die benötigte Variante der Aussage für Terme ist

$$\mathfrak{M}[\![E\sigma]\!]\eta = \mathfrak{M}[\![E]\!]\eta_\sigma.$$

□

Wir halten abschließend fest, dass unser formales Deduktionssystem zur gerade definierten Semantik passt:

Satz 4.18 (Korrektheit)**.** *Wenn* $\Phi \vdash \psi$*, dann auch* $\Phi \models \psi$*.*

Beweis. Wie schon im Fall der Aussagenlogik führen wir den Beweis über die Länge der Herleitung von ψ aus Φ, mit Fallunterscheidung über die zuletzt angewandte Regel und Verallgemeinerung der Induktionsbehauptung auf in Unterbeweisen aus lokalen Annahmen hergeleitete Aussagen. Die Regeln für Gleichheit sind einfach; die Regeln für $\exists$ sind aus denen für $\forall$ hergeleitet, so dass wir sie hier ignorieren können. Wir behandeln die Eliminationsregel für $\forall$, um die Verwendung des Substitutionslemmas zu illustrieren: Wenn im letzten Beweisschritt bei lokalen Annahmen Φ' die Formel $\psi[E/x]$ aus $\forall x.\, \psi$ hergeleitet wird, dann gilt per Induktionsvoraussetzung $\Phi' \models \forall x.\, \psi$. Sei nun $\mathfrak{M}, \eta \models \Phi'$; dann gilt nach Voraussetzung $\mathfrak{M}, \eta \models \forall x.\, \psi$. Zu zeigen ist $\mathfrak{M}, \eta \models \psi[E/x]$. Dies ist per Substitutionslemma äquivalent zu $\mathfrak{M}, \eta[x \mapsto \mathfrak{M}[\![E]\!]\eta] \models \psi$, was aber unmittelbar nach Definition der Semantik aus $\mathfrak{M}, \eta \models \forall x.\, \psi$ folgt. □

5 Unifikation

Unifikation bezeichnet das Problem, ob und wie zwei gegebene Terme durch Substitution strukturell gleich gemacht, sprich *unifiziert* werden können. Mit anderen Worten ist Unifikation Gleichungslösen mit rein syntaktischen Mitteln: Wir können z.B. eine Lösung für die Gleichung $f(g(x)) = f(y)$ angeben, ohne zu wissen, was f und g für Abbildungen sind, nämlich $y = g(x)$. Unifikation spielt eine zentrale Rolle in der logischen Programmierung und in automatischen Deduktionsalgorithmen, z.B. in der prädikatenlogischen Resolution (siehe Kapitel 7). Als einfaches Beispiel überlege man sich, wie man die (implizit allquantifizierten) Gleichungen

$$x \cdot y = y \cdot x \quad \text{und} \quad z \cdot (u + v) = z \cdot u + z \cdot v$$

in einer Schlussfolgerung verknüpfen könnte.

5.1 Problemstellung

Definition 5.1 (Gleichung, Unifikatoren, Allgemeinheitsvergleich)**.** Eine *Gleichung* $E \doteq D$ ist ein Paar (E, D) von Termen. Ein *Gleichungssystem* ist eine Menge S von Gleichungen; die freien Variablen von S sind die der in S vorkommenden Gleichungen, also $FV(S) = \bigcup_{(E \doteq D) \in S} FV(E) \cup FV(D)$. Eine Substitution σ ist ein *Unifikator* von $E \doteq D$, wenn $E\sigma = D\sigma$, wobei wir (wie auch bisher schon) mit dem undekorierten Gleichheitszeichen '$=$' *syntaktische* Gleichheit bezeichnen, d.h. mit der voranstehenden Gleichung meinen wir, dass $E\sigma$ und $D\sigma$ wörtlich identische Terme sind. Ein *Unifikator* eines Gleichungssystems S ist eine Substitution, die Unifikator aller Gleichungen in S ist. Das System S ist *unifizierbar*, wenn es einen Unifikator hat. Wir bezeichnen mit

$$\mathsf{Unif}(S) = \{\sigma \mid \sigma \text{ ist Unifikator von } S\}$$

die Menge aller Unifikatoren von S. Eine Substitution σ_1 ist *allgemeiner als* σ_2, und σ_2 heißt dann umgekehrt eine *Spezialisierung von* σ_1, wenn eine Substitution τ existiert mit $\sigma_1\tau = \sigma_2$, wenn also σ_2 durch Einsetzen aus σ_1 hervorgeht.

Ein Unifikator σ von S ist *allgemeinster Unifikator* (mgu, für *most general unifier*) *von* S ($\sigma = mgu(S)$), wenn σ allgemeiner als jeder Unifikator von S ist.

Beispiel 5.2. Zur Gleichung

$$add(suc(x), y) \doteq add(y, suc(z))$$

haben wir z.B. den Unifikator $\sigma = [suc(x)/y, x/z]$. Man kann sich überzeugen, dass dies sogar ein mgu ist. In jedem Fall ist z.B. $\sigma' = [suc(suc(z))/y, suc(z)/z, suc(z)/x]$ ein weiterer Unifikator, und σ ist allgemeiner als σ': wir haben

$$\sigma' = \sigma[suc(z)/x].$$

Unifikatoren sind abgeschlossen unter Spezialisierung, d.h. eine Spezialisierung eines Unifikators ist selbst wieder ein Unifikator:

$$\sigma \in \mathsf{Unif}(S) \Rightarrow \sigma\tau \in \mathsf{Unif}(S).$$

Es folgt, dass wir *alle* Unifikatoren eines Gleichungssystems kennen, sobald wir einen mgu gefunden haben; dann sind nämlich die Unifikatoren gerade die Spezialisierungen des mgu:

Lemma 5.3. *Wenn $\sigma = mgu(S)$, dann gilt $\mathsf{Unif}(S) = \{\sigma\tau \mid \tau \text{ Substitution}\}$.*

Der mgu ist im allgemeinen nicht eindeutig bestimmt; z.B. hat die Gleichung $f(x) = f(g(y))$ sowohl $[g(y)/x]$ als auch $[g(z)/x, z/y]$ als mgu. Es gilt aber:

Lemma 5.4 (Eindeutigkeit des mgu)**.** *Der mgu ist eindeutig bis auf injektive Umbenennung von Variablen.*

Beweis. Seien σ, σ' mgu eines Gleichungssystems S. Per Definition existieren dann Substitutionen τ, τ' mit

$$\sigma' = \sigma\tau \quad \sigma = \sigma'\tau',$$

also

$$\sigma = \sigma\tau\tau'$$

– d.h. τ' macht τ rückgängig (genauer gesagt gilt dies auf Variablen, die in Termen der Form $\sigma(x)$ mit $x \in FV(S)$ vorkommen, und nur die interessieren uns). Da man durch Substitution nie aus einem komplexen Term wieder eine Variable machen kann, folgt damit, dass τ eine Umbenennung ist, und da man die Identifikation zweier Variablen nicht wieder rückgängig machen kann, folgt, dass τ injektiv ist. □

Wir beweisen nun die *Existenz* eines mgu durch Angabe eines Algorithmus:

5.2 Unifikationsalgorithmus von Martelli/Montanari

Folgende Definition fängt einen Begriff von *Lösung* eines Gleichungssystems formal ein, den wir im Grunde seit der Mittelstufe kennen:

Definition 5.5 (Gelöstes Gleichungssystem)**.** Ein Gleichungssystem S heißt *gelöst*, wenn jede Gleichung in S von der Form $x \doteq E$ ist, wobei x nur dieses eine Mal in S vorkommt (formal: $x \notin FV(E) \cup FV(S \backslash \{x \doteq E\})$). Zwei Gleichungssysteme S_1, S_2 sind *äquivalent*, wenn $\mathsf{Unif}(S_1) = \mathsf{Unif}(S_2)$. Ein zu S äquivalentes gelöstes Gleichungssystem heißt *gelöste Form* von S.

Wenn wir ein Gleichungssystem in diesem Sinne gelöst haben, kennen wir seinen mgu:

Lemma 5.6. *Wenn $S' = \{x_1 \doteq E_1, \ldots, x_n \doteq E_n\}$ eine gelöste Form von S ist, dann gilt $[E_1/x_1, \ldots, E_n/x_n] = mgu(S)$.*

Beweis. Da S' gelöst ist, ist $\sigma = [E_1/x_1, \ldots, E_n/x_n]$ Unifikator von S' und damit auch von S. Wenn σ' Unifikator von S ist, dann auch von S'; es gilt also $\sigma'(x_i) = E_i\sigma' = \sigma(x_i)\sigma'$ für alle i. Wenn $y \notin \{x_1, \ldots, x_n\}$, dann gilt $\sigma(y)\sigma' = y\sigma' = \sigma'(y)$. Insgesamt gilt also $\sigma' = \sigma\sigma'$, d.h. σ ist allgemeiner als σ'. □

Der Algorithmus (ursprünglich Herbrand, später Martelli/Montanari) besteht nun in erschöpfender Anwendung der folgenden Umformungsregeln (ohne Vorgabe einer bestimmten Reihen-

folge der Regelanwendung):

(delete):
$S \cup \{x \doteq x\} \quad \to S$

(decomp):
$S \cup \{f(E_1, \ldots, E_n) \doteq f(D_1, \ldots, D_n)\} \quad \to S \cup \{E_1 \doteq D_1, \ldots, E_n \doteq D_n\}$

(conflict) :
$S \cup \{f(E_1, \ldots, E_n) \doteq g(D_1, \ldots, D_k)\} \quad \to \bot \qquad (\text{für } f \neq g)$

(orient) :
$S \cup \{E \doteq x\} \quad \to S \cup \{x \doteq E\} \qquad (E \text{ keine Variable})$

(occurs) :
$S \cup \{x \doteq E\} \quad \to \bot \qquad (\text{für } x \in FV(E), x \neq E)$

(elim) :
$S \cup \{x \doteq E\} \quad \to S[E/x] \cup \{x \doteq E\} \qquad (\text{für } x \notin FV(E), x \in FV(S))$

Wenn $\bot$ erreicht wird, gibt der Algorithmus „nicht unifizierbar“ aus; ansonsten berechnet er, wie wir noch zeigen, eine gelöste Form und damit einen mgu von S.

Beispiele:

- $f(x, g(y)) \doteq f(g(z), z)$
- $f(x, g(x), h(y)) \doteq f(k(y), g(z), z)$
- $f(x, g(x)) \doteq f(z, z)$

Wir zeigen zunächst Terminierung. Dazu erinnern wir an den Begriff des *Terminationsmaßes*: Wir ordnen jedem Zustand (hier: jedem Gleichungssystem S so wie aktuell umgeformt) ein Maß $\nu(S)$ zu, so dass $\nu(S)$ in jedem Schritt abnimmt. Wenn wir als Wertebereich des Maßes eine geordnete Menge wählen, in der es keine unendlichen absteigenden Ketten

$$x_0 > x_1 > x_2 > \ldots$$

gibt, beweist das, dass immer nur endlich viele Zustände durchlaufen werden, bevor der Prozess stoppt. Ein Beispiel einer solchen Menge ist die Menge $\mathbb{N}^n$ aller n-Tupel von natürlichen Zahlen in *lexikographischer Ordnung* $\preceq$. Diese definieren wir durch Rekursion über n wie folgt: Wir setzen

- $() \preceq ()$
- für $n > 0$: $(a_1, \ldots, a_n) \preceq (b_1, \ldots, b_n)$ genau dann, wenn $a_1 \leqslant b_1$ und, falls $a_1 = b_1$, außerdem $(a_2, \ldots, a_n) \preceq (b_2, \ldots, b_n)$.

Es gilt nun in der Tat

Satz 5.7. *Die lexikographische Ordnung ist eine* Wohlordnung*, d.h. es gibt keine unendliche absteigende Kette*

$$\vec{a}^0 \succ \vec{a}^1 \succ \vec{a}^2 \succ \ldots \tag{4}$$

Beweis. Induktion über n, mit trivialem Induktionsanfang. Wir nehmen zwecks Herleitung eines Widerspruchs an, wir hätten eine unendliche absteigende Kette (4), mit $\vec{a}^i = (a_1^i, \ldots, a_n^i)$. Dann gilt

$$a_1^0 \geqslant a_1^1 \geqslant a_1^2 \geqslant \ldots.$$

Diese Kette wird, da es in $\mathbb{N}$ keine unendlichen absteigenden Ketten gibt, irgendwann *stationär*, d.h. es existiert k mit $a_1^i = a_1^k$ für alle $i \geqslant k$. Dann gilt aber per Definition

$$(a_2^k, \ldots, a_n^k) \succ (a_2^{k+1}, \ldots, a_n^{k+1}) \succ (a_2^{k+2}, \ldots, a_n^{k+2}) \succ \ldots,$$

im Widerspruch zur Induktionsvoraussetzung, die besagt, dass es in $\mathbb{N}^{n-1}$ keine unendlichen absteigenden Ketten gibt. □

Im vorliegenden Fall verwenden wir als Terminationsmaß das Tripel $(n_0 - n_1, n_2, n_3)$ in lexikographischer Ordnung, wobei

- n_0 = Anzahl der *ursprünglich* im System vorkommenden Variablen
- n_1 = Anzahl der Variablen x, die *erledigt* sind, d.h. als linke Seite einer Gleichung vorkommen und sonst nirgends.
- n_2 = Anzahl Symbole in aktuellen System
- n_3 = Anzahl Gleichungen der Form $E \doteq x$ im aktuellen System mit E keine Variable.

Dieses Maß nimmt offenbar in jedem Umformungsschritt ab, so dass der Algorithmus terminiert. Die erste Komponente $n_0 - n_1$ ist nichtnegativ, da der Algorithmus nie neue Variablen hinzufügt, also alle erledigten Variablen von vornherein im System vorgekommen sind.

Wenn der Algorithmus ohne Erreichen von $\bot$ terminiert, also keine Regel mehr anwendbar ist, dann ist das so erreichte System S gelöst: wenn $E \doteq D$ eine Gleichung in S ist, dann kann E nicht zusammengesetzt sein (sonst greift eine der Regeln *(decomp)*, *(conflict)* oder *(orient)*). Also ist E eine Variable x. Wegen der Regeln *(delete)*, *(occurs)* und *(elim)* kommt dann x weder in D noch in den restlichen Gleichungen vor.

Es bleibt zu zeigen, dass die Regeln das gegebene Gleichungssystem stets in ein äquivalentes transformieren, wobei $\bot$ das unlösbare System repräsentiert. Das ist klar in allen Fällen bis vielleicht auf *(occurs)*. Eine Gleichung der Form $x \doteq E$ mit $x \in FV(E)$ und $x \neq E$ ist aber offenbar in der Tat nicht unifizierbar: Da E mindestens ein Vorkommen von x und mindestens ein weiteres Symbol enthält, ist der Term $E\sigma$ stets echt größer als der Term $x\sigma$.

6 Normalformen in Logik erster Stufe

Wir erinnern daran, dass wir für Aussagenlogik in Abschnitt 3 verschiedene Normalformen eingeführt haben, insbesondere die Negationsnormalform (NNF) und die konjunktive Normalform (CNF). Die NNF haben wir in Abschnitt 4.4 bereits auf die Logik erster Stufe ausgedehnt; zur Erinnerung: Eine Formel in Logik erster Stufe (mit $\vee$ und $\exists$ als eigenständigen Symbolen) ist in NNF, wenn in ihr Negationen nur direkt vor atomaren Formeln (also Gleichungen oder Prädikatenanwendungen) vorkommen, und wir formen eine gegebene Formel in NNF um, indem

wir die für die Aussagenlogik in Abschnitt 3 angegebenen Umformungsregeln (de Morgansche Regeln, Entfernung von Doppelnegationen) sowie die Äquivalenzen

$$\neg\forall x.\,\phi \equiv \exists x.\,\neg\phi \qquad \neg\exists x.\,\phi \equiv \forall x.\,\neg\phi$$

erschöpfend anwenden. Wir führen nun weitere Normalisierungen ein, die sich spezifisch mit der Form des Vorkommens von Quantoren befassen.)

6.1 Pränexe Normalform

Unsere erste Normalisierung betrifft die Position der Quantoren, die wir an den Anfang der Formel ziehen:

Definition 6.1. Eine *pränexe Normalform* ist eine Formel der Form

$$Q_1 x_1.\ \ldots Q_n x_n.\,\phi$$

mit $Q_1, \ldots, Q_n \in \{\forall, \exists\}$ und ϕ quantorenfrei (wo bei eine Formel wie erwartet *quantorenfrei* heißt, wenn sie keine Quantoren enthält, also eine aussagenlogische Kombination von atomaren Formeln ist).

Satz 6.2. *Zu jeder Formel in Prädikatenlogik erster Stufe lässt sich eine äquivalente pränexe Normalform berechnen.*

Beweis. Wir können annehmen, dass die Formel bereits in NNF ist. Die Berechnung erfolgt dann durch erschöpfende Anwendung der Umformungsregeln

$$\begin{aligned} \phi \wedge \exists x.\,\psi &\equiv \exists x.\,\phi \wedge \psi & \qquad \phi \wedge \forall x.\,\psi &\equiv \forall x.\,\phi \wedge \psi \\ \phi \vee \exists x.\,\psi &\equiv \exists x.\,\phi \vee \psi & \qquad \phi \vee \forall x.\,\psi &\equiv \forall x.\,\phi \vee \psi, \end{aligned}$$

wobei für die Korrektheit dieser Umformungen $x \notin FV(\phi)$ vorausgesetzt wird, was durch geeignete Umbenennung der gebundenen Variablen x stets erreicht werden kann. Es sind jeweils die symmetrischen Fälle mit gemeint. Jede Umformung beseitigt einen *Fehlstand*, d.h. ein Vorkommen eines Booleschen Konnektivs oberhalb eines Quantors, so dass das Verfahren mit einer Formel, auf die keine Umformungsregel mehr anwendbar ist, terminiert; eine solche ist eine pränexe Normalform. □

Beispiel 6.3. Die Formel

$$\forall x.\,(\forall y.\,L(y,x)) \to \exists y.\,M(x,y)$$

wird wie folgt umgeformt:

$$\begin{aligned} &\forall x.\,(\forall y.\,L(y,x)) \to \exists y.\,M(x,y) \\ &\equiv \forall x.\,\neg(\forall y.\,L(y,x)) \vee \exists y.\,M(x,y) \\ &\equiv \forall x.\,(\exists y.\,\neg L(y,x)) \vee \exists y.\,M(x,y) \\ &\equiv \forall x \exists y.\,(\exists y.\,\neg L(y,x)) \vee\ M(x,y) \\ &\equiv \forall x \exists y \exists y'.\,\neg L(y',x) \vee\ M(x,y). \end{aligned}$$

Man beachte insbesondere den letzten Umformungsschritt, in dem wir die quantifizierte Variable in $\exists y.\,\neg L(y,x)$ in y' umbenennen, da wir die Formel $M(x,y)$, in der y frei vorkommt, unter den Existenzquantor ziehen wollen.

6.2 Skolemform

Als nächstes eliminieren wir den Existenzquantor:

Definition 6.4. Eine *Skolemform* ist eine pränexe Normalform, die nur Allquantoren enthält.

Es gibt ersichtlich nicht zu jeder Formel eine äquivalente Skolemform; z.B. ist jede Skolemform über der leeren Signatur Σ äquivalent zu entweder $\top$ oder $\bot$ oder $\forall x, y.\, x = y$ (warum?), und damit nicht äquivalent zu $\exists x, y.\, x \neq y$. Es gilt jedoch

Definition 6.5. Formeln ϕ, ψ heißen *erfüllbarkeitsäquivalent*, wenn ϕ genau dann erfüllbar ist, wenn ψ erfüllbar ist.

Satz 6.6. *Zu jeder Formel in Prädikatenlogik erster Stufe lässt sich eine erfüllbarkeitsäquivalente Skolemform berechnen.*

Diese *Skolemisierung* beruht auf der Ersetzung von Existenzquantoren durch frische Konstanten, basierend auf der Beobachtung, dass $\exists x.\, \phi$ genau dann erfüllbar ist, wenn $\phi[c/x]$ für eine frische Konstante c (eine *Skolemkonstante*) erfüllbar ist. Im allgemeinen hängen die für x einzusetzenden Werte noch von vorhergehenden allquantifizierten Variablen ab, so dass man es mit Skolemfunktionen statt nur Skolemkonstanten zu tun hat. Formal:

Beweis (Satz 6.6). Man berechnet zunächst eine pränexe Normalform

$$Q_1 x_1 . \ldots Q_n x_n .\, \chi,$$

und beseitigt dann die Existenzquantoren durch wiederholte Anwendung der erfüllbarkeitsäquivalenten Umformung

$$\forall x_1 \ldots \forall x_k \exists y.\, \phi \rightsquigarrow \forall x_1 \ldots \forall x_k .\, \phi[f(x_1, \ldots, x_k)/y]$$

(Achtung: Nur auf ganze Formeln, nicht auf Teilformeln innerhalb größerer Formeln! Umformung von Teilformeln in erfüllbarkeitsäquivalente Formeln ist insbesondere im allgemeinen nicht selbst eine erfüllbarkeitsäquivalente Umformung.) mit jeweils einem frischen Funktionssymbol f, das wir als *Skolemfunktkion* bezeichnen. (Bei der Transformation von Formelmengen darf jedes f nur zur Transformation einer Formel verwendet werden.)

Wir bezeichnen die linke Seite dieser Umformung mit ψ und die rechte mit $\bar{\psi}$; wir müssen zeigen, dass ψ und $\bar{\psi}$ erfüllbarkeitsäquivalent sind:

Wenn einerseits $\mathfrak{M}, \eta \models \bar{\psi}$,, dann auch $\mathfrak{M}, \eta \models \psi$, da für jedes $\eta' = \eta[x_1 \mapsto m_1, \ldots, x_k \mapsto m_k]$ $\mathfrak{M}[\![f(x_1, \ldots, x_k)]\!]\eta'$ per Substitutionslemma einen Zeugen für den Existenzquantor für x_j liefert.

Wenn umgekehrt $\mathfrak{M}, \eta \models \phi$, dann kann man $\mathfrak{M}$ zu einem Modell $\overline{\mathfrak{M}}$ der um die Skolemfunktionen vergrößerten Signatur mit $\overline{\mathfrak{M}}, \eta \models \bar{\phi}$ *erweitern*, indem man für jedes Tupel $(m_1, \ldots, m_k) \in M^k$ den Wert $\mathfrak{M}[\![f]\!](m_1, \ldots, m_k) = m$ so *wählt*, dass

$$\overline{M}, \eta[x_1 \mapsto m_1, \ldots, x_k \mapsto m_k, y \mapsto m] \models \phi$$

gilt (was allerdings am sogenannten *Auswahlaxiom* der Mengenlehre hängt); dann gilt per Substititionslemma wie verlangt auch

$$\overline{M}, \eta[x_1 \mapsto m_1, \ldots, x_k \mapsto m_k] \models \phi[f(x_1, \ldots, x_n)/y].$$

Es folgt $\overline{M}, \eta \models \bar{\psi}$. □

(In der Praxis kann man die Skolemisierung in einem Schritt durchführen, indem man in $\phi = Q_1 x_1 \ldots Q_n x_n(\chi)$ alle Existenzquantoren $\exists x_j$ streicht und jede existenzquantifizierte Variable x_j durch

$$f_j(x_{j_1}, \ldots, x_{j_k})$$

substituiert, wobei f_j ein jeweils neu eingeführtes k-stelliges Funktionssymbol ist und $j_1, \ldots, j_k$ diejenigen Indizes $\leqslant j$ sind, für die $Q_j = \forall$; d.h. $\{j_1, \ldots, j_k\} = \{i \leqslant j \mid Q_i = \forall\}$)

Beispiel 6.7. Wir führen die Normalisierung aus Beispiel 6.3 fort:

$$\forall x \exists y \exists y'. \neg L(y', x) \vee M(x, y)$$

ist erfüllbarkeitsäquivalent zu seiner Skolemisierung

$$\forall x. \neg L(f_3(x), x) \vee M(x, f_2(x)).$$

Bemerkung 6.8. Das im Beweis von Satz 6.2 verwendete Verfahren zur Berechnung einer pränexen Normalform beinhaltet Wahlfreiheiten hinsichtlich der jeweils als nächstes anzuwendenden Umformung, und verschiedene Umformungsabfolgen können verschiedene pränexe Normalformen liefern. Z.B. kann man $(\forall x. P(x)) \vee \exists y. Q(y)$ in jeweils zwei Schritten sowohl zu $\forall x \exists y. P(x) \vee Q(y)$ als auch zu $\exists y \forall x. P(x) \vee Q(y)$ umformen. Zweiteres ist für Zwecke der Skolemisierung günstiger, da dann die Skolemfunktion für den Existenzquantor ein Argument weniger (nämlich keins) hat. Aus ähnlichen Gründen kann es vorteilhaft sein, vor der Bildung von pränexen Normalformen die Reihenfolge in Konjunktionen oder Disjunktionen geeignet zu vertauschen.

6.3 Klauselform

Man kann nach unseren Resultaten über Normalformen in Aussagenlogik jede pränexe Normalform $Q_1 x_1 \ldots Q_n x_n(\phi)$ so weiter normalisieren, dass ϕ in CNF ist. Wenn die Formel zusätzlich in Skolemform ist, also $Q_i = \forall$ für alle i, dann kann man die führenden Quantoren in der Notation weglassen, so dass man also alle Variablen als implizit allquantifiziert ansieht. Für die so erhaltene CNF verwendet man dann üblicherweise die bereits eingeführte Schreibweise als Menge von Klauseln. Eine solche Klauselmenge heißt *Klauselform* bzw. *ist in Klauselform.*

Beispiel 6.9. Die Formel aus Beispiel 6.7 hat bereits eine CNF als quantorenfreien Anteil; sie lautet in Klauselform (unter Weglassung der äußeren Mengenklammern)

$$\{\neg L(f_3(x), x),\ M(x, f_2(x))\}$$

(besteht also nur aus einer Klausel).

7 Resolution in Prädikatenlogik erster Stufe

Das Resolutionsverfahren für Aussagenlogik lässt sich auf Prädikatenlogik erster Stufe erweitern, ist dort allerdings nur noch ein Halbentscheidungsverfahren für Unerfüllbarkeit (d.h. für erfüllbare Eingabeformeln terminiert es möglicherweise nicht). Das Verfahren arbeitet mit Formeln in Klauselform wie oben eingeführt, d.h. die Eingabe ist eine konjunktiv gelesene endliche

Menge von Klauseln, die wiederum endliche Mengen von Literalen sind. Klauseln werden dabei implizit als über alle vorkommenden Variablen allquantifiziert gelesen. Z.B. steht die Klausel

$$\{\neg P(x,y), \neg Q(y), R(x)\}$$

für die Formel

$$\forall x, y.\, \neg P(x,y) \vee \neg Q(y) \vee R(x),$$

äquivalenterweise für

$$\forall x, y.\, P(x,y) \wedge Q(y) \to R(x).$$

Die Schlussregel, auf der das Verfahren beruht, kombiniert die aussagenlogische Resolutionsregel

$$\frac{C_1 \cup \{A\} \qquad C_2 \cup \{\neg A\}}{C_1 \cup C_2}$$

mit der Spezialisierungsregel

$$\frac{C}{C\sigma},$$

die aus einer Klausel C eine Substitutionsinstanz für eine Substitution σ folgert, sowie mit der '*Faktorisierungsregel*'

$$\frac{C \cup \{A, A\}}{C \cup \{A\}}$$

(die natürlich eigentlich nichts tut, da ja mengentheoretisch $\{A, A\} = \{A\}$ gilt). Die kombinierte Regel, *Resolution mit impliziter Faktorisierung*, verwendet Unifikation. Sie lautet

$$(RIF)\frac{C_1 \cup \{A_1, \ldots, A_n\} \qquad C_2 \cup \{\neg B\}}{C_1\sigma \cup C_2\sigma} \qquad (\sigma = mgu(A_1, \ldots, A_n, B)).$$

(Dies ist dann die *einzige* Regel des Systems!) Hierbei schreiben wir kurz $mgu(A_1, \ldots, A_n, B)$ für den mgu des Systems $\{A_i \doteq B \mid i = 1, \ldots, n\}$. Bei Anwendung der Regel werden zuerst nötigenfalls die Variablen in den beiden ursprünglichen Klauseln so umbenannt, dass die Klauseln disjunkte Variablenmengen haben (aus Lesbarkeitsgründen nehmen wir diese Umbenennung nicht in die Notation der Regel mit auf). Das ist offenbar zulässig, da die Klauseln ja allquantifizierte Formeln darstellen, und vergrößert die Menge der Unifikatoren.

Der Algorithmus zum Beweis der Gültigkeit einer Formel ϕ lautet damit

1. Bilde $\neg\phi$
2. Transformiere $\neg\phi$ in Klauselform
3. Wende (RIF) an, bis $\square$ erreicht ist.

(Achtung: anders als bei der aussagenlogischen Resolution kann man hier i.a. nicht erwarten, dass, wenn ϕ nicht gültig ist, irgendwann eine Formelmenge erreicht wird, auf die (RIF) nicht mehr anwendbar ist; d.h. der Algorithmus terminiert in diesem Fall eventuell nicht.)

Beispiel 7.1. Die Negation der Formel

$$P(a) \wedge (\forall x.\, P(x) \to P(f(x))) \to \exists x.\, P(f(f(x)))$$

wird in Klauselform zu

$$(1)\ \{P(a)\} \qquad (2)\ \{\neg P(x), P(f(x))\} \qquad (3)\ \{\neg P(f(f(x)))\}.$$

Resolution von (1) mit (2) liefert

$$(4)\ \{P(f(a))\},$$

Resolution von (4) mit (2) liefert

$$(5)\ \{P(f(f(a)))\},$$

und Resolution mit (3) liefert schließlich $\square$, womit die ursprüngliche Formel bewiesen ist.

Weiteres Beispiel: Wissenswertes über Erlangen.

Bemerkung 7.2. Ohne Faktorisierung ist der Resolutionsalgorithmus unvollständig: Gegenbeispiel: Die aus den Klauseln $\{P(x), P(y)\}, \{\neg P(z), \neg P(w)\}$ bestehende Klauselmenge ist ersichtlich widersprüchlich, aber es lässt sich aus ihr mittels einer Variante von (RIF), in der wir $n = 1$ festlegen (also keine implizite Faktorisierung einbauen) nicht die leere Klausel herleiten. (Im Beispiel liegt das unter anderem daran, dass wir vor jedem Resolutionsschritt die Variablenmengen der beiden Seiten disjunkt machen; wenn aber keine Disjunktheit verlangt, ist das letztlich auch nur eine Form von Faktorisierung.)

7.1 Herbrand-Modelle

Wir zeigen nun die *Vollständigkeit* des Resolutionsalgorithmus, d.h. dass der Algorithmus auf jeder *unerfüllbaren* Klauselform mit der Antwort 'unerfüllbar' terminiert. (Auf *erfüllbaren* Klauselformen terminiert der Algorithmus eventuell nicht.) Der Schlüsselbegriff ist hierbei der des sogenannten *Herbrandmodells*; Herbrandmodelle zeichnen sich dadurch aus, dass in ihnen jedes Element durch einen Term ohne Variablen bezeichnet wird. Insbesondere sind Herbrandmodelle damit abzählbar; ihre Konstruktion impliziert daher den berühmten *Satz von Löwenheim-Skolem*, der besagt, dass jede erfüllbare Formelmenge in Prädikatenlogik ein höchstens abzählbares Modell hat (sofern die Signatur höchstens abzählbar ist).

Zur technischen Vereinfachung schränken wir ab jetzt auf Prädikatenlogik ohne „=“ ein; d.h. als atomare Formeln gibt es nur noch Prädikatenanwendungen $P(E_1, \ldots, E_n)$. Für den Rest des Abschnitts sei eine Signatur Σ gegeben; o.E. nehmen wir an, dass Σ mindestens eine Konstante enthält (dass das o.E. ist, liegt daran, dass per Definition alle Modelle nichtleeren Träger haben).

Definition 7.3 (Herbrand-Universum)**.** Das *Herbrand-Universum* U_Σ ist die Menge aller *Grundterme* (Ground Terms), d.h. Terme ohne Variablen:

$$U_\Sigma = \{E \mid E\ \Sigma\text{-Term}, FV(E) = \varnothing\}.$$

Für $E \in U_\Sigma$ und ein Σ-Modell $\mathfrak{M}$ schreiben wir $\mathfrak{M}[\![E]\!]$ statt $\mathfrak{M}[\![E]\!]\eta$ für die Auswertung von E in $\mathfrak{M}$, da Belegungen für die Auswertung geschlossener Terme irrelevant sind.

Beispiel 7.4. Bei $\Sigma = \{s/1, z/0, O/2\}$ haben wir $U_\Sigma = \{z, s(z), s(s(z)), \ldots\}$.

Definition 7.5 (Herbrand-Modell)**.** Ein *Herbrand-(Σ-)Modell* ist ein Σ-Modell $\mathfrak{M}$ mit folgenden Eigenschaften:

- $M = U_\Sigma$

- Für $f/n \in \Sigma$ und $E_1, \ldots, E_n \in U_\Sigma$ gilt

$$\mathfrak{M}[\![f]\!](E_1, \ldots, E_n) = f(E_1, \ldots, E_n)$$

Beispiel 7.6. Mit Σ wie oben gilt also in Herbrand-Σ-Modellen $\mathfrak{M}$ z.B. $\mathfrak{M}[\![s]\!](s(z)) = s(s(z))$.

Lemma 7.7 (Auswertungslemma für Terme)**.** *Sei $\mathfrak{M}$ ein Herbrand-Σ-Modell. Dann gilt*

$$\mathfrak{M}[\![E]\!]\eta = E\eta$$

für jeden Term E. Insbesondere gilt $\mathfrak{M}[\![E]\!] = E$ für Grundterme E.

Zum Verständnis der Aussage des Lemmas beachte man, dass eine Belegung η in $\mathfrak{M}$ gleichzeitig eine Substitution ist, eben eine Abbildung von Variablen auf Terme (bei Unterschlagung der hier nicht so wichtigen Endlichkeitsbedingung an $\mathrm{Dom}(\eta)$).

Beweis. Induktion über E. □

Definition 7.8 (Grundinstanz)**.** Sei ϕ eine Formel. Eine *Grundinstanz* (Ground Instance) von ϕ ist eine Formel der Form $\phi\sigma$, wobei σ eine *Grundsubstitution* (Ground Substitution) ist, d.h. für alle $x \in FV(\phi)$ ist $\sigma(x)$ ein Grundterm; äquivalenterweise ist $\phi\sigma$ eine geschlossene Formel, also $FV(\phi\sigma) = \varnothing$. Für eine Menge Φ von Formeln bezeichnet $E(\Phi)$ die Menge der Grundinstanzen von Formeln aus Φ, d.h.

$$E(\Phi) = \{\phi\sigma \mid \phi \in \Phi, FV(\phi\sigma) = \varnothing\}.$$

Lemma 7.9 (Auswertungslemma für Formeln)**.** *Sei ϕ eine Formel und $\mathfrak{M}$ ein Herbrand-Modell. Dann gilt, unter erneuter Verwechslung von Substitutionen und Belegungen,*

$$\mathfrak{M}, \eta \models \phi \iff \mathfrak{M} \models \phi\eta.$$

Man beachte dabei, dass oben $\phi\eta$ eine Grundinstanz von ϕ ist.

Beweis. Nach dem Substitutionslemma gilt zunächst

$$\mathfrak{M}, \eta' \models \phi \iff \mathfrak{M} \models \phi\eta,$$

wobei $\eta'(x) = \mathfrak{M}[\![\eta(x)]\!]$. Nach dem Auswertungslemma für Terme gilt aber $\eta' = \eta$. □

Definition 7.10 (Universeller Abschluss)**.** Der *universelle Abschluss* $\forall\phi$ einer Formel ϕ mit $FV(\phi) = \{x_1, \ldots, x_n\}$ ist die Formel $\forall x_1 \ldots \forall x_n(\phi)$. Wir dehnen diesen Begriff auf Formelmengen Φ aus per $\forall\Phi = \{\forall\phi \mid \phi \in \Phi\}$.

In Worten: Der universelle Abschluss von ϕ allquantifiziert alle freien Variablen in ϕ.

Lemma 7.11 (Grundinstanzlemma)**.** *Sei $\mathfrak{M}$ ein Herbrand-Σ-Modell und ϕ eine Formel. Dann gilt*

$$\mathfrak{M} \models \forall\phi \iff \mathfrak{M} \models \phi\sigma \text{ für jede Grundinstanz } \phi\sigma \text{ von } \phi.$$

Beweis. Gemäß der Semantik des Allquantors gilt $\mathfrak{M} \models \forall\phi$ genau dann, wenn $\mathfrak{M}, \eta \models \phi$ für alle Belegungen η gilt. Letzteres ist nach dem Auswertungslemma äquivalent zur rechten Seite der behaupteten Äquivalenz. □

Entscheidend ist nunmehr

Satz 7.12 (Herbrand-Vollständigkeit)**.** *Sei Φ eine Menge von quantorenfreien Formeln über Σ. Dann sind äquivalent:*

1. *$\forall\Phi$ ist erfüllbar.*
2. *Es gibt ein ein Herbrandmodell $\mathfrak{M}$ mit $\mathfrak{M} \models \forall\Phi$.*
3. *$E(\Phi)$ ist aussagenlogisch erfüllbar.*

Dabei meinen wir in 3 mit *aussagenlogisch erfüllbar*, dass die Menge aussagenlogischer Formeln, die man erhält, wenn man alle atomaren Unterformeln als Atome mit merkwürdigen Namen ansieht, erfüllbar ist.

Beweis. 1. $\implies$ 3. und 2. $\implies$ 1. sind trivial. Wir zeigen 3. $\implies$ 2. Sei also $\kappa \models E(\Phi)$. Wir konstruieren dann ein Herbrand-Modell $\mathfrak{M}$ mit $\mathfrak{M} \models \forall\Phi$ per

$$\mathfrak{M}[\![P]\!] = \{(E_1, \ldots, E_n) \mid \kappa(P(E_1, \ldots, E_n)) = \top\}$$

für $P/n \in \Sigma$. Um zu zeigen, dass tatsächlich $\mathfrak{M} \models \forall\Phi$ gilt, reicht es nach dem Grundinstanzlemma zu zeigen, dass $\mathfrak{M} \models E(\Phi)$. Man zeigt durch Induktion über quantorenfreies ϕ, dass für jede Grundinstanz $\phi\sigma$

$$\mathfrak{M} \models \phi\sigma \iff \kappa \models \phi\sigma \tag{5}$$

gilt. Damit folgt dann die Behauptung, da ja $\kappa \models E(\Phi)$. Die Booleschen Fälle in der Induktion für (5) sind (wie meistens) leicht, und für atomare Formeln gilt die Äquivalenz nach der Konstruktion von $\mathfrak{M}$ und dem Auswertungslemma für (geschlossene) Terme:

$$\begin{aligned}
\mathfrak{M} \models P(E_1, \ldots, E_n)\sigma &\iff (\mathfrak{M}[\![E_1\sigma]\!], \ldots, \mathfrak{M}[\![E_n\sigma]\!]) \in \mathfrak{M}[\![P]\!] && \text{(Semantik)}\\
&\iff (E_1\sigma, \ldots, E_n\sigma) \in \mathfrak{M}[\![P]\!] && \text{(Auswertungslemma)}\\
&\iff \kappa(P(E_1\sigma, \ldots, E_n\sigma)) = \top && \text{(Definition von } \mathfrak{M}[\![P]\!]\text{)}\\
&\iff \kappa(P(E_1, \ldots, E_n)\sigma) = \top && \text{(Definition der Substitution)}\\
&\iff \kappa \models P(E_1, \ldots, E_n)\sigma && \text{(Semantik)}
\end{aligned}$$

(da ϕ quantorenfrei ist, gibt es keine weiteren Fälle). □

Der Satz gilt nicht für beliebige Formelmengen, z.B.

Beispiel 7.13. Man betrachte

$$\begin{aligned}
\Sigma &= \{a/0, P/1\}\\
\Phi &= \{\neg P(a), \exists x(P(x))\}\\
U_\Sigma &= \{a\}.
\end{aligned}$$

Die Formelmenge Φ ist offenbar in keinem einelementigen Modell erfüllbar, insbesondere also in keinem Herbrand-Modell, da Herbrandmodelle Grundbereich $U_\Sigma = \{a\}$ haben. Andererseits ist Φ aber offenbar erfüllbar, z.B. wähle $M = \{0, 1\}$, $\mathfrak{M}[\![a]\!] = 0$, $\mathfrak{M}[\![P]\!] = \{1\}$.

Da Herbrandmodelle über abzählbarem Σ abzählbar sind, folgt hieraus wie angekündigt

Satz 7.14 (Löwenheim/Skolem)**.** *Jede erfüllbare Menge von Sätzen über einer höchstens abzählbaren Signatur hat ein höchstens abzählbares Modell.*

Beweis. Durch Skolemisierung der gegebenen Formelmenge Ψ erhalten wir eine Formelmenge $\forall\Phi$ mit Φ quantorenfrei, über einer vergrößerten, aber weiterhin abzählbaren Signatur. Per Herbrand-Vollständigkeit hat $\forall\Phi$ ein abzählbares Modell (nämlich ein Herbrandmodell); dieses erfüllt dann auch Ψ. □

Als weitere Anwendung der Herbrand-Vollständigkeit erben wir den Kompaktheitssatz von der Aussagenlogik:

Satz 7.15. *Logik erster Stufe ist* kompakt*, d.h. wenn Φ eine Menge von Sätzen ist, so dass jede endliche Teilmenge von Φ erfüllbar ist, dann ist Φ erfüllbar.*

Beweis. Wir können annehmen, dass Φ aus Skolemformen besteht, d.h. $\Phi = \forall\Phi_0$ für eine Menge Φ_0 von quantorenfreien Formeln. Per Herbrand-Vollständigkeit ist dann Φ erfüllbar genau dann, wenn $E(\Phi_0)$ aussagenlogisch erfüllbar ist. Jede endliche Teilmenge Ψ von $E(\Phi_0)$ ist enthalten in einer Menge der Form $E(\bar{\Psi})$ für eine endliche Teilmenge $\bar{\Psi}$ von Φ_0. Nach Voraussetzung ist $\bar{\Psi}$ erfüllbar, damit auch $E(\bar{\Psi})$ und somit erst recht auch $\Psi \subseteq E(\bar{\Psi})$, d.h. $E(\Phi_0)$ ist endlich erfüllbar, also nach Kompaktheit der Aussagenlogik erfüllbar. □

Ferner erhalten wir Halbentscheidbarkeit des Gültigkeitsproblems der Prädikatenlogik erster Stufe:

Satz 7.16. *Es gibt einen Algorithmus, der alle gültigen Formeln in Logik erster Stufe aufzählt.*

Beweis. Dual zeigen wir, dass es einen Algorithmus gibt, der die *unerfüllbaren* Formeln auflistet. Wir können uns auf Skolemformen ϕ einschränken; es reicht, einen Algorithmus anzugeben, der mit der Antwort „unerfüllbar“ terminiert, wenn ϕ unerfüllbar ist, und ansonsten entweder „erfüllbar“ antwortet oder nicht terminiert. Wir erzeugen einfach nacheinander alle Grundinstanzen von ϕ; sobald wir eine aussagenlogisch unerfüllbare Menge erreichen, antworten wir „unerfüllbar“; falls die Menge der Grundinstanzen unendlich und aussagenlogisch erfüllbar ist, terminiert der Algorithmus also nicht. Im Sonderfall, dass die Menge der Grundinstanzen endlich und aussagenlogisch erfüllbar ist, antworten wir „erfüllbar“. □

7.2 Vollständigkeit der Resolution

Wir zeigen nunmehr wie angekündigt die Vollständigkeit der prädikatenlogischen Resolution. Wir reduzieren diese mittels Herbrandvollständigkeit auf die schon gezeigte Vollständigkeit der aussagenlogischen Resolution. Hierzu benötigen wir eine technische Aussage über das Verhältnis der beiden Resolutionsverfahren, für die wir wiederum die genaue Gestaltung der prädikatenlogischen Resolutionsregel eingehender diskutieren müssen.

Wir erinnern daran, dass die Anwendung von (RIF) auf den mgu als Substitution beschränkt ist. Wir bezeichnen mit (lRIF) die *liberalisierte RIF-Regel*, in der wir diese Einschränkung aufgeben und stattdessen beliebige Unifikatoren von $A_1, \ldots, A_n, B$ zulassen. Dies ändert offenbar nicht die Stärke des Systems:

Lemma 7.17. *Wenn sich aus einer Klauselmenge mittels (lRIF) die leere Klausel herleiten lässt, dann auch mittels (RIF).*

Beweis. Man zeigt durch Induktion über die Länge der Herleitung, dass jede mittels (lRIF) herleitbare Klausel eine Substitutionsinstanz einer mittels (RIF) herleitbaren Klausel ist. Damit folgt die Behauptung, da eine Klausel, die die leere Klausel als Substitutionsinstanz hat, selbst leer sein muss. □

Wir bezeichnen ferner mit (RI2F) (*resolution with implicit two-sided factoring*) die Regel

$$\frac{C, A_1, \ldots, A_n \qquad \neg B_1, \ldots, \neg B_k, D}{C\sigma, D\sigma} \quad \sigma = mgu(A_1, \ldots, A_n, B_1, \ldots, B_k)$$

Lemma 7.18. *(RI2F) ist aus (lRIF) herleitbar.*

Beweis. Wir lesen aus dem Beweis des Lemmas über gelöste Formen von Gleichungssystemen (Lemma 5.6) ab, dass wir $\sigma\sigma = \sigma$ annehmen können. Wir wenden (lRIF) mit der Substitution σ auf $A_1, \ldots, A_n, \neg B_1$ an (das ist bei (lRIF) eben erlaubt, obwohl σ i.a. nicht der mgu von $A_1, \ldots, A_n, B_1$ ist) und erhalten

$$C\sigma, \neg B_2\sigma, \ldots, \neg B_k\sigma, D\sigma.$$

Wir wenden dann (lRIF) mit der Substitution σ an auf $A_1, \ldots, A_n, B_2\sigma$ und erhalten (da $\sigma\sigma = \sigma$)

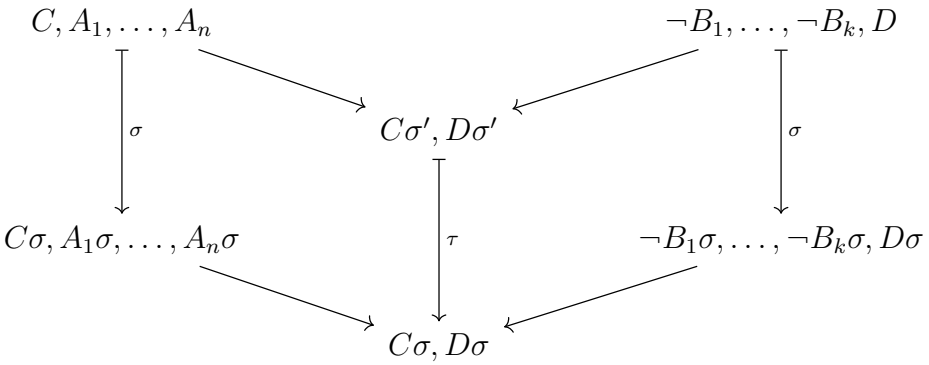
$$C\sigma, \neg B_3\sigma, \ldots, \neg B_k\sigma, D\sigma.$$

Wir fahren so fort und erhalten schließlich $C\sigma, D\sigma$. □

Bemerkung 7.19. Dagegen ist (RIF) nicht herleitbar aus der einfachen Resolutionsregel ohne Faktorisierung, Gegenbeispiel: Die bereits in Beispiel 7.2 erwähnte aus den Klauseln $\{P(x), P(y)\}, \{\neg P(z), \neg P(w)\}$ bestehende Klauselmenge.

Lemma 7.20 (Lifting-Lemma)**.** *Seien $C, A_1, \ldots, A_n$ und $\neg B_1, \ldots, \neg B_k, D$ prädikatenlogische Klauseln, und sei σ eine Grundsubstitution mit $A_i\sigma = B_j\sigma$ für alle i, j. Dann ist die zugehörige aussagenlogische Resolvente $C\sigma, D\sigma$ Grundinstanz einer prädikatenlogischen Resolvente von $C, A_1, \ldots, A_n$ und $\neg B_1, \ldots, \neg B_k, D$ nach (RI2F).*

Beweis. Setze $\sigma' = mgu(A_1, \ldots, A_n, B_1, \ldots, B_k)$. Dann gilt $\sigma = \sigma'\tau$ für eine Grundsubstitution τ, und $C\sigma, D\sigma$ ist unter τ Grundinstanz der prädikatenlogischen Resolvente $C\sigma', D\sigma'$:



□

Bemerkung 7.21. Der Beweis des Lifting-Lemmas ist also eher trivial; man beachte aber, dass das entsprechende Lemma für die Regel (RIF) mit nur einseitiger Faktorisierung *nicht* gilt: Gegeben prädikatenlogische Klauseln und σ wie im Lemma ist die aussagenlogische Resolvente $C\sigma, D\sigma$ i.a. nicht Grundinstanz einer Resolvente von $C, A_1, \ldots, A_n$ und $\neg B_1, \ldots, \neg B_k, D$ nach (RIF), da in einer solchen Resolvente bei $k > 1$ stets noch Literale der Form $\neg B_i\sigma$ verbleiben.

Damit gilt

Satz 7.22 (Vollständigkeit der prädikatenlogischen Resolution)**.** *Wenn Φ eine unerfüllbare Menge von prädikatenlogischen Klauseln ist, dann existiert eine Herleitung der leeren Klausel $\square$ aus Φ mittels prädikatenlogischer Resolution.*

Beweis. Nach Herbrandvollständigkeit ist $E(\Phi)$ aussagenlogisch unerfüllbar; nach der Vollständigkeit der aussagenlogischen Resolution existiert also eine Herleitung von $\square$ aus Grundinstanzen mittels aussagenlogischer Resolution. Indem wir das Lifting-Lemma auf alle Schritte dieses Resolutionsbeweises anwenden, erhalten wir eine Herleitung einer Klausel C aus Φ mittels (RI2F), und damit nach Lemma 7.18 auch per (lRIF), so dass $\square$ eine Grundinstanz von C ist. Dann ist aber $C = \square$, also per Lemma 7.17 auch per (RIF) herleitbar. $\square$

8 Quantorenelimination

Erfüllbarkeit von Formeln in Prädikatenlogik ist im allgemeinen unentscheidbar; dies zeigt man z.B. durch Reduktion des Postschen Korrespondenzproblems (siehe Skript „Ontologien im Semantic Web“). Erfüllbarkeit unter bestimmten Theorien kann dagegen in günstigen Fällen entscheidbar sein. Oft basieren Entscheidungsverfahren auf der Methode der *Quantorenelimination*, d.h. der äquivalenten Ersetzung von Formeln durch quantorenfreie Formeln.

Definition 8.1 (Theorie)**.** Eine *Theorie* $\mathcal{T} = (\Sigma, \Phi)$ besteht aus einer Signatur Σ und einer Menge Φ von Σ-Sätzen, ihren *Axiomen*. Ein *Modell* von $\mathcal{T}$ ist ein Σ-Modell $\mathfrak{M}$ mit $\mathfrak{M} \models \Phi$. Eine Σ-Formel ψ heißt *erfüllbar unter* $\mathcal{T}$, wenn ψ in einem Modell von $\mathcal{T}$ erfüllbar ist, d.h. wenn $\Phi \cup \{\psi\}$ erfüllbar ist. Wir schreiben $\mathcal{T} \models \psi$, wenn $\neg\psi$ unerfüllbar unter $\mathcal{T}$ ist, d.h. wenn $\Phi \models \psi$.

Beispiel 8.2. Die Theorie $\mathcal{T}_<$ der *dichten linearen Ordnungen ohne Endpunkte* hat die Signatur

$$\Sigma_< = \{</2\},$$

d.h. ein binäres Prädikat $<$, geschrieben in Infixnotation, sowie die Axiome

$\forall x.\, \neg(x < x)$	(Irreflexivität)
$\forall x, y, z.\, x < y \wedge y < z \rightarrow x < z$	(Transitivität)
$\forall x, y.\, x < y \vee x = y \vee y < x$	(Trichotomie)
$\forall x, y.\, x < y \rightarrow \exists z.\, x < z \wedge z < y$	(Dichte)
$\forall x \exists y.\, y < x$	
$\forall x \exists y.\, x < y$	(Endpunktfreiheit).

Ein Modell dieser Theorie sind z.B. die rationalen Zahlen mir der üblichen Interpretation von $<$; ein weiteres Modell sind die reellen Zahlen.

Wir werden mittels Quantorenelimination zeigen, dass Erfüllbarkeit unter $\mathcal{T}_<$ entscheidbar ist; man sagt kurz, dass $\mathcal{T}_<$ *entscheidbar* ist. Der allgemeine Ansatz ist hierbei wie folgt.

Definition 8.3. Eine Theorie $\mathcal{T} = \{\Sigma, \Phi\}$ *hat Quantorenelimination*, wenn für jede Σ-Formel ϕ eine quantorenfreie Σ-Formel ϕ' berechenbar ist, so dass $\mathcal{T} \models \phi \leftrightarrow \phi'$.

Fakt 8.4. *Wenn $\mathcal{T} = (\Sigma, \Phi)$ Quantorenelimination hat und Erfüllbarkeit von quantorenfreien Formeln unter $\mathcal{T}$ entscheidbar ist, dann ist $\mathcal{T}$ entscheidbar.*

Quantorenelimination lässt sich auf einen Spezialfall reduzieren:

Lemma 8.5. *Wenn für jede Σ-Formel ϕ der Form $\exists x(\alpha_1 \wedge \cdots \wedge \alpha_n)$ mit Literalen α_i eine quantorenfreie Formel ϕ' mit $\mathcal{T} \models \phi \leftrightarrow \phi'$ berechenbar ist, dann hat $\mathcal{T}$ Quantorenelimination.*

Beweis. Wir zeigen per Induktion über ψ, dass für jede Σ-Formel ψ eine äquivalente quantorenfreie Formel berechenbar ist. Wir nehmen dabei an, dass ψ nur $\exists$ enthält (d.h. $\forall$ durch $\exists$ definiert ist). Für Atome ist nichts zu zeigen, und die Booleschen Fälle sind trivial. Sei also ψ von der Form $\exists x.\, \chi$. Nach Induktionsvoraussetzung existiert ein quantorenfreies χ' mit $\mathcal{T} \models \chi \leftrightarrow \chi'$. Wir bringen χ' in DNF $\chi' \equiv \chi'_1 \vee \cdots \vee \chi'_n$, wobei die χ'_i Konjunktionen von Literalen sind. Per Vertauschung von $\exists$ mit $\vee$ (s.o.) gilt dann $\mathcal{T} \models (\exists x.\, \chi) \leftrightarrow ((\exists x.\, \chi'_1) \vee \cdots \vee (\exists x. \chi'_n))$, und die Quantoren auf der rechten Seite sind nach Voraussetzung eliminierbar. □

Damit zeigen wir

Satz 8.6. *Die Theorie der dichten linearen Ordnungen ohne Endpunkte hat Quantorenelimination.*

Beweis. Sei $\phi = \exists x.\, \alpha_1 \wedge \cdots \wedge \alpha_n$ mit Literalen α_i. Wir gehen schrittweise vor:

1. Wir eliminieren zunächst Negation: Per Irreflexivität, Transitivität und Trichotomie gilt $\mathcal{T} \models (\neg y = z) \leftrightarrow (y < z \vee z < y)$ und $\mathcal{T} \models (\neg(y < z) \leftrightarrow (y = z \vee z < y)$.

2. Damit erreichen wir eine Formel, die keine Negationen mehr, dafür aber wieder Disjunktionen enthält. Wie im Beweis von Lemma 8.5 formen wir ϕ dann in eine Disjunktion von existenzquantifizierten Formeln des richtigen Formats um, und machen mit einer solchen existenzquantifizierten Formel weiter.

3. Damit erreichen wir eine Formel $\psi = \exists x.\, \beta_1 \wedge \cdots \wedge \beta_n$ mit *Atomen* β_i. Wir eliminieren nun $=$:

   - Wenn $\beta_i = (y = y)$, dann streichen wir β_i.
   - Wenn $\beta_i = (y = x)$ oder $\beta_i = (x = y)$, dann streichen wir β_i und ersetzen in die verbleibenden β_j jeweils durch $\beta_j[y/x]$. Abschließend streichen wir den Existenzquantor (x kommt ja nun unter ihm nicht mehr vor) und sind fertig.

   Falls wir noch nicht fertig sind, ziehen wir von den verbleibenden Atomen nunmehr diejenigen, die x nicht erwähnen, vor den Existenzquantor (s.o.), unter dem dann kein $=$ mehr vorkommt.

4. Falls unter den verbleibenden β_j $x < x$ vorkommnt, ersetzen wir ψ durch $\bot$ (per Irreflexivität).

5. Der verbleibende existenzquantifizierte Anteil hat nunmehr (nach einfacher Umformung) die Form
$$\exists x.\ \bigwedge_{i=1}^{n} u_i < x \wedge \bigwedge_{j=1}^{m} x < v_j, \tag{6}$$
wobei die u_i und die v_j von x verschiedene Variablen sind. Diese Formel ist in $\mathcal{T}_<$ äquivalent zu
$$\bigwedge_{i=1}^{n}\bigwedge_{j=1}^{m} u_i < v_j. \tag{7}$$
Dabei ist die Implikation (6) $\Longrightarrow$ (7) klar per Transitivität. Für die Umkehrung bemerken wir, dass es per Trichotomie unter den u_i ein größtes geben muss; sei dies u_{i_0} (formal beweist man per Induktion eine Disjunktion mit n Disjunkten, eins für jedes in Frage kommende u_i). Entsprechend gibt es unter den v_j ein kleinstes, v_{j_0}. Dann gilt nach Voraussetzung $u_{i_0} < v_{j_0}$, so dass per Dichte x existiert mit $u_{i_0} < x$ und $x < v_{j_0}$; dieses x ist dann per Transitivität ein Zeuge für (6). Diese Argumentation setzt natürlich voraus, dass $n, m > 0$. Für $n > 0, m = 0$ und für $n = 0, m > 0$ führt man ein ähnliches Argument mittels Endpunktfreiheit durch; der Fall $n = m = 0$ ist trivial.

□

Um nun tatsächlich Entscheidbarkeit von $\mathcal{T}_<$ zu bekommen, fehlt uns noch die Entscheidbarkeit der Erfüllbarkeit quantorenfreier Formeln unter $\mathcal{T}$. Wir beobachten zunächst, dass bei der Quantorenelimination für $\mathcal{T}_<$ auch $\neg$ eliminiert wird. Ferner können wir mit Disjunktionen stets umgehen, indem wir zunächst in DNF transformieren und dann die Disjunkte der DNF der Reihe nach auf Erfüllbarkeit prüfen (natürlich hören wir damit auf, sobald wir einen erfüllbaren Disjunkt gefunden haben). Das verbleibende Problem ist, die Erfüllbarkeit einer Konjunktion atomarer Formeln der Form $x = y$ oder $x < y$ zu entscheiden. Da wir jetzt nur noch erfüllbarkeitsäquivalent umformen müssen, können wir atomare Formeln $x = y$ jeweils entfernen und in der verbleibenden Formel x durch y substituieren (das ist nicht mehr äquivalent, da ja die Aussage $x = y$ verloren geht, aber offenbar erfüllbarkeitsäquivalent). Zur Vereinfachung der Notation schreiben wir die verbleibende Konjunktion einfach als Menge C von Formeln der Form $x < y$. Folgender Algorithmus entscheidet Erfüllbarkeit von C unter $\mathcal{T}_<$:

1. Wenn in C Formeln $x < y$, $y < z$ enthalten sind mit $x < z$ nicht in C, füge $x < z$ zu C hinzu und mache bei Schritt 1 weiter.
2. Falls C eine Formel der Form $x < x$ enthält, antworte „unerfüllbar“, sonst „erfüllbar“.

Schritt 1 terminiert, da nur endlich (genauer: quadratisch) viele Formeln $x < z$ hinzugefügt werden können. Per Transitivität und Irreflexivität ist klar, dass der Algorithmus recht hat, wenn er „unerfüllbar“ antwortet. Um zu sehen, dass er auch im anderen Fall recht hat, zeigen wir

Lemma 8.7. *Wenn mit $x < y$ und $y < z$ stets auch $x < z$ in C ist und C keine Formel der Form $x < x$ enthält, dann ist C erfüllbar unter $\mathcal{T}_<$.*

Beweis. Nach Voraussetzung ist $<$ eine strikte partielle Ordnung auf den Variablen. Man kann dann die Variablen topologisch sortieren, d.h. in einer Folge $x_1, \ldots, x_n$ anordnen, so dass $i < j$, wenn $x_i < x_j$ in C ist. Dann $(\mathbb{Q}, <), \eta \models C$ mit $\eta(x_i) = i$. [Siehe AuD; zur Erinnerung: beim topologischen Sortieren entnimmt man einfach wiederholt ein minimales Element aus der partiellen Ordnung und fügt es als nächstes Element in eine Liste ein.] □

9 Vollständigkeit der Prädikatenlogik erster Stufe

Zur Vereinfachung arbeiten wir in Prädikatenlogik ohne Gleichheit (=) und ohne Funktionssymbole positiver Stelligkeit (aber mit Konstanten). Wir beweisen die Umkehrung des Korrektheitssatzes, d.h.

Satz 9.1 (Vollständigkeit der Prädikatenlogik erster Stufe)**.** *Sei Φ eine Menge von geschlossenen Formeln und ψ eine geschlossene Formel über einer Signatur Σ in Prädikatenlogik erster Stufe. Wenn ψ logische Folgerung aus Φ ist ($\Phi \models \psi$), dann ist ψ aus Φ herleitbar ($\Phi \vdash \psi$).*

Bemerkung 9.2. Eine Version des Satzes für beliebige Formeln (d.h. auch solche mit freien Variablen) ist leicht reduzierbar auf die obige Version, indem man einfach Variablen durch Konstanten ersetzt.

Wie schon im Falle der Aussagenlogik ist der Beweis der Vollständigkeit wesentlich schwieriger als der der Korrektheit. Der Satz wurde zuerst von Gödel bewiesen. Der Beweis, wie wir ihn hier präsentieren, geht auf Leon Henkin zurück; er basiert auf einer Reduktion der Prädikatenlogik erster Stufe auf die Aussagenlogik, für die wir die Vollständigkeit ja schon gezeigt haben. Im Groben besteht der Beweis aus den folgenden Schritten:

(A) Wir erweitern zunächst die Signatur Σ zu einer Signatur Σ_H, die für jede Formel $\phi(x)$ (mit höchstens einer freien Variablen x) über Σ_H eine *Zeugenkonstante* $c_{\phi(x)}$ enthält, gedacht als Name für ein $\phi(x)$ erfüllendes Element, wenn ein solches existiert. Ebenso erweitern wir die gegebene Formelmenge Φ um eine unendliche Menge $\mathscr{H}$ zusätzlicher Axiome, die *Henkin-Theorie*. Diese enthält insbesondere für jede Formel $\phi(x)$ ein Axiom $(\exists x.\, \phi(x)) \to \phi(c_{\phi(x)})$.

(B) (*Henkin-Elimination*) Wir beweisen dann für jede Formel ρ über Σ (die also keine Zeugenkonstanten erwähnt), dass aus $\Phi \cup \mathscr{H} \vdash \rho$ bereits $\Phi \vdash \rho$ folgt, in Worten: wenn sich ρ aus Φ unter Zuhilfenahme der Henkin-Theorie herleiten lässt, dann kann man ρ schon aus Φ allein herleiten .

(C) (*Henkin-Konstruktion*) Wir fassen nun Formeln in Logik erster Stufe als bloße aussagenlogische Formeln auf, indem wir alle Teilformeln der Form $\forall x.\, \phi$, $\exists x.\, \phi$ oder $P(\dots)$ als Atome ansehen. Aus einer Wahrheitsbelegung κ mit $\kappa \models \Phi \cup \mathscr{H}$ (*aussagenlogische* Erfülltheit) konstruieren wir dann ein Σ-Modell $\mathfrak{M}_\kappa$ mit $\mathfrak{M}_\kappa \models \Phi$ (*prädikatenlogische* Erfülltheit).

(Zu (C): Z.B. ist die Formel $(\forall x.\, \phi) \wedge \exists x.\, \neg\phi$ aussagenlogisch strukturgleich zur Formel $A \wedge B$, insbesondere zwar als prädikatenlogische Formel unerfüllbar, aber aussagenlogisch erfüllbar. Dagegen ist die Formel $(\forall x.\phi) \wedge \neg\forall x.\phi$ schon aussagenlogisch unerfüllbar, da sie strukturgleich zur aussagenlogischen Formel $A \wedge \neg A$ ist.) Damit läuft der Beweis des Vollständigkeitssatzes dann wie folgt: Wie schon im Falle der Aussagenlogik reicht es zu zeigen, dass jede konsistente Menge Φ von Sätzen in Logik erster Stufe erfüllbar ist. Per Henkin-Elimination ist mit Φ auch $\Phi \cup \mathscr{H}$ konsistent — als Menge von Formeln in Logik erster Stufe, und damit erst recht als Menge von aussagenlogischen Formeln, da ja nach der Uminterpretation allenfalls weniger Information und weniger deduktive Mittel als vorher zur Verfügung stehen, um einen Widerspruch herzuleiten. Per Vollständigkeit der Aussagenlogik ist damit $\Phi \cup \mathscr{H}$ erfüllbar als Menge aussagenlogischer Formeln; nach der Henkinkonstruktion folgt dann, dass Φ auch als Menge von Formeln in Logik erster Stufe erfüllbar ist.

Im einzelnen werden die Schritte der Beweisstrategie wie folgt umgesetzt.

(A) (Henkin-Theorie) Wir brauchen für jede Formel $\phi(x)$ eine Zeugenkonstante $c_{\phi(x)}$, *aber*: Zeugenkonstanten erzeugen neue Formeln, z.B. $\exists y.\, P\left(y, c_{\phi(x)}\right)$. Die Lösung besteht darin, die Hinzufügung von Zeugenkonstanten zu iterieren. Wir definieren also eine aufsteigende Folge $\Sigma = \Sigma_0 \subseteq \Sigma_1 \subseteq \Sigma_2 \subseteq \ldots$ von Signaturen, wobei jeweils Σ_{i+1} die vorhergehende Signatur Σ_i um neue Zeugenkonstanten $c_{\phi(x)}$ für alle Σ_i-Formeln $\phi(x)$ (mit $FV(\phi) = \{x\}$) erweitert, und setzen $\Sigma_H = \bigcup_{i=0}^{\infty} \Sigma_i$. Wir können dann für jedes $c_{\phi(x)}$ seinen *Geburtstag*

$$\min\{i \mid c_{\phi(x)} \in \Sigma_i\}$$

definieren. Damit sieht man auch, dass Σ_H in der Tat für jede Formel $\phi(x)$ über Σ_H eine Zeugenkonstante $c_{\phi(x)}$ enthält: wenn i der Geburtstag der jüngsten Zeugenkonstante in $\phi(x)$ ist, dann enthält $\Sigma_{i+1} \subseteq \Sigma_H$ eine Zeugenkonstante $c_{\phi(x)}$.

Die Henkin-Theorie $\mathscr{H}$ ist dann definiert als die (unendliche) Menge aller Instanzen der folgenden beiden Axiomenschemata:

H1: $(\exists x.\, \phi(x)) \to \phi(c_{\phi(x)})$

H2: $\phi(c) \to \exists x.\, \phi(x)$

(B) (Henkin-Elimination) Formeln das Typs H2 sind beweisbar in Logik erster Stufe und können daher offensichtlich in Beweisen in Logik erster Stufe als Annahmen weggelassen werden. Es bleibt zu zeigen, dass alle Instanzen von H1 eliminierbar sind. Wir wählen in einem gegebenen Beweis von ρ unter den verwendeten Instanzen von H1 die mit dem jüngsten $c_{\phi(x)}$. Wir bezeichnen die Menge aller anderen im Beweis verwendeten Instanzen von H1 mit $\mathscr{H}_0$, so dass

$$\Phi \cup \mathscr{H}_0 \cup \{(\exists x.\, \phi(x)) \to \phi(c_{\phi(x)})\} \vdash \rho.$$

Da $(\exists x.\, \phi(x)) \to \phi(c_{\phi(x)})$ sowohl aus $\neg\exists x.\, \phi(x)$ als auch aus $\phi(c_{\phi(x)})$ herleitbar ist, folgt

$$\Phi \cup \mathscr{H}_0 \cup \{\neg\exists x.\, \phi(x)\} \vdash \rho \text{ und } \Phi \cup \mathscr{H}_0 \cup \{\phi(c_{\phi(x)})\} \vdash \rho.$$

Da $c_{\phi(x)}$ in Φ, $\mathscr{H}_0$ und ρ nicht vorkommt (in $\mathscr{H}_0$ deswegen nicht, weil $c_{\phi(x)}$ die jüngste vorkommende Zeugenkonstante ist), folgt aus dem rechten Teil per ($\exists$E)

$$\Phi \cup \mathscr{H}_0 \cup \{\exists x.\, \phi(x)\} \vdash \rho,$$

also per Fallunterscheidung über $(\exists x.\, \phi(x)) \vee \neg\exists x.\, \phi(x)$ letztlich $\Phi \cup \mathscr{H}_0 \vdash \rho$. Damit ist eine Instanz von H1 eliminiert; Iterieren des Verfahrens eliminiert alle Instanzen von H1.

(C) (Henkin-Konstruktion): Wir setzen

$$\begin{aligned} M_\kappa &= \text{Menge der Konstanten in } \Sigma_{\mathscr{H}} \\ \mathfrak{M}_\kappa[\![c]\!] &= c \\ \mathfrak{M}_\kappa[\![P]\!] &= \{(c_1, \ldots, c_n) \mid \kappa(P(c_1, \ldots, c_n)) = \top\} \end{aligned}$$

Die Behauptung $\mathfrak{M}_\kappa \models \Phi$ folgt dann unmittelbar aus

Lemma 9.3 (Wahrheitslemma)**.** *Für jede geschlossene Formel ρ gilt $\mathfrak{M}_\kappa \models \rho$ genau dann, wenn $\kappa \models \rho$.*

Beweis. Induktion über die Größe von ρ. Wir nehmen an, dass in ρ nur Existenzquantoren vorkommen; das ist für Zwecke eines Vollständigkeitsbeweises deswegen zulässig, weil die Äquivalenz von $\forall x.\, \phi$ und $\neg\exists x.\, \neg\phi$ in unserem System herleitbar ist und alle logischen Konstrukte herleitbare Äquivalenz bewahren. Der Induktionsanfang ist $\rho = P(c_1, \ldots, c_n)$ (da ρ geschlossen ist, kommen keine Variablen unter den Argumenten von P vor); für solche Formeln gilt die Behauptung nach Konstruktion von $\mathfrak{M}_\kappa$. Die Booleschen Fälle ($\neg, \wedge$) sind klar.

Es bleibt der Fall $\rho = (\exists x.\, \phi(x))$; wir handeln die Implikationen getrennt ab:

„$\Rightarrow$“: Sei $\mathfrak{M}_\kappa \models \exists x.\, \phi(x)$. Da jedes Element von M_κ die Interpretation einer Konstanten ist, existiert dann per Substitutionslemma $c \in \Sigma_{\mathscr{H}}$ mit $\mathfrak{M}_\kappa \models \phi(c)$. Da $\phi(c)$ geschlossen und kleiner als $\exists x.\, \phi(x)$ ist, folgt nach Induktionsvoraussetzung $\kappa \models \phi(c)$. Mit H2 erhalten wir schließlich $\kappa \models \exists x.\, \phi(x)$.

„$\Leftarrow$“: Sei $\kappa \models \exists x.\, \phi(x)$. Nach H1 folgt dann $\kappa \models \phi(c_{\phi(x)})$; da $\phi(c_{\phi(x)})$ geschlossen und kleiner als $\exists x.\, \phi(x)$ ist, folgt nach Induktionsvoraussetzung $\mathfrak{M}_\kappa \models \phi(c_{\phi(x)})$ und damit $\mathfrak{M}_\kappa \models \exists x.\, \phi(x)$. □

Es folgt nunmehr erneut

Korollar 9.4 (Kompaktheit)**.** *Jede* endlich erfüllbare *Formelmenge* Φ *(d.h. jede endliche Teilmenge von* Φ *ist erfüllbar) ist erfüllbar*

Beweis. Da Beweise endliche Objekte sind, gilt die analoge Eigenschaft mit „konsistent“ statt „erfüllbar“; nach Vollständigkeit und Korrektheit sind aber Konsistenz und Erfüllbarkeit äquivalent. □

Beispiel 9.5. Sei Φ eine Axiomatisierung der natürlichen Zahlen in Logik erster Stufe (z.B. die Peano-Axiome, mit 0 und Sukzessorfunktion s sowie einem Axiomenschema für Induktion). Dann ist $\Phi \cup \{c \geqslant s^n(0) | n \in \mathbb{N}\}$ endlich erfüllbar, somit nach Kompaktheit erfüllbar — d.h. wir können in Logik erster Stufe durch keine noch so raffinierte Axiomatisierung die Existenz von Nicht-Standard-Zahlen ausschließen.

Bemerkung 9.6. Vollständigkeit ist durchaus keine selbstverständliche Eigenschaft einer Logik. Man unterscheidet im allgemeinen zwischen *starker Vollständigkeit*, wie wir sie oben für Prädikatenlogik erster Stufe bewiesen haben, und *schwacher Vollständigkeit*. Eine Logik, mit logischer Folgerungsrelation $\models$, ist per Definition stark vollständig, wenn aus $\Phi \models \psi$ stets $\Phi \vdash \psi$ folgt (*Korrektheit* ist die umgekehrte Implikation), und schwach vollständig, wenn jede gültige Formel herleitbar ist, d.h. wenn aus $\models \psi$ stets $\vdash \psi$ folgt. Wenn eine Logik korrekt und stark vollständig ist und außerdem das Beweissystem *finitär* ist, d.h. wenn jede Regel nur endlich viele Prämissen hat (das schließt Regeln aus, die z.B. aus $\phi(n)$ für alle konkreten natürlichen Zahlen n die Formel $\forall x \in \mathbb{N}.\, \phi(x)$ schließen), dann ist die Logik, mit der gleichen Argumentation wie oben, *kompakt*, d.h. jede endlich erfüllbare Formelmenge ist erfüllbar.

In *Prädikatenlogik monadisch zweiter Stufe* (MSO) hat man zusätzlich zu Quantoren über Individuenvariablen wie in Prädikatenlogik erster Stufe noch Variablen für Teilmengen des Grundbereichs, also für unäre Prädikate, und Quantoren darüber. Wir nennen die neuen Variablen *Mengenvariablen* und schreiben sie $P, Q, \ldots$; wir verwenden sie ansonsten wie Prädikatensymbole. Sei dann Φ die Formelmenge über der Signatur $\Sigma = \{0, suc_{\searrow}, \geqslant\}$ bestehend aus den

Formeln

$$\forall P.\,(P(0) \wedge (\forall x.\,P(x) \to P(s(x)))) \to \forall x.\,P(x)$$
$$\forall x.\,\neg s(x) = 0$$
$$\forall x, y.\,s(x) = s(y) \to x = y$$
$$\forall x, y.\,x \geqslant y \leftrightarrow \exists z.\,x = y + z.$$

Dann definiert Φ eindeutig (bis auf irrelevante Umbenennungen der Elemente des Grundbereichs) die natürlichen Zahlen mit der üblichen Lesart von $\geqslant$: das zweite und dritte Axiom stellen sicher, dass man lauter verschiedene Elemente $s^n(0)$ für $n \geqslant 0$ hat, das erste Axiom sorgt dafür, dass dies tatsächlich alle Elemente sind, und das letzte Axiom definiert $\geqslant$. Damit ist dann offenbar

$$\Phi \cup \{c \geqslant s^n(0) \mid n \in \mathbb{N}\}$$

endlich erfüllbar, aber nicht erfüllbar. Somit hat MSO keine stark vollständige finitäre Axiomatisierung. (Nach dem deutlich schwerer zu beweisenden Gödelschen Unvollständigkeitssatz ist MSO auch nicht schwach vollständig.)

A Mathematische Notation

Für Schlüsse zwischen mathematischen Aussagen schreiben wir $\Rightarrow$, zum Beispiel gilt für alle natürlichen Zahlen $n \in \mathbb{N}$:

$$n + 2 \text{ ist Primzahl} \quad \Longrightarrow \quad n \text{ ist ungerade}$$

Für mathematische Äquivalenzen schreiben wir $\Leftrightarrow$, zum Beispiel gilt für alle natürlichen Zahlen n:

$$n \text{ ist gerade} \quad \Longleftrightarrow \quad n + 2 \text{ ist gerade}$$

A.1 Mengen

Grundlegende Relation zwischen Mengen ist das Elementverhältnis $\in$. Die Formel $x \in X$ liest sich „x ist Element der Menge X“. Mengen sind allein durch ihre Elemente bestimmt, d.h. zwei Mengen sind gleich, wenn sie die gleichen Elemente haben. Übliche Relationen und Operationen auf Mengen sind

- *Teilmengenbeziehung/Inklusion:* A ist *Teilmenge* von B, Schreibweise: $A \subseteq B$, wenn aus $x \in A$ stets $x \in B$ folgt, d.h. B mindestens alle Elemente von A enthält. A heißt *echte Teilmenge* von B, wenn $A \subseteq B$ und $A \neq B$.

- *Vereinigung:* $A \cup B$ ist die Menge mit $x \in A \cup B$ genau dann, wenn $x \in A$ oder $x \in B$.

- *Durchschnitt:* $A \cap B$ ist die Menge mit $x \in A \cap B$ genau dann, wenn $x \in A$ und $x \in B$.

- *Disjunktheit:* Mengen A und B sind *disjunkt*, wenn $A \cap B = \varnothing$.

- *Mengendifferenz:* $A \backslash B$ ist die Menge mit $x \in A \backslash B$ genau dann, wenn $x \in A$, aber nicht $x \in B$.

- *Mengenkomprehension:* Wenn A eine Menge ist und $P(x)$ eine Aussage über x, so ist $\{x \in A \mid P(x)\}$ eine Teilmenge von A, die Menge derjenigen Elemente x von A, die $P(x)$ erfüllen.

- *Große Vereinigung/Großer Durchschnitt:* Wenn $\mathfrak{A}$ eine Menge von Mengen ist, dann schreibt man

$$\bigcup \mathfrak{A} = \{x \mid x \in A \text{ für ein } A \in \mathfrak{A}\}$$
$$\bigcap \mathfrak{A} = \{x \mid x \in A \text{ für alle } A \in \mathfrak{A}\}.$$

- *Potenzmenge:* Für eine Menge A ist die *Potenzmenge* $\mathcal{P}(A)$ die Menge aller Teilmengen von A:

$$\mathcal{P}(A) = \{B \mid B \subseteq A\}.$$

- *Kreuzprodukt/kartesisches Produkt:* Wenn A und B Mengen sind, dann schreibt man

$$A \times B = \{(x, y) \mid x \in A, y \in B\}.$$

- *Disjunkte Vereinigung:* Wenn A und B Mengen sind, dann schreibt man

$$A \cup B = (\{0\} \times A) \cup (\{1\} \times B).$$

für deren disjunkte Vereinigung. Wenn A und B bereits disjunkt sind (also $A \cap B = \varnothing$, d. h. keine gemeinsamen Elemente haben), dann entspricht $A \cup B$ genau $A \cup B$. Falls hingegen A und B gemeinsame Elemente haben, dann kommen diese zweimal in $A \cup B$ vor. Wenn A und B endlich sind, dann gilt deshalb $|A \cup B| = |A| + |B|$.

Ein häufig verwendetes Verfahren zum Beweis der Gleichheit zweier Mengen A, B ist, beide Inklusionen zwischen A und B zu zeigen, d.h. $A \subseteq B$ und $B \subseteq A$.

A.2 Relationen und Funktionen

Eine (*zweistellige* oder *binäre*) *Relation* zwischen Mengen A und B ist eine Teilmenge $R \subseteq A \times B$. Für $x \in A$, $y \in B$ schreibt man üblicherweise xRy als Abkürzung für $(x, y) \in R$. Allgemeiner betrachtet man oft auch Relationen anderer Stelligkeit; z.B. ist eine *einstellige* (*unäre*) Relation auf A einfach eine Teilmenge von A, und eine *dreistellige* (*ternäre*) Relation eine Teilmenge von $A \times A \times A$.

Eine Relation $f \subseteq A \times B$ heißt

- *linkstotal*, wenn für jedes $x \in A$ ein $y \in B$ existiert mit $(x, y) \in f$, und
- *rechtseindeutig*, wenn für $x, y, y' \in A$ aus $(x, y), (x, y') \in f$ stets $y = y'$ folgt.

Eine linkstotale und rechtseindeutige Relation heißt *Abbildung* oder *Funktion* von A nach B, Notation: $f\colon A \to B$. Die beiden Eigenschaften rechtfertigen die Schreibweise $f(x)$ für das eindeutig bestimmte y mit $(x, y) \in f$. Für $C \subseteq A$ bezeichnet dann $f[C]$ das *Bild* von C unter f, d.h.

$$f[C] = \{y \in B \mid f(x) = y \text{ für ein } x \in C\},$$

und für $D \subseteq B$ bezeichnet $f^{-1}[D]$ das *Urbild* von D, d.h.

$$f^{-1}[D] = \{x \in A \mid f(x) \in D\}.$$

Allgemein hat man für Mengenkomprehensionen der Form $\{y \mid f(x) = y \text{ für ein } x \text{ mit } P(x)\}$ die Kurzform $\{f(x) \mid P(x)\}$, z.B. in dieser Schreibweise

$$f[C] = \{f(x) \mid x \in C\}.$$

Eine Abbildung $f : A \to B$ heißt *injektiv*, wenn für alle $x_1, x_2 \in A$ aus $f(x_1) = f(x_2)$ bereits $x_1 = x_2$ folgt. Ferner ist f *surjektiv*, wenn für jedes $y \in B$ ein $x \in A$ existiert mit $f(x) = y$. Eine Abbildung ist *bijektiv*, wenn sie sowohl injektiv als auch surjektiv ist. Z.B. ist die durch $f(n) = 2n$ definierte Abbildung $f : \mathbb{N} \to \mathbb{N}$ injektiv, aber nicht surjektiv; die Abbildung $f : \mathbb{R} \to \mathbb{R}$ mit $f(x) = x^3 - x^2$ ist surjektiv, aber nicht injektiv; und die Abbildung $f : \mathbb{Z} \to \mathbb{Z}$ mit $f(x) = x+1$ ist bijektiv. Typische Beispiele injektiver Abbildungen sind ferner Abbildungen der Form $i\colon B \to A$ mit $i(x) = x$ für $B \subseteq A$; eine typische Klasse surjektiver Abbildungen sind *Quotientenabbildungen* $q\colon A \to A/\!\sim$, $q(x) = [x]_\sim$ für Äquivalenzrelationen $\sim \subseteq A \times A$.

Die *identische Abbildung* $id_A : A \to A$ ist definiert durch $id_A(x) = x$; diese Abbildung ist bijektiv. Gegeben Abbildungen $f : A \to B$ und $g : B \to C$ ist die *Komposition* $g \circ f : A \to C$ definiert durch $g \circ f(x) = g(f(x))$. Wenn $f : A \to B$ eine bijektive Abbildung ist, dann existiert eine *inverse Abbildung* $g : B \to A$ mit $g \circ f = id_A$, $f \circ g = id_B$.

A.3 Ordnungen und Äquivalenzen

Eine Relation $R \subseteq A \times A$ ist

- *reflexiv*, wenn xRx für alle $x \in A$;
- *symmetrisch*, wenn für alle $x, y \in A$ aus xRy yRx folgt;
- *transitiv*, wenn für alle $x, y, z \in A$ aus xRy und yRz xRz folgt;
- *antisymmetrisch*, wenn für alle $x, y \in A$ aus xRy und yRx bereits $x = y$ folgt.

Eine Relation $R \subseteq A \times A$ heißt *Äquivalenzrelation* (auf A), wenn sie reflexiv, symmetrisch und transitiv ist, und *(partielle) Ordnung* (auf A), wenn sie reflexiv, transitiv und antisymmetrisch ist. Beispiel: Auf den ganzen Zahlen ist die durch

$$nRm \text{ genau dann, wenn } n - m \text{ durch 3 teilbar ist,}$$

definierte Relation eine Äquivalenzrelation; auf den natürlichen Zahlen ist die Relation „teilt“ eine Ordnung (man überlege sich, warum das auf den ganzen Zahlen nicht stimmt). Eine Ordnung R auf A heißt *total* oder *linear*, wenn für alle $x, y \in A$ xRy oder yRx gilt. Nicht alle Ordnungen sind total, z.B. ist $\subseteq$ eine Ordnung auf $\mathcal{P}(A)$, und wenn z.B. $x, y \in A$, $x \neq y$, so gilt weder $\{x\} \subseteq \{y\}$ noch $\{y\} \subseteq \{x\}$. Auch die Teilbarkeitsordnung ist nicht total.

Wenn $\leqslant$ eine Ordnung auf A ist, so heißt ein Element $x \in A$ *größtes* Element von A, wenn $y \leqslant x$ für alle $y \in A$. Im Gegensatz hierzu heißt x *maximal*, wenn für alle $y \in A$ aus $x \leqslant y$ bereits $x = y$ folgt. Eine Menge kann viele maximale Elemente haben. Z.B. sind in der Menge aller echten Teilmengen von A, geordnet wieder durch $\subseteq$, alle Mengen der Form $A \backslash \{x\}$ für $x \in A$ maximal.

Wenn $\sim$ eine Äquivalenzrelation auf A ist, dann bezeichnet man mit $[x]_\sim$ oder, wenn $\sim$ aus dem Kontext klar ist, einfach $[x]$ für die *Äquivalenzklasse* von $x \in A$, gegeben durch

$$[x]_\sim = \{y \in A \mid x \sim y\}.$$

Die Äquivalenzklassen bilden eine *disjunkte Zerlegung* von A, d.h. jedes Element von A gehört zu genau einer Äquivalenzklasse. Insbesondere sind je zwei Äquivalenzklassen entweder gleich oder disjunkt. Mit

$$A/\sim = \{[x]_\sim \mid x \in A\}$$

bezeichnet man die *Quotientenmenge von A modulo $\sim$*.

B Induktion

Induktion ist das Prinzip der Reduktion einer Aussage über ein Objekt auf gleichartige Aussagen über „einfachere“ Objekte in einem jeweils geeigneten Sinn. Wenn Objekte nur endlich oft „einfacher werden“ können, erreicht man so nach endlich vielen Reduktionen ein Objekt, das nicht mehr einfacher werden kann; wenn man außerdem für solche Objekte die entsprechende Aussage beweisen kann („Induktionsanfang“), hat man die Aussage für alle Objekte bewiesen. Das Prinzip ist das gleiche wie bei der Programmierung rekursiver Funktionen – dort ruft man (jedenfalls dann, wenn man an Terminierung interessiert ist) eine Funktion rekursiv mit

Argumenten auf, die „einfacher“ als das ursprüngliche Argument sind, und hat Basisfälle, in denen keine rekursiven Aufrufe stattfinden.

Wir fassen kurz die wesentlichen im weiteren benötigten Induktionsprinzipien zusammen. Wir verweisen auch auf den auf der Veranstaltungshomepage verfügbaren Text von Thomas Voß.

B.1 Induktion über natürliche Zahlen

Das vermutlich bekannteste Induktionsprinzip ist die Induktion über natürliche Zahlen (die, mit gewisser Variation, immer in irgendeiner Form entweder als Axiom postuliert oder konstruktiv sichergestellt wird). In der einfachsten Form lautet das Prinzip wie folgt. Wenn eine Aussage $P(n)$ über natürliche Zahlen n (die *Induktionsbehauptung*) die beiden Eigenschaften

1. *Induktionsanfang*: P gilt für die 0 (als Formel: $P(0)$) und
2. *Induktionsschritt*: für jede natürliche Zahl n folgt $P(n+1)$ aus $P(n)$, als Formel:

$$\forall n \in \mathbb{N}.\, (P(n) \to P(n+1))$$

   (man bezeichnet hier $P(n)$ als die *Induktionsvoraussetzung*)

erfüllt, dann gilt P für jede natürliche Zahl, als Formel:

$$\forall n \in \mathbb{N}.\, P(n).$$

(Wir verwenden hier später formal eingeführte Notation vorwegnehmend, insbesondere Allquantifizierung $\forall x.\, \phi$, zu lesen „für alle x gilt ϕ“. Der Geltungsbereich des Quantors reicht immer so weit nach rechts wie möglich; im Moment setzen wir aber zur Vermeidung von Missverständnissen gelegentlich zusätzliche Klammern.)

Als Beispiel diene hier folgende einfache Identität:

$$\sum_{i=1}^{n}(2i-1) = n^2.$$

Zum besseren Abgleich mit dem allgemeinen Induktionsprinzip bezeichnen wir diese Aussage mit $P(n)$. Man beweist $\forall n.\, P(n)$ durch Induktion über n:

- Induktionsanfang: Es gilt $\sum_{i=1}^{0}(2i-1) = 0 = 0^2$ (also $P(0)$).
- Induktionsschritt: Sei $n \in \mathbb{N}$, so dass $\sum_{i=1}^{n}(2i-1) = n^2$ (also $P(n)$); zu zeigen ist dann $P(n+1)$, also $\sum_{i=1}^{n+1}(2i-1) = (n+1)^2$. Man rechnet wie folgt:

$$\begin{aligned}\sum_{i=1}^{n+1}(2i-1) &= \sum_{i=1}^{n}(2i-1) + 2(n+1) - 1 \\ &\overset{IV}{=} n^2 + 2(n+1) - 1 \\ &= n^2 + 2n + 1 = (n+1)^2.\end{aligned}$$

  Hierbei haben wir mit IV den Umformungsschritt markiert, in dem die Induktionsvoraussetzung $P(n)$ angewendet wird.

Course-Of-Values Induction Nicht immer führt Induktion nach obigem Schema zum Ziel. Wenn wir z.B. den Fundamentalsatz der Arithmetik

Jede positive natürliche Zahl ist ein endliches Produkt von Primzahlen

beweisen wollen, wird uns die Annahme, dass n ein Produkt von Primzahlen ist, erkennbar nicht weiterhelfen beim Beweis der Behauptung, dass $n+1$ ein Produkt von Primzahlen ist (im Gegenteil teilen ja die Primzahlen, aus denen n zusammengesetzt ist, $n+1$ gerade *nicht*). Stattdessen verwenden wir folgendes stärkere Induktionsprinzip:

Satz B.1 (Course-of-Values Induction)**.** *Sei $P(n)$ eine Aussage über natürliche Zahlen.*[2] *Wenn für jedes n aus $\forall k < n.\, P(k)$ bereits $P(n)$ folgt, als Formel:*

$$\forall n.\, ((\,\forall\, k < n.\, P(k)) \to P(n)\,), \tag{$*$}$$

so gilt P für jede natürliche Zahl (als Formel: $\forall n\, P(n)$).

Beweis. In der Tat lässt sich diese Prinzip mittels normaler Induktion über n beweisen. Dies ist gleichzeitig eine Illustration des Prinzip der *Verstärkung des Induktionsziels*: Wenn sich eine Aussage $P(n)$ nicht durch Induktion beweisen lässt, kommt man oft weiter, wenn man stattdessen eine *stärkere* Aussage $Q(n)$ (d.h. eine Aussage $Q(n)$, so dass für jedes n die Aussage $P(n)$ aus $Q(n)$ folgt) per Induktion beweist. Man hat dann zwar im Induktionsschritt mehr zu zeigen, hat aber dazu eine stärkere Induktionsvoraussetzung zur Verfügung. Auch im vorliegenden Fall kann man sich überzeugen, dass die Induktionsvoraussetzung $P(n)$ unter den Annahmen des Satzes nicht ausreicht, um $P(n+1)$ zu folgern (dazu braucht man $P(k)$ für alle $k < n+1$, die Induktionsvoraussetzung liefert dies aber nur für den Fall $k = n$). Stattdessen beweisen wir unter den Annahmen des Satzes die stärkere Aussage

$$\forall\, k \leqslant n.\, P(k)$$

durch Induktion über n:

- Induktionsanfang: nach der Annahme $(*)$ können wir $P(0)$ folgern, wenn $P(k)$ für alle natürlichen Zahlen $k < 0$ gilt. Da es keine solchen Zahlen gibt, ist dies der Fall, also gilt $P(0)$. Damit gilt natürlich auch $\forall\, k \leqslant 0.\, P(k)$.
- Induktionsschritt: Es gelte $\forall\, k \leqslant n.\, P(k)$ (dies ist die Induktionsvoraussetzung); zu zeigen ist $\forall\, k \leqslant n+1.\, P(k)$. Für die meisten k folgt dies sofort aus der Induktionsvoraussetzung; zu zeigen bleibt $P(n+1)$. Nach der Annahme des Satzes reicht es dazu aus, zu zeigen, dass $P(k)$ für alle $k < n+1$ gilt; das ist aber gerade unsere Induktionsvoraussetzung $\forall\, k \leqslant n.\, P(k)$. □

Mit diesem Prinzip beweisen wir nun den eingangs erwähnten Fundamentalsatz der Arithmetik: Sei $n > 0$. Wir nehmen an, jede Zahl $k < n$ sei ein Produkt von endlich vielen Primzahlen (Sonderfälle hierbei per Konvention: 1 ist ein Produkt von 0 Primzahlen, und jede Primzahl ist Produkt aus einer einzigen Primzahl). Wir müssen zeigen, dass dann n selbst ein Produkt endlich vieler Primzahlen ist. Wir unterscheiden dazu zwei Fälle: Wenn n selbst prim ist oder

[2] Wir sind hier, wie schon vorher, ungenau bezüglich der Ausdrucksmittel, die zur Formulierung von P zur Verfügung stehen, insofern bleibt der Satz hier zum Teil informell. Man könnte mit weiter unten eingeführtem Wissen z.B. verlangen, dass P eine Formel in Logik erster Stufe ist, die nur 0 und Nachfolger erwähnt.

$n = 1$, dann ist n per eben vereinbarter Konvention ein endliches Produkt von Primzahlen. Andernfalls ist n zusammengesetzt, also $n = km$ mit $k, m < n$. Nach Induktionsvoraussetzung sind dann k und m endliche Produkte von Primzahlen, also $k = p_1 \dots p_r$, $m = q_1 \dots q_s$ mit $p_1, \dots, p_r, q_1, \dots, q_s$ Primzahlen; damit ist auch $n = km = p_1 \dots p_r q_1 \dots q_s$ ein Produkt endlich vieler Primzahlen.

B.2 Backus-Naur-Form

Die Backus-Naur-Form (BNF) ist eine verbreitete Art, die Syntax von formalen Sprachen (genauer: sogenannten kontextfreien Sprachen, s. BFS) mittels sogenannter *Grammatiken* darzustellen. Man arbeitet mit zwei Alphabeten T und N von *terminalen* und *nichtterminalen* Symbolen; wir beschränken uns hier der Einfachheit halber auf den Fall mit nur einem nichtterminalen Symbol. Eine BNF (oder einfach eine *Grammatik*) hat dann die Form

$$n \ ::= \ B_1 \mid \dots \mid B_m$$

mit $N = \{n\}$ und $B_1, \dots, B_m \in (N \cup T)^*$; die B_i heißen *Alternativen*. Eine Alternative

$$B_i = w_0 n w_1 n \dots n w_k \text{ mit } w_0, \dots, w_k \in T^*$$

lesen wir als eine Regel zur Einführung von *Instanzen* von n:

> wenn $v_1, \dots, v_k$ Instanzen von n sind, dann auch $w_0 v_1 w_1 v_2 \dots v_k w_k$.

Die so entstehende Regelmenge lesen wir *induktiv*, d.h. ein Wort über T ist dann eine Instanz von n, wenn sich dies durch endlich viele Regelanwendungen herleiten lässt. Insbesondere sind alle Instanzen von n endliche Wörter.

Unser erstes Beispiel ist die Grammatik

$$\phi, \psi ::= \bot \mid A \mid \phi \wedge \psi \mid \neg\, \phi \qquad (A \in \mathscr{A}),$$

wobei wir zwecks leichterer Notation zwei verschiedene Namen ϕ, ψ für dasselbe Nichtterminal verwenden. Dabei ist $\mathscr{A}$ eine gegebene Menge von *Atomen*, d.h. nicht weiter unterteilbaren Aussagen. Wir haben hier also $T = \mathscr{A} \cup \{\bot, \wedge, \neg\}$ (bzw. $T = \mathscr{A} \cup \{\bot, \wedge, \neg,), (\}$, wobei wir aber Klammern in der Grammatik implizit lassen und Terme nur bei Bedarf klammern). Instanzen von ϕ nennen wir *(aussagenlogische) Formeln.* Dies entspricht in der Lesart als Regeln dem Regelsystem

1. $\bot$ und $A \in \mathscr{A}$ sind Formeln.
2. Wenn ϕ eine Formel ist, dann auch $\neg\phi$.
3. Wenn ϕ und ψ Formeln sind, dann auch $\phi \wedge \psi$.

Z.B. ist $A \wedge \neg\bot$ eine Formel; durch Rückwärtsanwenden der Regeln sieht man dies wie folgt:

- $A \wedge \neg\bot$ ist eine Formel, denn (3):
- $A \in \mathscr{A}$ ist eine Formel (1) und $\neg\bot$ ist eine Formel, denn (2):
- $\bot$ ist eine Formel (1).

B.3 Strukturelle Induktion

Man kann nun Induktion nicht nur über den natürlichen Zahlen verwenden, sondern auch über im wesentlichen allen endlichen azyklischen Datenstrukturen (und sogar noch allgemeineren Objekten, was hier aber zu weit führt) – insbesondere z.B. über mittels einer Grammatik definierten Objekten, wie etwa aussagenlogischen Formeln.

Wir stellen uns ein solches Objekt dabei eher als eine baumförmige Struktur als einen flachen String vor (d.h. wir stellen uns z.B. Formeln fertig geparst vor). In ihrer einfachsten Form besagt strukturelle Induktion dann, dass jede Eigenschaft, die für alle Blätter gilt und sich von direkten Kindern auf Elternknoten vererbt, für alle Bäume gilt.

Formal stellt sich dies wie folgt dar: Aus einer BNF

$$n ::= B_1 \mid \cdots \mid B_m$$

erhalten wir ein Induktionsprinzip zum Beweis einer Eigenschaft P für alle Instanzen von n, in dem m verschiede Induktionsschritte durchzuführen sind, einer für jedes B_i. Der Induktionsschritt für $B_i = w_0 n w_1 n \ldots n w_k$ verlangt, dass man unter der Annahme, dass Instanzen $v_1, \ldots, v_k$ von n bereits P erfüllen (Induktionsvoraussetzung), zeigt, dass auch die neu erzeugte Instanz

$$w_0 v_1 w_1 \ldots v_k w_k$$

die Eigenschaft P erfüllt. Wenn n nicht in B_i vorkommt, B_i also nur aus terminalen Symbolen besteht, ist der Induktionschritt für B_i natürlich eher eine Art Induktionsanfang (von denen es dann mehrere geben kann), da man keine Induktionsvoraussetzung hat. Die Rechtfertigung dieses Induktionsprinzips, d.h. der Beweis der Tatsache, dass man nach Durchführung aller Induktionsschritte tatsächlich folgern kann, dass $P(w)$ für alle Instanzen w von n gilt, ist per Course-of-Values-Induktion über die Länge von w, bei Fallunterscheidung über die Regel, mit der man w erzeugt hat.

Ein erstes Beispiel dieses Prinzips ist die eingangs diskutierte Induktion über natürliche Zahlen. Wir können nämlich die natürlichen Zahlen als die Instanzen der Grammatik

$$n ::= z \mid s(n)$$

ansehen – diese sind z, $s(z)$, $s(s(z))$, $s(s(s(z)))$, ... Dann haben wir gemäß den obigen Vorschriften beim Beweis einer Eigenschaft P für alle Instanzen von n zwei Induktionsschritte, einen für jede Alternative der Grammatik:

- z: Hier kommt n nicht vor, zu zeigen ist also einfach $P(z)$. Dies entspricht dem üblichen Induktionsanfang (z steht für 0).
- $s(n)$: Hier ist zu zeigen, dass, wenn n die Eigenschaft P hat, dann auch $s(n)$, wobei jetzt n als Platzhalter für eine beliebige Instanz steht. Da s für die Nachfolgerfunktion steht, entspricht dies genau dem üblichen Induktionsschritt.

Für aussagenlogische Formeln erhalten wir ein strukturelles Induktionsprinzip mit vier Induktionsschritten (von denen zwei in Wirklichkeit Induktionsanfänge sind): Um zu zeigen, dass eine Eigenschaft $P(\phi)$ für alle aussagenlogischen Formeln ϕ gilt, zeigt man

- $P(\bot)$;

- $P(A)$ für alle $A \in \mathscr{A}$;
- wenn $P(\phi)$, dann auch $P(\neg\phi)$; und
- wenn $P(\phi)$ und $P(\psi)$, dann auch $P(\phi \wedge \psi)$.

Wir verwenden dieses Prinzip ganz entsprechend auch zur *rekursiven Definition* von Funktionen, wie etwa in der folgenden Definition.

Definition B.2 (Atome einer Formel)**.** Die Menge $\mathrm{At}(\phi)$ der in ϕ vorkommenden Atome ist rekursiv definiert durch

$$\begin{aligned} \mathrm{At}(A) &= \{A\} \\ \mathrm{At}(\top) &= \varnothing \\ \mathrm{At}(\neg\phi) &= \mathrm{At}(\phi) \\ \mathrm{At}(\phi \wedge \psi) &= \mathrm{At}(\phi) \cup \mathrm{At}(\psi) \end{aligned}$$

Das Schema der rekursiven Aufrufe ist dasselbe wie das der Induktionsvoraussetzungen im Induktionsprinzip, d.h. rekursive Aufrufe erfolgen immer auf die Bestandteile des aktuellen Arguments – bei der Klausel für $\mathrm{At}(\phi \wedge \psi)$ z.B. auf ϕ und ψ. Als Beispiel beweisen wir die Behauptung

> *Für jede Formel ϕ ist die Menge* $\mathrm{At}(\phi)$ *endlich*

durch Induktion über ϕ:

- $\mathrm{At}(\bot) = \varnothing$ ist endlich.
- Für $A \in \mathscr{A}$ ist $\mathrm{At}(A) = \{A\}$ endlich.
- Sei $\mathrm{At}(\phi)$ endlich; dann ist auch $\mathrm{At}(\neg\phi) = \mathrm{At}(\phi)$ endlich.
- Seien $\mathrm{At}(\phi)$ und $\mathrm{At}(\psi)$ endlich; dann ist auch $\mathrm{At}(\phi \wedge \psi) = \mathrm{At}(\phi) \cup \mathrm{At}(\psi)$ endlich, da die Vereinigung zweier endlicher Mengen wieder eine endliche Menge ist.

C Verschiedene Systeme Formalen Schließens

Man unterscheidet verschiedene Typen von formalen Deduktionssystemen, u.a. nach der Gewichtung zwischen Axiomen und Regeln:

Hilbert Viele Axiome, wenig Regeln; meist nur *modus ponens*

$$(mp)\ \frac{\phi \quad \phi \to \psi}{\psi}$$

Gentzen Regeln für *Sequenten* $\phi_1, \ldots, \phi_n \vdash \psi_1, \ldots, \psi_k$, zu lesen als „die Konjunktion der ϕ_i impliziert die Disjunktion der ψ_j“. Axiome (in der Grammatik mit nur $\bot, \wedge, \neg$) sind nur Sequenten der Form $\phi, \cdots \vdash \phi, \ldots$ oder $\bot, \cdots \vdash \ldots$.

Natürliches Schließen Variante des Sequentenkalküls, in der nur eine Formel rechts von $\vdash$ steht[3] und die linke Seite implizit gelassen wird; stattdessen hat man lokale Mengen von Annahmen, die in hierarchisch strukturierten Beweisen rekursiv aufgebaut werden.

D Literatur

U. Schöning Logik für Informatiker, Spektrum akademischer Verlag, 2000

J. Barwise, J. Etchemendy Language, Proof & Logic, CSLI, 2000
(deutsche Version verfügbar)

M. Huth, M. Ryan Logic in Computer Science, CUP, 2000

S. Reeves, M. Clarke Logic for Computer Science, Addison-Wesley, 1990

E Anwendungen von Logik in der Informatik

Logik als Problem für Informatiker Die Frage, ob eine Aussage aus anderen folgt, wird, sobald man die Begriffe von *Aussage* und *Folgerbarkeit* (*logischer Konsequenz*) formal festgelegt hat, ein algorithmisches Problem, das man maschinell zu lösen bestrebt ist. Es gibt heutzutage eine ganze Anzahl laufend weiterentwickelter *Theorembeweiser*, die in der Lage sind, logische Folgerungen entweder vollautomatisch oder *halbautomatisch*, d.h. mit gelegentlicher Benutzerinteraktion, durchzuführen. Vollautomatisch sind z.B. viele Beweiser für Prädikatenlogik erster Stufe (die wir später noch kennenlernen) wie SPASS oder Vampire sowie SMT-Solver (SMT für *Satisfiability modulo Theories* wie Microsofts Z3. Wichtige halbautomatische Beweiser sind unter anderem *Isabelle*, das unter anderem eine zentrale Rolle beim Beweis der keplerschen Vermutung gespielt hat, und *Coq*.

Ein besonders zentrales Problem der Informatik, sowohl in der Theorie als auch in der Praxis, ist das sogenannte SAT-Problem (SAT wie *Satisfiability*), das sich letztlich als das Schlussfolgerungsproblem der *Aussagenlogik* verstehen lässt, die wir im ersten Teil der Veranstaltung kennenlernen. Die überhaupt größte offene Frage der sogenannten *Komplexitätstheorie* ist, ob jedes auf einem „magischen Rechner“, der Lösungen zu Problemen richtig raten kann und sie dann nur noch überprüfen muss, effizient (d.h. innerhalb einer Zeitschranke, die nur polynomiell von der Inputgröße abhängt) lösbare Problem auch auf einem „normalen“ Rechner effizient lösbar ist, d.h. in verbreiteter Notation, ob $P = NP$. (Alle hoffen, dass das nicht der Fall ist, weil sonst große Teile der aktuellen Kryptographie nutzlos werden.) Diese Frage ist eines der sieben Millenniums-Problemen der Clay Foundation, und das einzige informatische Problem auf der Liste. Das P-vs.-NP-Problem ist, wie man zeigen kann, äquivalent zur Frage, ob das SAT-Problem effizient lösbar ist. Gleichzeitig findet SAT-Solving breite praktische Anwendung, z.B. in Systemkonfiguration, Hardwaredesign und -verification sowie Kryptanalyse.

[3] Für sehr Interessierte: Wenn man weiter nachliest, mag zunächst irritieren, dass die Beschränkung auf nur eine Formel auf der rechten Seite gerade das ist, was Gentzens intuitionistischen Sequentenkalkül LJ vom klassischen Kalkül LK unterscheidet. In der Tat machen wir unser System gewissermaßen mit Gewalt klassisch, indem wir ausdrücklich die Doppelnegationsregel hinzunehmen, während diese in LK aus deutlich natürlicher wirkenden Regeln herleitbar ist.

Logik als Programmierparadigma Programmiersprachen folgen sehr verschiedenen *Programmierparadigmen*, die auf oberster Ebene in *imperative* und *deklarative* Paradigmen unterschieden werden. Erstere sind allgemein verbreiteter – insbesondere sind C und Java imperativ –, letztere werden aber als eleganter, verständlicher und leichter wartbar angesehen. (Letztlich werden Paradigmen auch oft gemischt, etwa in Scala oder Python.) Die hauptsächlichen deklarativen Paradigmen sind zum einen die funktionale Programmierung, wie in Haskell, ML und zum Teil eben Scala oder Python, sowie die logische Programmierung wie etwa in Prolog. Logische Programme bestehen letztlich aus Formeln in einer geeignet eingeschränkten Logik, und die Ausführung des Programms besteht vereinfacht ausgedrückt in der Herleitung von Schlussfolgerungen aus diesen Formeln.

Logik als Abfrageformalismus Eine relationale Datenbank ist nichts anderes als ein Modell in Prädikatenlogik erster Stufe, und gängige Abfragesprachen wie SQL oder Datalog sind im Wesentlichen nur andere Syntax für Formeln in Prädikatenlogik erster Stufe.

Logik als Wissensrepräsentationsformalismus Ein Weg, Weltwissen maschinell zu repräsentieren und zu verarbeiten, sind logikbasieree Wissensreoräsentationssprachen wie die Web Ontology Language OWL, ein W3C-Standard. In diesen Sprachen lassen sich Dinge ausdrücken wie „Ein Elefant ist ein großes graues Tier mit einem Rüssel und vier Beinen“, in etwa wie folgt:

$$\mathsf{Elephant} \sqsubseteq \mathsf{Animal} \sqcap \mathsf{Large} \sqcap \mathsf{Gray} \sqcap (\exists\, \mathsf{hasPart}.\, \mathsf{Trunk}) \sqcap {=}4\, \mathsf{hasPart}.\, \mathsf{Leg}.$$

(Die formale Lesart dieser Formel ist für diese Diskussion nicht wichtig; informell lesen wir $\sqcap$ als „und“, $\sqsubseteq$ als Implikation, also als wenn-dann-Beziehung, $\exists$ als „es gibt (mindestens) ein“ und $= n$ as „es gibt genau n“.) In OWL und verwandten Sprachen werden z.B. Begriffssammlungen, sogenannte *Ontologien*, in der Medizin und der Unternehmensorganisation verfasst.

Logik als Entwicklungsmethode Sicherheitskritische Software, z.B. für Flugzeuge, Autos oder Kraftwerke, wird idealerweise mittels *formaler Methoden* entwickelt und verifiziert, in denen die Anforderungen an die Korrektheit der Software in geeigneten Logiken formuliert werden. Ein einfaches Beispiel ist die Spezifikation einer Sortierfunktion, die (in geeigneter logischer Notation) verlangen würde, dass die Ausgabeliste a) sortiert ist und b) dieselben Einträge enthält wie die Eingabeliste (aber bewusst nichts darüber sagt, wie das anzustellen wäre). Die Sicherheitsüberprüfung des letztendlichen Programms besteht dann darin, mittels automatischer oder halbautomatischer Werkzeug (insbesondere mittels der oben genannten Theorembeweiser) zu bestätigen, dass das Programm – oder die Hardware – diese Anforderungen erfüllt. Gerade in der Hardwareverifikation ist die Verwendung formaler Methoden seit dem äußerst kostspieligen *Pentium-Bug* Standard. Aktuell gängige logikbasierte formale Werkzeuge und Methoden sind insbesondere

- die *B Method*, die z.B. bei der Entwicklung autonomer Metro-Linien und der Ariadne-Rakete zum Einsatz kam;
- verschiedene (relativ code-nahe) Design-by-Contract-Sprachen wie Spec#, Dafny und Boogie (alle Microsoft);
- statische Analysewerkzeuge wie FramaC und Facebook Infer.

F Symbolverzeichnis

Verwendete Symbole in der Reihenfolge ihrer Definition: