Grundlagen der Logik und Logikprogrammierung

Vorlesungsskript Sommersemester 2013
FAU Erlangen, Department Informatik, Lehrstuhl 8 (Theoretische Informatik)

Aufbauend auf einer Mitschrift der Vorlesung von Lutz Schröder im Sommersemester 2012 von

Johannes Schilling (dario@zerties.org)
Dominik Paulus (dominik@d-paulus.de)
Ulrich Rabenstein (ulrich.rabenstein@studium.uni-erlangen.de)
Tobias Polzer

teilweise überarbeitet von Lutz Schröder

Disclaimer Es werden keine Garantien betreffs der Richtigkeit des Materials und der Präsentationsqualität übernommen; es handelt sich i.w. um eine durch den Veranstalter nur grob angepasste studentische Niederschrift.

Inhaltsverzeichnis

0.1 Literatur

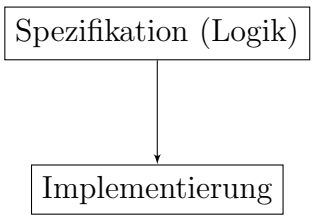
U. Schöning Logik für Informatiker, Spektrum akademischer Verlag, 2000

J. Barwise, J. Etchemendy Language, Proof & Logic, CSLI, 2000

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

0.2 Logik in der Informatik

- Logik als Problem für Informatiker
  - SAT
  - Automatisches/Halbautomatisches Theorembeweisen
- Logik als Programmierparadigma
  - Prolog
  - Mercury
- Logik als Abfrageformalismus
  - SQL
  - Datalog
- Logik als (Wissens-) Repräsentationsformalismus
  - Ontologien
  - Semantic Web
  - OWL
- Logik als Entwicklungsmethode



1 Aussagenlogik

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

$$(\text{z. B. EsRegnet} \to \text{HabeSchirm} \vee \text{WerdeNass})$$

und deren Wahrheitswerte, hier klassisch: $\{\underbrace{\bot}_{\text{falsch}}, \underbrace{\top}_{\text{wahr}}\}$

1.0.1 Grundelemente einer Logik

1. *Syntax*: „Was kann ich hinschreiben“
2. *Semantik*: „Was bedeutet das“
3. Beweismethoden, Algorithmen

1.1 Syntax

1.1.1 Backus-Naur-Form

Die Backus-Naur-Form (BNF) ist eine verbreitete Art, die Syntax von formalen Sprachen (genauer: sogenannten kontextfreien Sprachen, s. BFS) darzustellen. Eine Beschreibung in BNF besteht aus mehreren Klauseln, jede Klausel beschreibt eine Beziehung z.B. der Form „jedes a ist entweder ein B oder ein C“ in der Schreibweise

$$a \ ::= \ B \mid C.$$

Die rechte Seite besteht aus beliebig vielen Alternativen, durch die die linke ersetzt werden kann, jeweils durch einen senkrechten Strich getrennt. Die Alternativen B, C etc. können zusammengesetzt sein aus Symbolen, die entweder *terminal* sind, d.h. tatsächliche Zeichen in den zu erzeugenden Termen darstellen und nicht weiter expandier werden, oder *nicht-terminal* wie a im obigen Beispiel, d.h. linke Seiten von Klauseln der Grammatik, die dann gemäß diesen Klauseln weiter expandiert werden. Eine solche Grammatik definiert die Menge aller Terme, die durch endliche viele Anwendungen der Ersetzungsregeln der BNF erzeugt werden können; solche Terme nennen wir *Instanzen von a*. Wenn die Grammatik z.B.

$$\begin{aligned} a &::= a + a \mid b * a \mid 1 \\ b &::= 0 \mid 1 \mid \$(a) \end{aligned}$$

lautet, können wir durch die Serie von Expansionen

$$a \to b * a \to 0 * a \to 0 * (b * a) \to 0 * (\$(a) * a) \to 0 * (\$(1) * 1)$$

die Instanz $0*(\$(1)*1)$ von a erzeugen. Man beachte hier, dass wir Klammern in der Grammatik implizit lassen, aber wie im eben durchgeführten Beispiel Terme bei Bedarf klammern.

1.1.2 Syntax der Aussagenlogik

Wir definieren die Menge der aussagenlogischen Formeln ϕ, ψ durch die Grammatik

$$\varphi, \psi ::= \bot \mid A \mid \varphi \wedge \psi \mid \neg\, \varphi$$

Dabei ist $A \in \mathscr{A}$ ein Atom, d.h. eine nicht weiter unterteilbare Aussage. Wir vereinbaren:

- $\neg$ bindet am stärksten
- $\top = \neg\bot$
- $(\varphi \vee \psi) = \neg(\neg\varphi \wedge \neg\psi)$
- $\varphi \to \psi = \neg\varphi \vee \psi$
- $\varphi \leftrightarrow \psi = (\varphi \to \psi) \wedge (\psi \to \varphi)$

1.2 Semantik der Aussagenlogik

1.2.1 Informelle Semantik

Die Aussprache und intuitive Bedeutung der logischen Operatoren ist

$\varphi \wedge \psi$	„ϕ und ψ“
$\neg\varphi$	„nicht ϕ“
$\varphi \vee \psi$	„ϕ oder ψ“
$\varphi \to \psi$	„wenn ϕ, dann ψ“
$\varphi \leftrightarrow \psi$	„genau dann φ, wenn ψ“
$\top$	„Wahr“
$\bot$	„Falsch“

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 – z.B. 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.

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
$\bot$	Falschheit, konstant falsche Aussage, Absurdität

1.2.2 Wahrheitsbelegungen und Erfülltheit

Definition 1 (Wahrheitsbelegung, Erfülltheit)**.** Eine *Wahrheitsbelegung* ist eine Abbildung $\kappa : \mathscr{A} \to \underbrace{2}_{\{\top,\bot\}}$, legt also Wahrheitswerte für alle Atome fest.

Wir definieren die Erfüllheitsrelation $\kappa \vDash \varphi$ (lies: κ erfüllt φ) rekursiv durch

(i) $\kappa \nvDash \bot$

(ii) $\kappa \vDash A \Leftrightarrow \kappa(A) = \top$

(iii) $\kappa \vDash \varphi \wedge \psi \Leftrightarrow \kappa \vDash \varphi$ und $\kappa \vDash \psi$

(iv) $\kappa \vDash \neg\varphi \Leftrightarrow \kappa \nvDash \varphi$

In Worten: κ erfüllt ein Atom A, wenn es A den Wert „wahr“ zuweist; κ erfüllt die Konjunktion zweier Formeln, wenn κ beide Formeln erfüllt; κ erfüllt die Negation einer Formel ϕ, wenn κ die Formel ϕ nicht erfüllt.

Erfülltheit verhält sich für die abgeleiteten Operatoren wie folgt:

$$
\begin{aligned}
&\kappa \vDash \top \text{ stets}\\
&\kappa \vDash \bot \text{ nie}\\
\kappa \vDash \varphi \vee \psi &\iff (\kappa \vDash \varphi \text{ oder } \kappa \vDash \psi)\\
\kappa \vDash \varphi \to \psi &\iff (\text{Falls } \kappa \vDash \varphi, \text{ so auch } \kappa \vDash \psi)\\
\kappa \vDash \varphi \leftrightarrow \psi &\iff (\kappa \vDash \varphi \text{ genau dann wenn } \kappa \vDash \psi)
\end{aligned}
$$

Beispiel 2 ($(A \vee \neg B) \to B$)**.** κ ordne den Atomen A und B die Werte $\kappa(A) = \top$ und $\kappa(B) = \bot$ zu; Kurzschreibweise: $\kappa = [A \mapsto \top, B \mapsto \bot]$. Dann gilt

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

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 $\kappa \nvDash (A \vee \neg B) \to B$.

Eine andere Wahrheitsbelegung κ_2, die den Atomen andere Wahrheitswerte zuordnet, kann die Formel aber erfüllen, z. B. mit $\kappa_2 = [A \mapsto \top, B \mapsto \top]$ gilt $\kappa_2 \vDash (A \vee \neg B) \to B$.

1.3 Logische Konsequenz

Eine logische Konsequenz oder logische Folgerung ist die korrekte Ableitung einer neuen Formel aus einer Menge als gültig vorausgesetzter Formeln.

Für die formale Definition definieren wir zunächst die Erfülltheit einer Menge $\Phi \subseteq F$ von Formeln: Eine Wahrheitsbelegung κ erfüllt die Menge Φ genau dann, wenn κ alle Formeln φ erfüllt, die in Φ enthalten sind, d.h.

$$\kappa \vDash \Phi \ :\iff \ \forall \varphi \in \Phi.\, \kappa \vDash \varphi$$

Definition 3 (logische Konsequenz)**.** Sei $\Phi \subseteq F$ eine Menge von Formeln. Eine Formel $\psi \in F$ ist eine *logische Konsequenz* von Φ, wenn für alle Wahrheitsbelegungen $\kappa : \mathscr{A} \to 2$ gilt, dass, falls $\kappa \vDash \Phi$, so auch $\kappa \vDash \psi$. Man schreibt für „ψ ist logische Konsequenz von Φ“ auch $\Phi \vDash \psi$. In Symbolen:

$$\Phi \vDash \psi \ :\iff \ \forall \kappa.\, (\kappa \vDash \Phi \implies \kappa \vDash \psi).$$

Daraus lassen sich nun einige neue Begriffe und Definitionen ableiten.

Definition 4 (Gültigkeit einer Formel ψ)**.** Eine Formel ψ ist *gültig*, wenn sie aus der leeren Menge von Annahmen folgt:

$$\models \psi : \iff \emptyset \vDash \psi \iff \forall \kappa : \kappa \vDash \psi,$$

d.h. also wenn alle Wahrheitsbelegungen ψ erfüllen. Wir nennen ψ in diesem Fall auch *tautologisch*/eine *Tautologie*.

Definition 5 (Erfüllbarkeit einer Menge von Formeln Φ)**.** Eine Formelmenge Φ ist *unerfüllbar*, wenn sich aus ihr ein Widerspruch herleiten lässt, d.h. wenn Falsum eine logische Konsequenz von Φ ist: $\Phi \vDash \bot$. Dies ist gleichbedeutend damit, dass keine Wahrheitsbelegung Φ erfüllt: $\forall \kappa : \kappa \nvDash \Phi$.

Andernfalls, d.h. wenn $\exists \kappa : \kappa \vDash \Phi$, heißt Φ *erfüllbar*.

Definition 6 (Logische Äquivalenz zweier Formeln φ und ψ)**.** Zwei Formeln φ, ψ sind logisch äquivalent ($\varphi \equiv \psi$), wenn $\varphi \leftrightarrow \psi$ gültig ist:

$$\varphi \equiv \psi : \iff \models \phi \leftrightarrow \psi.$$

Lemma 7. *Eine Formel ψ ist genau dann logische Konsequenz einer Formelmenge Φ, wenn die Vereinigung von Φ und der Negation von ψ unerfüllbar ist:*

$$\Phi \vDash \psi \iff (\Phi \cup \{\neg\psi\}) \vDash \bot.$$

Beweis. Wir zeigen zwei Implikationen:

„$\Longleftarrow$“: Sei $\kappa \models \Phi$; wir müssen $\kappa \models \psi$ zeigen. Beweis durch Widerspruch: wenn $\kappa \not\models \psi$, dann per Definition $\kappa \models \neg\psi$, also $\kappa \models \Phi \cup \{\neg\psi\}$. Letzteres ist aber nach Annahme unerfüllbar, Widerspruch.

„$\Longrightarrow$“: Diese Richtung zeigen wir durch *Kontraposition*: allgemein ist $A \implies B$ äquivalent zu $\neg B \implies \neg A$. Wir nehmen also die Negation der rechten Seite an und zeigen dann die Negation der linken Seite. Sei also $\Phi \cup \{\neg\psi\} \not\models \bot$. Dann existiert κ mit $\kappa \models \Phi \cup \{\neg\psi\}$. Insbesondere gilt dann $\kappa \models \Phi$, aber $\kappa \not\models \psi$, also $\Phi \not\models \psi$.

□

Beispiel 8 (Erfüllbarkeit und logische Konsequenz)**.**

Erfüllbar: $A \to \neg A$ (Für die Belegung $\kappa(A) = \bot$)

Unerfüllbar: $A \wedge \neg A$

Gültig: $A \vee \neg A$, $(A \wedge B) \to A$

Logische Konsequenz: $\{A \to B, A\} \vDash B$ (diesen Schluss nennt man „modus ponens“)

1.4 Wahrheitstafeln

Eine Wahrheitstafel ist eine tabellarische Auflistung der Wahrheitswerte einer Formel in Abhängigkeit von den Wahrheitswerten der (endlich vielen!) in ihr vorkommenden Atome. Mit Hilfe von Wahrheitstafeln kann z.B. die logische Äquivalenz zweier Formeln entschieden werden (zwei Formeln sind im wesentlichen dann logisch äquivalent, wenn sie dieselbe Wahrheitstafel haben) oder eine logische Folgerung überprüft werden (eine Formel ϕ ist im wesentlichen dann eine logische Folgerung aus ψ, wenn in jeder Zeile der Wahrheitstafel für ψ, in der für ψ „wahr“ steht, auch für ϕ „wahr“ steht). 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.

Beispiel 9. Wahrheitstafel von $A \to B = \neg A \vee B$:

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

$\neg A \vee A$ ist gültig:

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

$A \to \neg A$ ist erfüllbar:

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

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

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

$B \vee \neg B \equiv A \vee \neg A$: bei beiden Formeln steht nur $\top$ in der Wahrheitstafel.

Im letzten Beispiel sieht man, dass zwei Formeln trotz unterschiedlicher verwendeter Atome äquivalent sein können.

Definition 10 (Atome einer Formel)**.** Die Menge $\mathrm{At}(\varphi)$ der in φ vorkommenden Atome ist rekursiv definiert durch

- $\mathrm{At}(A) = \{A\}$
- $\mathrm{At}(\neg\varphi) = \mathrm{At}(\varphi)$

- $\mathrm{At}(\varphi \wedge \psi) = \mathrm{At}(\varphi) \cup \mathrm{At}(\psi)$

Beispiel 11. 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 12. *Die Erfülltheit $\kappa \vDash \varphi$ hängt nur von den Belegungen der Atome von φ, also von den Werten $\kappa(A)$ für $A \in \mathrm{At}(\varphi)$ ab; d.h. wenn κ' sich bezüglich aller Atome $\mathrm{At}(\varphi)$ gleich verhält wie κ, so erfüllt $\kappa' \varphi$ genau dann wenn κ dies tut. Formal: Wenn $\kappa, \kappa' : \mathscr{A} \to 2$ mit $\kappa(A) = \kappa'(A)$ für alle $A \in \mathrm{At}(\varphi)$, dann gilt*

$$\kappa \vDash \varphi \iff \kappa' \vDash \varphi.$$

Der Beweis des Lemmas illustriert gleichzeitig ein in der Logik wichtiges Beweisprinzip, die *strukturelle Induktion*; wir verschieben ihn daher auf Abschnitt 2. Die Semantik von φ ist also bestimmt durch endliche Tabellierung von $\kappa \vDash \varphi$ für alle $\kappa : A_0 \to 2$ mit $A_0 \subseteq \mathscr{A}$ endlich, $\mathrm{At}(\varphi) \subseteq A_0$.

Lemma 13. *$\varphi \equiv \psi$ genau dann wenn φ, ψ identische Wahrheitstafeln über $\mathrm{At}(\varphi) \cup \mathrm{At}(\psi)$ haben.*

1.5 Logische Äquivalenzen

Im folgenden geben wir eine Übersicht wichtiger logischer Äquivalenzen.

- $\neg\neg\varphi \equiv \varphi$ (Doppelnegationselimination)
- $\left.\begin{array}{l}\neg(\varphi \wedge \psi) \equiv (\neg\varphi \vee \neg\psi)\\ \neg(\varphi \vee \psi) \equiv (\neg\varphi \wedge \neg\psi)\end{array}\right\}$ (De Morgansche Gesetze)
- $\left.\begin{array}{l}\varphi \wedge (\psi \vee \chi) \equiv (\varphi \wedge \psi) \vee (\varphi \wedge \chi)\\ \varphi \vee (\psi \wedge \chi) \equiv (\varphi \vee \psi) \wedge (\varphi \vee \chi)\end{array}\right\}$ (Distributivgesetze)
- $\left.\begin{array}{l}(\varphi \wedge \psi) \wedge \chi \equiv \varphi \wedge (\psi \wedge \chi)\\ \varphi \vee (\psi \vee \chi) \equiv (\varphi \vee \psi) \vee \chi\end{array}\right\}$ (Assoziativgesetze)
- $\left.\begin{array}{l}\chi \wedge \top \equiv \chi\\ \chi \vee \bot \equiv \chi\end{array}\right\}$ (Neutrale Elemente)

2 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.

2.1 Induktion über natürliche Zahlen

Das vermutlich bekannteste Induktionsprinzip ist die Induktion über natürliche Zahlen. In der einfachsten Form lautet das Prinzip wie folgt. Wenn eine Aussage $P(n)$ über natürliche Zahlen n die Eigenschaften

- für die 0 gilt ($P(0)$) (*Induktionsanfang*) und
- für jede natürliche Zahl n folgt $P(n+1)$ aus $P(n)$ ($\forall n \in \mathbb{N}.\, (P(n) \implies P(n+1))$ (*Induktionsschritt*; man bezeichnet ferner $P(n)$ als die *Induktionsvoraussetzung*)

erfüllt, dann gilt P für jede natürliche Zahl ($\forall n \in \mathbb{N}.\, P(n)$).

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-Over-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 14 (Course-over-Values Induction)**.** *Sei $P(n)$ eine Aussage über natürliche Zahlen.*[1] *Wenn für jedes n aus*

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

$P(n)$ folgt ($\forall n.\, (\forall k < n.\, P(k)) \implies P(n)$), so gilt P für jede natürliche Zahl ($\forall n.\, P(n)$).

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 $xQ(n)$ mit $\forall n.\, Q(n) \implies P(n)$) per Induktion beweist. Man hat dann zwar im Induktionsschritt mehr zu zeigen, hat aber dazu eine stärkere Induktionsannahme zur Verfügung. Auch im vorliegenden Fall kann man sich überzeugen, dass die Induktionsannahme $P(n)$ unter den Annahmen des Satzes nicht ausreicht, um $P(n+1)$ zu folgern (dazu braucht man $P(k)$ für alle $k < n+1$, die Induktionsannahme liefert dies aber nur für den Fall $k = n$). Stattdessen beweisen wir unter den Annahmen des Satzes die stärkere Aussage

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

durch Induktion über n:

- Induktionsanfang: nach 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 $\forall k \leq 0.\, P(0)$.
- Induktionsschritt: Es gelte $\forall k \leq n.\, P(n)$; zu zeigen ist $\forall k \leq n+1.\, P(n)$. Für die meisten k folgt dies sofort aus der Induktionsannahme; 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 Induktionsannahme $\forall k \leq n.\, P(n)$.

□

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 $n = 1$, dann ist n per eben vereinbarter Konvention ein endliches Produkt von Primzahlen.

[1] 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.

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.

2.2 Strukturelle Induktion

Man kann nun Induktion nicht nur über den natürlichen Zahlen verwenden, sondern auch über im wesentlichen allen endlichen azyklischen Datenstrukturen – insbesondere z.B. über mittels einer Grammatik definierte Objekte, wie etwa aussagenlogische 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.

Für die formale Diskussion nehmen wir der Einfachheit halber an, wir hätten es mit einer Grammatik mit nur einem nichtterminalen Symbol zu tun (wie z.B. der Grammatik für aussagenlogische Formeln), also einer Grammatik der Form

$$a ::= B_1 \mid \dots \mid B_n$$

wobei die B_i aus terminalen Symbolen und a zusammengesetzt sind. Wir erhalten dann ein Induktionsprinzip zum Beweis einer Eigenschaft P für alle Instanzen von a mit n Induktionsschritten, einer für jedes B_i. Der durchzuführende Induktionsschritt für B_i verlangt, dass man unter der Annahme, dass alle in B_i vorkommenden Instanzen von a bereits P erfüllen (Induktionsvoraussetzung), zeigt, dass auch B_i (die neu erzeugte Instanz von a) P erfüllt. Wenn a 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.

Ein erstes Beispiel dieses Prinzip 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 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.

Als zweites Beispiel entnehmen wir der Grammatik für aussagenlogische Formeln,

$$\varphi, \psi ::= \bot \mid A \mid \varphi \wedge \psi \mid \neg\, \varphi$$

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 $P(\phi \wedge \psi)$.

Als Beispiel reichen wir jetzt den Beweis von Lemma 12 nach (Erfülltheit von Formeln hängt nur von der Wahrheitsbelegung in ihr vorkommender Atome ab), allerdings der besseren Lesbarkeit wegen mit einer etwas vereinfachten Aussage. Wir benötigen (hier und immer wieder) eine Schreibweise zur Abänderung von Abbildungen: Wenn $f : X \to Y$ eine Abbildung ist und $x_0 \in X$, $y_0 \in Y$, dann bezeichnet $f[x_0 \mapsto y_0]$ die Abbildung mit

$$f[x_0 \mapsto y_0](x) = \begin{cases} y_0 & \text{falls } x = x_0 \\ f(x) & \text{sonst.} \end{cases}$$

Wir wenden diese Schreibweise z.B. auf Wahrheitsbelegungen, also Abbildungen $\mathscr{A} \to \{\bot, \top\}$, an.

Lemma 15. *Wenn ϕ eine aussagenlogische Formel ist, κ eine Wahrheitsbelegung, $B \notin \mathrm{At}(\phi)$, und $b \in \{\bot, \top\}$, dann gilt*

$$\kappa \models \phi \quad \textit{genau dann, wenn} \quad \kappa[B \mapsto b] \models \phi.$$

(In Worten: Auf in ϕ nicht vorkommenden Atomen können wir die Wahrheitsbelegung abändern, ohne die Erfülltheit von ϕ zu beeinflussen).

Beweis. Strukturelle Induktion über ϕ.

$\bot$*:* Es gilt $\kappa \models \bot$ genau dann, wenn $\kappa[A \mapsto b] \models \bot$ (es gilt nämlich beides nicht).

A *(mit $A \in \mathscr{A}$):* Es gilt per Definition $\kappa \models A$ genau dann, wenn $\kappa(A) = \top$. Da nach Voraussetzung $B \notin FV(A)) = \{A\}$, gilt $B \neq A$, also $\kappa[B \mapsto b](A) = \kappa(A)$. Damit gilt $\kappa(A) = \top$ genau dann, wenn $\kappa[B \mapsto b](A) = \top$, was per Definition wiederum äquivalent ist zu $\kappa[B \mapsto b] \models A$.

$\neg\phi$*:* In diesem Fall besagt die Induktionsvoraussetzung, dass die Behauptung des Lemmas für ϕ gilt. (Weiter unten sagen wir das nicht mehr so explizit.) Da nach Voraussetzung $A \notin FV(\neg\phi) = FV(\phi)$, folgt hieraus, dass $\kappa \models \phi$ genau dann, wenn $\kappa[B \mapsto b] \models \phi$. Negieren beider Seiten einer Äquivalenz gibt wieder eine Äquivalenz, so dass also $\kappa \not\models \phi$ genau dann, wenn $\kappa[B \mapsto b] \not\models \phi$. Per Definition der Semantik von $\neg$ erhalten wir $\kappa \models \neg\phi$ genau dann, wenn $\kappa[B \mapsto b] \models \neg\phi$.

$\phi \wedge \psi$*:* In diesem Fall besagt die Induktionsvoraussetzung, dass die Behauptung des Lemmas für ϕ und ψ gilt. Nach Voraussetzung gilt $A \notin FV(\phi \wedge \psi) = FV(\phi) \cup FV(\psi)$, also $A \notin FV(\phi)$ und $A \notin FV(\psi)$. Damit gilt also nach Induktionsvoraussetzung $\kappa \models \phi$ genau dann, wenn $\kappa[B \mapsto b] \models \phi$, und $\kappa \models \psi$ genau dann, wenn $\kappa[B \mapsto b] \models \psi$. Per Definition der Semantik von $\wedge$ haben wir damit

$$\begin{aligned} \kappa \models \phi \wedge \psi &\iff \kappa \models \phi \text{ und } \kappa \models \psi \\ &\iff \kappa[B \mapsto b] \models \phi \text{ und } \kappa[B \mapsto b] \models \psi \\ &\iff \kappa[B \mapsto b] \models \phi \wedge \psi. \end{aligned}$$

□

Bemerkung 16. Es fehlt natürlich noch eine Rechtfertigung des Prinzips der strukturellen Induktion. Intuitiv kann man sich vorstellen, dass es sich einfach um Course-over-Values Induction über die textuelle Größe von Instanzen handelt; z.B. wird im obigen Induktionschritt für $\phi \wedge \psi$ die Behauptung für $\phi \wedge \psi$ zurückgeführt auf die Behauptung für ϕ und für ψ, also Formeln, die echt kleiner sind als $\phi \wedge \psi$. Um hieraus ein formal korrektes Argument zu machen, muss man strenggenommen noch zeigen, dass alle Instanzen einer Grammatik endliche Größe haben; das zeigt man z.B. durch Induktion über die Länge der Ableitung. Wenn man diesen Beweis wirklich durchführt, wird er allerdings technisch nicht ganz unkompliziert: man muss z.B. aus einer Ableitung von $\phi \wedge \psi$

$$\phi \to \phi \wedge \psi \to \ldots,$$

Ableitungen von ϕ und ψ herausgreifen (die dann echt kürzer sind als die Ableitung von $\phi \wedge \psi$), was offenbar geht, aber notationsmäßig einigen Aufwand erzeugt.

Ein technisch einfacheres, aber abstrakteres Argument ist das folgende: die Menge der Instanzen einer Grammatik kann man auch definieren als die kleinste Menge, die unter den Klauseln der Grammatik abgeschlossen ist (also z.B. mit ϕ und ψ stets auch $\phi \wedge \psi$ enthält etc.). Diese Definition *ist* das Prinzip der strukturellen Induktion: diese besteht gerade darin, zu zeigen, dass auch die Zieleigenschaft P unter den Klauseln der Grammatik abgeschlossen ist (z.B. also, dass, wenn ϕ und ψ P erfüllen, dann auch $\phi \wedge \psi$); damit muss sie also die Menge aller Instanzen enthalten, die ja die *kleinste* Menge mit dieser Eigenschaft ist. Zu zeigen ist dann strenggenommen noch, dass die auf die neue Weise definiert Menge der Instanzen einer Grammatik mit der alten Definition (über Ableitbarkeit in endlich vielen Schritten) übereinstimmt. Man zeigt dazu zum einen durch Induktion über die Länge einer endlichen Ableitung, dass jeder in endlich vielen Schritten ableitbare Term eine Instanz gemäß der neuen Definition ist[2], und zum anderen durch strukturelle Induktion, dass jede Instanz gemäß der neuen Definition in endlich vielen Schritten ableitbar ist.

3 Normalformen

3.0.1 Negationsnormalform (NNF)

Definition 17 (Negationsnormalform NNF von φ)**.** Eine aus Atomen sowie $\neg, \wedge, \vee$ gebildete Formel φ ist genau dann in *NNF*, wenn die Negation $\neg$ in φ nur direkt vor Atomen vorkommt. π ist *NNF von* φ, wenn π in NNF und $\pi \equiv \varphi$.

Durch wiederholte Anwendung der Gesetze

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

lässt sich jeder Term in NNF bringen; Details s. Übung. Atome sind in NNF.

Beispiel 18 (Terme in NNF)**.** $\neg A \vee (B \vee \neg C)$ ist in NNF;
$\neg(A \vee \neg B)$ nicht ($\neg$ vor der Klammer, Klammer kein Atom)

[2]Hinter dieser Phrase verbirgt sich aber offenbar wieder ein Herausgreifen von Teilableitungen

Alternativ ist NNF(φ) *rekursiv* definiert durch

$$NNF(\varphi) = \begin{cases} NNF(A) = A \\ NNF(\varphi \wedge \pi) = NNF(\varphi) \wedge NNF(\pi) \\ NNF(\varphi \vee \pi) = NNF(\varphi) \vee NNF(\pi) \\ NNF(\neg A) = \neg A \\ NNF(\neg\neg\varphi) = NNF(\varphi) \\ NNF(\neg(\varphi \wedge \pi)) = NNF(\neg\varphi) \vee NNF(\neg\pi) \\ NNF(\neg(\varphi \vee \pi)) = NNF(\neg\varphi) \wedge NNF(\neg\pi) \end{cases}$$

Es lässt sich dann induktiv zeigen, dass NNF(φ) $\equiv \varphi$.

Beispiel 19 ($\varphi :- \neg(\neg(A \vee \neg B) \wedge C)$)**.** NNF($\neg(\neg(A \vee \neg B) \wedge C)$) = NNF($\neg\neg(A \vee \neg B)$)$\vee$ NNF($\neg C$) =
NNF($A \vee \neg B$) $\vee \neg C = A \vee \neg B \vee \neg C$

3.0.2 Konjunktive Normalformen (CNF)

Eine Formel in *Konjunktiver Normalform* ist eine Konjunktion von Disjunktionen von Literalen. Sie hat also die allgemeine Form

$$\bigwedge_{D \in C} \left(\bigvee_{L \in D} L \right)$$

Die Menge aller konjunktiven Normalformen ist formal definiert durch die folgende Grammatik:

Literale $L ::= A \mid \neg A \qquad A \in \mathscr{A}$

Klauseln $C ::= \bot \mid L \mid L \vee C$

CNFs $\varphi ::= \top \mid \varphi \mid C \wedge \varphi$.

Meistens, insbesondere zum Zwecke der Repräsentation im Rechner, verwenden wir eine alternative Darstellung von CNFs als Mengen:

Klauseln sind endliche Mengen von Literalen, d. h. die Disjunktion $L_1 \vee \cdots \vee L_n$ wird repräsentiert als die Menge $\{L_1, \ldots, L_n\}$ (was von der Reihenfolge und eventuellen Wiederholungen abstrahiert!). Die leere Klausel repräsentiert $\bot$ und wird als $\square$ notiert.

CNFs sind endliche Mengen von Klauseln, wobei die leere Menge $\top$ repräsentiert: $\top \triangleq \emptyset$; $D_1 \wedge \cdots \wedge D_n \triangleq \{D_1, \ldots, D_n\}$

Definition 20 (CNF C einer Formel φ)**.** Sei φ eine Formel. Eine CNF φ' heißt *CNF von* φ, wenn $\varphi \equiv \varphi'$.

Lemma 21. *Sei φ eine Formel. Dann hat φ eine CNF* CNF(φ).

Beweis. Wir können nach obigem annehmen, dass φ bereits in NNF ist. Man kann dann φ durch wiederholtes Anwenden des *Distributivgesetzes*

$$(\psi \wedge \chi) \vee \rho \equiv (\psi \vee \rho) \wedge (\chi \vee \rho) \tag{1}$$

(und seiner symmetrischen Form $\rho \vee (\psi \wedge \chi) \equiv (\rho \vee \psi) \wedge (\rho \vee \chi)$) in CNF bringen: Wir zeigen, dass wiederholte Anwendung der Gleichung (von links nach rechts) in beliebiger Abfolge (d.h. auf beliebige Teilformeln der gegebenen Formel) stets mit einer Formel terminiert, auf die die Gleichung nicht mehr anwendbar ist. Da die Umformung NNF wieder in NNF transformiert, ist eine solche Formel eine NNF, auf die das Distributivgesetz nicht mehr anwendbar ist, und damit notwendigerweise eine CNF.

Um zu zeigen, dass die Anwendung des Distributivgesetzes terminiert, definieren wir rekursiv eine Bewertung r von Formeln in NNF durch natürliche Zahlen:

$$\begin{aligned} r(A) = r(\neg A) &= 1 \\ r(\psi \wedge \chi) &= r(\psi) + r(\chi) \\ r(\psi \vee \chi) &= r(\psi)^2 r(\chi)^2 \end{aligned}$$

(so dass insbesondere $r(\psi)$ ganzzahlig und ≥ 1 ist für alle ψ). Wir kürzen ab $n = r(\psi)$, $m = r(\chi)$ und $k = r(\rho)$. Dann haben wir als Bewertung der linken Seite von (1)

$$(n+m)^2 k^2 = n^2 k^2 + 2nmk^2 + m^2 k^2$$

und als Bewertung der rechten Seite

$$n^2 k^2 + m^2 k^2,$$

also um $2nmk^2$ (mindestens 2) weniger. Da alle in der Definition von r verwendeten Operationen streng monoton sind, nimmt die Bewertung auch dann echt ab, wenn das Distributivgesetz auf eine Teilformel angewendet wird. Da die Bewertung ganzzahlig und ≥ 1 ist, kann sie aber nur endlich oft abnehmen, d.h. wir können das Distributivgesetz stets nur endlich oft anwenden. □

Alternativ definieren wir $\mathrm{CNF}(\varphi)$ rekursiv:

$$\begin{aligned} \mathrm{CNF}(\varphi \wedge \psi) &= \mathrm{CNF}(\varphi) \wedge \mathrm{CNF}(\psi) \\ \mathrm{CNF}(L) &= L \\ \mathrm{CNF}(\varphi \vee \psi) &= \bigwedge_{j=1}^{k} \bigwedge_{i=1}^{n} (D_i \vee E_j) \text{ mit } \mathrm{CNF}(\varphi) = \bigwedge_{i=1}^{n} D_i \text{ und } \mathrm{CNF}(\psi) = \bigwedge_{i=1}^{k} E_i. \end{aligned}$$

Man zeigt durch Indukion über φ, dass $\varphi \equiv \mathrm{CNF}(\varphi)$ und dass $\mathrm{CNF}(\varphi)$ in der Tat eine CNF ist. Der einzig interessante Teil der Induktion ist der Induktionsschritt für $\varphi \vee \psi$, der auf wiederholtem Anwenden des Distributivgesetzes („Ausmultiplizieren“) beruht:

$$\begin{aligned} \varphi \vee \psi &\equiv (\bigwedge_{i=1}^{n} D_i) \vee (\bigwedge_{j=1}^{k} E_j) \\ &\overset{Distr.}{\equiv} \bigwedge_{j=1}^{k} ((\bigwedge_{i=1}^{n} D_i) \vee E_j) \\ &\overset{Dist.}{\equiv} \bigwedge_{j=1}^{k} \bigwedge_{i=1}^{n} (D_i \vee E_j) = \mathrm{CNF}(\varphi \vee \psi). \end{aligned}$$

Problem an CNF($\varphi \vee \psi$): $n \cdot k$ Klauseln, z.B. CNF($(A_1 \wedge B_1) \vee \cdots \vee (A_n \wedge B_n)$) hat 2^n Klauseln. Durch Einführung zusätzlicher Atome lässt sich der Blowup[3] polynomiell halten, dann aber $\overline{\text{CNF}}(\varphi)$ nur noch *erfüllbarkeitsäquivalent* zu φ ($\overline{\text{CNF}}(\varphi)$ erfüllbar $\Leftrightarrow$ φ erfüllbar).

Beispiel 22.

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

3.1 Resolution

Das Resolutionsverfahren ist ein Algorithmus zur Entscheidung der Erfüllbarkeit einer CNF (Klauselmenge C). Er basiert auf der Resolutionsregel.

Definition 23 (Resolutionsregel)**.**

$$\text{(Res)}\ \frac{D_1 \cup \{A\} \qquad D_2 \cup \{\neg A\}}{D_1 \cup D_2} \tag{2}$$

Diese Regel wird im Resolutionsverfahren auf die Menge der Klauseln in einer CNF angewendet.

Beispiel 24.

$$\{D, B, \neg C\}, \{D, C\}, \{\neg D, B\}, \{\neg C, B, \neg A\}, \{C, B, \neg A\}, \{\neg B, \neg A\}, \{\neg B, A\}$$

Regeln dieser Form werden so interpretiert, dass, wenn in der Menge alle Elemente über dem Strich enthalten sind, das Element unter dem Strich zur Menge hinzugefügt werden darf. $D_1 \cup D_2$ hat hier den Namen *Resolvente*.

3.1.1 Formale Schlussregeln

Regeln bestehen i.a. aus Voraussetzungen (*Prämissen*) und einem Schluss daraus (*Konklusion*). Die Prämissen stehen dabei über einer waagrechten Linie, die einem Bruchstrich nicht unähnlich sieht, die Konklusion darunter. Die Mechanik der Anwendung so einer Regel besteht darin, dass, aobald die Prämissen hergeleitet sind, auch die Konkusion hergeleitet werden darf. Typischerweise werden die Regeln benannt, der Name wird in Klammern auf Höhe des „Bruchstrichs“ vor die Regel geschrieben.

$$\text{(Und)}\ \frac{A \qquad B}{A \wedge B}$$

Im Fall der Resolutionsregel bedeutet „Herleiten“ Aufnahme einer neuen Klausel in eine CNF (Klauselmenge). Das folgende Lemma besagt, dass Anwendung der Resolutionsregel in diesem Sinne die Erfüllbarkeit einer CNF nicht beeinflusst.

Lemma 25. *Sei $D_1 \cup \{A\}, D_2 \cup \{\neg A\} \in \varphi$. Dann gilt: φ ist genau dann erfüllbar, wenn $\varphi \cup \{D_1 \cup D_2\}$ erfüllbar ist.*

[3] Als *Blowup* bezeichnet man das Verhältnis zwischen Größe der Eingabe und Größe der Ausgabe eines Algorithmus. Oft verwendet bei Algorithmen, die ein Problem in ein anderes Transformieren.

Beweis. „⇐“ trivial, da linke Seite Teilmenge der rechten Seite
„⇒“ Sei $\kappa \vDash \varphi$, dann $\kappa \vDash D_1 \cup \{A\}, \kappa \vDash D_2 \cup \{\neg A\}$
Fall 1: $\kappa(A) = \top \Rightarrow \kappa \vDash D_2 \Rightarrow \kappa \vDash D_1 \cup D_2$
Fall 2: $\kappa(A) = \bot \Rightarrow \kappa \vDash D_1 \Rightarrow \kappa \vDash D_1 \cup D_2$ □ □

Basierend auf obigem Lemma und obiger Regel können wir den folgenden Algorithmus definieren, der eine Aussage über die Erfüllbarkeit einer Formel, die in konjunktiver Normalform vorliegt, trifft:

Definition 26 (Resolutionsverfahren)**.** Eingabe: CNF ϕ. Ausgabe: „ja“, wenn ϕ erfüllbar, „nein“ sonst.

Verwende ϕ als globale Variable:

1. If $\square \in \varphi$ return „nein“.
2. Suche $D_1 \cup \{A\}, D_2 \cup \{\neg A\} \in \varphi$, $D_1 \cup D_2 \notin \varphi$. Falls keine solchen D_1, D_2 existieren, return „ja“.
3. $\varphi := \varphi \cup \{D_1 \cup D_2\}$, gehe zu Schritt 1.

Beispiel 27.

$$\{D, B, \neg C\}, \{D, C\}, \{\neg D, B\}, \{\neg C, B, \neg A\}, \{C, B, \neg A\}, \{\neg B, \neg A\}, \{\neg B, A\}$$
$$\{A, \neg B\}, \{A, B\}$$

Wir beweisen die totale Korrektheit des Algorithmus:

Terminierung. Der Algorithmus arbeitet ausschließlich auf der Menge $\mathrm{At}(\phi)$ der in φ vorkommenden Atome, d.h. die Anwendung der Regel führt nie neue Atome ein. In Mengendarstellung gibt es nur endlich viele unterschiedliche Klauseln über $\mathrm{At}(\varphi)$, nämlich $4^{|\,\mathrm{At}(\varphi)|}$ (jedes Atom kann sowohl als positives als auch als negatives Literal jeweils vorkommen oder nicht vorkommen). Insbesondere kann also nur endlich oft eine Klausel hinzugefügt werden, die nicht bereits in der aktuellen CNF enthalten ist. □

Korrektheit (Antwort „nein“ ist richtig). Klar per Lemma 25. □

Wir sagen, dass ϕ *resolutionsabgeschlossen (ra.)* ist, wenn mit $D_1 \cup \{A\}, D_2 \cup \{\neg A\} \in \phi$ stets $D_1 \cup D_2 \in \phi$ gilt. Der Algorithmus bricht also genau dann bei Schritt 2 ab, wenn ϕ ra. ist.

Vollständigkeit (Antwort „ja“ ist richtig). Sei φ ra., $\square \notin \varphi$. ZZ: φ erfüllbar.

Induktion über $|\,\mathrm{At}(\varphi)|$:

Induktionsanfang: Ohne Atome kann ϕ als Klauselmenge nur leer (also wahr) sein, da die einzige Klausel ohne Atome, nämlich $\square$, n.V. nicht in ϕ enthalten ist.

Induktionsschritt: Wähle $A \in \mathrm{At}(\varphi)$. Sei

$$\varphi/A = \{\, D \setminus \{\neg A\} \mid A \notin D \in \varphi \,\}$$
$$\varphi/\neg A = \{\, D \setminus \{A\} \mid \neg A \notin D \in \varphi \,\}.$$

φ/A ist der noch als erfüllbar zu zeigende Rest, wenn wir A annehmen, entsprechend für $\varphi/\neg A$ und $\neg A$. Es gilt $\mathrm{At}(\varphi/A) \not\ni A \notin \mathrm{At}(\varphi/\neg A)$).

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

Ohne Einschränkung[4] sei $\square \notin \varphi/A$.

Nun ist auch φ/A ra.: Sei $E_1 \cup \{B\}, E_2 \cup \neg\{B\} \in \phi/A$ (insbesondere $B \neq A$). Dann ex. $D_1 \cup \{B\}, D_2 \cup \{\neg B\} \in \phi$ mit $E_1 \cup \{B\} = D_1 \cup \{B\} \setminus \{\neg A\}$, $E_2 \cup \{\neg B\} = D_2 \cup \{\neg B\} \setminus \{\neg A\}$. Da ϕ ra., folgt $D_1 \cup D_2 \in \varphi$; da $A \notin D_1 \cup D_2$, folgt $D_1 \cup D_2 \setminus \{\neg A\} \in \varphi/A$.

Da $A \notin At(\varphi/A)$, folgt nach IV, dass φ/A erfüllbar ist, d.h. es existiert κ mit $\kappa \vDash \varphi/A$. Sei $D \in \varphi$, wir behaupten $\kappa[A \to \top] \vDash D$.

Fall 1: $A \in D$ ✓
Fall 2: $A \notin D \Rightarrow D \setminus \{\neg A\} \in \varphi/A \Rightarrow \kappa \vDash D \setminus \{\neg A\} \Rightarrow \kappa[A \to \top] \vDash D$
$\Rightarrow \kappa[A \to \top] \vDash \varphi$ $\square$ $\square$

4 Unifikation

Unifikation bezeichnet dass Problem, wie – und ob überhaupt – zwei Terme strukturell gleich gemacht, sprich *Unifiziert* werden können. Man benötigt diesen Begriff und die entsprechenden Verfahren z.B., um Herleitungen in logischen Programmiersprachen wie Prolog zu definieren und durchzuführen. Wir formalisieren zunächst den Begriff des Terms.

Was ein Term ist, hängt zunächst einmal davon ab, welche Symbole wir verwenden dürfen, um ihn aufzubauen. Dies wird über die Wahl einer *Signatur* festgelegt.

Definition 28. Eine *Signatur* ist eine Menge Σ von *Funktionssymbolen* gegebener Stelligkeit. Wir schreiben $f/n \in \Sigma$, wenn f ein n-stelliges Funktionssymbol in Σ ist. Dabei ist $n \geq 0$ eine natürliche Zahl. Insbesondere kann es auch 0-stellige Funktionssymbole geben; diese bezeichnen wir auch als *Konstanten*.

Gegeben eine Signatur Σ und eine (feste) abzählbar unendliche Menge V von *Variablen* definiert man dann die Menge der Terme E (für *expression*) durch die Grammatik

$$E ::= X \mid f(E_1, \ldots, E_n) \qquad (X \in V, f/n \in \Sigma).$$

Wir definieren die Menge $Vars(E)$ der in E vorkommenden Variablen rekursiv durch

$$Vars(X) = \{X\}$$
$$Vars(f(E_1, \ldots, E_n) = \bigcup_{i=1}^{n} Vars(E_i).$$

Beispiel 29. Wenn wir $\Sigma_{Nat} = \{0/0, suc/1\}$ wählen, bekommen wir als Terme z.B. Terme ohne Variablen der Form

$$suc(\ldots(suc(0)..)$$

[4] „ohne Einschränkung“ wird gleichbedeutend mit „ohne Beschränkung der Allgemeinheit“ verwendet, ist aber kürzer.
Konkret heißt es hier, dass der Beweis mit dem Fall $\square \in \varphi/\neg A$ ganz genauso geführt werden kann, und deshalb nur die Hälfte der Fallunterscheidung aufgeführt wird.

oder Terme mit einer Variablen X der Form

$$suc(\ldots(suc(X)..).$$

Wenn wir eine größere Signatur $\Sigma_{Natadd} = \{0/0, suc/1, add/2\}$ wählen, erhalten wir zusätzlich Terme wie

$$E = add(suc(0), add(suc(X), Y)).$$

Hier haben wir $Vars(E) = \{X, Y\}$.

Definition 30 (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 Wert zuordnet. Mit $E\sigma$ bezeichnen wir die Anwendung von σ auf E. Dabei wird jede Variable wie in σ angegeben ersetzt: $X\sigma = \sigma(X)$, $f(E_1, \ldots, E_n)\sigma = f(E_1\sigma, \ldots, E_n\sigma)$. Eine Substitution σ heißt eine *Umbenennung*, wenn $\sigma(x)$ für alle x eine Variable ist.

Die Substitution $[X_1 \mapsto E_1, \ldots, X_n \mapsto E_n]$ 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 $\Phi\,[\,] \equiv \Phi$. Für Substitutionen σ, τ bezeichnet $\sigma\tau$ die Substitution mit $(\sigma\tau)(X) = \sigma(X)\tau = \tau(\sigma(X))$, d.h. die Substitution, die entsteht, wenn man zuerst σ und dann τ ausführt.

Beispiel 31. Mit $\sigma = [X \mapsto add(Z, Y), Y \mapsto suc(X)]$ haben wir $\mathrm{Dom}(\sigma) = \{X, Y\}$ und z.B.

$$add(add(X, Z), add(X, Y))\sigma = add(add(add(Z, Y), Z), add(add(Z, Y), suc(X)).$$

Definition 32 (Gleichung, Unifikatoren, Allgemeinheitsvergleich)**.** Eine *Gleichung* $E \doteq D$ ist ein Paar (E, D) von Termen. Ein *Gleichungssystem* ist eine Menge von Gleichungen. Eine Substitution σ ist ein *Unifikator* von $E \doteq D$, wenn $E\sigma \equiv D\sigma$. Dreifaches Gleichheitszeichen ($\equiv$) meint syntaktische, nicht nur semantische Gleichheit. Ein Unifikator eines Gleichungssystem S ist eine Substitution, die Unifikator aller Gleichungen in S ist. Das System S ist *unifizierbar*, wenn es einen Unifikator hat.

$\mathsf{Unif}(S) = \{\ \sigma \mid \sigma \text{ ist Unifikator von S}\ \}$ ist die Menge aller Unifikatoren von S. Eine Substitution σ_1 ist *allgemeiner* als σ_2, wenn eine Substitution τ existiert mit $\sigma_1\tau = \sigma_2$, also wenn σ_2 durch eine ‘Spezialisierung’ τ aus σ_1 hervorgeht.

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

Beispiel 33. Zur Gleichung

$$add(suc(X), Y) \doteq add(Y, suc(Z))$$

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

$$\sigma' = \sigma[X \mapsto suc(Z)].$$

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

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

Dies können wir nutzen, um alle Unifikatoren eines Gleichungssystems zu finden, wenn wir einen *allgemeinsten Unifikator* gefunden haben:

Lemma 34. *Wenn $\sigma = mgu(S)$, dann gilt $\mathsf{Unif}(S) = \{\sigma\tau \mid \tau \text{ Subst. }\}$.*

Kennt man also den allgemeinsten Unifikator von S, so erhält durch Spezialisierung gerade alle Unifikatoren von S.

Setze nun für Substitutionen σ

$$Vars(\sigma) = \bigcup_{x \in V} Vars(\sigma(x)).$$

Lemma 35 (Eindeutigkeit des MGU)**.** *Der MGU ist eindeutig bis auf eindeutige und bijektive Umbenennung von Variablen, d.h. wenn σ, σ' allgemeinste Unifikatoren von S sind, dann gilt:*

1. *Es existiert ein auf $Vars(\sigma)$ eindeutig bestimmtes τ mit $\sigma' = \sigma\tau$;*
2. *dieses τ ist eine Umbenennung und bijektiv als Abb. $Vars(\sigma) \to Vars(\sigma')$.*

Beweis. τ existiert, da σ MGU und σ' Unifikator, ebenso existiert τ' mit $\sigma'\tau' = \sigma$.

τ *eindeutig auf* $x \in Vars(\sigma(y))$: wenn auch $\sigma' = \sigma\overline{\tau}$, dann
$\sigma(y)\tau = \sigma(y)\overline{\tau}$, also notwendig $\tau(x) = \overline{\tau}(x)$ (formal: Induktion über $\sigma(y)$).

$\tau : Vars(\sigma) \to Vars(\sigma')$ *bijektiv:* Es gilt

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

also per Eindeutigkeit $\tau\tau'(x) = x$ für $x \in Vars(\sigma)$, analog $\tau'\tau(x) = x$ für $x \in Vars(\sigma')$. Es folgt, dass τ und τ' Umbenennungen sind, und dann gegenseitig invers als Abbildungen zwischn □

Beweis der Existenz eines MGU durch Angabe eines Algorithmus, der einen solchen MGU findet:

4.0.2 Unifikationsalgorithmus von Martelli/Montanari

Definition 36 (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. 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.

Lemma 37. *Wenn $S' = \{X_1 \doteq E_1, \ldots, X_n \doteq E_n\}$ gelöste Form von S ist, dann ist $\sigma = [X_1 \mapsto E_1, \ldots, X_n \mapsto E_n$ MGU von S.*

Beweis. Da S' gelöst ist, ist σ Unifikator von S' und damit auch von S. Wenn σ' Unifikator von S ist, dann auch von S'; es gilt damit $\sigma'(X_i) = E_i\sigma'$, 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 Rewrite-Regeln:

$$S \cup (X \doteq X) \to S$$

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

$$S \cup f(E_1, \ldots, E_n) \doteq g(D_1, \ldots, D_k) \to \bot$$

$$S \cup (E \doteq X) \to S \cup (X \doteq E)$$

$$S \cup (X \doteq E) \to \begin{cases} \bot & (X \in \mathit{Vars}(E)) \\ S[E/X] \cup (X \doteq E) & (X \notin \mathit{Vars}(E), X \in \mathit{Vars}(S)) \end{cases}$$

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.

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* $\prec$. Diese definieren wir durch Rekursion über n wie folgt: Wir setzen

- $() \leq ()$
- für $n > 0$: $(a_1, \ldots, a_n) \preceq (b_1, \ldots, b_n)$ genau dann, wenn $a_1 \leq b_1$ 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 38. *Die lexikographische Ordnung ist eine* Wohlordnung*, d.h. es gibt keine unendliche absteigende Kette*

$$\vec{a}^0 \succ \vec{a}^1 \succ \vec{a}^2 \succ \ldots \tag{3}$$

Beweis. Induktion über n, mit trivialem Induktionsanfang. Wir nehmen zwecks Herleitung eines Widerspruchs an, wir hätten eine unendliche absteigende Kette (3), mit $\vec{a}^i = (a_1, \ldots, a_n)$. Dann gilt

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

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 $i \geq k$. 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_1, n_2, n_3) in lexikographischer Ordnung, wobei

- n_1 = Anzahl Variablen mit mehr als einem Vorkommen
- n_2 = Anzahl Symbole
- n_3 = Anzahl Gleichungen der Form $E = X$ mit E keine Variable.

Diese Maß nimmt offenbar in jedem Umformungsschritt ab, so dass der Algorithmus terminiert.

Wenn der Algorithmus ohne Erreichen von $\perp$ 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)* 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 $\perp$ das unlösbare System repräsentiert. Das ist klar in allen Fällen bis vielleicht auf *(occurs)*. Eine Gleichung der Form $X \doteq E$ mit $X \in \mathit{Vars}(E)$ ist aber offenbar nicht unifizierbar: der Term $E\sigma$ ist stets größer als der Term $X\sigma$.

Beispiele:

- $f(X, g(Y)) \doteq f(g(Z), Z)$
- $f(X, g(X), b) \doteq f(a, g(Z), Z)$
- $f(X, g(X)) \doteq f(Z, Z)$

5 Prolog

Logisches Programmieren = Programmieren durch *Deklaration logischer Zusammenhänge* und *Anfragen nach deren Konsequenz*.

Definition 39. Wir erweitern im folgenden den Begriff der *Signatur* dahingehend, dass eine Signatur Σ auch Prädikatensymbole $p, q, \ldots$ enthalten kann. Wie Funktionssymbole haben auch Prädikatensymbole eine Stelligkeit; wir schreiben $p/n \in \Sigma$ um anzudeuten, dass p ein n-stelliges Prädikatensymbol in Σ ist. Ein *Atom* ist dann ein Ausdruck der Form

$$p(E_1, \ldots, E_n)$$

mit Termen $E_1, \ldots, E_n$ und $p/n \in \Sigma$.

Definition 40 (definite Klausel, Zielklausel, Hornklausel, Programm, Anfrage)**.** Ein *Literal* ist ein Atom $A = p(E_1, \ldots, E_k)$ oder die Negation $\neg A$ eines Atoms. Eine *Klausel* ist eine Menge D von Literalen. D ist eine *Hornklausel*, wenn D höchstens ein positives Literal enthält. *definit*, wenn D genau ein positives Literal enthält, und eine *Zielklausel*, wenn D kein positives Literal enthält. Schreibweise:

def. Klausel $\{A_0, \neg A_1, \ldots, \neg A_n\} \equiv A_0 \leftarrow A_1, \ldots, A_n$

Zielklausel $\{\neg A_1, \ldots, \neg A_n\} \equiv \leftarrow A_1, \ldots, A_n$

Ein *Programm* ist eine endliche Menge P von definiten Klauseln (*Regeln* bzw., wenn $n = 0$, *Fakten*). Eine Anfrage ist eine *Zielklausel* $Q = \leftarrow A_1, \ldots, A_n$

Der Prolog-Interpreter auf eine Anfrage Q die allgemeinsten Substitutionen σ, so dass sich $Q\sigma$ aus dem Programm herleiten lässt. Intuitiv versteht sich dies wie folgt: man liest die Variablen im Programm als implizit allquantifiziert; damit kann man insbesondere P zu $P\sigma$ spezialisieren und dann ggf. $Q\sigma$ einfach durch Verkettung von Implikationen herleiten.

Prolog funktioniert nach der sog. *closed world assumption* (CWA), d.h. wenn eine Aussage A aus einem Programm P nicht explizit folgt, z.B. weil A überhaupt nicht erwähnt wird, so gilt A auch nicht.

Beispiel 41 (Verwandtschaften)**.** $P = \begin{cases} \text{Vater(Hans, Paula)} \\ \text{Mutter(Paula, Fritz)} \\ \text{Großvater(X, Y)} \leftarrow \text{Vater(X, Z), Mutter(Z, Y)} \end{cases}$

$Q_1 = \leftarrow$ Großvater(Hans, X): $[\mathit{Fritz}/X]$
$Q_2 = \leftarrow$ Großvater(Y, Paula): schlägt fehl, da sich keine entsprechende Aussage aus dem Programm ableiten lässt – das illustriert die CWA.

Formaler gesprochen verwendet Prolog zur (Rückwärts)transformation von Zielen (beginnend mit der Anfrage) mittels *Regeln* (inkl. Fakten) die *Resolutionsregel*

$$\text{(Res)}\ \frac{\leftarrow A_1, \ldots, A_n \qquad B_0 \leftarrow B_1, \ldots, B_m \in P}{\leftarrow A_1\sigma, \ldots, A_{i-1}\sigma, \underbrace{B_1\sigma, \ldots, B_m\sigma}_{A_i}, Ai+1\sigma, \ldots, A_n\sigma}$$

wobei σ allgemeinster Unifikator von A_i und B_0 ist. Dabei werden die Variablen der B_i so umbenannt, dass sie in den A_j nicht vorkommen; in der Praxis wird für jede Regelanwendung ein fortlaufender Index vergeben (so dass aus Variablen $X, Y, \ldots$ in den B_i bei der k-ten Regelanwendung Variablen $X_k, Y_k, \ldots$ werden).

Beispiel 42 (Listen)**.** $\Sigma = \{ \underbrace{[]/0}_{\text{leere Liste}}, \underbrace{[\_|\_]/2}_{\text{cons}}, \underbrace{0/0}_{\text{Konstante 0}}, \underbrace{1/0}_{\text{Konstante 1}} \}$

Prolog hat keine Konzepte wie „Rückgabewerte“ oder „Funktionsaufruf“, deshalb drücken wir das Anhängen eines Elements an eine Liste und das Ergebnis in einem dreistelligen Prädikat aus: append($[], X, X$) (lies: append([], X) = X)
append($[X|Y], Z, [X|W]$) $\leftarrow$ append(Y, Z, W)

Notation Zur Vereinfachung führen wir folgende abkürzenden Schreibweisen ein:

$$[a_1, a_2, \ldots, a_n] = [a_1|[a_2| \ldots a_n|[]] \ldots]$$

$$[a_1, a_2, \ldots, a_n|l] = [a_1|[a_2| \ldots a_n|l]]$$

Beispiel 43 (Auswertung von logischen Ausdrücken durch Prolog)**.**

$Q_1 : \leftarrow append([0,1],[1,0],X)$
Antwort: $X \mapsto [0,1,1,0]$

$Q_2 : \leftarrow append([0,1],X,[0,1,1])$
$\leftarrow append([1],X,[1,1])$
$\leftarrow append([],X,[1])$
Antwort: $X \mapsto [1]$
Hierbei wurde in Schritt 1 und 2 gegen die zweite Regel gematcht.

Das System liefert die Resultate *aller* erfolgreichen Ableitungen zurück und arbeitet dazu mit Backtracking, z.B.
$Q :\leftarrow append(X, Y, [0,1])$

a) $\leftarrow append([],[0,1],[0,1])$, Resultat: $[X \mapsto [], Y \mapsto [0,1]]$;

b) $append([0|X_1], Y, [0,1])$, per Substitution: $[X \mapsto [0|X_1]]$
$append(X_1, Y, [1])$

b.i) $[X_1 \mapsto [], Y \mapsto [1]]$, Resultat : $[X \mapsto [0]]$

b.ii) $X_1 \mapsto [1|X_2]$
$\leftarrow append(X_2, Y, [])$, fertig per Substitution $[X_2 \mapsto [], Y \mapsto []]$, Ergebnis $X \mapsto [0,1]$

Anmerkung: Beim realen Programmieren braucht man häufig Test auf Ungleichheit, $X \neq Y$. Dies kann in den Aufgabenlösungen ohne weitere Diskussion der Semantik verwendet werden.

5.1 SLD-Resolution

(Linear resolution for Definite clauses with Selection function) Transformation von Zielen (Anfragen) mittels *Regeln* (inkl. Fakten). Die *Selection Function* gibt abhängig von aktuellem Ziel und angewandter Regel an, welches Literal des aktuellen Ziels transformiert werden soll. Hier verwenden wir die *Prolog-Regel*, die besagt, dass immer das erste Literal des aktuellen Ziels transformiert wird, also

$$(Res)\frac{\leftarrow A_1, \ldots, A_n \qquad B_0 \leftarrow B_1, \ldots, B_m \in P}{\leftarrow B_1\sigma, \ldots, B_m\sigma, A_2\sigma, \ldots, A_n\sigma} \quad (\sigma = mgu(A_1, B_0)$$

Beispiel 44. Auf die Anfrage

$$\leftarrow append(X, [0|Y], [1, 0, 1|Z])$$

soll die Regel

$$append([X_1|Y_1], Z_1, [X_1|W_1]) \leftarrow append(Y_1, Z_1, W_1).$$

angewendet werden. Wir suchen also eine Substitution σ, die die linke Seite der Regel und (das erste Literal der) Anfrage gleich macht; wir verwenden

$$\sigma = mgu(append([X_1|Y_1], Z_1, [X_1|W_1]), append(X, [0|Y], [1, 0, 1|Z]),$$

also

$$\sigma = \{X_1 \mapsto 1, X \mapsto [1|Y_1], Z_1 \mapsto [0, 1]|Z], Y \mapsto [1|Z], W_1 \mapsto [0, 1|Z]\}.$$

Definition 45 (SLD-Herleitung)**.** Gegeben $P, \mathscr{R}$: Eine SLD-*Herleitung* von Ziel Q_0 ist eine Sequenz

$$\delta = Q_0 \overset{D_0}{\rightsquigarrow} Q_1 \overset{D_1}{\rightsquigarrow} Q_2 \ldots$$

mit $D_i \in P$ und Zielen Q_i, so dass $(\text{Res}) \dfrac{Q_i \qquad D_i}{Q_{i+1}}$ mit einer Substitution $\sigma(Q_i, D_i)$

Feinheit: Die Variablen in D_i werden umbenannt nach dem Schema $X \mapsto X_i$, um Konflikte mit den Variablen von Q_i zu vermeiden.

Terminologie Sei $\delta = Q_0 \overset{D_0}{\rightsquigarrow} Q_1 \overset{D_1}{\rightsquigarrow} Q_2 \ldots \overset{D_n}{\rightsquigarrow} Q_{n+1}$ eine *endliche* Herleitung.

Dann ist $\sigma(\delta) = \sigma(Q_0, D_0) \ldots \sigma(Q_n, D_n)$ die von δ berechnete Substitution.

Eine Herleitung δ ist eine *Refutation*, wenn die leere Klausel $\square$ hergeleitet werden kann, d.h. δ Refutation $\iff Q_{n+1} = \square$. In diesem Fall ist $\sigma(\delta)|_{Vars(Q_0)}$, also $\sigma(\delta)$, eingeschränkt auf die in $VarsQ_0$ vorkommenden Atome, eine *Antwort*.

δ heißt *fehlgeschlagen*, wenn für alle Regeln $B_0 \leftarrow B_1, \ldots, B_n \in P$ bei $Q_{n+1} = \leftarrow A_1, \ldots, A_n$ gilt, dass A_1 nicht unifizierbar mit B_0 ist, also keine weitere Regel mehr anwendbar ist.

Allgemein (auch wenn δ nicht endlich ist) gilt, dass δ *vollständig* $\iff$ δ Refutation, δ fehlgeschlagen oder δ unendlich.

Beispiel 46 (Abstammung)**.** Die Sprechweise für D sei "descendant", also "Kind von", die Sprechweise für $\rightsquigarrow$ sei "wird transformiert zu".

$$D(X, Y) \leftarrow D(X, Z), D(Z, Y) \qquad \text{Konstanten: } a, b, c$$

Nun wollen wir die Abstammungsbeziehung von a und b untersuchen:

$$\begin{aligned}
&\leftarrow D(a,a) \rightsquigarrow \\
&\leftarrow D(a,Z_1), D(Z_1,a) \rightsquigarrow \\
&\leftarrow D(a,Z_2), D(Z_2,Z_1); D(Z_1,a) \\
&\leftarrow D(a,Z_{n+1}), \ldots, D(Z_2,Z_1); D(Z_1,a) \\
&\rightsquigarrow \infty
\end{aligned}$$

Wir versinken hier also in der Endlosrekursion.

Beispiel 47 (Listenoperation *append*)**.** Unser Programm besteht aus zwei Regeln, einem Basisfall R_{nil} und einem rekursiven Fall R_{cons}.

$$\begin{aligned}
R_{nil}: &\ \text{append}([], X, X) \\
R_{cons}: &\ \text{append}([X|Y], Z, [X|W]) \leftarrow\ \text{append}(Y, Z, W)
\end{aligned}$$

Nun wollen wir wissen, in welche Teillisten man $[0,1]$ zerlegen kann. Wir beginnen also mit der Ergebnisliste $[0,1]$ und fragen Prolog, welche möglichen Belegungen für die Teillisten X und Y es gibt.

$$\begin{aligned}
&\leftarrow\ \text{append}(X, Y, [0,1]) \overset{R_{cons}}{\rightsquigarrow} \\
&\leftarrow\ \text{append}(Y_0, Z_0, [1]) \overset{R_{nil}}{\rightsquigarrow} \\
&\leftarrow \square
\end{aligned}$$

$\Rightarrow$ Refutation, weil es keine weiteren Unterziele mehr gibt.

Da die Resolution eine Refutation ergeben hat, gibt es eine Antwort, die aus der angesammelten Substitution besteht: Die Unifikation der ursprünglichen Anfrage mit der linken Seite der R_{cons}-Regel hat die Substitution $[X \mapsto [0|Y_0], Y \mapsto Z_0, X_0 \mapsto 0, W_0 \mapsto [1]]$ erzeugt, die Unifikation der Unteranfrage mit der linken Seite der R_{nil}-Regel zusätzlich noch $[Y_0 \mapsto [], X_1 \mapsto [1], Z_0 \mapsto [1]]$.

Einsetzen der zweiten Substitution in die erste bringt uns zu der Substitution

$$[X \mapsto [0|[]], Y \mapsto [1], X_0 \mapsto 0, W_0 \mapsto [1]]$$

Einschränkung auf die Variablen X und Y (alle anderen sind nach der Definition für eine Antwort egal) führt zu der Antwort $[X \mapsto [0], Y \mapsto [1]]$

6 Formale Deduktion in Aussagenlogik

`FIX: Prinzip der Schlussregel erklären`

Hilbert Viele Axiome, wenig Regeln; meist nur *modus ponens*: (mp) $\frac{\varphi \quad \varphi \to \psi}{\psi}$ [5]

Gentzen Regeln für *Sequenten* $\varphi_1, \ldots, \varphi_n \vdash \psi_1, \ldots, \psi_k$, zu lesen als 'die Konjunktion der φ_i impliziert die Disjunktion der ψ_j. Axiome sind nur Sequenten der Form $\varphi, \cdots \vdash \varphi, \ldots$.

[5] Gödel, Escher, Bach; logische Schildkröte vs. Achilles

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

Wir betrachten im folgenden ein System natürlichen Schließens, den Fitch-Kalkül.

Beispiel 48 (Regeln für eine auf Konjunktion beschränkte Logik)**.** $(\wedge I)\frac{\varphi \quad \psi}{\varphi \wedge \psi}$ $(\wedge E1)\frac{\varphi \wedge \psi}{\varphi}$ $(\wedge E2)\frac{\varphi \wedge \psi}{\psi}$

Die Regel $(\wedge I)$ („Und-Introduktion“) erlaubt es uns, sofern wir bereits φ und ψ geschlossen haben, auch $\varphi \wedge \psi$ zu schließen, die Regeln $(\wedge E1)$ und $(\wedge E2)$ („Und-Elimination“) erlauben es, aus einer bereits geschlossenen Konjunktion deren Teilsätze zu schließen.

Notation: Wir schreiben $\Phi \vdash \varphi$, wenn φ per Regeln aus Annahmen in Φ (Φ Menge von Formeln) rein syntaktisch herleitbar ist.

Schreibweise:
Baumartig:
z.B. $\{\varphi \wedge \psi\} \vdash \psi \wedge \varphi$:

$$\dfrac{(\wedge E2)\dfrac{\varphi \wedge \psi}{\varphi} \qquad (\wedge E1)\dfrac{\varphi \wedge \psi}{\psi}}{\psi \wedge \varphi}(\wedge I)$$

Der Übersichtlichkeit halber verwenden wir im folgenden eine lineare Darstellung des Beweisablaufs:

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

Wir notieren in jedem Schritt die gezogene Konklusion (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.

Innerhalb eines Beweises ist es möglich, dass ein Unterbeweis als Prämisse verwendet wird, es ergibt sich daraus eine hierarchische Struktur. Unterbeweise haben *lokale* Annahmen, die ebenso wie oben durch einen waagerechten Strich abgetrennt werden. Solche Unterbeweise werden benötigt, wenn wir unser Regelwerk auf Negation und $\bot$ erweitern:

$$(\neg I)\ \dfrac{\begin{array}{|l} \varphi \\ \hline \quad . \\ \quad . \\ \quad \bot \end{array}}{\neg\varphi} \qquad (\bot\ \mathrm{I})\ \dfrac{\varphi \quad \neg\varphi}{\bot} \qquad (\bot\ \mathrm{E})\ \dfrac{\bot}{\Phi} \qquad (\neg\ \mathrm{E})\ \dfrac{\neg\neg\varphi}{\varphi}$$

Beispiel 49 ($\vdash \neg(\varphi \wedge \neg\varphi)$)**.**

1	$\vert$	
2	$\vert\quad\vert$	$\varphi \wedge \neg\varphi$
3	$\vert\quad\vert$	φ $\quad (\wedge E1), 1$
4	$\vert\quad\vert$	$\neg\varphi$ $\quad (\wedge E1), 1$
5	$\vert\quad\vert$	$\bot$ $\quad (\bot I), 2, 3$
6	$\vert$	$\neg(\varphi \wedge \neg\varphi)$ $\quad (\neg I), 1-4$

Erweiterung auf Disjunktion und Implikation

$$(\vee I1)\frac{\varphi}{\varphi \vee \psi} \qquad (\vee I2)\frac{\psi}{\varphi \vee \psi} \qquad (\vee E)\frac{\begin{array}{ccc} \varphi & \psi & \\ \vdots & \vdots & \varphi \vee \psi \\ \chi & \chi & \end{array}}{\chi}$$

$$(\to E)\frac{\varphi \to \psi \qquad \varphi}{\psi} \qquad (\to I)\frac{\begin{array}{c} \varphi \\ \vdots \\ \psi \end{array}}{\varphi \to \psi}$$

Herleitung von $(\to I)$ **bei Codierung** $\varphi \to \psi = \neg(\neg\neg\varphi \wedge \neg\psi)$

$\vert\quad \neg\neg\varphi \wedge \neg\psi$	
$\vert\quad \neg\neg\varphi$	$(\wedge E_1)$
$\vert\quad \varphi$	$(\neg E)$
$\vert\quad \vdots$	
$\vert\quad \bot$	Annahme
$\neg(\neg\neg\varphi \wedge \neg\psi)$	$(\neg I)$

Herleitung von $(\to E)$ bei Codierung $\varphi \to \psi = \neg\varphi \vee \psi$

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

Satz 50. *(Korrektheit)*

$$\Phi \vdash \psi \Rightarrow \Phi \vDash \psi$$

Beweis. Formal gesehen müssen wir zunächst die Behauptung verallgemeinern auf Aussagen ψ, die in Unterbeweisen gefolgert werden; wir behaupten, dass für solche Aussagen $\Phi' \models \psi$ gilt, wobei Φ' aus allen in dem betreffenden Unterbeweis aktiven Annahmen besteht (inklusive Φ). Wir beweisen die verallgemeinerte Behauptung per Induktion über die Länge n der Herleitung von ψ. Hierbei verwenden wir *course-over-values induction*, d.h. wir beweisen, dass die Behauptung für $n = k$ gilt, wenn sie für alle $n \leq k$ gilt.

Wir unterscheiden dann nach der zuletzt angewandten Regel, z.B. a) $\wedge I$, b) $(\neg I)$: Der Unterbeweis in der Prämisse bedeutet, dass $\Phi \cup \{\varphi\} \vdash \bot$, 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 $\Phi \cup \{\varphi\} \vDash \bot$; mit anderen Worten, $\Phi \cup \{\varphi\}$ ist unerfüllbar. Damit folgt wie verlangt $\Phi \vDash \neg\varphi$: Wenn $\kappa \vDash \Phi$, dann $\kappa \nvDash \psi$, also $\kappa \vDash \neg\psi$. Die anderen Regeln für $\wedge$, $\neg$ und $\bot$ sind noch wesentlich einfacher; die Regeln für $\vee$ und $\to$ sind dann ebenfalls korrekt, da wir sie ja aus denen für $\wedge$, $\neg$ und $\bot$ herleiten. □

6.1 Vollständigkeit

In Umkehrung zur Korrektheit gilt auch

Satz 51. *Vollständigkeit*

$$\Phi \vDash \psi \Rightarrow \Phi \vdash \psi$$

Definition 52. Eine Formelmenge Φ 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 $\subseteq$ unter den konsistenten Mengen ist, d.h.

1. Φ ist konsistent, und
2. wenn Ψ konsistent ist und $\Phi \subseteq \Psi$, dann folgt $\Phi = \Psi$.

Beweis. Strategie des Vollständigkeitsbeweises ist

(A) Reduziere auf

Φ konsistent $\Rightarrow$ Φ erfüllbar:

Sei $\Phi \vDash \psi$; dann ist $\Phi \cup \{\neg\psi\}$ unerfüllbar, also nach obigem inkonsistent, also $\Phi \cup \{\neg\psi\} \vdash \bot$. Mittels Regel $(\neg I)$ folgt $\Phi \vdash \neg\neg\psi$, und mittels $(\neg E)$ $\Phi \vdash \psi$.

Slogan: Vollständigkeitsbeweise bestehen in *Modellkonstruktionen.*

(B) Reduziere (A) auf den Fall, dass Φ maximal konsistent ist, per

Lindenbaumlemma: Φ konsistent $\Rightarrow$ es existiert $\overline{\Phi}$ max. kons mit $\Phi \subseteq \overline{\Phi}$.

(Damit: Φ konsistent $\Rightarrow$ es ex. $\overline{\Phi}$ wie im Lemma $\Rightarrow$ $\overline{\Phi}$ erfüllbar $\Rightarrow$ Φ erfüllbar.)

□

Dazu:

Lemma 53. *(C) Sei Φ konsistent, dann $\Phi \cup \{\psi\}$ konsistent oder $\Phi \cup \{\neg\psi\}$ konsistent.*

(C). 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 $(\bot I)$ $\Phi \vdash \bot$, d.h. Φ ist inkonsistent. □

(Lindenbaumlemma). **a) Zorn** Vereinigungen aufsteigender Ketten von konsistenten Mengen sind konsistent, also hat die mittels $\subseteq$ geordnete Menge der konsistenten Mengen oberhalb einer gegebenen nach dem Zornschen Lemma maximale Elemente.

oder b) Sei $\varphi_1, \varphi_2, \varphi_3, \ldots$ Aufzählung aller Formeln. Konstruiere Kette $\Phi = \Phi_0 \subseteq \Phi_1 \subseteq \Phi_2 \subseteq \ldots$ konsistenter Formelmengen per $\Phi_{i+1} = \begin{cases} \Phi_i \cup \{\varphi_i\} & \text{wenn konsistent} \\ \Phi_i \cup \{\neg\varphi_i\} & \text{sonst (konsistent nach (C))} \end{cases}$

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

1. $\overline{\Phi}$ ist konsistent: Nimm an $\overline{\Phi} \vdash \bot$. 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 Indices i erhalten wir ein Φ_i, das *alle* verwendeten Annahmen enthält. Dann $\Phi_i \vdash \bot$, im Widerspruch zur Konsistenz von Φ_i.

2. $\overline{\Phi}$ maximal: Sei Ψ konsistent und $\overline{\Phi} \subseteq \Psi$. Zu zeigen ist dann $\Psi \subseteq \overline{\Phi}$. Sei also $\psi \in \Psi$. Es existiert n mit $\psi = \phi_n$. Da $\Phi_n \cup \{\phi_n\} \subseteq \Psi$, ist $\Phi_n \cup \{\phi_n\}$ konsistent, also $\phi_n \in \Phi_{n+1} \subseteq \Phi$.

□

Es bleibt nach Reduktion (B) zu zeigen, dass jede maximal konsistente Menge Φ erfüllbar ist. Wir halten folgende Eigenschaften maximal konsistenter Mengen fest:

Lemma 54. *(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. ad (1): Klar.

ad (2): „⇒“ klar (verwendet $(\bot I)$), „⇐“: Sei $\psi \notin \Phi$. Dann gilt $\Phi \cup \{\psi\} \supsetneq \Phi$. Da Φ maximal konsistent ist, ist $\Phi \cup \{\psi\}$ inkonsistent; nach (C) folgt, dass $\Phi \cup \{\neg\psi\}$ konsistent ist, und per maximaler Konsistenz von Φ folgt $\Phi \cup \{\neg\psi\} \subseteq \Phi$, also $\neg\psi \in \Phi$.

ad (3): „⇒“: Per Maximalität von Φ reicht es, zu zeigen, dass $\Phi \cup \{\phi\}$ und $\Phi \cup \{\psi\}$ konsistent sind. Z.B.: Nimm an, dass $\Phi \cup \{\phi\} \vdash \bot$. Per $(\wedge E1)$ gilt $\Phi \vdash \phi$, also schon $\Phi \vdash \bot$ im Widerspruch zur Konsistenz von Φ. Analog zeigt man, dass $\Phi \cup \{\psi\}$ konsistent ist.

„⇐“ läuft analog mit $(\wedge I)$. □

Um die verlangte erfüllende Wahrheitsbelegung κ für eine maximal konsistente Menge Φ zu erhalten, setzen wir nun

$$\kappa(A) = \top \iff A \in \Phi.$$

Lemma 55. *(Wahrheitslemma)* $\kappa \vDash \psi \iff \psi \in \Phi$.

Beweis. Induktion über ψ per Definition und Hintikka-Eigenschaften, z.B.: $\kappa \vDash \neg\psi \iff \kappa \nvDash \psi \overset{\text{IV}}{\iff} \psi \notin \Phi \overset{\text{Hintikka}}{\iff} \neg\psi \in \Phi$ □

Korollar 56. *(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 Vollständigkeit ist Erfüllbarkeit gleichbedeutend mit Konsistenz. Konsistenz hat 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. □

6.2 Anwendungen des Kompaktheitssatzes

Definition 57. 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 für jede Kante $\{v, w\} \in E$ $c(v) \neq c(w)$ gilt.

Satz 58. *Ein (möglicherweise unendlicher) Graph ist k-färbbar, wenn alle seine endlichen Untergraphen k-färbbar sind.*

Beweis. 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:

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

– d.h. jeder Knoten hat mindestens eine Farbe.

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

– d.h. kein Knoten hat mehr als eine Farbe.

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

– d.h. keine zwei adjazenten Knoten haben die gleiche Farbe.

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.

7 Prädikatenlogik erster Stufe

7.0.1 Terminologie

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*, typischerweise 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.

Definition 59 (Syntax der Prädikatenlogik erster Stufe)**.** Die Syntax der Prädikatenlogik hängt ab von einem Vorrat an Symbolen für Konstanten sowie für Prädikate und Funktionen gegebener Stelligkeit. Dieser Symbolvorrat wird oft als die *Sprache* bezeichnet; wir bevorzugen hier zur Vermeidung von Verwechslungen den Ausdruck *Signatur*. Formal besteht eine Signatur $\Sigma = (P_\Sigma, F_\Sigma)$ aus

- einer Menge P_Σ von *Prädikatensymbolen* und

- einer Menge F_Σ von *Funktionssymbolen.*

Jedes Funktions- oder Prädikatensymbol s hat eine endliche *Stelligkeit* $\mathrm{ar}(s) \geq 0$. Für $s \in P_\Sigma \cup F_\Sigma$ und $\mathrm{ar}(s) = n$ schreiben wir oft $s/0 \in \Sigma$, wobei wir typischerweise dadurch die Unterscheidung zwischen Funktionen- und Prädikatensymbolen sicherstellen, dass wir erstere mit Kleinbuchstaben und letztere mit Großbuchstaben bezeichnen. Ein nullstelliges Funktionssymbol $c/0 \in \Sigma$ heißt auch *Konstante*; nullstellige Prädikatensymbole entsprechen gerade den Atomen der Aussagenlogik.

Diese Daten bestimmen nun zunächst den Begriff des *Terms*: Wir unterstellen einen Vorrat V an Variablen; Terme $E, D, \ldots$ sind dann gegeben durch die BNF

$$E ::= X | f(E_1, \ldots, E_n) \qquad (X \in V, f/n \in \Sigma).$$

Die Syntax von Formeln ist dann gegeben durch die folgende BNF:

$$\varphi ::- E = D \mid P(E_1, \ldots, E_n) \mid \neg\varphi \mid \varphi_1 \wedge \varphi_2 \mid \forall X.\varphi \qquad (X \in V, P/n \in \Sigma).$$

Der Existenzquantor $\exists$ wird dann definiert durch $\exists X.\varphi := \neg\forall X.\neg\varphi$, außerdem $\bot \equiv \neg\forall X.X = X$. Weitere aussagenlogische Junktoren wie Implikation, Disjunktion etc. werden wie gewohnt definiert.

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

Wie immer folgt der Syntax-Definition die Semantik. Ein verbreitetes Prinzip der Logik ist, dass *Erfülltheit* von Formeln relativ zu einem *Modell* definiert wird, 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 60 (Σ-Modell)**.** Ein Σ-*Modell* $\mathfrak{M}$ besteht aus

- Einer Menge M („Universum“, „(Grund)bereich“, „Träger“ ...)
- Für ein n-stelliges Funktionssymbol $f/n \in \Sigma$ eine Interpretation in $\mathfrak{M}$, gegeben durch eine Funktion $\mathfrak{M}[\![f]\!] : M^n \to M$
- Für ein n-stelliges Prädikatensymbol $P/n \in \Sigma$ eine Teilmenge $\mathfrak{M}[\![P]\!] \subseteq M^n$

Eine *Umgebung* η (in $\mathfrak{M}$) ist eine Abbildung $\eta : V \to M$.

Definition 61 (Erfülltheit)**.** Zu einem Term E definieren wir seine Bedeutung $\mathfrak{M}[\![E]\!]\eta \in M$ rekursiv durch

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

Die Erfülltheit einer Formel φ (bestehend aus n Termen) auf einem Modell und einer Umgebung $\mathfrak{M}, \eta \models \varphi$ ist rekursiv definiert durch

$$\begin{aligned}
\mathfrak{M}, \eta \models (E = D) &\iff \mathfrak{M}[\![E]\!]\eta = \mathfrak{M}[\![D]\!]\eta \\
\mathfrak{M}, \eta \models P(E_1, \ldots, E_n) &\iff (\mathfrak{M}[\![E_1]\!]\eta, \ldots, \mathfrak{M}[\![E_n]\!]\eta) \in \mathfrak{M}[\![P]\!] \\
\mathfrak{M}, \eta \models \forall X.\varphi &\iff \text{Für alle } m \in M \text{ gilt } \mathfrak{M}, \eta[X \mapsto m] \models \varphi
\end{aligned}$$

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

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

Wir definieren rekursiv die Menge aller *freien Variablen* in einer Formel wie folgt; alle Variablen, die nicht nach dieser Definition frei sind, sind *gebunden*.

Definition 62 (Freie Variable)**.** $\forall X.\varphi$ *bindet* X, welches damit keine freie Variable mehr ist. Die freien Variablen eines Terms bzw. einer Formel sind

$$\begin{aligned}
FV(\mathrm{E}) &= VarsE \\
FV(\mathrm{E} = \mathrm{D}) &= FV(\mathrm{E}) \cup FV(\mathrm{D}) \\
FV(\mathrm{P}(\mathrm{E}_1, \ldots, \mathrm{E}_n)) &= \bigcup_{i=1}^{n} FV(\mathrm{E}_i) \\
FV(\neg\varphi) &= FV(\varphi) \\
FV(\varphi \wedge \pi) &= FV(\varphi) \cup FV(\pi) \\
FV(\forall \mathrm{X}.\varphi) &= FV(\varphi) \setminus \{X\}
\end{aligned}$$

Z.B. gilt $FV(\mathrm{X} = \mathrm{Y} \wedge \forall \mathrm{Y}.\mathrm{Y} = \mathrm{W}) = \{X, Y, W\}$.

Definition 63 (Satz)**.** Eine Formel φ heißt *Satz*, wenn $FV(\varphi) = \emptyset$, wenn also alle Variablen gebunden sind.

Nun wollen wir zeigen, dass die freien Variablen das tun, was wir haben wollten, als wir sie definiert haben, dass nämlich 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 $\forall X.\, X = Y \vee \neg\forall X.\, X = Y$).

Lemma 64. *Sei* $\eta_1(X) = \eta_2(X)$ *für alle* $X \in FV(\varphi)$*, dann* $\mathfrak{M}, \eta_1 \models \varphi \iff \mathfrak{M}, \eta_2 \models \varphi$.

Lemma 64. Da in Formeln Terme vorkommen, müssen wir zunächst eine entsprechende Aussage für Terme beweisen, nämlich, dass für alle Terme E und alle Umgebungen η_1, η_2 mit $\eta_1(X) = \eta_2(X)$ für alle $X \in FV(\mathrm{E})$

$$\mathfrak{M}[E]\eta_1 = \mathfrak{M}[E]\eta_2 \tag{4}$$

gilt. 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 (4) abgehandelt, z.B. $\mathfrak{M}, \eta_1 \models E = D \iff \mathfrak{M}[\![E]\!]\eta_1 = \mathfrak{M}[\![D]\!]\eta_1 \overset{(4)}{\iff} \mathfrak{M}[\![E]\!]\eta_2 =$

$\mathfrak{M}[\![D]\!]\eta_2 \iff \mathfrak{M}, \eta_2 \models E = D$. Die Booleschen Fälle sind trivial. Es bleibt der Allquantor:

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

Zur Anwendung der Induktionsvoraussetzung ist hierbei zu zeigen, dass

$$\eta_1[X \mapsto m](Y) = \eta_2[X \mapsto m](Y) \text{ für alle } Y \in FV(\varphi) \subseteq FV(\forall \mathrm{X}.\varphi) \cup \{X\}$$

Dass diese Gleichheit gilt, ist klar dadurch, dass die Gleichheit für $FV(\forall \mathrm{X}.\varphi)$ gegeben ist, und falls $Y = X$, so wird Y auf beiden Seiten der Gleichung auf m gesetzt, somit auch gleich und wir können schreiben $\mathfrak{M}, \eta_2 \models \forall X.\varphi$. □ □

Damit ist für einen *Satz* φ (der ja keine freien Variablen hat) $\mathfrak{M}, \eta \models \varphi$ von η unabhängig, schreibe $\mathfrak{M} \models \varphi$.

Analog wie schon für aussagenlogische Formeln definieren wir für eine Menge Φ von Formeln $\mathfrak{M}, \eta \models \Phi \iff \mathfrak{M}, \eta \models \varphi$ für alle $\varphi \in \Phi$. Logische Konsequenz, Erfüllbarkeit, Gültigkeit, logische Äquivalenz werden analog wie bisher definiert, bis auf Ersetzung der Wahrheitsbelegung κ durch $\mathfrak{M}, \eta$ (oder nur $\mathfrak{M}$ bei Sätzen). 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 65 (Logische Konsequenz)**.** $\exists X.\forall Y.\mathrm{Loves}(Y, X) \models \forall Y.\exists X.L(Y, X)$

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

$$\mathfrak{M}, [X \mapsto m] \models \forall Y.L(Y, X). \tag{5}$$

Sei $n \in M$; zu zeigen ist $\mathfrak{M}, [Y \mapsto n] \models \exists X.L(Y, X)$. Wegen (5) gilt $\mathfrak{M}, [Y \mapsto n, X \mapsto m] \models L(Y, X)$, woraus die Behauptung folgt. □

$\forall Y.\exists X.L(Y, X) \not\models \exists X.\forall Y.L(Y, X)$: Gegenmodell z.B. $M = \{0, 1\}$, $\mathfrak{M}[\![L]\!] = \{(1, 0), (0, 1)\}$.

Beispiel 66 (Logische Äquivalenz)**.**

$$\neg\forall X.\varphi \equiv \exists X.\neg\varphi \qquad \neg\exists X.\varphi \equiv \forall X.\neg\varphi$$

Mittels der Äquivalenzen im Beispiel lässt sich die Konstruktion der NNF auf prädikatenlogische Formeln verallgemeinern; dazu müssen natürlich $\forall$ und $\exists$ als vollberechtigte Bestandteile der Grammatik von Formeln angesehen werden (ebenso wie schon im aussagenlogischen Fall $\wedge$ und $\vee$).

Wichtig ist zunächst

Lemma 67 (Substitutionslemma)**.** *Es gilt*

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

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

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

□

7.1 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

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

Damit beweist man andere erwartete Eigenschaften von Gleichheit, etwa Symmetrie

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

und Transititvität

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

Die Regeln für $\forall$ formalisieren bekannte Schlussweisen: Die $\forall$-Elimination lautet

$$\forall\mathrm{E}\ \frac{\forall X.\,\phi}{\phi[c/X]}$$

also umgangssprachlich „Wenn für alle X ϕ gilt, dann gilt ϕ auch für E“.

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

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

wobei c eine *frische* Konstante ist, d.h. bisher nirgendwo vorkommt (dies wird durch die Box um das c angedeeutet).

Die Einführungsregel für $\exists$ ist ebenso intuitiv:

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

D.h. „Wenn ϕ für ein c gilt, dann gibt es ein X, für das ϕ gilt“.
Etwas gewöhnungsbedürftiger ist die Eliminationsregel:

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

wiederum mit der Seitenbedingung, dass c frisch ist.

Beweis von (∃ I) Als erste Anwendung der Regeln für ∀ können wir die Regeln für ∃ per $\exists X.\,\phi \equiv \neg\forall X.\,\neg\phi$ herleiten:

1	$\phi[c/X]$	
2	$\quad\forall X.\,\neg\phi$	
3	$\quad\neg\phi[c/X]$	(∀E, 2)
4	$\quad\bot$	(⊥I 1, 3)
5	$\neg\forall X.\,\neg\phi$	(¬I 1-4)
6	$\exists X.\,\phi$	

Beweis von (∃ E)

1	$\neg\forall X.\,\neg\phi$	
2	$\quad\boxed{c}\ \ \phi[c/X]$	
3	$\quad\vdots$	
4	$\quad\psi$	
5	$\quad\neg\psi$	
6	$\quad\boxed{c}$	
7	$\quad\quad\phi[c/X]$	
8	$\quad\quad\psi$	(2, 3)
9	$\quad\quad\bot$	(⊥I 4,7)
10	$\quad\neg\phi[c/X]$	(¬I 6-8)
11	$\quad\forall X.\,\neg\phi$	(∀I 5-9)
12	$\quad\bot$	(⊥I 1,10)
13	$\neg\neg\psi$	(¬I 4-11)
14	ψ	(¬E, 12)

Folgerung der Existenz aus der Allquantifiziertheit $(\forall X.\, P(X)) \to (\exists X.\, P(X))$

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

Wir nehmen also einfach *irgendein* c, dass laut Bedingung $P(X)$ erfüllt, und können damit dann sagen es existiert ein Ausdruck c, der $P(c)$ erfüllt.

Transitivität des $\forall$-Quantors

1	$\forall X.\, (P(X) \to Q(X))$	
2	$\forall Z.\, (Q(Z) \to R(Z))$	
3	$\boxed{\text{d}}$ $P(d)$	
4	$P(d) \to Q(d)$	($\forall$E, 1)
5	$Q(d)$	($\to$E 3, 4)
6	$Q(d) \to R(d)$	($\forall$E 2)
7	$R(d)$	($\to$E 5, 6)
8	$\forall X.\, (P(X) \to R(X))$	($\forall$I* 3-7)

(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; dies sieht man am besten daran, dass die Umkehrung keine logische Folgerung (per Korrektheit also auch nicht herleitbar) ist, was man leicht intuitiv versteht, wenn man für P z.B. *likes* liest.

1	$\exists X.\, \forall Y.\, P(X, Y)$	
2	$\boxed{\text{c}}$ $\forall Y.\, P(c, Y)$	
3	$\boxed{\text{d}}$	
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,2–6)

8 Vollständigkeit der Prädikatenlogik erster Stufe

Zur Vereinfachung hier: ohne Gleichheit (=), ohne Funktionssymbole positiver Stelligkeit (aber mit Konstanten). Wir beweisen die Umkehrung des Korrektheitssatzes, d.h.

Satz 68 (Vollständigkeit der Prädikatenlogik erster Stufe)**.** *Sei Φ eine Menge von Sätzen und ψ ein Satz über einer Signatur Σ in Prädikatenlogik erster Stufe. Wenn ψ logische Folgerung aus Φ ist ($\Phi \vDash \psi$), dann ist ψ aus Φ herleitbar ($\Phi \vdash \psi$).*

Bemerkung 69. Eine Version des Satzes für beliebige Formeln (d.h. solche mit freien Variablen) ist leicht reduzierbar auf die obige Version, indem man einfach Variablen durch Konstanten ersetzt.

Wie schon im Falle der Auszsagenlogik ist der Beweis der Vollständigkeit wesentlich schwieriger als der der Korrektheit. 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 $\phi(X)$ über Σ_H eine *Zeugenkonstante* $c_{\varphi(X)}$ 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 $(\exists X.\, \varphi(X)) \to \varphi(c_{\varphi(X)})$.

(B) (*Henkin-Elimination*) Wir beweisen dann 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 alleine herleiten .

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

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 als Menge von Formeln in Logik erster Stufe erfüllbar ist.

Im einzelnen werden die Schritte der Beweisstrategie wie folgt umgesetzt.

(A) Wir brauchen für jede Formel $\varphi(X)$ eine Zeugenkonstante $c_{\varphi(x)}$, *aber*: Zeugenkonstanten erzeugen neue Formeln, z.B. $\exists y.P\left(y, c_{\varphi(c)}\right)$. Die Lösung besteht darin, die Hinzufügung von Zeugenkonstanten zu iterieren. Wir definieren also eine aufsteigende Folge $\Sigma = \Sigma_0 \subseteq \Sigma_1 \subseteq \Sigma_2 \subseteq \dots$ von Signaturen und setzen $\Sigma_H = \bigcup_{i=0}^{\infty} \Sigma_i$. Wir können dann für jedes $c_{\varphi(x)}$ einen *Geburtstag* $min\{i | c_{\varphi(x)} \leftarrow \Sigma_i\}$ 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 drei Axiomenschemata:

H1: $(\exists x.\varphi(x)) \to \varphi(c_{\varphi(c)})$

H2: $\varphi(c) \to \exists x.\varphi(x)$

H3: $\neg\forall x.\varphi(x) \leftrightarrow \exists x.\neg\varphi(x)$

(B) H2 und H3 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 die Instanz von H1 mit dem jüngstem $c_{\varphi(x)}$. Wir bezeichnen die Menge aller anderen im Beweis vorkommenden Instanzen von H1 mit $\mathscr{H}_0$, so dass

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

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

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

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 nun per ($\exists$E)

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

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

(C): 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 dem

Wahrheitslemma: Für jeden Satz ρ gilt $\mathfrak{M}_\kappa \vDash \rho$ genau dann, wenn $\kappa \vDash \rho$.

Beweis(Wahrheitslemma). Induktion über die *Größe* von ρ, die wir definieren, indem wir 3 für $\forall x$, und 1 für $\exists X, \neg, \wedge$ zählen. Der Induktionsanfang ist $\rho = P(c_1,\ldots,c_n)$; für solche Formeln gilt die Behauptung nach Konstruktion von $\mathfrak{M}_\kappa$. Die Booleschen Fälle ($\neg, \wedge$) sind klar. Der Fall $\rho \equiv \forall X.\,\varphi(x)$ wird unter Ausnutzung der Definition der Größe von ρ durch Umformung von $\forall$ in $\neg\exists\neg$ erledigt; im Detail:

$$\begin{aligned} \mathfrak{M}_\kappa \vDash \forall X.\,\varphi(X) &\Leftrightarrow \mathfrak{M}_\kappa \nvDash \underbrace{\overbrace{\exists X.\,\neg}^{2}\,\varphi(X)}_{\textit{kleiner als } \underbrace{\forall X}_{3}.\,\varphi(X)} \\ &\overset{IV}{\Leftrightarrow} \kappa \nvDash \exists X.\,\neg\varphi(X) \\ &\overset{H3}{\Leftrightarrow} \kappa \nvDash \neg\forall X.\,\varphi(X) \\ &\Leftrightarrow \kappa \vDash \forall X.\,\varphi(X). \end{aligned}$$

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

„$\Rightarrow$“: Sei $\mathfrak{M}_\kappa \models \exists X.\, \phi(X)$. Dann existiert $c \in \Sigma_{\mathscr{H}}$ mit $\mathfrak{M}_\kappa \vDash \varphi(c)$. Da die Formel $\varphi(X)$ kleiner als $\exists X.\, \varphi(X)$ ist, folgt nach Induktionsvoraussetzung $\kappa \vDash \varphi(c)$. Mit H2 erhalten wir schließlich $\kappa \vDash \exists X.\, \varphi(X)$.

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

Korollar 70 (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 71. Sei Φ eine Axiomatisierung der natürlichen Zahlen in Logik erster Stufe (z.B. die Peano-Axiome, mit einem Axiomenschema für Induktion). Ferner bezeichne $\mathbb{N}$ das Standardmodell der natürlichen Zahlen (z.B. die Menge aller Terme der Form $suc(suc(\dots(0)\dots))$). Dann ist $\Phi \cup \{c > k | k \in \mathbb{N}\}$ 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. (In Logik zweiter Stufe, in der wir z.B. über Teilmengen des Grundbereichs quantifizieren können, geht das, wenn wir die sogenannte Standardsemantik annehmen, d.h. insbesondere dass ein Quantor über Teilmengen wirklich über *alle* Teilmengen des Grundbereichs läuft; eine Konsequenz hieraus ist, dass diese Logik nicht vollständig ist.)