

Grundlagen der Logik und Logikprogrammierung

Vorlesungsskript Wintersemester 2013/14

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, mit weiteren Beiträgen von

Andreas Schieb

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 Mitschrift.

Inhaltsverzeichnis

0.1	Literatur	4
0.2	Logik in der Informatik	4
1	Aussagenlogik	5
1.0.1	Grundelemente einer Logik	5
1.1	Syntax	5
1.1.1	Backus-Naur-Form	5
1.1.2	Syntax der Aussagenlogik	6
1.2	Semantik der Aussagenlogik	6
1.2.1	Informelle Semantik	6
1.2.2	Wahrheitsbelegungen und Erfülltheit	7
1.3	Logische Konsequenz	7
1.4	Wahrheitstafeln	9
1.5	Logische Äquivalenzen	10
2	Induktion	11
2.1	Induktion über natürliche Zahlen	11
2.2	Strukturelle Induktion	13
3	Normalformen	16
3.0.1	Negationsnormalform (NNF)	16
3.0.2	Konjunktive Normalformen (CNF)	16
3.1	Resolution	18
3.1.1	Formale Schlussregeln	19
4	Unifikation	20
4.0.2	Unifikationsalgorithmus von Martelli/Montanari	23
5	Prolog	26
5.1	SLD-Resolution	28
6	Formale Deduktion in Aussagenlogik	29
6.1	Vollständigkeit	33
6.2	Anwendungen des Kompaktheitssatzes	35
7	Prädikatenlogik erster Stufe	36
7.1	Natürliches Schließen in Prädikatenlogik	40

8	Quantorenelimination	42
9	Vollständigkeit der Prädikatenlogik erster Stufe	45

0.1 Literatur

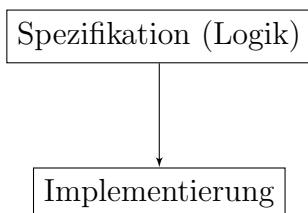
U. Schöning Logik für Informatiker, Spektrum akademischer Verlag, 2000

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, \dots ohne Rücksicht auf deren innere Struktur.

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

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

1.0.1 Grundelemente einer Logik

- *Syntax*: „Was kann ich hinschreiben?“
- *Semantik*: „Was bedeutet das?“
- *Beweismethoden*: „Wie ziehe ich daraus Schlüsse?“

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 expandiert 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 eine Familie von Mengen, nämlich für jedes Nichtterminal a die Menge aller Terme, die aus a 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

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

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

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

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

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

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 “
$\varphi \rightarrow \psi$	„wenn ϕ , dann ψ “
$\varphi \leftrightarrow \psi$	„genau dann φ , wenn ψ “
\top	„Wahr“
\perp	„Falsch“

Hierbei ist „oder“ als *inklusive 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 $\phi \rightarrow \psi$ stets wahr – z.B. auch dann, wenn ψ falsch ist. Ferner ist $\phi \rightarrow \psi$ 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
\rightarrow	Implikation
\leftrightarrow	Bimplikation, Äquivalenz
\top	Wahrheit, konstant wahre Aussage
\perp	Falschheit, konstant falsche Aussage, Absurdität

1.2.2 Wahrheitsbelegungen und Erfülltheit

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

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

- (i) $\kappa \not\models \perp$
- (ii) $\kappa \models A \Leftrightarrow \kappa(A) = \top$
- (iii) $\kappa \models \varphi \wedge \psi \Leftrightarrow \kappa \models \varphi$ und $\kappa \models \psi$
- (iv) $\kappa \models \neg\varphi \Leftrightarrow \kappa \not\models \varphi$

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:

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

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

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

Es gilt $\kappa \models A \vee \neg B \iff (\kappa \models A \text{ oder } \kappa \models \neg B)$. Nun gilt $\kappa(A) = \top$, also $\kappa \models A$, somit $\kappa \models A \vee \neg B$.

Es gilt aber $\kappa \not\models B$ (da $\kappa(B) = \perp$), also $\kappa \not\models (A \vee \neg B) \rightarrow B$.

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 $\kappa_2 \models (A \vee \neg B) \rightarrow B$.

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.

$$\kappa \models \Phi : \iff \forall \varphi \in \Phi. \kappa \models \varphi$$

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 $\kappa : \mathcal{A} \rightarrow 2$ gilt, dass, falls $\kappa \models \Phi$, so auch $\kappa \models \psi$. Man schreibt für „ ψ ist logische Konsequenz von Φ “ auch $\Phi \models \psi$. In Symbolen:

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

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:

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

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: $\Phi \models \perp$. Dies ist gleichbedeutend damit, dass keine Wahrheitsbelegung Φ erfüllt: $\forall \kappa : \kappa \not\models \Phi$.

Andernfalls, d.h. wenn $\exists \kappa : \kappa \models \Phi$, 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:

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

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:*

$$\Phi \models \psi \iff (\Phi \cup \{\neg\psi\}) \models \perp.$$

Beweis. Wir zeigen zwei Implikationen:

„ \Leftarrow “: 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.

„ \Rightarrow “: 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 $\Phi \cup \{\neg\psi\} \not\models \perp$. 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: $A \rightarrow \neg A$ (Für die Belegung $\kappa(A) = \perp$)

Unerfüllbar: $A \wedge \neg A$

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

Logische Konsequenz: $\{A \rightarrow B, A\} \models B$ (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. Wahrheitstafeln liefern Entscheidungsalgorithmen für Erfüllbarkeit, Gültigkeit, logische Konsequenz und logische Äquivalenz in der Aussagenlogik; diese sind aber offenbar in der Praxis nicht skalierbar, da stets die gesamte (exponentiell große) Wahrheitstafel erzeugt werden muss. Konkret:

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

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

A	B	$\neg A$	$\neg A \vee B$
\perp	\perp	\top	\top
\perp	\top	\top	\top
\top	\perp	\perp	\perp
\top	\top	\perp	\top

$\neg A \vee A$ ist gültig:

A	$\neg A$	$\neg A \vee A$
\perp	\top	\top
\top	\perp	\top

$A \rightarrow \neg A$ ist erfüllbar:

A	$\neg A$	$A \rightarrow \neg A$
\perp	\top	\top
\top	\perp	\perp

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

A	B	$A \rightarrow B$	$\neg(A \rightarrow B)$
\perp	\perp	\top	\perp
\perp	\top	\top	\perp
\top	\perp	\perp	\top
\top	\top	\top	\perp

$\{A \rightarrow B, A\} \models B$:

A	B	$A \rightarrow B$
\perp	\perp	\top
\perp	\top	\top
\top	\perp	\perp
\top	\top	\top

(hier muss man Erfülltheit von B nur in der letzten Zeile prüfen, da nur dort die beiden Annahmen A und $A \rightarrow B$ erfüllt sind).

$B \vee \neg B \equiv A \vee \neg A$: 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 $\text{At}(\varphi)$ der in φ vorkommenden Atome ist rekursiv definiert durch

$$\begin{aligned}\text{At}(A) &= \{A\} \\ \text{At}(\top) &= \emptyset \\ \text{At}(\neg\varphi) &= \text{At}(\varphi) \\ \text{At}(\varphi \wedge \psi) &= \text{At}(\varphi) \cup \text{At}(\psi)\end{aligned}$$

Beispiel 11. Wir berechnen $\text{At}((A \wedge B) \wedge \neg A)$:

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

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

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

$$\kappa \models \varphi \iff \kappa' \models \varphi.$$

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 $\kappa \models \varphi$ für alle $\kappa : A_0 \rightarrow 2$ mit $A_0 \subseteq \mathcal{A}$ endlich, $\text{At}(\varphi) \subseteq A_0$.

Wir präzisieren mittels der At-Notation das Wahrheitstafelverfahren für logische Äquivalenz:

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

1.5 Logische Äquivalenzen

Im folgenden geben wir eine Übersicht wichtiger logischer Äquivalenzen.

- $\neg\neg\varphi \equiv \varphi$ (Doppelnegationselimination)
- $\left. \begin{aligned} \neg(\varphi \wedge \psi) &\equiv (\neg\varphi \vee \neg\psi) \\ \neg(\varphi \vee \psi) &\equiv (\neg\varphi \wedge \neg\psi) \end{aligned} \right\} \text{ (De Morgansche Gesetze)}$

- $\left. \begin{array}{l} \varphi \wedge (\psi \vee \chi) \equiv (\varphi \wedge \psi) \vee (\varphi \wedge \chi) \\ \varphi \vee (\psi \wedge \chi) \equiv (\varphi \vee \psi) \wedge (\varphi \vee \chi) \end{array} \right\}$ (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)
- $\left. \begin{array}{l} \chi \wedge \top \equiv \chi \\ \chi \vee \perp \equiv \chi \end{array} \right\}$ (Neutrale Elemente)
- $\varphi \wedge \psi \equiv \psi \wedge \varphi$ (Kommutativität)
- $\varphi \wedge \varphi \equiv \varphi$ (Idempotenz)

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:

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

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

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

Jede positive natürliche Zahl ist ein endliches Produkt von Primzahlen

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

Satz 14 (Course-of-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

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

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 $\forall k \leq 0. P(k)$.

¹Wir 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.

- 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. 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 geparkt 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 Induktionsschritt für B_i natürlich eher eine Art Induktionsanfang (von denen es dann mehrere geben kann), da man keine Induktionsvoraussetzung hat.

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

$$n ::= Z \mid S(n)$$

ansetzen – diese sind $Z, S(Z), S(S(Z)), S(S(S(Z))), \dots$. 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,

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

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

- $P(\perp)$;
- $P(A)$ für alle $A \in \mathcal{A}$;
- 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 $f : X \rightarrow Y$ 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 $\mathcal{A} \rightarrow \{\perp, \top\}$, an.

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

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

(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 .

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

A (mit $A \in \mathcal{A}$): Es gilt per Definition $\kappa \models A$ genau dann, wenn $\kappa(A) = \top$. Da nach Voraussetzung $B \notin \text{At}(A) = \{A\}$, 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 $A \notin \text{At}(\neg\phi) =$

$\text{At}(\phi)$, 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 $A \notin \text{At}(\phi \wedge \psi) = \text{At}(\phi) \cup \text{At}(\psi)$, also $A \notin \text{At}(\phi)$ und $A \notin \text{At}(\psi)$. 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-of-Values Induction über die textuelle Größe von Instanzen handelt; z.B. wird im obigen Induktionsschritt 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 $\phi \wedge \psi$

$$\phi \rightarrow \phi \wedge \psi \rightarrow \dots,$$

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 definierte 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 ist², und zum anderen durch strukturelle Induktion, dass jede Instanz gemäß der neuen Definition in endlich vielen Schritten ableitbar ist.

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

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. Wir halten explizit fest:

Satz 18. *Jede Formel φ hat eine NNF.*

Beispiel 19 (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)

Alternativ ist $NNF(\varphi)$ *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 $NNF(\varphi) \equiv \varphi$.

Beispiel 20.

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

3.0.2 Konjunktive Normalformen (CNF)

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

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

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

Literale $L ::= A \mid \neg A \quad A \in \mathcal{A}$

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

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 $L_1 \vee \dots \vee L_n$ wird repräsentiert als die Menge $\{L_1, \dots, L_n\}$ (was von der Reihenfolge und eventuellen Wiederholungen abstrahiert!). Die leere Klausel repräsentiert \perp und wird als \square notiert.

CNFs sind endliche Mengen von Klauseln, wobei die leere Menge \top repräsentiert: $\top \hat{=} \emptyset$; $D_1 \wedge \dots \wedge D_n \hat{=} \{D_1, \dots, D_n\}$

Definition 21 (CNF C einer Formel φ). Sei φ eine Formel. Eine CNF φ' heißt *CNF von φ* , wenn $\varphi \equiv \varphi'$.

Lemma 22. *Sei φ eine Formel. Dann hat φ eine CNF $\text{CNF}(\varphi)$.*

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

$$(\psi \wedge \chi) \vee \rho \equiv (\psi \vee \rho) \wedge (\chi \vee \rho) \quad (1)$$

(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. \square

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

$$\text{CNF}(\varphi \wedge \psi) = \text{CNF}(\varphi) \wedge \text{CNF}(\psi)$$

$$\text{CNF}(L) = L$$

$$\text{CNF}(\varphi \vee \psi) = \bigwedge_{j=1}^k \bigwedge_{i=1}^n (D_i \vee E_j) \text{ mit } \text{CNF}(\varphi) = \bigwedge_{i=1}^n D_i \text{ und } \text{CNF}(\psi) = \bigwedge_{i=1}^k E_i.$$

Man zeigt durch Induktion über φ , dass $\varphi \equiv \text{CNF}(\varphi)$ und dass $\text{CNF}(\varphi)$ 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:

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

Problem an $\text{CNF}(\varphi \vee \psi)$: $n \cdot k$ Klauseln, z.B. $\text{CNF}((A_1 \wedge B_1) \vee \dots \vee (A_n \wedge B_n))$ hat 2^n Klauseln. Durch Einführung zusätzlicher Atome lässt sich der Blowup³ polynomiell halten, dann ist aber $\overline{\text{CNF}(\varphi)}$ nur noch *erfüllbarkeitsäquivalent* zu φ ($\overline{\text{CNF}(\varphi)}$ erfüllbar $\Leftrightarrow \varphi$ erfüllbar).

Beispiel 23.

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

3.1 Resolution

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

Definition 24 (Resolutionsregel).

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

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

Beispiel 25.

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

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*.

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

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, sobald die Prämissen hergeleitet sind, auch die Konklusion hergeleitet werden darf. Typischerweise werden die Regeln benannt, der Name wird in Klammern auf Höhe des „Bruchstrichs“ vor die Regel geschrieben.

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

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 26. *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.*

Beweis. „ \Leftarrow “ trivial, da linke Seite Teilmenge der rechten Seite

„ \Rightarrow “ Sei $\kappa \models \varphi$, dann $\kappa \models D_1 \cup \{A\}, \kappa \models D_2 \cup \{\neg A\}$

Fall 1: $\kappa(A) = \top \Rightarrow \kappa \models D_2 \Rightarrow \kappa \models D_1 \cup D_2$

Fall 2: $\kappa(A) = \perp \Rightarrow \kappa \models D_1 \Rightarrow \kappa \models D_1 \cup D_2 \quad \square$

Basierend auf obigem Lemma und obiger Regel können wir den folgenden Algorithmus formulieren, der die Erfüllbarkeit einer CNF entscheidet:

Definition 27 (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 28.

$$\begin{aligned} &\{D, B, \neg C\}, \{D, C\}, \{\neg D, B\}, \{\neg C, B, \neg A\}, \{C, B, \neg A\}, \{\neg B, \neg A\}, \{\neg B, A\} \\ &\{A, \neg B\}, \{A, B\} \end{aligned}$$

Wir beweisen die totale Korrektheit des Algorithmus:

Terminierung. Der Algorithmus arbeitet ausschließlich auf der Menge $\text{At}(\varphi)$ 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 $\text{At}(\varphi)$, nämlich $4^{|\text{At}(\varphi)|}$ (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. \square

Korrektheit (Antwort „nein“ ist richtig). Klar per Lemma 26. □

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 $|\text{At}(\varphi)|$:

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 $A \in \text{At}(\varphi)$. Sei

$$\begin{aligned}\varphi/A &= \{ D \setminus \{\neg A\} \mid A \notin D \in \varphi \} \\ \varphi/\neg A &= \{ D \setminus \{A\} \mid \neg A \notin D \in \varphi \}.\end{aligned}$$

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

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 Einschränkung⁴ sei $\square \notin \varphi/A$.

Nun ist auch φ/A ra.: Sei $E_1 \cup \{B\}, E_2 \cup \neg\{B\} \in \varphi/A$ (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 $E_1 \cup E_2 = D_1 \cup D_2 \setminus \{\neg A\} \in \varphi/A$.

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

Fall 1: $A \in D$ ✓

Fall 2: $A \notin D \Rightarrow D \setminus \{\neg A\} \in \varphi/A \Rightarrow \kappa \models D \setminus \{\neg A\} \Rightarrow \kappa[A \rightarrow \top] \models D$.

□

4 Unifikation

Unifikation bezeichnet das 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 29. 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

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.

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

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

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

$$\begin{aligned} Vars(X) &= \{X\} \\ Vars(f(E_1, \dots, E_n)) &= \bigcup_{i=1}^n Vars(E_i). \end{aligned}$$

Beispiel 30. Wenn wir $\Sigma_{Nat} = \{0/0, suc/1\}$ wählen, bekommen wir als Terme z.B. Terme ohne Variablen der Form

$$suc(\dots(suc(0)..))$$

oder Terme mit einer Variablen X der Form

$$suc(\dots(suc(X)..).$$

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

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

endlich ist, d.h. σ verändert nur eine endliche Anzahl an Variablen. $\text{Dom}(\sigma)$ („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: $X\sigma = \sigma(X)$, $f(E_1, \dots, E_n)\sigma = f(E_1\sigma, \dots, E_n\sigma)$. Eine Substitution σ heißt eine *Umbenennung*, wenn $\sigma(x)$ für alle x eine Variable ist.

Die Substitution $[X_1 \mapsto E_1, \dots, X_n \mapsto E_n]$ 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 $E[\] \equiv E$. 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 32. Mit $\sigma = [X \mapsto add(Z, Y), Y \mapsto suc(X)]$ haben wir $\text{Dom}(\sigma) = \{X, Y\}$ und z.B.

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

Definition 33 (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.

$\text{Unif}(S) = \{ \sigma \mid \sigma \text{ ist Unifikator von } S \}$ 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* ($\sigma = \text{mgu}(S)$, *most general unifier*), wenn σ allgemeiner als jeder Unifikator von S ist.

Beispiel 34. Zur Gleichung

$$\text{add}(\text{suc}(X), Y) \doteq \text{add}(Y, \text{suc}(Z))$$

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

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

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

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

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

Lemma 35. *Wenn $\sigma = \text{mgu}(S)$, dann gilt $\text{Unif}(S) = \{ \sigma\tau \mid \tau \text{ Subst. } \}$.*

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

Setze nun für Substitutionen σ

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

Lemma 36 (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 $\text{Vars}(\sigma)$ eindeutig bestimmtes τ mit $\sigma' = \sigma\tau$;*
2. *dieses τ ist eine Umbenennung und bijektiv als Abb. $\text{Vars}(\sigma) \rightarrow \text{Vars}(\sigma')$.*

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

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

$\tau : \text{Vars}(\sigma) \rightarrow \text{Vars}(\sigma')$ *bijektiv*: Es gilt

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

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

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

4.0.2 Unifikationsalgorithmus von Martelli/Montanari

Definition 37 (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 $\text{Unif}(S_1) = \text{Unif}(S_2)$. Ein zu S äquivalentes gelöstes Gleichungssystem heißt *gelöste Form* von S .

Lemma 38. *Wenn $S' = \{X_1 \doteq E_1, \dots, X_n \doteq E_n\}$ gelöste Form von S ist, dann ist $\sigma = [X_1 \mapsto E_1, \dots, X_n \mapsto E_n]$ 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' . \square

Der Algorithmus (ursprünglich Herbrand, später Martelli/Montanari) besteht nun in erschöpfender Anwendung der folgenden Rewrite-Regeln:

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

$$S \cup f(E_1, \dots, E_n) \doteq f(D_1, \dots, D_n) \rightarrow S \cup \{E_1 \doteq D_1, \dots, E_n \doteq D_n\}$$

$$S \cup f(E_1, \dots, E_n) \doteq g(D_1, \dots, D_k) \rightarrow \perp$$

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

$$S \cup (X \doteq E) \rightarrow \begin{cases} \perp & (X \in \text{Vars}(E)) \\ S[E/X] \cup (X \doteq E) & (X \notin \text{Vars}(E), X \in \text{Vars}(S)) \end{cases}$$

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

Beispiele:

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

- $f(X, g(X)) \doteq f(Z, Z)$

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

$$x_0 > x_1 > x_2 > \dots$$

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

Es gilt nun in der Tat

Satz 39. *Die lexikographische Ordnung ist eine Wohlordnung, d.h. es gibt keine unendliche absteigende Kette*

$$\vec{a}^0 \succ \vec{a}^1 \succ \vec{a}^2 \succ \dots \quad (3)$$

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

$$a_1^0 \geq a_1^1 \geq a_1^2 \geq \dots$$

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

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

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

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 $X \in \text{Vars}(E)$ ist aber offenbar nicht unifizierbar: der Term $E\sigma$ ist stets größer als der Term $X\sigma$.

5 Prolog

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

Definition 40. Wir erweitern im folgenden den Begriff der *Signatur* dahingehend, dass eine Signatur Σ auch Prädikatensymbole p, q, \dots 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

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

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

Definition 41 (definite Klausel, Zielklausel, Hornklausel, Programm, Anfrage). Ein *Literal* ist ein Atom $A = p(E_1, \dots, E_k)$ 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:

$$\begin{array}{ll} \text{definite Klausel:} & \{A_0, \neg A_1, \dots, \neg A_n\} \equiv A_0 \leftarrow A_1, \dots, A_n \\ \text{Zielklausel:} & \{\neg A_1, \dots, \neg A_n\} \equiv \leftarrow A_1, \dots, A_n \end{array}$$

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

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

$Q_1 = \leftarrow \text{Großvater(Hans, X)}$: $[Fritz/X]$

$Q_2 = \leftarrow \text{Großvater(Y, Paula)}$: 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*

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

wobei σ allgemeinsten 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, \dots in den B_i bei der k -ten Regelanwendung Variablen X_k, Y_k, \dots werden).

Beispiel 43 (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: $\text{append}([], X, X)$ (lies: $\text{append}([], X) = X$)
 $\text{append}([X|Y], Z, [X|W]) \leftarrow \text{append}(Y, Z, W)$

Notation Zur Vereinfachung führen wir folgende abkürzenden Schreibweisen ein:

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

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

Beispiel 44 (Auswertung von logischen Ausdrücken durch Prolog).

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

$Q_2 : \leftarrow \text{append}([0, 1], X, [0, 1, 1])$
 $\leftarrow \text{append}([1], X, [1, 1])$
 $\leftarrow \text{append}([], X, [1])$
 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.

$Q : \leftarrow \text{append}(X, Y, [0, 1])$

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

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

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

b.ii) $X_1 \mapsto [1|X_2]$

$\leftarrow \text{append}(X_2, Y, [])$, 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

Der Ausdruck *SLD-Resolution* steht für *Linear resolution for Definite clauses with Selection function*. Er bezeichnet die bei der Ausführung von Prolog-Programmen durchgeführte Transformation von Zielen (Anfragen) mittels *Regeln* (inklusive 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

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

Beispiel 45. Auf die Anfrage

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

soll die Regel

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

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

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

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 46 (SLD-Herleitung). Gegeben P, \mathcal{R} : Eine *SLD-Herleitung* von Ziel Q_0 ist eine Sequenz

$$\delta = Q_0 \xrightarrow{D_0} Q_1 \xrightarrow{D_1} Q_2 \dots$$

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

Wie oben schon angedeutet, werden dabei die Variablen in D_i per $X \mapsto X_i$ umbenannt, um Konflikte mit den Variablen von Q_i zu vermeiden.

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

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

Eine Herleitung δ ist eine *Refutation*, wenn sie in der leeren Klausel endet, d.h. wenn $Q_{n+1} = \square$. 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 $B_0 \leftarrow B_1, \dots, B_n \in P$ bei $Q_{n+1} = \leftarrow A_1, \dots, A_n$ gilt, dass A_1 nicht unifizierbar mit B_0 ist, also keine weitere Regel mehr anwendbar ist.

(Allgemein heißt δ *vollständig*, wenn δ eine Refutation, fehlgeschlagen oder unendlich ist.)

Beispiel 47 (Abstammung). Wir versuchen, ein Prädikat D („Descendant“) zu programmieren per

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

Nun wollen wir testen, ob a von sich selbst abstammt:

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

Wir versinken hier also in einer Endlosrekursion.

Beispiel 48 (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. Prolog findet dann beispielsweise folgende Refutation:

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

Die entsprechende Antwort ergibt sich aus der kumulierten Substitution: 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

Systeme formalen Schließens dienen der rein syntaxbasierten *Herleitung* von Formeln, die einfach als zu manipulierende Zeichenketten angesehen werden. Ein Deduktionssystem besteht typischerweise aus *Axiomen*, also Formeln, die ohne weitere Voraussetzungen als hergeleitet hingeschrieben werden können, sowie aus *Regeln*, die festlegen, wie man aus bereits hergeleiteten Formeln bzw. schon durchgeführten Herleitungen neue Formeln herleitet. Im einfachsten Fall hat eine Regel die Form

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

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

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

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

$$(mp) \frac{\varphi \quad \varphi \rightarrow \psi}{\psi}$$

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

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, die in hierarchisch strukturierten Beweisen rekursiv aufgebaut werden.

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

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

Beispiel 49 (Regeln für eine auf Konjunktion beschränkte Logik).

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

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

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. $\{\varphi \wedge \psi\} \vdash \psi \wedge \varphi$:

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

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

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. Die lokale Annahme verschwindet mit dem Ende des Unterbeweises (daher der Name), d.h. sie steht zum einen außerhalb des Unterbeweises nicht als Prämisse für Regelanwendungen zur Verfügung, zum anderen ist aber der äußere Beweis auch von ihr unabhängig, d.h. liefert Herleitungen mit entsprechend weniger Annahmen (also stärkere Aussagen). Solche Unterbeweise werden benötigt, wenn wir unser Regelwerk auf Negation und \perp erweitern:

$(\neg I)$	φ \cdot \cdot \perp	$(\perp I) \frac{\varphi \quad \neg\varphi}{\perp}$	$(\perp E) \frac{\perp}{\Phi}$	$(\neg E) \frac{\neg\neg\varphi}{\varphi}$
	$\neg\varphi$			

Beispiel 50 ($\vdash \neg(\varphi \wedge \neg\varphi)$).

1	$\varphi \wedge \neg\varphi$	
2	φ	$(\wedge E1), 1$
3	$\neg\varphi$	$(\wedge E1), 1$
4	\perp	$(\perp I), 2, 3$
5	$\neg(\varphi \wedge \neg\varphi)$	$(\neg I), 1 - 4$

Erweiterung auf Disjunktion und Implikation

$$\begin{array}{c}
 \varphi \quad \psi \\
 \vdots \quad \vdots \quad \varphi \vee \psi \\
 (\vee I1) \frac{\varphi}{\varphi \vee \psi} \quad (\vee I2) \frac{\psi}{\varphi \vee \psi} \quad (\vee E) \frac{\chi \quad \chi}{\chi} \\
 \varphi \\
 \vdots \\
 (\rightarrow E) \frac{\varphi \rightarrow \psi \quad \varphi}{\psi} \quad (\rightarrow I) \frac{\psi}{\varphi \rightarrow \psi}
 \end{array}$$

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

$$\begin{array}{l}
 \begin{array}{|l}
 \neg\neg\varphi \wedge \neg\psi \\
 \hline
 \neg\neg\varphi \\
 \varphi \\
 \vdots \\
 \perp
 \end{array} \\
 \neg(\neg\neg\varphi \wedge \neg\psi) \quad (\neg I)
 \end{array}
 \begin{array}{l}
 (\wedge E_1) \\
 (\neg E) \\
 \\
 \text{Annahme}
 \end{array}$$

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

$$\begin{array}{l}
 1 \quad \begin{array}{|l} \varphi \end{array} \\
 2 \quad \begin{array}{|l} \neg\varphi \vee \psi \end{array} \\
 3 \quad \begin{array}{|l} \neg\varphi \end{array} \\
 4 \quad \begin{array}{|l} \perp \end{array} \quad (\perp I) \ 1, 3 \\
 5 \quad \begin{array}{|l} \psi \end{array} \quad (\perp E) \\
 6 \quad \begin{array}{|l} \psi \end{array} \\
 7 \quad \begin{array}{|l} \psi \end{array} \\
 8 \quad \psi \quad (\vee E) \ 3-5, 6-7, 2
 \end{array}$$

Satz 51. (Korrektheit)

$$\Phi \vdash \psi \Rightarrow \Phi \models \psi$$

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 $n < k$ gilt.

Wir unterscheiden dann nach der zuletzt angewandten Regel, z.B. a) $\wedge I$ (hier im Skript ausgelassen), b) $(\neg I)$: Der Unterbeweis in der Prämisse bedeutet, dass $\Phi \cup \{\varphi\} \vdash \perp$, 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 $\Phi \cup \{\varphi\} \vDash \perp$; mit anderen Worten, $\Phi \cup \{\varphi\}$ ist unerfüllbar. Damit folgt wie verlangt $\Phi \vDash \neg\varphi$: Wenn $\kappa \vDash \Phi$, dann $\kappa \not\vDash \psi$, also $\kappa \vDash \neg\psi$. Die anderen Regeln für \wedge , \neg und \perp sind noch wesentlich einfacher; die Regeln für \vee und \rightarrow sind dann ebenfalls korrekt, da wir sie ja aus denen für \wedge , \neg und \perp herleiten. \square

6.1 Vollständigkeit

In Umkehrung zur Korrektheit gilt auch

Satz 52. *Vollständigkeit*

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

Definition 53. Eine Formelmenge Φ heißt *konsistent*, wenn $\Phi \not\vDash \perp$, 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

$$\Phi \text{ konsistent} \Rightarrow \Phi \text{ erfüllbar:}$$

Sei $\Phi \vDash \psi$; dann ist $\Phi \cup \{\neg\psi\}$ unerfüllbar, also nach obigem inkonsistent, also $\Phi \cup \{\neg\psi\} \vdash \perp$. 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

$$\textit{Lindenbaumlemma: } \Phi \text{ konsistent} \Rightarrow \text{es existiert } \bar{\Phi} \text{ max. kons mit } \Phi \subseteq \bar{\Phi}.$$

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

\square

Dazu:

Lemma 54. (C) Sei Φ konsistent, dann $\Phi \cup \{\psi\}$ konsistent oder $\Phi \cup \{\neg\psi\}$ konsistent.

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

Beweis (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 $\varphi_1, \varphi_2, \varphi_3, \dots$ Aufzählung aller Formeln. Konstruiere Kette $\Phi = \Phi_0 \subseteq \Phi_1 \subseteq \Phi_2 \subseteq \dots$ 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 $\bar{\Phi} := \bigcup_{i=0}^{\infty} \Phi_i \supseteq \Phi$. Zu zeigen:

1. $\bar{\Phi}$ ist konsistent: Nimm an $\bar{\Phi} \vdash \perp$. Der Beweis ist endlich, verwendet also nur endlich viele Annahmen aus $\bar{\Phi}$. Jede dieser Annahmen kommt in einem Φ_i vor; durch Wahl des größten unter diesen endlich vielen Indizes i erhalten wir ein Φ_i , das *alle* verwendeten Annahmen enthält. Dann $\Phi_i \vdash \perp$, im Widerspruch zur Konsistenz von Φ_i .
2. $\bar{\Phi}$ maximal: Sei Ψ konsistent und $\bar{\Phi} \subseteq \Psi$. Zu zeigen ist dann $\Psi \subseteq \bar{\Phi}$. 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 $\phi_n \in \Phi_{n+1} \subseteq \bar{\Phi}$.

\square

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

Lemma 55. (*Hintikka-Eigenschaften*) Sei Φ maximal konsistent. Dann gilt

1. $\perp \notin \Phi$
2. $\neg\psi \in \Phi \iff \psi \notin \Phi$
3. $\phi \wedge \psi \in \Phi \iff \phi \in \Phi \text{ und } \psi \in \Phi$

Beweis. ad (1): Klar.

ad (2): „ \implies “ klar (verwendet $(\perp I)$), „ \impliedby “: 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): „ \implies “: Per Maximalität von Φ reicht es, zu zeigen, dass $\Phi \cup \{\phi\}$ und $\Phi \cup \{\psi\}$ konsistent sind. Z.B.: Nimm an, dass $\Phi \cup \{\phi\} \vdash \perp$. Per $(\wedge E1)$ gilt $\Phi \vdash \phi$, also schon $\Phi \vdash \perp$ im Widerspruch zur Konsistenz von Φ . Analog zeigt man, dass $\Phi \cup \{\psi\}$ konsistent ist.

„ \impliedby “ läuft analog mit $(\wedge I)$. \square

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 56. (*Wahrheitslemma*) $\kappa \models \psi \iff \psi \in \Phi$.

Beweis. Induktion über ψ per Definition und Hintikka-Eigenschaften, z.B.: $\kappa \models \neg\psi \iff \kappa \not\models \psi \iff \psi \notin \Phi \stackrel{\text{Hintikka}}{\iff} \neg\psi \in \Phi$ \square

Korollar 57. (*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 $\Phi \vdash \perp$ genau dann, wenn es eine endliche Teilmenge $\Phi_0 \subseteq \Phi$ gibt mit $\Phi_0 \vdash \perp$. Das ist aber klar, da ein Beweis von \perp aus Φ nur endlich viele der Annahmen in Φ verwendet. \square

6.2 Anwendungen des Kompaktheitssatzes

Definition 58. Ein (ungerichteter) Graph mit Knotenmenge V und Kantenmenge E heißt k -färbbar, wenn es eine Abbildung $c : V \rightarrow \{1, \dots, k\}$ (*Colouring/Färbung*) gibt, so dass für jede Kante $\{v, w\} \in E$ $c(v) \neq c(w)$ gilt.

Satz 59. 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 $v \in V$, $i \in \{1, \dots, k\}$ ein, mit der Lesart „Knoten v hat Farbe i “ (also $c(v) = i$). Wir definieren Formelmengen Φ_1, Φ_2, Φ_3 wie folgt:

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

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

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

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

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

– 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 $V_0 = \{v \mid \exists i. A_{v,i} \in \text{At}(\Psi)\}$ 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. \square

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

Ein *Prädikat* ist eine Eigenschaft, die Individuen haben (oder nicht haben) können, d.h. eine Aussage mit Parametern, für die Individuen einzusetzen sind; Beispiel: $\text{positiv}(n)$ oder $\text{verheiratetMit}(x, y)$. Diese Parametrisierung unterscheidet Prädikate von den Atomen der Aussagenlogik, die nur schlechthin wahr oder falsch sein können (wie z.B. 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 60 (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* $\text{ar}(s) \geq 0$. Für $s \in P_\Sigma \cup F_\Sigma$ und $\text{ar}(s) = n$ schreiben wir oft $s/n \in \Sigma$, wobei wir typischerweise dadurch die Unterscheidung zwischen Funktionen- und Prädikatensymbolen sicherstellen, dass wir erstere mit Kleinbuchstaben und letztere mit Großbuchstaben bezeichnen. Ein nullstelliges Funktionsymbol $c/0 \in \Sigma$ 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, \dots sind dann gegeben durch die BNF

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

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

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

Die durch die erste zwei Klauseln konstruierte Formeln heißen *atomare Formeln*. Der Existenzquantor \exists wird dann definiert durch $\exists X.\varphi := \neg\forall X.\neg\varphi$, außerdem $\perp \equiv \neg\forall X.X = X$. 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 61 (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 $\mathfrak{M}[[f]] : M^n \rightarrow M$
- 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 $\eta : V \rightarrow M$.

Definition 62 (Erfülltheit). Zu einem Term E definieren wir seine Bedeutung $\mathfrak{M}[[E]]\eta \in M$ rekursiv durch

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

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

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

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

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

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

$$\begin{aligned} FV(E) &= \text{Vars } E \\ FV(E = D) &= FV(E) \cup FV(D) \\ FV(P(E_1, \dots, E_n)) &= \bigcup_{i=1}^n FV(E_i) \\ FV(\neg \varphi) &= FV(\varphi) \\ FV(\varphi \wedge \psi) &= FV(\varphi) \cup FV(\psi) \\ FV(\forall X. \varphi) &= FV(\varphi) \setminus \{X\} \end{aligned}$$

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

Definition 64 (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 65. 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 65. 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 $X \in FV(E)$

$$\mathfrak{M}[[E]]_{\eta_1} = \mathfrak{M}[[E]]_{\eta_2} \quad (4)$$

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. $\mathfrak{M}, \eta_1 \models E = D \iff \mathfrak{M}[[E]]_{\eta_1} = \mathfrak{M}[[D]]_{\eta_1} \stackrel{(4)}{\iff} \mathfrak{M}[[E]]_{\eta_2} = \mathfrak{M}[[D]]_{\eta_2} \iff \mathfrak{M}, \eta_2 \models E = D$. Die Booleschen Fälle sind trivial. Es bleibt der Allquantor:

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

Zur Anwendung der Induktionsvoraussetzung ist hierbei zu zeigen, dass

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

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

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}, η (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 66 (Logische Konsequenz). $\exists X.\forall Y.Loves(Y, X) \models \forall Y.\exists X.L(Y, X)$

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

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

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. □

$\forall Y. \exists X. L(Y, X) \not\equiv \exists X. \forall Y. L(Y, X)$: Gegenmodell z.B. $M = \{0, 1\}$, $\mathfrak{M} \llbracket L \rrbracket = \{(1, 0), (0, 1)\}$.

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

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

Der entscheidende Unterschied zwischen $\forall Y. \exists X. L(Y, X)$ und $\exists X. \forall Y. L(Y, X)$ ist also der, dass im Erfülltheitsspiel für $\forall Y. \exists X. L(Y, X)$ Abaelard den ersten Zug macht, in dem für $\exists X. \forall Y. L(Y, X)$ dagegen Eloise.

Beispiel 67 (Logische Äquivalenz).

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

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 68 (Substitutionslemma). *Es gilt*

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

wobei $\eta_\sigma(X) = \mathfrak{M} \llbracket \sigma(X) \rrbracket \eta$ für $X \in V$.

(Man beachte, dass $\eta_\sigma(X) = \eta(X)$ für $X \notin \text{Dom}(\sigma)$: dann haben wir nämlich $\mathfrak{M} \llbracket \sigma(X) \rrbracket \eta = \mathfrak{M} \llbracket X \rrbracket \eta = \eta(X)$.)

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

$$\mathfrak{M} \llbracket E\sigma \rrbracket \eta = \mathfrak{M} \llbracket E \rrbracket \eta_\sigma.$$

□

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

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

Damit beweist man andere erwartete Eigenschaften von Gleichheit, etwa Symmetrie

$$\begin{array}{l|l}
 1 & E = D \\
 2 & \hline E = E & (\text{=I}) \\
 3 & D = E & (\text{=E}) 1,2
 \end{array}$$

und Transitivität

$$\begin{array}{l|l}
 1 & E = D \\
 2 & D = F \\
 3 & \hline E = F & (\text{=E}) 1,2
 \end{array}$$

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

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

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

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

wobei c eine *frische* Konstante ist, d.h. außer in dem betreffenden Unterbeweis nirgendwo vorkommt (dies wird durch die Box um das c angedeutet).

Die Einführungsregel für \exists ist ebenso intuitiv:

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

D.h. „Wenn ϕ für ein c gilt, dann gibt es ein X , für das ϕ gilt“.

Etwas gewöhnungsbedürftiger ist die Eliminationsregel:

$$(\exists\text{E}) \frac{\exists X. \phi \quad \begin{array}{|l} \boxed{c} \phi[c/X] \\ \vdots \\ \psi \end{array}}{\psi}$$

wiederum mit der Seitenbedingung, dass c frisch ist; dies bedeutet hier insbesondere, dass c nicht in ψ vorkommen darf.

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

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

Beweis von $(\exists \text{ E})$

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

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

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

It.

Transitivität der Subsumption

1	$\forall X. (P(X) \rightarrow Q(X))$	
2	$\forall Z. (Q(Z) \rightarrow R(Z))$	
3	<div style="border-left: 1px solid black; padding-left: 10px;"> $\boxed{d} \quad P(d)$ </div>	
4	<div style="border-left: 1px solid black; padding-left: 10px;"> $P(d) \rightarrow Q(d)$ </div>	($\forall E$) 1
5	<div style="border-left: 1px solid black; padding-left: 10px;"> $Q(d)$ </div>	($\rightarrow E$) 3, 4
6	<div style="border-left: 1px solid black; padding-left: 10px;"> $Q(d) \rightarrow R(d)$ </div>	($\forall E$) 2
7	<div style="border-left: 1px solid black; padding-left: 10px;"> $R(d)$ </div>	($\rightarrow E$) 5, 6
8	$\forall X. (P(X) \rightarrow R(X))$	($\forall I^*$) 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, da sie, wie oben gesehen, keine logische Folgerung, per Korrektheit also auch nicht herleitbar ist.

1	$\exists X. \forall Y. P(X, Y)$	
2	<div style="border-left: 1px solid black; padding-left: 10px;"> $\boxed{c} \quad \forall Y. P(c, Y)$ </div>	
3	<div style="border-left: 1px solid black; padding-left: 10px;"> <div style="border-left: 1px solid black; padding-left: 10px;"> \boxed{d} </div> </div>	
4	<div style="border-left: 1px solid black; padding-left: 10px;"> <div style="border-left: 1px solid black; padding-left: 10px;"> $P(c, d)$ </div> </div>	($\forall E$) 2
5	<div style="border-left: 1px solid black; padding-left: 10px;"> <div style="border-left: 1px solid black; padding-left: 10px;"> $\exists X. P(X, d)$ </div> </div>	($\exists I$) 4
6	<div style="border-left: 1px solid black; padding-left: 10px;"> $\forall Y. \exists X. P(X, Y)$ </div>	($\forall I$) 3–5
7	$\forall Y. \exists X. P(X, Y)$	($\exists E$) 2–6

Weitere Beispiele

$$\begin{aligned} \exists X. (\phi \vee \psi) &\equiv \exists X. \phi \vee \exists X. \psi \\ \phi \wedge \exists X. \psi &\equiv \exists X. (\phi \wedge \psi) && (X \notin FV(\phi)) \\ \phi \vee \exists X. \psi &\equiv \exists X. (\phi \vee \psi) && (X \notin FV(\phi)) \end{aligned}$$

8 Quantorenelimination

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

Definition 69 (Theorie). Eine *Theorie* $\mathcal{T} = (\Sigma, \Phi)$ besteht aus einer Signatur Σ und einer Menge Φ von Σ -Sätzen, ihren *Axiomen*. Ein *Modell* von \mathcal{T} ist ein Σ -Modell \mathfrak{M} mit $\mathfrak{M} \models \Phi$. Eine Σ -Formel ψ heißt *erfüllbar über \mathcal{T}* , wenn ψ in einem Modell von \mathcal{T} erfüllbar ist, d.h. wenn $\Phi \cup \{\psi\}$ erfüllbar ist. Wir schreiben $\mathcal{T} \models \psi$, wenn $\neg\psi$ unerfüllbar über \mathcal{T} ist, d.h. wenn $\Phi \models \psi$.

Beispiel 70. Die Theorie $\mathcal{T}_<$ der *dichten linearen Ordnungen ohne Endpunkte* hat die Signatur

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

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

$$\begin{array}{ll} \forall X. \neg(X < X) & \text{(Irreflexivität)} \\ \forall X, Y, Z. (X < Y \wedge Y < Z \rightarrow X < Z) & \text{(Transitivität)} \\ \forall X, Y. (X < Y \vee X = Y \vee Y < X) & \text{(Trichotomie)} \\ \forall X, Y. (X < Y \rightarrow \exists Z. (X < Z \wedge Z < Y)) & \text{(Dichte)} \\ \forall X. \exists Y. Y < X & \\ \forall X. \exists Y. X < Y & \text{(Endpunktfreiheit).} \end{array}$$

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

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

Definition 71. Eine Formel heißt *quantorenfrei*, wenn Sie, nun ja, keine Quantoren enthält. Eine Theorie $\mathcal{T} = \{\Sigma, \Phi\}$ hat *Quantorenelimination*, wenn für jede Σ -Formel ϕ eine quantorenfreie Σ -Formel ϕ' berechenbar ist, so dass $\mathcal{T} \models \phi \leftrightarrow \phi'$.

Fakt 72. Wenn $\mathcal{T} = (\Sigma, \Phi)$ *Quantorenelimination* hat und *Erfüllbarkeit von quantorenfreien Formeln über \mathcal{T} entscheidbar* ist, dann ist \mathcal{T} *entscheidbar*.

Quantorenelimination lässt sich auf einen Spezialfall reduzieren:

Lemma 73. Wenn für jede Σ -Formel ϕ der Form $\exists X. (\alpha_1 \wedge \dots \wedge \alpha_n)$ mit Literalen α_i eine quantorenfreie Formel ϕ' mit $\mathcal{T} \models \phi \leftrightarrow \phi'$ berechenbar ist, dann hat \mathcal{T} *Quantorenelimination*.

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

Damit zeigen wir

Satz 74. Die Theorie der *dichten linearen Ordnungen ohne Endpunkte* hat *Quantorenelimination*.

Beweis. Sei $\phi = \exists X. (\alpha_1 \wedge \dots \wedge \alpha_n)$ mit Literalen α_i . Wir gehen schrittweise vor:

1. Wir eliminieren zunächst Negation: Per Irreflexivität, Transitivität und Trichotomie gilt $\mathcal{T} \models (\neg Y = Z) \leftrightarrow (Y < Z \vee Z < Y)$ und $\mathcal{T} \models (\neg(Y < Z) \leftrightarrow (Y = Z \vee Z < Y))$.
2. Damit erreichen wir eine Formel, die keine Negationen mehr, dafür aber wieder Disjunktionen enthält. Wie im Beweis von Lemma 73 formen wir ϕ dann in eine Disjunktion von existenzquantifizierten Formeln um; es reicht offenbar aus, zu zeigen, dass wir aus einer solchen Formel den Quantor eliminieren können.
3. Damit erreichen wir eine Formel $\psi = \exists X. (\beta_1 \wedge \dots \wedge \beta_n)$ mit *Atomen* β_i . Wir eliminieren nun =:
 - Wenn $\beta_i = (Y = Y)$, dann streichen wir β_i .
 - Wenn $\beta_i = (Y = Z)$, dann streichen wir β_i und ersetzen in die verbleibenden β_j jeweils durch $\beta_j[Y \mapsto Z]$.
4. Von den verbleibenden Atomen ziehen wir nunmehr diejenigen, die X nicht erwähnen, vor den Existenzquantor (s.o.).
5. Falls unter den verbleibenden β_j $X < X$ vorkommt, ersetzen wir ψ durch \perp (per Irreflexivität).
6. Die verbleibende Formel hat nunmehr (nach einfacher Umformung) die Form

$$\exists X. \left(\bigwedge_{i=1}^n (U_i < X) \wedge \bigwedge_{j=1}^m (X < V_j) \right), \quad (6)$$

wobei die U_i und die V_j von X verschiedene Variablen sind. Diese Formel ist in $\mathcal{T}_<$ äquivalent zu

$$\bigwedge_{i=1}^n \bigwedge_{j=1}^m U_i < V_j. \quad (7)$$

Dabei ist die Implikation $(6 \implies 7)$ klar per Transitivität. Für die Umkehrung bemerken wir, dass es per Trichotomie unter den U_i ein größtes geben muss; sei dies U_{i_0} (formal beweist man per Induktion eine Disjunktion mit n Disjunkten, eins für jedes in Frage kommende U_i). Entsprechend gibt es unter den V_j ein kleinstes, V_{j_0} . Dann gilt nach Voraussetzung $U_{i_0} < V_{j_0}$, so dass per Dichte X existiert mit $U_{i_0} < X$ und $X < V_{j_0}$; dieses X ist dann per Transitivität ein Zeuge für (6). Diese Argumentation setzt natürlich voraus, dass $n, m > 0$. Für $n > 0, m = 0$ und für $n = 0, m > 0$ führt man ein ähnliches Argument mittels Endpunktfreiheit durch; der Fall $n = m = 0$ ist trivial.

□

Um nun tatsächlich Entscheidbarkeit von $\mathcal{T}_<$ zu bekommen, fehlt uns noch die Entscheidbarkeit der Erfüllbarkeit quantorenfreier Formeln über \mathcal{T} . Wir beobachten zunächst, dass bei der Quantorenelimination für $\mathcal{T}_<$ in Wirklichkeit nicht nur die Quantoren, sondern auch \neg und $=$ eliminiert werden. Ferner können wir mit Disjunktionen stets umgehen, indem wir zunächst in DNF transformieren und dann die Disjunkte der DNF der Reihe nach auf Erfüllbarkeit prüfen (natürlich hören wir damit auf, sobald wir einen erfüllbaren Disjunkt gefunden haben). Das verbleibende Problem ist, die Erfüllbarkeit einer Konjunktion atomarer Formeln der Form $X < Y$ zu zeigen. Zur Vereinfachung der Notation schreiben wir die Konjunktion einfach als Menge C von Formeln der Form $X < Y$. Folgender Algorithmus entscheidet Erfüllbarkeit von C über $\mathcal{T}_<$:

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

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

Lemma 75. *Wenn mit $X < Y$ und $Y < Z$ stets auch $X < Z$ in C ist und C keine Formel der Form $X < X$ enthält, dann ist C erfüllbar über $\mathcal{T}_<$.*

Beweis. Induktion über die Anzahl Variablen in C . Wähle eine in C vorkommende Variable und setze $C \setminus X = \{Y < Z \in C \mid Y, Z \text{ verschieden von } X\}$. Damit erfüllt $C \setminus X$ offenbar die Voraussetzung des Lemmas, ist also nach Induktionsvoraussetzung erfüllbar über $\mathcal{T}_<$, da es eine Variable weniger als C enthält. Seien also \mathfrak{M} ein Modell von $\mathcal{T}_<$ und η eine Umgebung in \mathfrak{M} mit $\mathfrak{M}, \eta \models C \setminus X$. Seien nun U_1, \dots, U_n die Variablen mit $U_i < X \in C$ und V_1, \dots, V_m die mit $X < V_j \in C$. Unter den $\eta(U_i)$ gibt es per Trichotomie einen größten Wert $l \in M$, und unter den $\eta(V_j)$ einen kleinsten, $r \in M$. Es gilt dann per Transitivität $l < r$ in \mathfrak{M} (wobei wir $\mathfrak{M}[\llbracket \cdot \rrbracket]$ einfach als $<$ schreiben), so dass per Dichte $x \in M$ existiert mit $l < x$ und $x < r$. Da $X < X \notin C$, gilt dann per Transitivität $\mathfrak{M}, \eta[X \mapsto x] \models C$. \square

9 Vollständigkeit der Prädikatenlogik erster Stufe

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

Satz 76 (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 ($\Phi \models \psi$), dann ist ψ aus Φ herleitbar ($\Phi \vdash \psi$).*

Bemerkung 77. 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 Aussagenlogik 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* $c_{\phi(X)}$ enthält, gedacht als Name für ein $\phi(X)$ erfüllendes Element, wenn ein solches existiert. Ebenso erweitern wir die gegebene Formelmenge Φ um eine unendliche Menge \mathcal{H} zusätzlicher Axiome, die *Henkin-Theorie*. Diese enthält insbesondere für jede Formel $\phi(X)$ ein Axiom $(\exists X. \phi(X)) \rightarrow \phi(c_{\phi(X)})$.

- (B) (*Henkin-Elimination*) Wir beweisen dann für jede Formel ρ über Σ (die also keine Zeugenkonstanten erwähnt), dass aus $\Phi \cup \mathcal{H} \vdash \rho$ 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 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 $\kappa \models \Phi \cup \mathcal{H}$ (*aussagenlogische Erfülltheit*) konstruieren wir dann ein Σ -Modell \mathfrak{M}_κ mit $\mathfrak{M}_\kappa \models \Phi$ (*Erfülltheit in Logik erster Stufe*).

Damit läuft der Beweis des Vollständigkeitsatzes 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 $\Phi \cup \mathcal{H}$ 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 $\Phi \cup \mathcal{H}$ erfüllbar als Menge aussagenlogischer Formeln; nach der Henkin-Konstruktion 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. $\exists y. P(y, c_{\varphi(c)})$. 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*

$$\min\{i \mid c_{\varphi(X)} \in \Sigma_i\}$$

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

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

$$\mathbf{H1:} \quad (\exists x. \varphi(x)) \rightarrow \varphi(c_{\varphi(c)})$$

$$\mathbf{H2:} \quad \varphi(c) \rightarrow \exists x. \varphi(x)$$

$$\mathbf{H3:} \quad \neg \forall x. \varphi(x) \leftrightarrow \exists x. \neg \varphi(x)$$

(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 \mathcal{H}_0 , so dass

$$\Phi \cup \mathcal{H}_0 \cup \{(\exists X. \varphi(X)) \rightarrow \varphi(c_{\varphi(X)})\} \vdash \rho.$$

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

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

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

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

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

(C): Wir setzen

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

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

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

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 $\rho = P(c_1, \dots, c_n)$; für solche Formeln gilt die Behauptung nach Konstruktion von \mathfrak{M}_κ . 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:

$$\begin{aligned} \mathfrak{M}_\kappa \models \forall X. \varphi(X) &\Leftrightarrow \mathfrak{M}_\kappa \not\models \underbrace{\exists X. \neg \varphi(X)}_{\substack{\text{kleiner als} \\ \forall X. \varphi(X)}} \\ &\stackrel{IV}{\Leftrightarrow} \kappa \not\models \exists X. \neg \varphi(X) \\ &\stackrel{H3}{\Leftrightarrow} \kappa \not\models \neg \forall X. \varphi(X) \\ &\Leftrightarrow \kappa \models \forall X. \varphi(X). \end{aligned}$$

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 $c \in \Sigma_{\mathcal{H}}$ mit $\mathfrak{M}_\kappa \models \varphi(c)$. Da die Formel $\varphi(c)$ kleiner als $\exists X. \varphi(X)$ ist, folgt nach Induktionsvoraussetzung $\kappa \models \varphi(c)$. Mit H2 erhalten wir schließlich $\kappa \models \exists X. \varphi(X)$.

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

Korollar 78 (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. \square

Beispiel 79. Sei Φ eine Axiomatisierung der natürlichen Zahlen in Logik erster Stufe (z.B. die Peano-Axiome, mit 0 und Sukzessorfunktion s sowie einem Axiomenschema für Induktion). Dann ist $\Phi \cup \{c > s^n(0) \mid n \in \mathbb{N}\}$ endlich erfüllbar, somit nach Kompaktheit erfüllbar — d.h. wir können in Logik erster Stufe durch keine noch so raffinierte Axiomatisierung die Existenz von Nicht-Standard-Zahlen ausschließen.

Bemerkung 80. Vollständigkeit ist durchaus keine selbstverständliche Eigenschaft einer Logik. Man unterscheidet im allgemeinen zwischen *starker Vollständigkeit*, wie wir sie oben für Prädikatenlogik erster Stufe bewiesen haben, und *schwacher Vollständigkeit*. Eine Logik, mit logischer Folgerungsrelation \models , ist per Definition stark vollständig, wenn aus $\Phi \models \psi$ stets $\Phi \vdash \psi$ folgt (*Korrektheit* ist die umgekehrte Implikation), und schwach vollständig, wenn jede gültige Formel herleitbar ist, d.h. wenn aus $\models \psi$ stets $\vdash \psi$ folgt. Wenn eine Logik korrekt und stark vollständig ist und außerdem das Beweissystem *finitär* ist, d.h. wenn jede Regel nur endlich viele Prämissen hat (das schließt Regeln aus, die z.B. aus $\phi(n)$ für alle konkreten natürlichen Zahlen n die Formeln $\forall x \in \mathbb{N}. \phi(x)$ schließen), dann ist die Logik, mit der gleichen Argumentation wie oben, *kompakt*, d.h. jede endlich erfüllbare Formelmengung ist erfüllbar.

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

$$\begin{aligned} \forall P. ((P(0) \wedge \forall x. (P(x) \rightarrow P(\text{suc}(x)))) \rightarrow \forall x. P(x)) \\ \forall x. \neg \text{suc}(x) = 0 \\ \forall x, y. (s(x) = s(y) \rightarrow x = y) \\ \forall x. ((0 < x \vee 0 = x) \wedge \neg x < 0) \\ \forall x, y. (s(x) < s(y) \leftrightarrow x < y). \end{aligned}$$

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

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

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